



LUKAS STANKEVIČIUS

LEARNABLE TEXT REPRESENTATION MODELS AND THEIR EVALUATION

DOCTORAL DISSERTATION

K a u n a s
2 0 2 6

KAUNAS UNIVERSITY OF TECHNOLOGY

LUKAS STANKEVIČIUS

LEARNABLE TEXT REPRESENTATION MODELS AND THEIR EVALUATION

Doctoral dissertation
Natural Sciences, Informatics (N 009)

2026, Kaunas

The dissertation has been prepared at the Department of Software Engineering of the Faculty of Informatics of Kaunas University of Technology in 2019–2025.

The doctoral right has been granted to Kaunas University of Technology together with Vytautas Magnus University and Vilnius Gediminas Technical University.

Research Supervisor:

Assoc. Prof. Dr. Mantas LUKOŠEVIČIUS (Kaunas University of Technology, Natural Sciences, Informatics, N 009).

Edited by: English language editor Dr. Armandas Rumšas (Publishing House *Technologija*), Lithuanian language editor Aurelija Gražina Rukšaitė (Publishing House *Technologija*).

Dissertation Defence Board of Informatics Science Field:

Prof. Dr. Hab. Rimantas BARAUSKAS (Kaunas University of Technology, Natural Sciences, Informatics, N 009) – **chairperson**;

Prof. Dr. Andrius KRIŠČIŪNAS (Kaunas University of Technology, Natural Sciences, Informatics, N 009);

Prof. Dr. Sanda MARTINČIĆ-IPŠIĆ (University of Rijeka, Croatia, Natural Sciences, Informatics, N 009);

Prof. Dr. Agnė PAULAUSKAITĖ-TARASEVIČIENĖ (Kaunas University of Technology, Natural Sciences, Informatics, N 009);

Prof. Dr. Simona RAMANAUSKAITĖ (Vilnius Gediminas Technical University, Natural Sciences, Informatics, N 009).

The dissertation defence will be held on 15 June 2026, at 10 a.m. at the Senate Hall of Kaunas University of Technology in the meeting of the Dissertation Defence Board of the Informatics Science Field.

Address: K. Donelaičio 73-302, LT-44249 Kaunas, Lithuania.

Phone +370 608 28 527; e-mail: doktorantura@ktu.lt.

The dissertation was sent out on 15 May 2026.

The dissertation is available on the website <http://ktu.edu> and at libraries of Kaunas University of Technology (Gedimino 50, Kaunas), Vytautas Magnus University (K. Donelaičio 52, Kaunas) and Vilnius Gediminas Technical University (Saulėtekio 14, Vilnius).

KAUNO TECHNOLOGIJOS UNIVERSITETAS

LUKAS STANKEVIČIUS

IŠMOKSTAMI TEKSTO REPREZENTACIJOS
MODELIAI IR JŲ EFEKTYVUMO TYRIMAS

Daktaro disertacija
Gamtos mokslai, informatika (N 009)

Kaunas, 2026

Disertacija rengta 2019–2025 metais Kauno technologijos universiteto Informatikos fakultete, Programų inžinerijos katedroje.

Doktorantūros teisė Kauno technologijos universitetui suteikta kartu su Vytauto Didžiojo universitetu ir Vilniaus Gedimino technikos universitetu.

Mokslinis vadovas:

doc. dr. Mantas LUKOŠEVIČIUS (Kauno technologijos universitetas, gamtos mokslai, informatika, N 009).

Redagavo: anglų kalbos redaktorius dr. Armandas Rumšas (leidykla „Technologija“), lietuvių kalbos redaktorė Aurelija Gražina Rukšaitė (leidykla „Technologija“).

Informatikos mokslo krypties disertacijos gynimo taryba:

prof. habil. dr. Rimantas BARAUSKAS (Kauno technologijos universitetas, gamtos mokslai, informatika, N 009) – **pirmininkas**;

prof. dr. Andrius KRIŠČIŪNAS (Kauno technologijos universitetas, gamtos mokslai, informatika, N 009);

prof. dr. Sanda MARTINČIĆ-IPŠIĆ (Rijekos universitetas, Kroatija, gamtos mokslai, informatika, N 009);

prof. dr. Agnė PAULAUŠKAITĖ-TARASEVIČIENĖ (Kauno technologijos universitetas, gamtos mokslai, informatika, N 009);

prof. dr. Simona RAMANAUSKAITĖ (Vilniaus Gedimino technikos universitetas, gamtos mokslai, informatika, N 009).

Disertacija bus ginama viešame Informatikos mokslo krypties disertacijos gynimo tarybos posėdyje 2026 m. birželio 15 d. 10 val. Kauno technologijos universiteto Senato salėje.

Adresas: K. Donelaičio g. 73-302, LT-44249 Kaunas, Lietuva.

Tel. (+370) 608 28 527; el. paštas doktorantura@ktu.lt.

Disertacija išsiųsta 2026 m. gegužės 15 d.

Su disertacija galima susipažinti interneto svetainėje <http://ktu.edu>, Kauno technologijos universiteto bibliotekoje (Gedimino g. 50, Kaunas), Vytauto Didžiojo universiteto bibliotekoje (K. Donelaičio g. 52, Kaunas) ir Vilniaus Gedimino technikos universiteto bibliotekoje (Saulėtekio al. 14, Vilnius).

CONTENT

LIST OF TABLES	7
LIST OF FIGURES	8
LIST OF ABBREVIATIONS AND TERMS	9
1 INTRODUCTION	14
Relevance of the Work	14
Object of the Thesis	15
The Aim and Tasks of the Thesis	15
Scientific Novelty	16
Practical Value	16
Document Structure	17
2 REVIEW OF TEXT REPRESENTATION METHODS	19
2.1 Statistical Methods	19
2.2 Word2vec Related Methods	23
2.3 Recurrent Neural Network Models	25
2.4 Transformer Models	30
2.4.1 Vanilla Transformer	31
2.4.2 BERT	32
2.5 Fine-Tuning Pre-Trained Transformer Model	35
2.6 Evaluation of Text Representation	37
2.6.1 Multi-task benchmarks	39
2.6.2 Probing embeddings	41
2.7 Summary of the Review	44
3 IMPROVING TEXT-LEVEL REPRESENTATIONS FROM PRE-TRAINED MODELS	46
3.1 Review of Enhancement Methods	46
3.1.1 Combining word vectors	46
3.1.2 Post-processing representations	54
3.1.3 Summary of the review	62
3.2 Methodology	62
3.2.1 Problem formulation	62
3.2.2 Transformer model	63
3.2.3 Aggregation methods	64
3.2.4 Post-processing methods	65
3.2.5 Static models	67

3.2.6	Text representation evaluation	68
3.3	Experiments and Results	73
3.3.1	Token aggregation and post-processing techniques	73
3.3.2	Improving RE baseline	82
3.3.3	Using prompts	83
3.3.4	BERT + Avg. model	85
3.4	Conclusions	87
4	TEXT-LEVEL TRANSFORMATIONS FOR LITHUANIAN LANGUAGE	90
4.1	Lithuanian NLP Resources	90
4.1.1	Datasets	90
4.1.2	Models	92
4.2	Lithuanian News Clustering	92
4.2.1	Methodology	92
4.2.2	Results	94
4.3	Abstractive Summarization of Lithuanian News	95
4.3.1	Methodology	96
4.3.2	Results	97
4.4	Grammatical Error Correction	100
4.4.1	Methodology	100
4.4.2	Evaluation	107
4.4.3	Experiments and results	108
4.5	Summarization and Grammatical Error Correction as Text Transfor- mation	117
4.5.1	Methodology	117
4.5.2	Results	118
4.6	Conclusions	120
5	CONCLUSIONS	123
	SANTRAUKA	125
	LIST OF REFERENCES	141
	CURRICULUM VITAE AND DESCRIPTION OF CREATIVE ACTIVITIES (CV)	181
	LIST OF SCIENTIFIC PAPERS AND SCIENTIFIC CONFERENCES	182
	APPENDICES	185

LIST OF TABLES

Table 1. Manual template search by adding additional text and MASK tokens	65
Table 2. Properties of tokenizers used for RE model experiment	68
Table 3. Dataset statistics for STS tasks	69
Table 4. Dataset statistics for classification tasks	70
Table 5. Dataset statistics for the short text clustering datasets	72
Table 6. Effect of different token aggregation and post-processing methods on multiple text embedding models	74
Table 7. Performance of post-processing with plain token averaging	75
Table 8. Effect of tokenization and random vector size for 10 random seed initializations	82
Table 9. Confusion matrix for types of pair counts after clustering for MCC calculation	94
Table 10. Languages and the training dataset statistics	101
Table 11. Alpha-word accuracy results for the diacritics restoration task . . .	109
Table 12. Alpha-word error ratio between the ByT5 and Dictionary methods for two word groups and models in different training stages	111
Table 13. The distribution of diacritics restoration errors for the different frequencies of the word	112
Table 14. The confusion matrix of the unseen word diacritics restoration per- formance by the ByT5 model	113
Table 15. Alpha-word accuracies for the simultaneous diacritics and typo- graphic error corrections.	114
Table 16. Evaluation for the separate error categories with models trained for one epoch	116
Table 17. Effect of summarization for Lithuanian news clustering.	118
Table 18. Effect of GEC for Lithuanian hate speech detection task.	119
Table 19. Training and inference times for Lithuanian text transformation models	119

LENTELIŲ SĄRAŠAS

20 lentelė. Tokenų agregavimo ir reprezentacijų apdorojimo įtaka įvairiems teksto modeliams	134
21 lentelė. Generatyvinių transformerių modelių mokymo užduotys	135
22 lentelė. Alfa-žodžių tikslumo rezultatai diakritų atkūrimo bei bendrai di- akritų ir klaviatūros įvedimo klaidų ištaisymo užduočiai	139

LIST OF FIGURES

Figure 1. Illustration of text representation pipeline with Transformer models	18
Figure 2. Relation between average Spearman correlation for STS tasks and IsoScore of Wikitext representations	78
Figure 3. Relation between average clustering accuracy and IsoScore of Wikitext representations	79
Figure 4. Alignment and uniformity of representations in relation to various representation methods	81
Figure 5. Layer-wise performance of templated and RE models	84
Figure 6. BERT + Avg. model by weighting fraction	86
Figure 7. Lithuanian news clustering results	95
Figure 8. Number of tokens in validation samples for different tokenizers	98
Figure 9. Results of training Lithuanian abstractive news summarization model	99
Figure 10. Distribution of generated typographical errors by category	104
Figure 11. Alpha-word accuracy improvement during diacritics restoration training	110
Figure 12. Alpha-word accuracy improvement during the simultaneous diacritics and typographical errors correction training	115
Figure 13. Training loss and $F_{0.5}$ score for both T5 and ByT5 on the Lithuanian Grammatical Error Correction (GEC) task.	116

LIST OF ABBREVIATIONS AND TERMS

- ABTT** All-But-The-Top – the numerical text representation improvement technique developed by the authors of [1];
- AI** Artificial Intelligence – the capability of computational systems to perform tasks typically associated with human intelligence;
- BERT** Bidirectional Encoder Representations from Transformers – A Transformer-based language model introduced by [2]; Famous for applying the transfer learning paradigm for Transformer architecture;
- BM25** Best Matching 25 – a popular scoring algorithm, usually employed by search engines to evaluate how well a document matches a specific search query;
- BoW** Bag of Words – a model of text which uses an unordered collection (a ‘bag’) of tokens;
- BPE** Byte-Pair Encoding – an algorithm for text compression and tokenization;
- ByT5** – a byte-level input seq2seq model by the authors of [3];
- CBOW** Continuous Bag of Words – a type of Word2vec architecture, where a model predicts probability for the center word, given the surrounding words as an input;
- CNN** Convolutional Neural Network – a type of FFNN that learns features via filter (or kernel) optimization;
- CoVe** Context Vectors – a contextualized recurrent model by the authors of [4];
- DCT** Discrete Cosine Transform – a numerical representation summarization technique by the authors of [5];
- Doc2vec** – Doc2vec model [6];
- ELMo** Embeddings from Language Models – a recurrent contextualized model by the authors of [7];
- EM** Expectation Maximization – an algorithm of an iterative approach used to estimate the parameters in statistical models that involve unobserved latent variables;
- embedding** – a mapping of text into a vector space that captures semantic meaning;
- epoch** – one complete pass through the entire training dataset;

fastText – a competitor to Word2vec architecture, representing words as subwords of character n-grams;

FFNN Feed Forward Neural Network – an artificial neural network in which information flows in a single direction – inputs are multiplied by weights to obtain outputs;

fine-tuning – a process by which a model that was designed for one task (the upstream task) is specialized to a new task (the downstream task);

GEC Grammatical Error Correction;

GLoVe Global Vectors for Word Representation – a model by the authors of [8] rivaling Word2vec;

GMM Gaussian Mixture Model – a probabilistic model that represents data as a weighted combination of multiple Gaussian distributions;

GPU Graphics Processing Unit;

GRU Gated Recurrent Unit – a gating mechanism used in RNNs, introduced by the authors of [9];

HODMD Higher-Order Dynamic Mode Decomposition;

IDF Inverse Document Frequency – a measure of how much information the word provides, i.e., how common or rare it is across all documents;

LDA Latent Dirichlet Allocation – a generative statistical model that explains how a collection of text documents can be described by a set of unobserved ‘topics’;

LLM Large Language Model;

LM Language Model – a model of the human brain’s ability to produce natural language;

LSA Latent Semantic Analysis;

LSTM Long Short-Term Memory – a type of RNN aimed at mitigating the vanishing gradient problem commonly encountered by traditional RNNs;

MCC Matthews Correlation Coefficient – a measure of the quality of binary (two-class) classifications;

MI Mutual Information – in probability theory and information theory, MI of two random variables is a measure of the mutual dependence between them;

MLM Masked Language Modeling – a type of task which masks tokens inside the input text for the model to predict; popularised by the BERT model;

MLP Multi-Layer Perceptron;

MMTEB Massive Multilingual Text Embedding Benchmark – a set of multilingual benchmarks for text representation evaluation by the authors of [10];

mT5 – multilingual T5-type model by the authors of [11];

MTEB Massive Text Embedding Benchmark – a set of benchmarks for text representation evaluation by the authors of [12];

MultiNLI Multi-Genre Natural Language Inference – an NLI dataset by the authors of [13];

NCE Noise Contrastive Estimation – a technique in NLP used to estimate the parameters of probability models without calculating the computationally expensive normalization constant;

n-gram – a sequence of n adjacent tokens in a particular order;

NLI Natural Language Inference – a canonical task in NLU with labeled directional relations between text fragments;

NLP Natural Language Processing – the processing of natural language information by a computer;

NLU Natural Language Understanding – a subset of NLP in AI that deals with machine reading comprehension;

NMF Nonnegative Matrix Factorization;

NMT Neural Machine Translation – an approach to machine translation that uses an artificial neural network to predict the likelihood of a sequence of words;

NSP Next Sentence Prediction – a type of task employed during BERT pre-training;

OOV out of vocabulary;

PCA Principal Component Analysis;

PLSA Probabilistic Latent Semantic Analysis;

post-processing – methods to improve representations after turning tokens into vector(s);

pre-processing – methods to improve representations before turning tokens into vector(s);

pre-training – usually, a long and expensive process of training the model to acquire the general knowledge, substantially improving the later fine-tuning stage;

PV-DBOW Distributed Bag of Words version of Paragraph Vectors – a type of Doc2vec model;

PV-DM Distributed Memory Model of Paragraph Vectors – a type of Doc2vec model;

RNN Recurrent Neural Network – artificial neural networks, designed for processing sequential data, such as text, speech, and time series, where the order of elements is important;

ROUGE Recall-Oriented Understudy for Gisting Evaluation – an automatic word-overlap-based evaluation metric [14];

SCDV Sparse Composite Document Vector;

SentenceBERT – a fine-tuned BERT model for sentence representations by the authors of [15];

seq2seq sequence-to-sequence;

SIF Smooth Inverse Frequency – a token weighting technique presented by the authors of [16];

SimCSE – a fine-tuned BERT model for sentence representations using contrastive learning by the authors of [17];

Skip-gram – a type of Word2vec architecture, where model predicts probabilities for surrounding words, given only the center word as an input;

SkipThought – a famous recurrent sentence representation model by the authors of [18];

SNLI Stanford Natural Language Inference – a NLI dataset presented by the authors of [19];

SOTA State Of The Art;

STLR Slanted Triangular Learning Rates – a dynamic learning rate technique used by the authors of [20];

STS Semantic Textual Similarity – is a measure of how closely two texts are related in meaning, going beyond simple word matching to capture concepts and intent;

STS-B Semantic Textual Similarity Benchmark – a STS task presented by the authors of [21];

SVD Singular Value Decomposition;

T5 – a seq2seq Transformer model by the authors of [22];

TF-IDF Term Frequency – Inverse Document Frequency;

the curse of dimensionality – the problem of data sparsity in high-dimensional spaces caused by exponential grow of possible states; then, the dimensionality is increased;

token – the most fundamental data unit in a processed text document; usually, a single word, character, or word parts;

tokenization – breaking down larger texts into smaller, manageable pieces called tokens;

Transformer – a revolutionary neural network architecture based on the multi-head attention mechanism;

ULMFiT Universal Language Model Fine-tuning – a recurrent model by the authors of [20]; it was one of the first to employ transfer learning for text domain;

WMD Word Mover’s Distance – a distance measure by the authors of [23];

Word2vec – Word2vec model [24, 25];

WRD Word Rotator’s Distance – a distance measure by the authors of [26].

1. INTRODUCTION

We live in an age of information. Digital text is growing at an unprecedented scale. Billions of webpages, news articles, social media posts, emails, research papers, and other forms of digital word are generated every single day. An individual can observe only a tiny fraction of this generated knowledge, and automatic processing is becoming ever more pertinent. Efficiently processing this vast amount of information is critical for search engines, automatic summarization, detection of fake news, inappropriate, or criminal content detection. However, it is not trivial.

Computers do not inherently understand human language. The bytes that encode text symbols carry no meaning beyond their assigned character labels. To bridge this gap, specialized methods are required to capture relationships between text fragments. Traditional approaches like Term Frequency – Inverse Document Frequency (TF-IDF) and n-grams relied on statistical techniques, which primarily modeled surface-level patterns rather than deep semantic meaning. This limitation significantly constrained their effectiveness in complex language tasks.

The rise of artificial neural networks enabled for more expressive patterns to be encoded. The authors of a well-known Word2vec model showed that semantic meaning can be embedded in numerical vector representations. Arithmetic operations such as $\text{king} - \text{man} + \text{woman} \approx \text{queen}$ became possible. This marked a major shift in text understanding. Many following works sought to improve these word representations, starting with shallow, recurrent, and, finally and recently, Transformer-type neural networks.

This dissertation focuses on Transformer-type architecture models and their ability to produce numerical text representation. Inspired by Word2vec, this work seeks to produce similarly arithmetic-able representations, just for whole sentences or paragraphs. As will be covered in the review (Chapter 2), the first pre-trained Transformer models were unable to sufficiently represent sentences despite being trained on huge amounts of texts, and they only performed particularly well on sentence-level tasks, which were then fine-tuned. In addition to the final numerical representation, the interest has also been directed towards other factors affecting the success of such models. In particular, the constrained case of applying Transformers models for the Lithuanian language is also covered in this dissertation.

Relevance of the Work

This dissertation began as an effort to advance the field of Natural Language Processing (NLP), which, at the time (2019), lagged behind computer vision in leveraging pre-trained deep learning architectures. However, in the years that followed, NLP rapidly caught up, transforming into one of the most dynamic areas of Artificial Intelligence (AI).

During this research, one of the most groundbreaking developments in AI was wit-

nessed, culminating in a well-known *ChatGPT* [27] – a system capable of engaging in human-like conversations by using deep neural networks. In just a few years, the scale of NLP models has grown by three orders of magnitude, leading to unprecedented advances. This surge in capability has not only reshaped AI research, but has also driven significant investment and sparked new groundbreaking applications across various domains.

Learnable text representation models, particularly the Transformers architecture, which is at the heart of this thesis research, were the main ‘culprit’ behind the unprecedented progress in NLP. Despite the intense pace of the NLP field, it is still the backbone element of all Large Language Models (LLMs) and the main method of choice for many NLP tasks.

The same representations as in this paper are particularly useful in complementing the current LLMs. Although LLMs are better with more popular knowledge, the authors of [28] showed that they struggle with less popular factual knowledge, and an effective way to mitigate it is to use retrieval augmentation. According to the authors of [29], pre-trained retrieval augmented Language Model (LM) can outperform even 50-times larger models. In the simple case of retrieval-augmented generation, the LLM is supplied additional text related to the given input, thus improving its context understanding. Pre-trained representations are the key to turn input text into a representative numerical vector and retrieve the most similar records from the database. This is just one of many tasks that uses text representation models.

Although it is hard to keep up with the fast pace the NLP endured during the recent years, in this dissertation, the new possibilities for the Lithuanian language were brought into fruition. In particular, Transformer models were applied to tasks that previously were thought to be impossible. It included abstractive summarization of Lithuanian news articles and, a sequence-to-sequence (seq2seq) style, like translation, GEC.

In conclusion, the relevance of this dissertation is currently even greater than at the time of the start of this work. The explosion of methods used here brought many new applications and possibilities, as well as the need for better learnable text representations for these new applications.

Object of the Thesis

The object of the dissertation is text representation pipelines based on pre-trained Transformer models, encompassing text-level transformations, representation extraction strategies, and their impact on the quality of numerical text representations.

The Aim and Tasks of the Thesis

This dissertation aims to investigate and improve text representations obtained from pre-trained Transformer models for English and Lithuanian languages. The tasks of the dissertation are the following:

1. Review existing learnable text representation methods and evaluation approaches.

2. Develop and investigate methods for extracting improved text representations from pre-trained Transformer models, including token-layer aggregation and post-processing techniques.
3. Implement and investigate Transformer-based text-level transformation methods, such as summarization and grammatical error correction, as preprocessing steps for text representation, with a focus on the Lithuanian language.

The main research questions are as follows:

1. How can the pre-trained Transformer representations be improved, and to what extent?
2. How effective can the Transformer-induced text-level transformations be for textual and numerical text representation?

Scientific Novelty

1. It has been demonstrated that substantial improvements in unsupervised sentence representations for unsupervised tasks can be achieved without additional contrastive or task-specific training, but, instead, through principled token-layer aggregation and post-processing of pretrained Transformer representations. Specific optimal combinations of token-layer aggregation and post-processing have been proposed for various Semantic Textual Similarity (STS), short text clustering and classification tasks.
2. A competitive sub-word tokenization random embedding baseline enhanced with the same shaping techniques has been introduced. This baseline allows disentangling the contribution of embedding space geometry from knowledge induced by pre-training, thereby demonstrating that a non-trivial portion of performance in similarity tasks can be explained by geometric properties alone.
3. The first Transformer model applications for abstractive Lithuanian news summarization and Lithuanian GEC tasks have been fine-tuned and shared publicly.
4. The application of a vocabulary-free byte-level Transformer (ByT5) for simultaneous diacritics restoration and typo correction has demonstrated near state-of-the-art performance with significantly reduced training resources, thus providing evidence for the effectiveness of byte-level modeling in 13 morphologically rich languages.

Practical Value

This dissertation has many practical applications. General text representations acquired by the current top deep-learning Transformer architecture is analyzed. These representations can be used in multiple unsupervised NLP tasks, including clustering, topic modeling, semantic search and retrieval, text similarity, and paraphrase detection, to

name a few. These representations can also be used as features for many other supervised tasks, such as classification. Unsupervised methods to improve text representations are proposed, thus benefiting all tasks that use them. The analyzed aggregation and post-processing techniques do not need any training and can be used for any rare-domain or low-resource setting. This is also the case for sensitive datasets that should not be used for training, as they pose a threat to surface information during inference. The proposed adequately processed random embeddings are a very good baseline for use in similar future works. It offers a mid-level performance, a simple implementation, and is able to showcase the performance difference from the learning context quality. A great practical value of this dissertation is the methodology for fine-tuning GEC models in different languages. It has been shown that, by using the methodology, one can achieve a 98.3 % average alpha-word accuracy on the diacritic restoration task over 13 benchmark languages and a greater than 94 % accuracy for the simultaneous diacritics restoration and typos correction. Such models can be used in real-world GEC systems to improve the typed text quality.

This dissertation also provides pre-trained models for the Lithuanian language. Trained Lithuanian news summarization and GEC models are publicly available¹. They serve a good starting position to be used in other tasks for fine-tuning. Due to training on Lithuanian news data, the models are already familiar with the Lithuanian language and the news domain. The follow-up work of [30] has already employed one of the models from this work. Pre-trained Transformer models represent general knowledge; therefore, they can be applied to a wide range of NLP tasks.

Document Structure

This thesis consists of 5 chapters. The first part, Chapter 1, i.e., the current chapter, introduces the topic. It covers the relevance of the research, the object of the dissertation, tasks, scientific novelty, thesis statement, and practical value of this work. The second Chapter 2 provides the theoretical background and review of text representation methods. It covers them historically, starting from statistical, shallow network ones, and finishing with the currently popular deep neural network Transformer architecture. The following Chapters 3 and 4 present the original contributions of the dissertation. Chapter 3 focuses on representation extraction from Transformer activation features; see the *vectorization method* process part in Fig. 1. Chapter 4 applies Transformer models as text-level transformations to the Lithuanian language; see the *text-level transformation* process part in Fig 1. The methodology and results are presented for each aspect separately. Finally, the thesis ends with conclusions in Chapter 5.

¹<https://huggingface.co/LukasStankevicius>

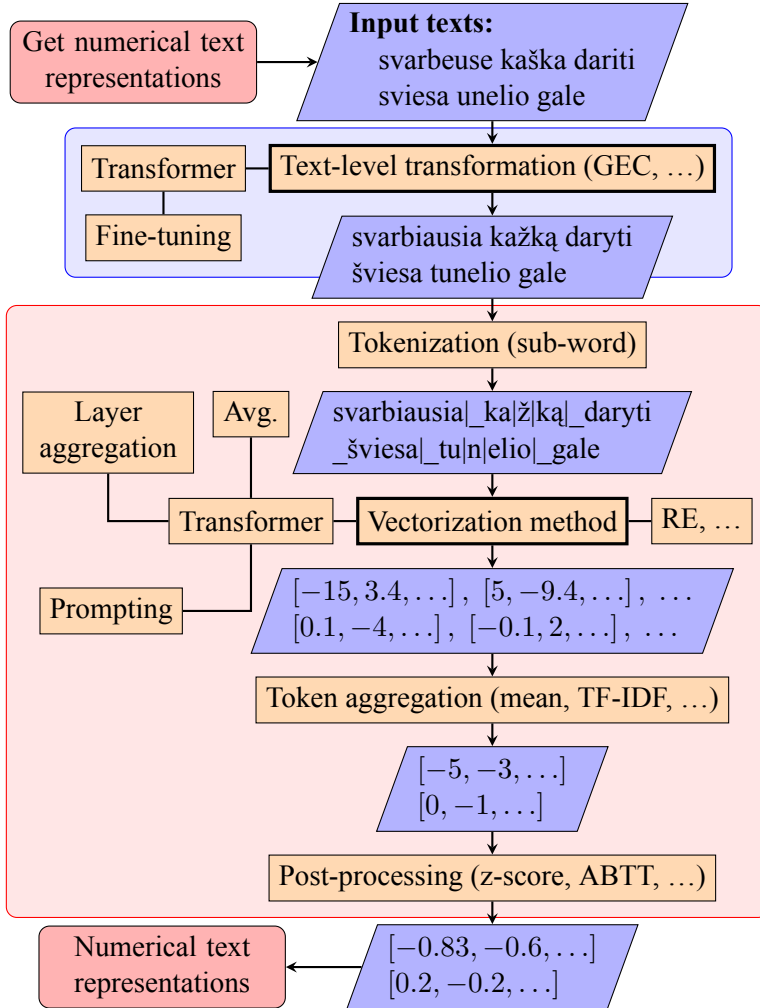


Figure 1. Illustration of text representation pipeline with Transformer models. The components enclosed in the red shaded region correspond to the core methodological contributions presented in Chapter 3, whereas the components in the blue shaded region are covered in Chapter 4.

2. REVIEW OF TEXT REPRESENTATION METHODS

The main aim of this chapter is to review the methods of text-level representations. The focus here will be on historic evolution, starting from the statistical, Word2vec-related, recurrent neural networks up to the Transformer models. The focus is on the factors that drove the field of text representations, regardless of time or technology. In addition, the pros and cons of each method will be noted, as well as the ways how their weaknesses have so far been addressed in the follow-up work.

2.1. Statistical Methods

The most ordinary way to represent the text is to use the statistics of its compounding pieces. In common sense, words such as ‘police’, ‘fine’, ‘guns’ should be common in crime-related news articles, and rare in topics such as computer news, with more frequent terms of ‘software’, ‘memory’, etc. This idea that “each language can be described in terms of a distributional structure” was proposed in the work of [31] in 1954, and it is more commonly known as a classic method of the Bag of Words (BoW). BoW is a simple and interpretable method. After collecting word counts from the two documents, one can manually inspect the differences and similarities. The final text representation is just a list of all words in both documents and the number of occurrences of each word in each document. One can clearly see that, for example, the word ‘police’ is used substantially more frequently in a text related to crime than in a text related to computer science.

The BoW representation comes with several drawbacks. As the name implies, it treats a document as an unordered collection of words, discarding the original word sequence. Although this issue can be mitigated by incorporating n-grams — sequences of multiple words — to capture some contextual information, doing so exacerbates another major limitation of BoW: the curse of dimensionality. Since words serve as the fundamental building blocks of text, and every language consists of a vast vocabulary, feature sets grow in proportion to the number of unique words in a language. In practice, this results in extremely high-dimensional and sparse feature spaces, where only a small fraction of the features have nonzero values.

To reduce the dimensionality of feature vectors, pre-processing methods are employed. The simplest of these is stemming. It simply stems the endings of words, like ‘writing’ to ‘writ’. Some different words with the same lemma can still be stemmed differently, like ‘wrote’ to ‘wrot’, so the other pre-processing method of lemmatization is usually employed, which transforms both words ‘writing’ and ‘wrote’ to ‘write’. Lastly, the third method of eliminating stop words discards very frequent words such as ‘the’ or ‘or’, which repeat in every text despite the representative thematic content. Rare tokens were omitted, or the ‘unknown token’ meaning was given to them. Repeated only several times, they cannot bring much significant information, as it will be superseded by noise. Both too rare and too frequent removals are called minimum and maximum

document frequency filtering. Overall, these are the usual pre-processing steps:

- stop words (i.e., ‘and’, ‘or’) removal;
- lower-casing;
- stemming (i.e., ‘beautifully’ to ‘beauty’);
- lemmatization (i.e., convert ‘went’ to ‘go’);
- mapping (i.e., any date to ‘#DATE’, or any number to ‘#NUMBER’);

It was empirically observed that the frequency of any word is inversely proportional to its rank in the frequency table [32]. Therefore, any calculation of BoW vectors would result in some words being more important solely because they are more frequent. To avoid this, the term occurrence count must be weighted. Words that are frequent in all documents should not be considered more important in a particular document. Only words that are more frequent in a particular document relative to the rest of texts can be considered as more important features. One of the most popular term weightings that addresses these issues is TF-IDF. Token t weight $W_{t,d}$ in document d is given by:

$$W_{t,d} = tf_d \cdot idf_{t,d} = tf_d \cdot \log \frac{n}{df_{t,d}} \quad (1)$$

where tf_d is token t count in document d , $df_{t,d}$ – count of documents having token t , n – number of all documents. TF-IDF weighting results in a decent BoW text representation.

The weighting of Best Matching 25 (BM25) [33] accounts for even more factors. First, it saturates the importance of a term. The document containing 400 occurrences of a ‘guns’ is not twice as relevant as the one with 200 such occurrences. Therefore, after some high count, the term relevance should saturate. The second factor that BM25 takes into account is the document length, where it tries to reward matches in short documents while penalizing them in long ones. By accounting for more factors and having several coefficients that can be fine-tuned to a specific dataset, BM25 is more complex than a simple TF-IDF and is mainly used by search engines and in retrieval-based NLP tasks.

In addition to weighting the features, one can also choose the granularity of text to use as terms. Classical BoW uses words as its features/terms. However, there are more options for which statistics to use. Going from the coarse- to fine-grained granularity there are paragraphs, sentences, words, syllables and letters, or individual characters with the full spectra of various UTF-8 symbols. If paragraphs are used as unique constituent pieces of the text, the same paragraph would not be repeated even in the same topic text. On the other hand, the same letters and characters are used in all the texts, and discriminating one from the other would be barely possible (excluding, of course, some very specific texts like the code). Therefore, one has to strike the balance of which degree of granularity to choose.

Modern works train a separate tokenization model that learns to segment the given text into sequential features. The popular Bidirectional Encoder Representations from Transformers (BERT) [2] Transformer model used WordPiece [34] tokenization. This algorithm first breaks text into words on whitespaces, as the usual classical algorithms, but then tries to break them further into word pieces. The other Byte-Pair Encoding (BPE) algorithm [35] builds vocabulary by iteratively merging the most frequent vocabulary pairs in the current iteration step. Such greedy and deterministic approach was later improved by unigram LM [36], which instead relies on segmentations with their probabilities. The authors of SentencePiece [37] implemented an efficient framework of the BPE and unigram approaches, which was then used in other popular modern models such as T5 [22] and mT5 [11]. In general, such modern data-driven text tokenization allows one to set a desired vocabulary size (the common size is 32 000) and at the same time optimize the minimum number of tokens in each sample. However, these approaches are tailored to deep-learning-based NLP models, which are good at learning nonlinear relationships, rather than pure feature-based BoW.

In addition to pre-processing and weighting, one can improve BoW representations by applying dimensionality reduction methods. One of the most known is Principal Component Analysis (PCA) [38, 39]. It is a linear transformation which works by projecting data into a set of orthogonal axes (principal components) so that the variance of the data is maximized along these new axes. For example, if given data is two-dimensional, but all points lie in a single arbitrary line, this line will be identified as a dominant principal component, and the final reduced data will have just this single dimension.

According to the authors of [40], the main dimensionality reduction methods for the text are topic models. It is straightforward to understand that reducing the dimensions of the BoW term-document matrix results in a smaller lower dimensionality of the concept-document matrix, reducing many term features into a few concept ones.

One such model which attempts to describe a BoW corpus by using a combination of concepts is Latent Semantic Analysis (LSA) [41]. In LSA, terms that have a high co-occurrence across documents are assumed to be conceptually related. The relationship between terms and concepts can be derived by factoring a matrix using Singular Value Decomposition (SVD). Given a term-document matrix $X \in \mathbb{R}^{|t| \times |d|}$, where $|t|$ is a number of terms (vocabulary size) and $|d|$ is a number of documents, SVD factorizes it into 3 distinct matrices:

$$X = \begin{matrix} U \\ \left[\begin{bmatrix} \vdots \\ \mathbf{u}_1 \\ \vdots \end{bmatrix} \dots \begin{bmatrix} \vdots \\ \mathbf{u}_l \\ \vdots \end{bmatrix} \right] \end{matrix} \cdot \begin{matrix} \Sigma \\ \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_l \end{bmatrix} \end{matrix} \cdot \begin{matrix} V^T \\ \left[\begin{bmatrix} \mathbf{v}_1 & \dots & \vdots & \mathbf{v}_l \end{bmatrix} \right] \end{matrix} \quad (2)$$

Matrices $U \in \mathbb{R}^{|t| \times l}$ and $V \in \mathbb{R}^{|d| \times l}$ are orthonormal. The diagonal elements of Σ

are singular values of the matrix X . l is the number of non-zero singular values. Each singular value corresponds to some column vector in U and a row vector in V^T . The dimensionality reduction is then performed by selecting the highest singular values k and the corresponding vectors in U and V^T and recalculating the now-modified X_k :

$$X_k = U_k \Sigma_k V_k^T \quad (3)$$

In practice, U_k is a term-concept matrix, Σ_k is a concept-concept matrix, and V_k^T is a concept-document matrix. X_k can then be used as approximation for document representation. On the other hand, U_k represents reduced document representation in the ‘concept’ space, while V_k^T can reveal terms comprising each concept.

Another method related to LSA is Nonnegative Matrix Factorization (NMF) [42, 43]. Similarly to SVD, the input matrix is factorized into other matrices. The main difference is that these other matrices are restricted to be only positive. When the desired lower dimension is k , the goal of NMF is to find the two matrices $W \in \mathbb{R}^{|t| \times k}$ and $H \in \mathbb{R}^{k \times |d|}$ having only nonnegative entries such that

$$X \approx WH \quad (4)$$

W is term-wise topic representation. Each column of W is a particular topic with row elements (terms) that describe it. H is a topic-wise document representation, where the columns of H are documents with row elements (topics) representation. It is the final reduced-dimension document representation. Positive values can then be easily interpreted (i.e., as probabilities).

Probabilistic Latent Semantic Analysis (PLSA) [44], the probabilistic extension of LSA, uses a probabilistic method instead of SVD to tackle the problem of document representation in a reduced topic or concept space. PLSA models probability of a specific document d and word w appearing simultaneously $P(d, w)$ (co-occurrence) as a mixture of conditionally independent multinomial distributions:

$$P(d, w) = \sum_z P(w|z)P(z)P(d|z) \quad (5)$$

Here, z corresponds to a particular topic, a higher level representation than a word. Equation (5) closely resembles the SVD used in LSA (see Equation (3)), but, instead of using linear algebra, the parameters are learned by using the Expectation Maximization (EM) algorithm.

An improvement over PLSA is the Latent Dirichlet Allocation (LDA) [45, 46]. It is a Bayesian model that uses document-topic and topic-word priors, both modeled as Dirichlet distributions. These properties make it more universal than PLSA as it allows LDA to generate new documents. However, computing the posterior distribution of the hidden variables given a document is intractable, so approximations such as Laplace approximation, variational approximation, or Markov chain Monte Carlo are then used [45].

2.2. Word2vec Related Methods

A substantial change in NLP occurred in 2013 with the introduction of the famous Word2vec method [24]. The authors presented a way to train dense word vectors and reported significant improvements in accuracy for the word similarity task. One of the most surprising properties of such embeddings is the ability to perform meaningful algebraic operations. Subtracting vector(“Man”) from vector(“King”) and adding vector(“Woman”) results in a vector that is closest to the vector(“Queen”) [47]. This is a ‘man-woman’ semantic relationship, but there are many more such as ‘country-capital’, currency or syntactic relationships as opposite, comparative, adjective to adverb, etc. This thesis was also inspired by Word2vec: the idea that a particular representation could be expressed by a single dense vector with the properties of meaningful algebraic operations.

The original Word2vec model [24] is very simple. It consists of just a single-layer Feed Forward Neural Network (FFNN) with stripped-off non-linearity part. The authors of [24] presented two architectures: Continuous Bag of Words (CBOW) and Skip-gram. Generally, the embeddings for each vocabulary word are initialized, and, during the training, these vectors are constantly updated to better reduce the training loss. The algorithm takes a fixed window of sequential words and tries to predict the center word, given the sum of the surrounding ones (CBOW), or tries to give large probabilities for surrounding words, given only the center word as an input (Skip-gram).

The follow-up work of [25] presented several improvements to the Skip-gram model. First, the authors replaced hierarchical softmax [48] with negative sampling, the simplification of Noise Contrastive Estimation (NCE) [49]. It allowed them to speed up the training by sampling only a small number of negatives for each positive training example. The second improvement was subsampling of frequent words, where each word w_i in the training set is discarded with a probability computed by the formula

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}} \quad (6)$$

where $f(w_i)$ is the frequency of word w_i , and t is the threshold. In general, the negative sampling and subsampling of frequent words allowed the authors of [25] to train the Skip-gram model much faster even when using more data, a total of 30 billion words, while surpassing other models in the quality of the learned representations.

Another popular dense word representation model is Global Vectors for Word Representation (GLoVe) [8]. It merged the advantages of both matrix factorization and shallow window-based methods. Models like LSA [41] operated on complete word co-occurrence statistics, while Word2vec [24, 25] used only several words from the given context window. Thus, the authors of GloVe [8] saw an opportunity to incorporate global statistics in Word2vec type training. The objective function of the GloVe

model is:

$$J = \sum_{i,j=1}^V f(X_{i,j})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log(X_{i,j}))^2 \quad (7)$$

where V is the size of vocabulary; f – weighting function; $X_{i,j}$ – the number of times word j occurs in the context of word i ; w_i, b_i – word vectors and their bias; $\tilde{w}_j, \tilde{b}_j$ – context word vectors and their bias. The weighting function is specially designed so that rare and frequent co-occurrences are not overweighted:

$$f(x) = \begin{cases} \left(\frac{x}{x_{max}}\right)^\alpha, & \text{if } x \leq x_{max} \\ 1, & \text{otherwise} \end{cases} \quad (8)$$

The authors of [8] empirically determined optimal values $\alpha = \frac{3}{4}$ and $x_{max} = 100$.

The first step in GLoVe training is the construction of matrix X , scanning all corpus words given some context window k . GLoVe weighted distances between words with a fraction of $1/d$, where d is the number of words apart. Once the word-word matrix X has been constructed, the model is trained similar to Word2vec, stochastically sampling non-zero elements from matrix X .

The third famous model is fastText [50]. The authors represent target words as subwords of character n -grams. Taking word ‘Vilnius’ and $n = 3$ as an example, it will be represented by [“<Vi”, “Vil”, “iln”, “lni”, “niu”, “ius”, “us>”, “<Vilnius>”]. FastText is trained as a Skip-gram model, which maximizes probabilities for the context words, given the target word. However, its target word is derived like in a CBOW Word2vec architecture, by averaging its subwords of character n -grams and the main target word itself. Such an averaging allows fastText to better account for morphology and infer unseen out of vocabulary (OOV) words. It is especially useful for languages with complex word formations, large vocabularies, and many rare words. The authors of [51] in 2018 reported that, given the same amount of training data, fastText outperformed other models, like GLoVe, in the rare words task of [52] and achieved 87 % accuracy in the word analogy tasks.

Word2vec-related models are definitely better than statistical ones. They eliminated the curse of dimensionality problem, as the target vectors became dense and generally of up to 300 elements. Meaningful algebraic operations became possible, and high accuracy was reached in the tasks of semantic and syntactic relationships. Yet, they still did not account for word order. The authors of [53] point to even more flaws associated with word vectors. Firstly, several embeddings of consecutive words cannot represent a joint phrase. One has to train the n -gram model to accurately acquire phrase embedding. Another problem is that downstream semantic analysis tasks cannot detect polarity. Words such as ‘good’ and ‘bad’ tend to learn similar embeddings despite being semantically opposite. The Word2vec model is also unable to detect polysemy, and such words end up with one representation for all different meanings. Lastly, it was also observed that “A general caveat for word embeddings is that they are highly

dependent on the applications in which it is used” [53]. This means that, for the best performance of word embeddings, the training and application data must align.

The authors of [6] proposed a Doc2vec model, an adaptation of the Word2vec algorithm from the word level to paragraph or document level representations. The main addition is that, for each text piece, a new random vector is initialized, together with word vectors, as in Word2vec. This new vector contributes to the training task by encapsulating the information relevant to that text piece and is later used as the representation of that text. There are two architectures of Doc2vec: Distributed Memory Model of Paragraph Vectors (PV-DM), and Distributed Bag of Words version of Paragraph Vectors (PV-DBOW). In PV-DM, the model receives context words and a paragraph vector as input, concatenates or averages them, and is tasked to predict the next (target) word, similarly to CBOW of Word2vec. In this mode, the additional document vector represents what is missing from the word vectors (hence the memory keyword in `glsterm:pvd`). During the training stage, the word vectors W , the paragraph vectors D , and the softmax parameters U and b are learned. If an unseen document needs to be inferred, only a new column has to be added to matrix D , freeze W , U , b , and the gradient descends to D . In this way, representations for new unseen documents can be acquired. In the PV-DBOW model, similar to the Skip-gram, the paragraph vector alone is used to maximize probabilities for context words. It is emphatically simpler because it does not use word vectors. It only learns softmax weights U (which correspond to words), biases b and document vectors D . Unlike PV-DM, context words are sampled randomly, and thus the word-ordering information in the context space is lost. The authors of [6] noted that the PV-DM architecture is consistently better than PV-DBOW, but combining the two gives the best results in text classification tasks. Several extensions of Doc2vec were also developed. The authors of [54] proposed the Doc2vecC model. Unlike Doc2vec, a document vector is derived not as a unique vector but as an average of the word vectors randomly sampled from the document. The authors also added data-dependent regularization that weights words’ embeddings on the basis of their informativeness or rareness. Whereas, the authors of [55] proposed Sent2Vec, which merged n-gram ideas from fastText [50] and CBOW training of Word2vec [24]. The authors of FastSent [56] proposed a method to represent each sentence as a sum of its word embeddings and trained the model to predict adjacent sentences given the target one.

Document-level representations can also be computed by simply averaging individual word vectors from the Word2vec, GLoVe, or fastText models. The use of weighting by frequency or pre-processing can further boost the performance of these representations. A detailed review of these methods will be presented in Section 3.1.

2.3. Recurrent Neural Network Models

Recurrent Neural Networks (RNNs) [57] are very suitable architectures for natural language tasks. First, they follow the sequential nature of language, where each word depends on those before it. Moreover, sentences or paragraphs are of unpredictable

length. This wastes inputs in regular FFNN as some inputs will have to be marked as empty. In RNN, on the other hand, the inputs are fed one by one, and the total length does not matter. The hidden state produced at the time step depends on all previous ones, and the final hidden state represents the whole sequence. These properties of recurrent neural networks made them popular for text representation.

Vanilla RNN are very hard to train. They tend to succumb to the vanishing and exploding gradient problem [58]. Therefore, a more robust variant of Long Short-Term Memory (LSTM) [59] was proposed. Unlike vanilla RNN, an LSTM maintains a separate cell state that retains information over long distances. As shown in Equation (9), it uses input, forget, and output gates to control how this information is stored, discarded, or outputted.

$$\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
\tilde{c}_t &= \tanh_c(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= o_t \odot \sigma_h(c_t)
\end{aligned} \tag{9}$$

Matrices W and U contain connection weights, b are bias vector parameters. All of these are learnable. σ is the sigmoid function, h_t – hidden state vector, x_t – input vector, and $\odot$ denotes element-wise multiplication (Hadamard product). Many famous NLP models used LSTM or a simpler version of Gated Recurrent Unit (GRU) [9], which lacks a context vector and an output gate, making it faster to train due to fewer parameters. One more technique to improve the training of multiple layer models is residual connections [60], which allows part of the input-output flow to skip the actual model layer while enabling better gradient feedback.

After the input sequence consumption, all the previous inputs of a recurrent neural network have to fit into the last hidden state to fully represent the whole sequence. However, in practice, usually only the last inputs are sufficiently remembered, thereby limiting the modeling of long-range dependencies. This was addressed by using: 1) the bidirectional [61] setting, in which hidden states of both forward and backward sequence processing are collected; and 2) the attention mechanisms [62], which, instead of relying only on the last hidden state, allow one to perform weighted selection on all previous hidden states.

A seq2seq setting [63] of RNN is often used to encapsulate the text representation. It was proven to be highly successful in Neural Machine Translation (NMT), reaching State Of The Art (SOTA) in 2014, and later being used in production by giant corporations like *Google* in their language translation products [34]. In NMT, the task is to translate the text sequence from one language to another. Therefore, two LSTMs are used – one, the encoder, to encode the first sequence into its numerical representation, and another, the decoder, to generate the target language element by element, given only the compact representation from the encoder. Although this setting is used

to translate text, the intermediate encoded output can be used as a representation for the first sequence, leaving only the encoder and discarding the decoder part entirely. All the techniques described above to train successful RNN models are well reflected in Google’s neural machine translation system [34] of 2016. It used the LSTM network with 8 encoder and 8 decoder layers (seq2seq setting), employed residual connections, attention connections and had bidirectional setting in the bottom encoder layer. One of the first famous recurrent models for the direct sentence representation task is SkipThought [18]. It is an ‘encoder – two decoders’ model composed of bidirectional GRUs [9]. The encoder produces the hidden state for the center sentence, while the first decoder is conditioned on this state to generate the previous sentence, whereas the second decoder deals with the next sentence. The authors showed that encoded representations performed well in multiple tasks. The main drawback was a lengthy month-long training, a large vector size of 4800, and the need for long texts of at least 3 sentences for training. The authors also had to additionally train vocabulary expansion from CBOW word vectors to their encoder’s vocabulary size of just 20,000 words. The following work explored different optimization objectives to learn sentence representations. The authors of the QuickThoughts model [64] replaced sentence generation with a classification objective. Their model consists only of the GRU encoder, which encodes the sentence and the candidate ones, and the classifier, which selects the target sentence. Such a setting without a decoder greatly reduced the training time while still allowing us to train a good sentence encoder. In relation to the sentence ordering task, the authors of [65] trained LSTM with the objective of ordering multiple sentences. The authors of [56] proposed a Sequential Denoising Autoencoder (SDAE) model. It is an LSTM-based encoder-decoder, but, instead of NMT, it uses a denoising objective. Original texts are simply corrupted by the random delete and word-swap operations, allowing the authors to apply the method to the sets of sentences in an arbitrary order. Two works explored discourse-based objectives. The authors of DiscSent [66] used paragraph-level coherence of ORDER, CONJUNCTION, and NEXT objectives. Other authors of DisSent [67] constructed a discourse marker prediction task, predicting tokens such as *because*, *and*, *if*, etc. with a classifier, given the bidirectional LSTM encoded text. Such a setting allowed one to mine a large amount of unsupervised data for training the encoder. One of the best-performing models was the supervised InferSent [68]. This bidirectional LSTM was trained on the Stanford Natural Language Inference (SNLI) dataset [19]. Although the dataset is small if compared to the previous unsupervised objectives, its quality of sentence similarity and dissimilarity information allowed InferSent to outperform previous models in sentence representation tasks. Finally, the authors of GenSen [69] proposed using multiple tasks, supervised and unsupervised, during training. It included SkipThought, NMT, constituency parsing and natural language inference tasks. Overall, the factors leading to better representation models were: 1) training on more data by utilizing unsupervised setting, 2) using high quality supervised data, or 3) using an efficient training objective. Usually, the goal is to train a model so that the generated sentence representations

would perform well on a wide range of tasks. However, there are even more domains than that, in particular – the language dimension. One of the best-known models in this dual setup, both a language- and task-agnostic representation, is the LASER model [70]. The authors used a total of 223 million parallel sentences, or bitexts, from 93 languages, to train the 5 layer biLSTM encoder – a single LSTM decoder model in a machine translation task. To accommodate many different tokens, tokenization was trained by BPE on the whole dataset. To force the encoder to learn the input sentence representation, no attention is used, and the output from the encoder, which is obtained by the max-pooling operation, is concatenated to decoder input embeddings at every timestep together with language ID embedding. Therefore, only the decoder knows the language in which it is operating, and multiple language input is possible to the encoder side. The authors showed that universal language agnostic sentence embeddings from LASER obtain strong results in multilingual similarity search, cross-lingual document classification, and bitext mining tasks.

Multiple recurrent neural network approaches tried to add context awareness to word vectors. Historically, word embeddings produced by methods like Word2vec or GLoVe did not account for word polysemy. As an example, the word ‘bank’ had the same representation in both contexts of ‘river bank’ or ‘federal bank’. This fault made these words’ vectors poorly representative. We, humans, distinguish polysemy by the context, and therefore many works tried to incorporate polysemy into their models. The authors of [71] added context to regular word vectors as LM embeddings. They first pre-trained both word embeddings and a neural LM on large unlabeled corpora. Next, they extracted word embeddings as a combination of regular word vectors and LM embeddings and finally fed them to the task-specific model. This way the second part of final word embeddings is context-aware. The authors of the Context Vectors (CoVe) model [4] produced vectors as outputs of the bidirectional 2-layer LSTM encoder. This encoder was trained on the machine translation task in a seq2seq setting. CoVe authors concatenated GLoVe and CoVe vectors and showed that this model performed better. A notable contextualized model is Embeddings from Language Models (ELMo) [7]. What is new in the ELMo architecture is that word representation is derived not just from the top layer of a bidirectional LSTM stack, but also from all the internal layers, not excluding input vectors, with weighting of layers learned depending on the task. For k -th sequence, the token task-specific vector is computed as follows:

$$\mathbf{ELMo}_k^{task} = \gamma^{task} \sum_{j=0}^L s_j^{task} \mathbf{h}_{k,j}^{LM} \quad (10)$$

where $\mathbf{h}_{k,j}^{LM}$ is a set of $2L + 1$ representations (L layers of forward LSTM hidden states, L layers of backward, and 1 for input embedding representation). s_j^{task} are softmax-normalized weights, and the scalar parameter γ^{task} allows the task model to scale the entire ELMo vector. ELMo reuses many elements from previous works of [72] and [73] with the addition of bidirectionality of LSTM and residual connection between 3 LSTM

layers. The first layer is context-insensitive, where token characters are transformed by 2048 character n-gram convolutional filters followed by two highway layers [74]. This allows the model to include tokens outside the training set. Compared to other models of the time, such as in [73], the architecture is less complex, allowing to train ELMo together with downstream tasks. ELMo became a popular model as an input to already existing downstream architectures. The previously used Word2vec, GLoVe or CoVe vectors can be easily exchanged with ELMo. The authors of ELMo [7] reported relative performance gains of 5.8 to 21 % [7] for various NLP tasks.

2.3.0.1 Transfer learning

In many NLP tasks, producing words or sentence representations is only a part of the entire task pipeline. For example, in classification tasks, one would feed text embeddings into classification models, which would have to be also trained on domain data. Only bare tasks such as similarity or clustering use just raw representations without additional overhead. Word2vec or GLoVe vectors are very convenient, as one can easily download already trained vectors and plug them into their models. However, as recurrent models, such as contextualized ELMo, became bigger and more complex, using them was not that straightforward. These vectors were just inputs to a task-specific, second complex multi-layer model, and constituted to a small fraction of the whole training time.

The Universal Language Model Fine-tuning (ULMFiT) [20] method brought robust inductive transfer learning for any NLP task. In a simple neural network, training weights are initialized randomly, and, during training they converge (hopefully) to generalize the optimization task. However, instead of random initialization one can use pre-trained weights. Such initial weights are already guided toward basins of attraction of minima and support better generalization [75]. By using pre-training, one obtains: 1) faster training; and 2) a better final model. The advantage is obvious and was already exploited in computer vision, fine-tuning ImageNet-like models.

The authors of the ULMFiT model used some novel techniques to pre-train their model. They used 3 stages of training: 1) LM pre-training (consuming the most resources); 2) LM fine-tuning to align weights to target task data distribution; and 3) target task (classifier) fine-tuning. During the fine-tuning stages, they presented a special type of *discriminative fine-tuning*, which allows the tuning of each layer with different learning rates. Different layers capture different types of information [76], and discriminative fine-tuning adapts to this diversity. The second technique presented is Slanted Triangular Learning Rates (STLR). Here, the learning rate has a sharp linear increase followed by a long linear decay period back to the initial learning rate. The final technique, named *gradual unfreezing*, mitigates risks of catastrophic forgetting [20]. The authors proposed to gradually unfreeze (allow weights to be trained) the model starting from the last layer. The first layers contain a more general information [76] and therefore must be affected by the last layer. The authors found that *Discriminative finetuning*, STLR and *gradual unfreezing* had the greatest impact during target task fine-tuning

when used together [20]. ULMFiT used the AWD-LSTM [77] LM which was SOTA at the time. It is a regular LSTM with various tuned dropout parameters. AWD-LSTM has no attention or shortcut mechanism and is unidirectional. However, the authors used bidirectionality in ensembling two ULMFiT models, one forward LSTM and another backward LSTM. Unlike ELMo, ULMFiT used word-based tokenization . After more than a year, a newer version, MultiFiT [78], was released with many changes to the original model. First, AWD-LSTM was replaced with a quasi-recurrent neural network (QRNN) [79]. It combines advantages of both Convolutional Neural Network (CNN) and LSTM. Like CNNs, QRNNs allow for parallel computation across both timestep and mini-batch dimensions. On the other hand, like RNNs, QRNNs allow the output to depend accordingly to the order of elements in the sequence. Despite their declared speed-up, QRNNs are less capable of sustaining complex long-term interactions such as character-level language modeling [80], which is not a case of MultiFiT. The next difference of MultiFiT from ULMFiT is that tokenization by words is changed to subwords. MultiFiT replaced the slanted triangular learning rate schedule of ULMFiT and the gradual unfreezing with the cosine variant of the one-cycle policy. The optimized learning rate curve with a maximum in the middle imparts regularization and prevents the model from overfitting. As a consequence of the high learning rate, the model is also trained much faster [81]. The last improvement of the architectural model over MultiFiT is the use of label smoothing [82]. In general, MultiFiT preserved only the discriminative fine-tuning technique of ULMFiT. MultiFiT outperformed previous best models, such as LASER [70], on zero-shot and supervised tasks. Many modifications of its predecessor, ULMFiT [20], show that the most significant contribution of both works is the establishment of transfer-learning in the field of NLP. Overall, it is seen that RNN’s development led to more and more complicated models. Additional methods, such as bidirectionality, were needed to maintain long-term input interaction. General pre-training was also slow, as input had to be processed sequentially. These drawbacks were mitigated by the newer Transformer deep learning architecture.

2.4. Transformer Models

Roots of the famous Transformer model lie in seq2seq recurrent neural network models. It was realized that a single last hidden encoder state carries the burden to encode all the information to the decoder. This eventually became a bottleneck. To avoid this, the authors of [62] proposed an attention mechanism. Now, instead of only the last hidden state, information of the weighted average of all hidden encoder states is transferred. Note that this weighting depends on a decoder state, – as each token generation process selects the information it needs most. Although it brought improvements for many tasks, it was undesirable for the goal of text representations. Many models, such as LASER [70], opted not to use it, so that they could retain a compressed sequence representation from the encoder. Attention implies that, for each new decoded token, the input sequence representation is different, and therefore not universal.

2.4.1. Vanilla Transformer

The authors of the revolutionary Transformer model [83] proposed a radical way of completely eliminating recurrence in seq2seq models with attention. The main building block in their Transformer architecture is Scaled Dot-Product Attention. Given an input sequence of token embeddings $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_T$, or in the matrix form $W \in \mathbb{R}^{T \times d_{model}}$, the first Q , K , and V matrices are calculated as learned linear projections from W . Then, attention output of the form $\text{Attention} \in \mathbb{R}^{T \times d_{model}}$ is:

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V \quad (11)$$

Here, d_{model} is the token embedding size (vector length). Transformer models use Multi-Head Attention consisting of several such independent Attentions in parallel. In general, in the encoder, each input sequence token is projected into the corresponding embedding, to which, positional embeddings are added. Then, these activations travel through multiple layers, each consisting of: 1) Multi-Head Attention with residual connection; 2) normalization; 3) two positionwise FFNN with residual connection; 4) normalization. The decoder has an additional Multi-Head Attention with residual connection, which, instead of attending to previous tokens of itself, attends to the all encoder tokens. In both the encoder and decoder flow of word embeddings, vector dimensions do not change – before and after each layer, they are of dimension $W \in \mathbb{R}^{T \times d_{model}}$. Only after all decoder layers, a linear layer projects these weights to logits, and, by using softmax, converts to class probabilities for the next token classification task.

The self-attention mechanism is better than recurrent networks in many aspects. First, it reduced the complexity of a single layer from $\mathcal{O}(nd^2)$ to $\mathcal{O}(n^2d)$. As a general rule, the length of the sequence n is much shorter than the number of words in the dictionary d . Consequently, the Transformer layers are less complex. Another improvement is the reduction of sequential operations from $\mathcal{O}(n)$ to just $\mathcal{O}(1)$. The RNN state is dependent on the previous state, and thus its calculations cannot be paralleled. The Transformer, on the other hand, is parallelizable, and this eventually resulted in the training of huge models up to 671 billion parameters, such as DeepSeek-V3 [84]. The third benefit of Transformer, compared to RNN, is its ability to attend to all other inputs instantly. This leads to very well-contextualized word vectors. All these properties allowed the authors of [83] to outperform previous RNN-based models in machine translation tasks while being trained an order of magnitude faster.

Many following works tried to improve the vanilla Transformer. The biggest issue is the quadratic time and memory complexity. Multiplication of the matrix of QK^T consumes n^2 time and memory. This limited the Transformer application for longer texts, such as documents or books. There were many works that addressed this issue, as surveyed by the authors of [85], and yet only a few modifications were accepted by the community. According to the authors of [86], layer normalization is applied before attention and feedforward blocks instead of after, and sinusoidal positional embeddings

are exchanged with relative attention with shared biases [87, 22, 88].

The vanilla Transformer of [83] was trained for a single machine translation task, and it took twelve hours on eight P100 Graphics Processing Units (GPUs). It is fast compared to the previous models, yet repeating the same training for other tasks is not practical. Later Transformer models like BERT [2] employed a transfer learning approach dissecting the costly general pre-training and downstream task fine-tuning processes.

2.4.2. BERT

The BERT [2] model is one of the first models that showed that the Transformer architecture can be astonishingly powerful. As an example, for a GLUE benchmark [89], it marked a 7.7 % point absolute improvement. BERT leveraged Transformer advantages and added some novel techniques that made it very popular.

The main contribution, as the very name BERT suggests, is a bidirectional training. Previously, language models were usually trained to predict the next word given all previous ones. BERT authors came up with a way to predict masked tokens instead of future ones. In this way, the model can attend not only to the right, but also to the left words. Specifically, during the pre-training task, 15 % of the token positions are chosen randomly to be later predicted by the model. 80 % of them are replaced with a special [MASK] token, 10 % with random tokens, and the remaining 10 % are left unchanged. Besides masked language modeling, during pre-training, BERT also uses the Next Sentence Prediction (NSP) task to achieve better performance also on sentence-level tasks. In ablation results, the authors of [2] showed that mMasked Language Modeling (MLM) outperforms left-to-right (LTR) modeling.

Similarly to ULMFiT [20], BERT is designed for transfer learning, i.e., is generally pre-trained (which is an expensive process) to be fine-tuned (a cheap process) on downstream tasks. The input sequences in BERT start with the token [CLS], which is used to train the NSP task and carries representation of the entire text sequence. Sentences are ended with the [SEP] token. In addition to token and position embeddings, segment embeddings are also added to distinguish between two separate sentences. The output from BERT is contextualized token representations, on top of which, a separate classification or regression head can be added to aid in downstream task fine-tuning. The downside of the BERT model is that the artificial token [MASK] has no applicability during fine-tuning. Also, masked tokens cannot know each other. These and other drawbacks were later exploited by newer models. However, the objective of MLM and its profound results made BERT one of the most remarkable models in NLP.

One of the first follow-up models was XLNET [90]. As the name suggests, it reused several ideas from Transformer-XL [91], in particular, the relative positional encoding and the segment recurrence mechanism. The authors combined the bidirectional capability of BERT with the autoregressive technology of Transformer-XL, which allows one to recurrently process arbitrary length sequences. Additionally, XLNET used relative segment encodings and dropped the objective of next sentence prediction. The main contribution of XLNET is the permutation language modeling objective. Here,

each sequence is trained in all possible factorization orders. It is generalized over BERT, which uses random masking. XLNET does not skip tokens that are otherwise masked in BERT. Because of changing the factorization order, it sees all training data including other targets in self-supervised language modeling. XLNET authors also had to overcome permutation-induced difficulties with a two-stream self-attention mechanism. Despite theoretical advantages, the goal of permutation language modeling is very exhaustive. The authors of XLNET [90] also used more data, and the total training cost of their XLNET-Large model was 512 TPU v3 chips for approximately 5.5 days. XLNET surpassed BERT in 20 evaluation tasks, but the complexity and cost of the model demotivated future researchers from using it.

Just a month after XLNET, a RoBERTa (Robustly Optimized BERT Pre-training Approach) model was proposed [92]. The authors found that removing next-sentence prediction loss matches or slightly improves downstream task performance and decided not to use it. The original BERT used static masking and avoided using the same mask for each training instance by duplicating data 10 times. In contrast, RoBERTa uses dynamic masking, where the masking pattern is generated when each time sequence is fed into a model. This way pre-training for more steps or with larger datasets is more efficient. Another difference is that RoBERTa is trained exclusively on full-length sequences, whereas BERT processes reduced-length sequences for 90 % of training updates and only uses full-length sequences in the remaining 10 %. The authors of RoBERTa achieved SOTA results on GLUE [89], RACE and SQuAD, outperforming both BERT and XLNET; yet, this can mostly be attributed to the use of more compute and data.

Soon (approximately 2 months) after the RoBERTa, the authors of [93] introduced the ALBERT model. This modification of BERT proposed several ways to reduce memory consumption and increase the training speed. The authors reasoned that the size of the input embedding dimension tying with the hidden state size of the model is suboptimal. Input vectors are context-independent, rarely used, and they take up a lot of space due to vast dictionaries. On the other hand, hidden states are context-dependent and produce final outputs for downstream tasks. The authors of ALBERT [93] factorized the original embedding matrix into two smaller matrices. Embedding parameters are reduced from $O(V \times H)$ to $O(V \times E + E \times H)$, where V is the dictionary size, H is the model hidden size and E is the word piece embedding size. If E is much smaller than H , a substantial reduction in parameters can be achieved. The second ALBERT model contribution regarding the model size is a cross-layer parameter sharing, which removed model size scaling with the number of layers and acted as some form of regularization. The authors showed that a not-shared 108 million parameter model average score of 82.3 is just slightly over the all-shared 12 million parameter model with an average score of 80.1. The third ALBERT contribution is the use of new inter-sentence coherence loss, which the authors called the sentence-order prediction (SOP). This loss uses consecutive segments as positive, and swapped-order consecutive segments as negative examples. The best ALBERT-xxlarge model with 233 million parameters

had 12 layers and a hidden size $H = 4096$, was trained with combined MLM and SOP losses, and did not use dropout. All ALBERT properties made it computationally more expensive but allowed it to outperform the larger, 334 million parameter, BERT-large model, achieving SOTA results at GLUE, RACE and SQuAD benchmarks as of September 23, 2019.

The authors of the ELECTRA [94] model proposed a two neural network approach, where a smaller language model, the generator, corrupts the input text by predicting masked tokens and the main encoder, the discriminator, and tries to classify which tokens were altered. The authors demonstrated that such a new self-supervised task for language representation is more compute-efficient and results in better performance on downstream tasks, comparing not only the scores of the tasks, but also the number of floating point operations during the pre-training.

There were even more BERT-related works that modified the initial BERT Transformer encoder, yet, instead, how the BERT-type encoder-only models were replaced by the decoder-only ones will be shortly covered below. Here, it is important to mention a popular T5 [22] seq2seq Transformer model. In addition to training the larger models on a larger data set, the authors of [22] also presented a text-to-text framework. It was novel in that it did not require placing additional neural network parts (called ‘heads’ in the original BERT paper [2]) on top of the pre-trained model to fine-tune it. Instead, the authors showed that the model can be fine-tuned to similar performance levels if the simple task description is added in the input sequence. This greatly simplified the fine-tuning, as only the existing weights were updated during the fine-tuning, instead of adding additional elements to the model. More generally, even a decoder-only architecture is often sufficient, as it can generate the next token conditioned on the input text, which itself can define the task. This paradigm shift led to the rise of prompt engineering [95], enabling fine-grained control over the model output. Finally, [96] demonstrated that a task-agnostic model could be fine-tuned to perform well even on unseen tasks, highlighting the power of instruction tuning and simple general encoder-type LLMs.

Pre-trained BERT and other Transformer models became a core of many NLP tasks. However, they did not produce a numerical representation for the entire sequence of text. Although the authors of BERT [2] reserved a special [CLS] token for that and used it for sentence-level task pre-training, other authors found it unnecessary for the downstream task performance and removed it in their models, as the next sentence prediction objective used to train it is considered too weak [97]. The authors of [15, 98, 99, 100] found that the [CLS] token in the BERT model produced too low quality text representations, even worse than averaged GLoVe vectors. How could such a powerful pre-trained model not have good features for text representation? One of the goals of this thesis is to find methods to extract these features from the pre-trained Transformer model like BERT.

2.5. Fine-Tuning Pre-Trained Transformer Model

As the BERT and other pre-trained models did not have the property of out-of-the-box good unsupervised universal sentence embeddings, multiple efforts were directed to fine-tune such encoding ability.

One of the first very successful models was SentenceBERT [15]. Similarly to InferSent, it was trained on Natural Language Inference (NLI) data from SNLI [19] and Multi-Genre Natural Language Inference (MultiNLI) [13] datasets. The authors used mean pooling to pool the whole text sequence representation from all contextualized token vectors from the BERT output. Unlike earlier attempts [101, 102] to use the Transformer model for the sentence representation task, the authors of [15] were the first to benefit by starting training from an already pre-trained BERT model. At the time, SentenceBERT achieved SOTA results on the most tasks of the SentEval [103] toolkit. Several works also use generative techniques to learn the text representation. The authors of TSDAE [104] used the seq2seq setting to train the encoder, capable of deriving the whole sequence representation, just as in RNN models based on machine translation. However, the authors had to cut the cross-attention connection from the decoder to the encoder, by replacing the keys and values in the attention block (11) with the final sentence representation derived from the encoder. TSDA is trained by encoding damaged input sentences (by deletions) into fixed-sized vectors and then decoding them back into the original input. Other authors of PaSeR [105] used data augmentation techniques such as synonym replacement, random deletion, and token reordering and a statistical key-phrase extraction algorithm RAKE [106] to find and mask important phrases in input sentences, with the goal of the encoder to reconstruct such phrases given only the encoder representation. The PaSeR model delivered only marginal gains for STS tasks, while TSDAE was greatly behind many models. However, TSDAE was able to outperform previous models on re-ranking, information retrieval, and paraphrase identification tasks.

After the SentenceBERT model, all the effort of the field was concentrated on learning sentence embeddings by contrastive learning. At the core here is the distance-based contrastive loss [107]. It works on positive and negative sample pairs, by encouraging representations from the first group to be as close as possible, while distancing the negative ones away from each other. This offers a lot of opportunities for unsupervised learning, given that a method of generating positives and negatives exists. Usually, positives are collected by various augmentation techniques on token, embedding, or feature level, while negatives are used as random unrelated sentences from the same batch.

One of the first and most recognized contrastive sentence representation models is SimCSE [17]. The authors showed a competitive unsupervised BERT and RoBERTa fine-tuning, while using only dropout [108] to generate positive sample pairs. They demonstrated that a simple dropout is a much better augmentation than cropping, deleting words, replacing synonyms, or using the BERT masked language modeling task to replace tokens. Unsupervised SimCSE managed to outperform SentenceBERT on STS

tasks. The authors of [17] also presented a supervised version of SimCSE, which is trained on NLI data (SNLI [19] and MultiNLI [13] datasets). They took advantage of the NLI structure, where each sentence has a corresponding other sentence with the ‘entailment’, ‘neutral’ or ‘contradiction’ label. Instead of formulating the task as a classification, like in SentenceBERT [15], the authors of [17] employed “entailment” as positives and “contradiction” as hard negatives. The resulting model resulted in significant SentEval tasks improvements to the previous SOTA results.

Multiple follow-up works tried to build on top of SimCSE and improve it. The authors of [109] noticed the bias of unsupervised SimCSE, where all positive pairs are of the same length, while negatives are almost always different. They added a simple additional repetition augmentation so that positives would also be of different lengths. Other authors of SCD [110] proposed to utilize the contrast arising from a high and low dropout rate applied to the same sentence. The authors of PromCSE [111] proposed to address the domain shift limitation with prompt-based contrastive learning and better exploit hard negatives by using energy-based Hinge loss. They trained a small-scale Soft Prompt, a set of trainable vectors appended to each layer and each token activation in the Transformer model. By employing the same training data as SimCSE, the authors managed to only improve the performance for STS tasks. The authors of WhitenedCSE [112] proposed to improve contrastive learning by augmenting the positive representations when using whitening. Given representations of sentences from the encoder, their method randomly groups features and applies whitening independently within each group. Then, a reshuffling operation is performed to recover the original feature arrangement, yet modified by the whitening. During training, the authors used multi-positive contrastive loss objective. Resulting representations achieved better alignment and uniformity properties while also performing better than SimCSE [17]. The authors of [113] proposed a DiffCSE model, which combined SimCSE and ELECTRA training objectives. The idea is that the representation produced by SimSCE during the training is also used in ELECTRA model’s discriminator to help discriminate the altered tokens. Both contrastive and classification losses are backpropagated to the encoder. The DiffSCE model scored higher than SimSCE in SentEval [103] benchmark tasks of semantic textual similarity and classification. The authors of RankCSE [114] proposed to account for ranking during the training. The idea is that, instead of only separating positive and negative ones, it is better to also distinguish high and moderate similarity levels. They proposed a training objective which consists of 3 parts: 1) contrastive learning, like in SimCSE [17]; 2) ranking consistency, which ensures that two positives with different dropout masks should have the same similarity ranking with other sentences; and 3) ranking distillation, which tries to follow the ranking knowledge from pre-trained teacher models. Despite the need for a parent model like SimCSE to distill the ranking information and an associated increase of the training time, the authors of RankCSE [114] showed it to perform better than SimCSE and DiffSCE in STS and SentEval [103] classification tasks. The authors of AoE [115] came up with the idea to improve optimization itself during the contrastive training. In particular, they

addressed cosine functions and their saturation zones. According to the authors, gradients can vanish if embeddings pairs are nearly aligned or anti-aligned. They added the second, angle objective, which minimizes the angle between complex embeddings of given positives. Real and imaginary parts are constructed by dividing usual embedding vectors into the corresponding parts. Their AoE model, trained by the supervised MultiNLI and SNLI data, improved upon previous models in many tasks considered. Many other methods were also used to generate positives for contrastive learning, such as back-translation [116, 117], adversarial attacks [97, 118, 119, 119], token shuffling and cutoff [97], synonym replacement [120], masked language modeling [121], or using separate models to encode two different versions of the same text [122, 123, 119, 124]. Other directions for improving contrastive learning include the incorporation of soft negatives [125], the use of multiple diverse positives [126], the use of location information [127], or the scheduling of training data according to its quality [128]. Overall, the best sentence representations are now acquired by using Transformer models fine-tuned by contrastive learning. Fine-tuning is the only viable approach to turn a pre-trained model into the text encoder. The important factor here is the fine-tuning dataset with positive and negative relations between samples, such as NLI. There were no such successful methods of extracting representations from only the pre-trained state of Transformer models. This task will be given more attention in Chapter 3.

2.6. Evaluation of Text Representation

To measure the performance of the Natural Language Understanding (NLU) system, an evaluation must be carried out. Usually, this is done automatically with a specially designed benchmark. It is the main ground that guides and spurs development, allowing researchers to focus on a simple measure and seek to improve it against the work of others.

The custom dataset is the main building block of any benchmark. Human annotators, usually experts or crowd workers, are used to compose it. The goal here is to have unbiased samples with a distribution that closely resembles real-life data. Any heuristic used can spoil the performance measurement by allowing the model to learn shortcuts. In total, a task needs training, validation, and testing splits to allow the model to first learn the structure and the goal. Usually, the number of samples is counted in thousands, and thus the cost of human labor is huge. In addition to the dataset, the benchmark also has a metric and a way to aggregate the performance [129].

There are two distinct groups of tasks related to text representation: 1) the STS task, used to intercept the meaning in textual representations; and 2) NLI, used to evaluate models on how well they work in NLU. They will be described in more detail below.

Semantic Textual Similarity Each STS task sample consists of two sentences and a score indicating how semantically equivalent these two sentences are. Evaluation is very simple: one has to convert sentences into numerical vector representations, calculate the cosine distance between the two sentences in each sample, and finally

calculate how this distance correlates to the annotated scores. Such simplicity turned STS tasks, such as STS from 2012-2016 annual contests [130, 131, 132, 133, 134, 21], into a very popular choice for automatic textual embedding evaluation.

Despite popularity, there are some downsides to the STS task. The authors of [135] criticize unsupervised evaluations based on Pearson and Spearman correlations, particularly for semantic textual similarity tasks. The authors show that simple normalization by the z norm can greatly improve the scores. The argument is that normalization is a simple operation that could easily be learned by a machine learning model and therefore should not affect the evaluation.

Another line of criticism is that “similarity tasks show little or negative correlation with the performance of downstream tasks” [136]. As a result, most researchers shifted reporting results on a single STS task to multi-task benchmarks with downstream transfer tasks, such as SentEval [103], which, in particular, has incorporated STS from 2012-2016 annual contests. In addition, new benchmarks are proposed that have stronger correlations with downstream tasks, such as EvalRank [136].

Natural Language Inference NLI is considered the most canonical task in NLU. Starting with the challenge *Recognizing Textual Entailment* (RTE) [137], the main boost in the field came with the SNLI corpus [19]. It contains 570,152 samples of premise and hypothesis pairs and the label indicating whether the pair of sentences is in contradiction, neutral, or in entailment relationship. The huge size of this corpus saw benefits in deep neural network architectures that now claim SOTA on this task. Some derivative works supplemented the original SNLI corpus [19]. One drawback is that it was created while using heuristics [129], which can be learned by models, so it is not an ideal dataset. The authors of MultiNLI [13] presented additional 433k samples from ten different genres of written and spoken English, covering almost the entire complexity of the language. The authors of [138] recasted 13 different common NLP datasets in a common context-hypothesis inference format. The resulting Diverse Natural Language Inference Collection (DNC) encompassed 7 semantic phenomena: event-factuality, named entity recognition, gendered anaphora resolution, sentiment analysis, relationship extraction, pun detection, and lexicosyntactic inference. The authors of ANLI [139] collected three rounds of data by using the *Human-And-Model-in-the-Loop* strategy. In each round, humans were asked to provide a hypothesis that convinces the model to misclassify the label, and the resulting samples were then used to train the new model.

According to Budanitsky and Hirst [140], the concept of ‘relatedness’ is more general than that of “similarity” as similarity is just one of many forms of relatedness. Therefore, the authors of [141] created a SICK dataset that not only has labels *entailed*, *contradiction* or *neutral*, but also has a 5-point scale score of relatedness.

The authors of [142] proposed to unify natural language inference with common sense reasoning. They proposed a SWAG dataset which challenges models to finish the given sentence with the next multiple options given. Later, the task was hardened to Hel-

laSwag with the authors of [143] proposing a more difficult set of samples achieved by adversarial filtering.

Natural language inference is a universal task that relates to a number of others. For example, in the question natural language inference task (QNLI) [144, 89], positive examples are question and sentence pairs that contain the correct answer. Quora Question Pairs (QQP) [145] seek to distinguish if two questions are duplicate, while the goal in the Microsoft Research Paraphrase Corpus (MRPC) [146] is to distinguish if the pair of sentences are semantically equivalent. These and other tasks require the system to succeed in natural language inference to solve them. Therefore, NLI is a good task to probe the quality of NLU systems.

2.6.1. Multi-task benchmarks

Evaluation on a single task is not practical. Models can quickly overfit it. Then, while exhibiting excellent results on a particular task, the model can be a total scapegrace on the others. Therefore, a consensus is being reached that models should be evaluated on a collection of tasks. Some more popular multitask benchmarks will be covered in the following sections.

2.6.1.1 SentEval

Introduced by the authors of [103], it is a toolkit to evaluate the quality of universal sentence representations. SentEval comprises entailment, semantic relatedness, paraphrase detection, caption-image retrieval, and various binary and multiclass classification tasks. General-purpose sentence embeddings are used as features of logistic regression or a Multi-Layer Perceptron (MLP) on top. In addition, an unsupervised evaluation is performed by using STS.

The authors of [135] criticize the ambiguity of logistic regression or the MLP option. They suggested reporting results for both classifiers. The authors also note that more challenging benchmarks must be adopted that do not allow simple baselines, such as random projection models [147], to achieve substantial performance. However, SentEval is considered the main benchmark for static feature-based sentence representations. It was the main benchmark used to evaluate sentence embeddings from fine-tuned Transformer models (see Section 2.5).

2.6.1.2 GLUE and SuperGLUE

These benchmarks, unlike SentEval, do not assess embeddings, but rather the models that themselves solve the presented tasks. GLUE [89] includes 9 English sentence understanding tasks. Two are with single-sentence inputs: Corpus of Linguistic Acceptability (CoLA) [148] and Stanford Sentiment Treebank (SST-2) [149], three tasks involve detecting semantic similarity (MRPC, QQP, STS-B), and the last four are of natural language inference. GLUE became a battleground for emerging Transformer-type [83] models, quickly oversaturating the benchmark. Thus, replacement of a more difficult set of SuperGLUE [150] was proposed. However, both benchmarks were sat-

urated within a year.

The authors of the ST5 model [128] also proposed using GLUE together with SentEval for the static embedding evaluation. They added the constraint that “each input must be independently encoded into a fixed embedding space representation that can then be fed to additional layers in order to make a prediction”. Such a restriction leaves a wider gap for improvement.

2.6.1.3 Dynamic benchmarks

There is a trend for new dynamic benchmarks to emerge. In contrast to static ones, they are supposed to be constantly updated, particularly while focusing on the current model weaknesses. In this way, the wide remaining margin for improvement is continually ensured. Such benchmarks also solve the problem that models trained on a most available text on the Internet may inadvertently also consume the evaluation data, thus violating train-test separation. An example of such a dynamic benchmark is Dynabench [151], which covers natural language inference, question answering, sentiment analysis, and hate speech detection. The other GEM benchmark [152] focuses on the natural language generation task. As automatic evaluation of text generation has problems of good correlation to human judgments, not only the dataset, but also the metrics and evaluation settings are proposed to be dynamic and subject to improvements in the future. For autoregressive language models, there are benchmarks consisting of more than 200 tasks [153].

2.6.1.4 MMLU

As models become more powerful, so has the task of becoming more proficient. The authors of [154] presented an MMLU benchmark that measures the multitask accuracy of a text model. It covers 57 tasks, spanning various subjects and areas, that only experts can solve. To succeed, such models must have extensive world knowledge (serve as knowledge bases) and expert-level problem-solving ability. The researchers framed all tasks in a unified multiple-choice question format, already seen in the decaNLP [155] benchmark. From the 2022 score of 67.5 % (average score of the Chinchilla model [156], trained with the massive 1.4T number of tokens) to the 2024 score of 86.4 % (score of GPT-4 [27],) this benchmark is also saturated. The authors of MMLU-Pro [157] improved the benchmark by eliminating trivial questions, expanding the choice options from 4 to 10, and adding more challenging questions focused on reasoning. These changes also reduced the sensitivity of model scores to prompt variations, allowing a better comparison of models that perform similarly [157].

2.6.1.5 Multilingualism

One of the current limitations of both models and tasks is the dependence on the popular English language. Therefore, multilingual benchmarks are needed. The authors of [158] proposed XTREME that evaluates the cross-lingual generalization capabilities

of multilingual representations in 40 languages and 9 tasks. It is designed to require reasoning about different levels of syntax or semantics. The benchmark consists of classification (NLI, paraphrase), structured prediction (part of speech tagging, named entity recognition), question answering, and retrieval tasks. The concurrent XGLUE [159] benchmark has similar tasks with the addition of generation tasks (question generation and news title generation), while GEM [152] covers only the generative ones. Although these benchmarks are designed to be multilingual, this is usually the case only for a few tasks out of many given (others have up to 10 times fewer supported languages). Moreover, multilingual benchmarks with only up to 40 languages are still far too small numbers in the shade of thousands of spoken languages.

2.6.1.6 MTEB

Text embeddings are very versatile and are used in many types of tasks. Recently, the authors of [12] developed the new Massive Text Embedding Benchmark (MTEB). They addressed the problem that most works evaluate text representations mainly on STS tasks, and proposed a benchmark consisting of 58 datasets divided into 8 groups: clustering, bitext mining, retrieval, STS, summarization, classification, pair classification, and reranking. The authors compared 33 models and found that “no particular text embedding method dominates in all tasks”, thus indicating the task-specific polarization of text embedding methods. At the time of writing, an expansion of Massive Multilingual Text Embedding Benchmark (MMTEB) was just created [10]. The authors of expanded MTEB went even further to 550 tasks and 1090 languages. Currently, the MMTEB benchmark is the most closely related to the topic of this dissertation.

2.6.2. Probing embeddings

There is a separate line of research that focuses on probing static representations. A simple classifier is trained on top of the frozen embedding and is thus considered to represent the quality of the frozen vector. From an information-theoretic perspective, training the probing classifier can be seen as estimating the Mutual Information (MI) between the intermediate representations and the target property [160]. Generally, each classifier focuses on a particular linguistic feature [161]. According to the survey work on the analysis methods in NLP of [162], in the literature, this method is called the “diagnostic classifiers”, the “auxiliary tasks”, and the “probing tasks”. It is related to the decoding of human brain activation patterns in cognitive neuroscience studies [163], in which, subjects are presented with a common language input, and the corresponding brain regions with distinguished activity patterns are searched.

Similarly to the human brain, artificial neural networks represent a black box. That is, only the input and output can be directly interpreted, and not the intermediate representations. Thus, probing tasks present a unique way to analyze inner embeddings. This was in particular helpful for earlier representation studies where the inner vectors of sequence-to-sequence models were only a byproduct of the main task such as machine translation.

One of the first works of [164] that probed sentence embeddings proposed to search for semantic information. The authors of [165] used probing tasks to acquire more fine-grained information about sentence vectors than the performance of the original task that produces these embeddings. In particular, they probed the way the sequence length, the items within it, and their order are encoded. The work of [166] proposed using probing classifiers to reveal how representations evolve across deep neural network layers. [167] used probes to test the hypothesis of how hierarchical compositional semantics are processed. By testing the content and negation probes, the researchers in [168] showed that the average of the word vectors encodes the word content surprisingly well, but cannot distinguish the presence of negation. The authors of [169] used probing tasks to reveal that the encoder of the NMT model, trained for the machine translation task, is much more capable than the auto-encoder and is capable of acquiring expressive representations for syntactic tasks. Several years later, the work of [170] used probes to show that an encoder of the bidirectional LM captures more useful features than the translation one. Word-level probes were used to show that contextual embeddings improve their lexical (context-independent) baselines in [171]. The survey of the abilities of the probing classifiers to better inspect deep neural networks and also compare them resulted in probes being very popular among researchers.

There are some suites of probing tasks available. The authors of [161] presented 10 probing tasks designed to capture simple linguistic features of sentences, covering surface, syntactic, and semantic information. Such a wide range of properties can provide broader insights, as in [172] it was shown that the controlling parameter of their discrete cosine transform method shifts the best performance from surface to synthetic and finally to semantic tasks. Similar probing sets were also crafted for token-level embeddings, as 16 tasks in [173].

2.6.2.1 Criticism

Despite the widespread adoption, probing methods are increasingly criticized [160]. First, it is not clear to what extent the classifier itself contributes to the accuracy, and what contribution the static representation makes. One of the first surprises was found in [170] that the untrained LSTM baseline performs very well on syntactic tagging tasks. The authors of [174] argue that a sufficiently expressive probe with enough training data can learn any task on top of any basic representation. To reduce the effect of the classifier, the first works in the field preferred linear probes. Yet, it does not seem reasonable, since the way information is extracted from frozen representations also depends on the classifier capabilities. The authors of [175] showed that selecting the best-performing probe is optimal to achieve the best estimate of MI between static embeddings and target labels. In [176], it was suggested to directly account for the complexity of the classifier by the proposed minimum description length probing (MDL). It is a minimum number of bits needed to transmit the labels knowing the representations, and it consists of a quality of fit of the model and the cost of transmitting the model itself. Therefore, the complexity of the model becomes accounted for.

A recommended way to weigh the effect that representation has on the accuracy of the classifier is to employ a relative baseline. The authors of [174] and [175] construct a control task ([175] calls it a control function, but [177] summarizes that both methodologies are identical) that decodes a random function from the representation of interest. Then, instead of accuracy, the difference between the accuracy of the target and control tasks is measured. The authors of [178] propose to calculate the performance of conditional probing. It is the accuracy difference of two probes: one fed with concatenated baseline representation and the one of interest; and the second probe fed with only the baseline embedding. In this way, only the contribution of the probed representation is measured.

Another weakness of probing tasks is the questionable existence of probed properties as the true building blocks in neural representations. The authors of [135] note that the correlations between the probing tasks and the downstream transfer tasks are not consistent, particularly when analyzing the work of [161, 179]. Surprisingly, in [172], a negative correlation was observed between the probe and the downstream transfer tasks. The authors of [180] constructed a synthetic task where a specific property c is just random noise in the input of the original task. Surprisingly, they showed that a probing classifier was able to probe frozen representations and distinguish the property c well above the chance level. They concluded that models learn to encode linguistic properties incidentally, and that probes should not be used to explain what the main task relies upon.

2.6.2.2 Disposing of the classifier

The problem that the probing classifier interferes with the evaluation of the representation can be solved by eliminating it. There are several works that proposed methods that probe representations in a non-parametric way.

The authors of [181] devised a clever technique to reveal BERT’s own syntax by collecting the model output. They mask all possible pairs of words one at a time to determine the impact each word has on the other. Later, the resulting sentence impact matrix can be aligned to ground-truth dependency trees and compared with. Although this technique is limited to dependency tree probing, the authors showed that BERT has learned its own syntax as a by-product of self-supervised pre-training.

Another method without classifiers, called subnetwork probing, was proposed by the authors of [182]. Instead of the usual linear or MLP classifier, a version of the original model with a subset of the weights set to zero is learned. An advantage of this method is that one can clearly see how many weights were pruned in each layer. This way, each deep neural network layer’s influence is clearly depicted.

Some work also makes great use of clustering. The authors of [183] modified the training of the probing classifiers so that they could extract the structure from the input representations. In this way, latent ontologies can be axed out of input features. They analyzed clusters and found, for example, that both the BERT and ELMo models group numeric/quantitative entities together. In other work of [184], it is argued that

different representations may need different classifiers, so they propose to directly analyze the structure of raw representations. By using regular probing tasks, they acquire representations and cluster them according to both the embedding and label. They obtain minimum distances between clusters of different labels and show that these distances correlate with the accuracy of classical probing classifiers. It is understandable because, for example, a small distance between two different label clusters in the representation space would be difficult for the classifiers to separate. This way, minimum distances between clusters can describe how well representations can be fitted. The use of clustering is also present in the work of [185], where the authors evaluated their representations by using eight short text clustering datasets. The clustering of embeddings is a promising direction in representation probing and evaluation.

Overall, there is no mature consensus on how the representations inside the model weights should be probed. Due to the wide criticism of probing, it is more resourceful to stick to the direct NLP tasks as the main representation evaluation metrics.

2.7. Summary of the Review

The review covered the evolution of text representation models, from statistical methods to Word2vec-style embeddings, RNNs, and Transformer-based architectures. Key factors that influence representation quality and model success were found, including:

1. Deep neural networks enable the capture of complex relationships in text.
2. Transformer models, which excel in scalability (via parallel computing) and effectively share information between tokens through the attention mechanism.
3. Subword tokenization, which effectively handles rare words, morphology, and out-of-vocabulary issues.
4. Pre-training, which decouples expensive general knowledge acquisition from the more efficient fine-tuning stage.
5. Contrastive learning, which improves the quality of the representation by optimizing the distances between the positive and negative samples.
6. NLI-type datasets, which serve as a rich resource for training models to understand semantic relationships.

In this dissertation, the first four factors will be delved into. First, a deep learning pre-trained Transformer model will be used to extract its acquired general knowledge in the form of text representations using various methods (Chapter 3). Next, Transformer models will be trained to induce text-level transformations for tasks in the Lithuanian language and investigate how it effects text representation (Chapter 4).

The literature review of text representation evaluation revealed that the most popular evaluation choices are STS and Transfer tasks from the SentEval benchmark. Although techniques to probe embeddings were also analyzed, yet there is no common practice

in the field of which probing methods are the right ones. Evaluation directly on tasks better resembles the performance of representations and has a higher practical value. This is especially important for the Lithuanian language, for which, Transformer applications will be covered in Chapter 4. In such a case, there are no benchmarks and tasks that are easily crafted, and the automatic means are the only viable solution which produces additional value.

3. IMPROVING TEXT-LEVEL REPRESENTATIONS FROM PRE-TRAINED MODELS

This chapter focuses on the learnable text representations of the popular pre-trained BERT [2] Transformer model. Many methods are presented to improve the pre-trained model outputs and evaluated on 3 different types of tasks. The first findings of this work were published in [186].

A comprehensive review of the literature on token aggregation and post-processing methods of numerical representations in Section 3.1 will be provided. Then, the methodology is presented in Section 3.2. In Section 3.3, results of the experiments are provided.

3.1. Review of Enhancement Methods

3.1.1. Combining word vectors

Many works were done on word-level vectors. At the time of writing, popular publicly accessible Word2vec [24] and GLoVe [8] embeddings can be easily downloaded. The raw outputs of the BERT model are also made up of contextualized vectors corresponding to each input token. However, aggregating to a single high-quality vector for the whole sequence is challenging. The Principle of Semantic Compositionality (usually called Frege’s principle) states that “the meaning of an expression is a function of, and only of, the meanings of its parts together with the method by which those parts are combined” [187]. Many scholars use this as a guide to how a sentence/paragraph/document or just a text vector should be formed. According to the principle, for much easier acquisition of word vectors, only the method of combination needs to be found. Therefore, in this subsection, the most popular candidate combination methods are reviewed. These methods produce a sequence vector from its multiple word vectors.

3.1.1.1 Averaging

As neural vectors became more effective than traditional approaches [188], it turned out that it was common to derive a vector for a sequence of words simply by adding or averaging individual vectors. In 2009, the authors of [189] reported that averaging is the most common method for composing phrase representations, where the vectors of individual words are averaged to obtain a single vector for the phrase. Despite its simplicity, the basic rule of averaging word vectors was shown to work very well [190, 191, 192]. Later work, such as [193, 194, 195], even showed that simple pooling methods, such as basic vector addition or averaging, match or outperform much more sophisticated methods for encoding the meaning of a text sequence. The authors of [196] reported that, in out-of-domain scenarios, simple architectures such as word averaging outperform complex LSTM models. In addition, they are even competitive

with systems tuned for particular tasks. Later work by [197] evaluated various compositional models and found that word vector averaging performed reasonably well in most supervised benchmarks. According to the authors of [15], even for the famous BERT Transformer model, to map the sentence to a single vector, the most popular approach is to average the word embeddings of the BERT output layer.

As averaging is well suited to acquiring sentence optimization, some methods take averaging into their structure to further optimize it. One such example is the deep average network (DAN) [198]. It takes the input as an averaged text sequence and passes it to one or more feedforward layers to finally perform linear classification. Another, C-PHRASE model of [199], is trained to predict the contexts of phrases from the additive combination of their elements. Such a design results in a useful property of C-PHRASE, that summing word vectors yields sequence representation. The authors of [200] found that one should use an average of all hidden states of LSTM, rather than using representation only from the last one. In a similar fashion, the averaging or summing is optimized in the works of the Siamese CBOW model [201], Sent2Vec model [55], and in other works [202, 203, 204].

A model derived from BERT, specially designed for sentence embeddings, SentenceBERT, also uses averaging to pool words from two sequences to a pair of fixed-size sentence embeddings. Later, by using the pair and its element-wise difference, the representation is passed to the softmax layer for classification and regression tasks. The authors of [205] tried to improve this configuration by replacing the averaging with a convolutional neural network. However, their improvement was limited to only better scores for the SALBERT model, which was still lower than SentenceBERT based on averaging. Therefore, calculating the mean embedding from multiple word vectors is still a solid approach to the inner workings of current sentence models.

Averaging is used in various scenarios. Some authors employ it to derive static embeddings from contextualized models. In [202], word vectors are learned by predicting them from the average of all contextual embeddings of words (except the target word) returned by the BERT encoder. Other authors of [206] derived static word embeddings from contextualized Transformer representations by averaging word embeddings for each word in 100k different contexts. Such derived static versions are of a better quality than classic Word2vec and share their benefits, such as having a computational cost which is tens of millions of times lower than using standard contextual embedding models [202]. The other scenario is to compose a new more expressive meta-embedding from different models and domains. As a means of composition, the authors of [207] proposed concatenation with some trainable methods. However, a later work [208] found that averaging word vectors and padding them with zeros to compensate for dimensionality mismatches is a surprisingly effective method.

Despite its popularity, averaging has inherent deficiencies. The most prominent one is the loss of word order information. For this reason, it is sometimes called a BoW representation (that is, imagine that words lie in the bag in no specific order). As shown in [198], models based on averaging are weaker in tasks that require syntax information.

The mean pooled static representations of the words “dog chased human” and “human chased dog” will be the same. In addition, simple summation can cause destructive interference, thus affecting valuable information. This is especially relevant in long documents. In addition to a neglected sentence structure and word order, the authors of [197] also note that individual word identities are lost, and noninformative words are more prominently represented than essential ones.

However, to some extent, drawbacks can still be overcome. Most tasks do not usually rely on the word order, and the number of occurrences of order-sensitive elements, such as double negations, is generally low. Word vectors have a rich representation, and some related knowledge can still be found in them. For example, the authors of [165] showed that the average of the word vectors retains information about the original sequence length. It is even more the case with contextualized Transformer representations, where, starting from the second layer, each hidden vector is expected to ‘know’ every other token vector and, therefore, to be aware of the word order.

3.1.1.2 Weighted average

Not all constituent words are of the same importance; therefore, the corresponding vectors should be weighted. As a surprising event in Information Theory has higher information content than an expected event [209], some specific words should also add more weight to the sentence vector than others.

In an early work [189], different weights for the members of the pairs of adjective–noun, noun–noun, or verb–object phrases can already be found. Later, work went from formal semantics to statistics-based measures, in particular, the TF-IDF scores. They can be used as features on their own, such as in [210], but it is better to use them as weights in neural word vectors. For example, the authors of [211] proposed the Composite Document Vector as a concatenation of Inverse Document Frequency (IDF)-graded weighted Word2vec vectors and TF-IDF features. The authors of [16] proposed another better weighting approach, called Smooth Inverse Frequency (SIF). Here, higher-frequency words are down-weighted smoothly. The authors of [197] evaluated various compositional models and found that weighted averaging, in particular SIF, resulted in better performance in unsupervised similarity tasks that outperformed all other models. The author of [212] presented the uSIF method, an improvement to SIF, omitting hyperparameter tuning and constructing weighting with both word frequency and word vector length information.

A drastic case of weighting – a total removal of some words – is also shown to improve results. The authors of an older work [191] removed the stop words from the sequence before averaging. Meanwhile, as shown in [97, 213], the current performance of the BERT Transformer model is also significantly improved in STS tasks if the most frequent tokens are removed. In other cases, the task itself focuses only on a few words, and others become redundant. The authors of [214] found that, for the relation extraction task, some sentence embedding methods work better with shorter spans of words than the original sentences. They performed sentence segmentation in a way that a

(sub)sentence would cover the identified entity mentions. Other authors of [214] also noted that SentenceBERT and Quickthought in spans or short segments containing two entity mentions are more clusterable than on the original sentences. These works also highlight the unequal contribution that each word has to the final sentence representation vector.

Some works use more complex word weighting schemes. For example, the authors of [200] proposed a form of attention to weight each hidden state of LSTM. Other authors of [215] proposed the idea that each word brings a novel orthogonal basis to the sentence. Therefore, the length of a projection in this direction can be converted into a word's weight for use in the averaging. They come up with the final weight consisting of three scores: novelty, significance, and corpus-wise uniqueness. The authors of [214] found that for the methods analyzed, such weighting of GLoVe embeddings made them the most clusterable. Similar work by the authors of [216] incorporated alignment and novelty scores and applied them to contextualized representations of SentenceBERT. Here, the alignment measures how well the word aligns with the neighboring ones, i.e., a well-aligned one is less informative and should be weighted less. The novelty, similar to [215], is expressed as the magnitude of the orthogonal component of the word to the subspace of neighboring words. Furthermore, the authors of [216] perform average weighting through all SentenceBERT layers and also weight each layer-aggregated vector via l_1 -normalized variance. This means that words that evolve faster across layers will receive higher weights since they have greater variance. Although complex to implement, such methods are reported to provide some minor performance improvements.

Surprisingly, the norm of word vectors has a large dispersion, as observed by the authors of [217, 218], and, during the average pooling operation, it acts as a weight of the word vector. The authors of [55] observed that their trained word vector in this way down-weights frequent tokens by itself, and this weighting follows the hypothesis of Luhn [219], a well-known information retrieval paradigm, stating that mid-rank terms are the most significant for discriminating content. By using these insights, the authors of [26] used the norm of word vectors as a proxy for the importance of words.

3.1.1.3 Clustering

As shown by [220], the average of word vectors leads to a loss of information. It is particularly significant for longer text sequences, such as documents. Therefore, the idea was developed to aggregate vectors in a more smooth and information-preserving manner. The authors of [221] clustered the word vectors into k groups by using k -means. Then, for a document, each cluster vector is obtained as a sum of its constituent vectors. The final representation is then the concatenation of the cluster vectors and the Inverse Cluster Frequency (IDF) values, which are calculated using the IDF values of the words present in the document. Other authors of [222] proposed a similar method, but viewed it as one that reduces the dimensionality from words to concepts. Concepts (as clusters) are created by clustering word vectors generated from Word2vec, and fre-

quencies of these concept clusters are used to represent document vectors. Finally, similarly to TF-IDF applied to BoW, a weighting scheme, concept frequency-inverse document frequency (CF-IDF), is applied to the acquired bag-of-concepts. The authors of [223] proposed the concepts of word containers and document containers to explain how such procedures bring benefits. Similarly to other works, they perform clustering of words into distinct clusters. Vectors belonging to the same cluster are averaged, while resulting representations among different clusters are concatenated. By using standard contextualized pre-trained Transformer models with frozen weights, the authors of [224], instead of a simple mean pooling, proposed training a categorical variational autoencoder and reported the performance improvement on some STS tasks. This way, similar to clustering, the lower intermediate compressed representation would carry the essential representation of the sequence. Overall, using clustering of independent word vectors allows the document representation to be expressed in a smaller space of concepts, whose concepts are otherwise erased due to destructive interference during the global averaging.

The authors of [225] improved the aforementioned method of [221] by using soft clustering and allowing a single word vector to participate in multiple clusters. More specifically, each word is represented as a $K \times d$ dimensional embedding, where each k^{th} row corresponds to the original word vector weighted by its probability distribution in the k^{th} cluster. The embedding of this word is weighted with the inverse document frequency of the word and summed with the embeddings of other words to form a document vector. As many values in such a vector were observed to be close to zero, sparseness is induced with a given minimal value threshold. The authors of [225] named their method the Sparse Composite Document Vector (SCDV).

The success of SCDV was supplemented by numerous other contributions. The authors of [226], instead of the Gaussian Mixture Models (GMMs), used K-SVD (an algorithm for designing overcomplete dictionaries for sparse representation) [227] for the topic modeling. Furthermore, they used a newer word vector algorithm Doc2VecC and a modern SIF weighting and removal of the top principal component of [16]. Other authors of [228], with their SCDV-MS method, also performed word sense disambiguation to dissect polysemous words into distinct vectors. In this way, the quality of the clusters improved. To induce sparsity, the authors also applied hard thresholding in an earlier stage in word cluster assignments and trained word vectors with Doc2vecC. Next, with the emergence of Transformer models, the authors of [229] utilized contextualized representations in their SCDV + BERT (CTXD) method. First, the corpus is contextualized following the technique of [230]. That is, each corpus word is clustered among its individual recurrence contexts into distinct clusters corresponding to the different meanings of the same word. After such a procedure is repeated for all words, the vector for each word will then be the centroid vector of the closest cluster. The authors of [229] then repeated the original SCDV procedure, and the result was an average improvement for STS tasks. Therefore, the idea of turning global averaging to local inside-cluster averaging did not change; only newer representation techniques

and models were employed.

3.1.1.4 Spectral methods

A text sequence comprising word embeddings can be interpreted as a multidimensional signal over time. Therefore, temporal summarization techniques can be employed. One such method is the Discrete Cosine Transform (DCT), originally presented in [5]. This invertible function maps an input sequence of N real numbers to the coefficients of the N number of orthogonal cosine basis functions. The DCT components are arranged in order of significance, with the first one being proportional to the simple average of the sequence. DCT is used in data compression by preserving only the most important coefficients.

In [172, 231], DCT was shown to outperform simple averaging of word embeddings. First, it can be attributed to the fact that DCT is structure-sensitive, as it captures signal dynamics. Second, the DCT has multiple coefficients, with the first one already corresponding to averaging; thus, it should capture more information and enrich the performance of probing classifiers. The authors of [172] apply DCT along the sequence of words for each embedding dimension. Then, they retain lower-order coefficients and concatenate them to obtain the overall feature patterns in the word sequence. One drawback of this technique is that short sentences must be padded with zeroed vectors. The following work [231] also showed good results for DCT in multilingual and cross-lingual settings.

Concerning the spectral decomposition of DCT, the authors of [232] proposed EigenSent, which utilizes Higher-Order Dynamic Mode Decomposition (HODMD). The method summarizes transitions in a sequence of words into one representative sequence embedding. The authors found that the best performance is achieved when such an embedding is concatenated with simple word vector averaging. In this way, information on both the dynamics and the scale of the sequence is captured.

3.1.1.5 Using special tokens

Modern Transformer models have several special tokens that can play an important role during representation aggregation. For example, BERT has [CLS] designed for the next sentence prediction task, and it is supposed to carry the meaning of the input text sequence. The authors of [123] tried to use this property and trained the model to learn to aggregate everything in this [CLS] token. Other special tokens are [SEP], used to separate two text inputs, [MASK], to concentrate the representation of the masked word, padding, and various sentinel tokens (as in T5). Despite their intended purpose, special tokens are generally ignored, as in [233, 99, 234], and a simple average is used instead. However, such tokens are important, as the authors of [235], after fine-tuning BERT, found a clear tendency for the earlier layers to pay attention to [CLS] and for later layers to pay attention to [SEP].

A prompt-based learning paradigm (see the survey [95]) can also be leveraged. Here, instead of fine-tuning the model to the downstream task, one reformulates it to look

like the one that was being solved during the model pre-training process. Therefore, the main effort goes to prompt engineering, that is, finding the most appropriate prompt for the given task. Such a strategy is especially suitable for very large, even colossal language models that are difficult to train.

The authors of [213] successfully used prompt engineering to better capture sentence embeddings with the BERT model. During manual prompt search, the best prompt found was “This sentence: "[X]" means [MASK].”, where [X] is replaced by the input sentence, and the output vector of the token [MASK] is used as a final representation of the input sequence. Then, they further optimized the template by fine-tuning it on NLI data with the contrastive objective, while the BERT model weights were frozen. The authors used the continuous template technique of [236], where each template token is treated as a vector and optimized by gradient descent. The final template was shown to outperform multiple untrained baselines. The authors also showed that different templates can be used effectively to represent the same sentence with different points of view during supervised contrastive learning. The following work by [125] adapted the idea of using prompt instead of mean pooling, and also used this representation extraction during contrastive learning.

3.1.1.6 Aggregating through layers

One special aggregation that is possible with newer contextualized models is through deep neural network layers. For simplicity, here, the Transformer model blocks are referred as layers.

One can imagine that “each layer will increasingly magnify small but significant differences” [198]. Therefore, the early layers capture more fundamental and low-level information [235], which dominates the learning of shallow lexical and meaning-related knowledge [237]. The authors of [224] reported that the word embeddings of the lower layers of BERT perform better than their upper layers on a word analogy task, and other authors of [206] showed that the first quarter of the layers of the models perform best in lexical semantic understanding. In the middle layers of the Transformers, the hidden states are the most transferable [173] and contain the most relevant information [171]. The authors of [238, 239] report that the middle layers of the multilingual Transformers are more multilingually aligned. According to the authors of [237], with the inclusion of context in the upper layers, the encoded concepts evolve into a linguistic hierarchy where the middle and upper layers have a better representation of the core linguistic and semantic concepts. Finally, as the author of [240] explains, upper-layer representations become more context-specific.

A general tendency, as shown by [122], is that the STS performance increases until the middle layers before decreasing toward the final ones. The authors of [122] even report that the final layer of most Transformer models produces the worst-performing representations. Most scholars agree that this is due to the overspecialization of the last layers in the pre-training task [241]. Therefore, the final layers of BERT are the most task-specific [242] (but not as much as LSTM, as analyzed in [173]). Even during the

fine-tuning phase, the authors of [235] determined that the last two layers encoded the highest share of task-specific features attributed to the score gain. Therefore, the last layers are specific to the training objective and cause problems if it differs substantially from the downstream task objective.

However, some work in [243, 244] found that it was beneficial to apply post-processing to the final layers. This enabled them to restore the apparent drop in representational quality in later layers and gave large performance improvements. It is thought that anisotropy, known as the narrow cone of embeddings in the vector space, may be to blame, which in [240] was found to increase from the earlier layers to the later layers. However, successful post-processing reveals that the last layers also possess relevant knowledge, but it is somewhat obscured in the raw outputs.

The question is how to compose a universal representation from the relevant pieces in multiple layers. One way is to try various combinations. The authors of [99] analyzed all possible two-layer combinations for BERT. They found that combining the top and bottom layers is better than using only the top layers. They tried to expand it to more layers, but stayed in a combination of three layers ($L1 + L2 + L12$). From the older works, one can mention the AdaSent model [245]. It operates on a hierarchy of representations derived from a pyramid-like multilayer network structure. The gating network is trained to adaptively select the most appropriate representation in the hierarchy for the given task. The authors of [246] proposed a similar gating system to dynamically decide which intermediate Transformer layers to use.

The contribution of each layer can also be accounted for by weighted average. The authors of ELMo [7], a contextualized model based on LSTM, present this configuration with additional weight parameters for each layer. Other authors investigating representations [183, 173, 171] call this weighting a scalar mix technique. The authors of [216], instead of learning layer weights for each word, compute them by using complex alignment and novelty measures. For a few-layer model like ELMo, other authors of [247] found it useful to concatenate representations across layers while averaging across tokens, rather than using only the top layer. Other authors of [123] extract relevant information from intermediate hidden representations of BERT by treating them as positive samples during contrastive learning and determining which final sentence embedded in the tuned model should be close.

In conclusion, there is no generally accepted method for taking advantage of representations across all layers in an unsupervised setting. In the other supervised case, the whole network is fine-tuned, and the whole hierarchy of representations is taught to surface representations according to the task at hand.

3.1.1.7 Other means of composition

Many other composition functions can be used to derive sentence embeddings. In an early work [189] on phrase composition, the authors also found that multiplicative and dilation models perform well. However, there are caveats; element-wise multiplication of sparse representations results in loss of information (this effect is especially strong

for more than two words), while dilation, just like weighted average, requires additional parameters. It should also be mentioned that max pooling is quite a popular operation to aggregate the output of the LSTM encoder, as shown in [70, 68], and also by the authors of [123], who found it to work best for all BERT layers for the STS benchmark task. According to the authors of [248], max pooling is also competitive with boundary-based methods (such as those that use embeddings of words in the boundaries) for the representation of the text span.

The input text sequence can be combined hierarchically. The authors of [249] used a recursive autoencoder to repeatedly contract two child words to a single parent, to arrive at a final single common parent representation of the whole sequence. Later, the authors of [149] extended this combination to specifically follow the structure of the parse tree. Such a recursive model allowed them to compose word vectors in a bottom-up approach up to a sentence level and outperform the simple averaging method. However, a drawback is that such a composition is limited to only sentences. Other authors of [195] used similar ideas for combinations, but did not learn any composition function. Instead, given a specific window length, the vectors of words in these windows are averaged, and the resulting embeddings are then max-pooled. This setup better preserves spatial information.

The common average operation can be generalized as a power mean, as proposed for information retrieval in [250]. The authors of [251] introduced this idea for combining word vectors. They showed that using the document vector as a concatenation of the power mean with different power values, particularly $p = \pm\infty$ (maximum and minimum), $p = 1$ (regular arithmetic mean), and other $p > 0$, substantially improves the representation of the document. Other authors of [252] also proposed a long vector representation. They employ self-attention to replace the max (or average) pooling from either RNN hidden states or convolved n-grams, resulting in the sentence embedding of the matrix form. However, as the authors of [135] criticize, concatenating representations is problematic. According to them, larger embeddings will always increase the performance of the linear model on top of them. Therefore, embeddings of the same size should always be compared. In light of this, concatenation is an undesirable operation.

3.1.2. Post-processing representations

The word and sentence vectors obtained by older Word2vec [25], GLoVe [8], and current SOTA Transformer techniques are not ideal. Therefore, additional processing techniques are applied to eliminate side effects and improve the performance of downstream tasks. In this subsection, the anisotropy will be described. A phenomenon believed to be responsible for this, the reasons for its appearance, and the techniques that attempt to improve the representation space shall be discussed below.

3.1.2.1 Isotropy

Isotropy is defined as uniformity in all orientations. In this work, the embedding spaces of words and sentences are discussed. According to the authors of [253], “distribution is isotropic when the variance is uniformly distributed across all dimensions”.

The anisotropic properties of independent word embeddings were first observed by the authors of [1]. They analyzed common embedding models and found that, in all cases, word vectors share a nonzero common vector; therefore, they are not zero-centered. The authors also found that the variance ratios of the first few components derived by PCA decay nearly exponentially. Furthermore, the variances explained by the leading components “encode the frequency of the word to a significant degree”. Therefore, they proposed eliminating the common mean vector and then removing the top principal components, computed on representations from the entire vocabulary. A similar procedure was also shown in an earlier work [16] to be a very good baseline. Here, words were combined by a frequency-weighted average, and then just the first principal component (yet dataset-specific, as computed on the entire dataset) was removed. As shown by the All-But-The-Top (ABTT) method of [1], such a simple procedure, eliminating the common mean vector and some top dominating directions from the word vectors, greatly improves both the isotropy and the downstream task performance (for sentence tasks, a simple average of pre-processed word vectors is used). As found by [254], it is already known that the isotropy of the target embedding is critical for the alignment of embeddings, which is important for areas such as domain adaptation, word embedding evaluation, and machine translation. However, the work [1] became the main stimulus for the research direction that attempts to improve the isotropy of word embeddings.

Word representation anisotropy is also observed in the latest contextualized models. The authors of [255] first observed this effect on the Transformer machine translation model [83], while the author of [240] found that this is the case for all ELMo [7], BERT [2], and GPT-2 [256] layers. All representations for randomly sampled words from these contextualized anisotropic spaces were observed to be highly similar, far from a zero average cosine similarity. Furthermore, the embeddings of any two words are positively correlated. Finally, it was observed that all word vectors in the representation space tend to occupy a narrow cone.

The research community offered some explanations for why anisotropy may occur, that will be covered in more detail below.

Word Frequencies The authors of [257] were among the first to observe that word embeddings encode a surprising degree of information about word frequencies. In [16], it was shown that simply representing sentences as a weighted average of words by their frequencies is a very competitive method. In particular, the authors computed the weighted average of the word vectors in the sentence (the weight of a word w is $a/(a + p(w))$, with a being a parameter and $p(w)$ the frequency) and then removed the projections of the average vectors on their first singular vector (“common component

removal”). Later, the authors of [1] suggested removing more than one component. They found that the top PCA directions encode word frequency information; therefore, the method was proposed to null these directions. As this bias is very strong in the representation space, it is inevitably related to the anisotropy phenomena.

Similar encoding of frequency information is also evident in contextualized embeddings of modern Transformer models. The authors of [258] investigated BERT embeddings and found that high-frequency words are all close to the origin and densely concentrated, while low-frequency words are far away and scattered sparsely. The work in [259] highlighted that anisotropy is the most pronounced in rare words. The authors of [260] also showed that the same applies to multilingual BERT embeddings; they have a biased structure towards word frequency.

This bias to word frequency reveals a weakness. The authors of [261] constructed a dataset using semantic relations extracted from WordNet [262] to test the semantic properties of words. By using that, they showed that the ability of BERT to understand words depends highly on their frequency, and therefore rare words are neglected. In [97], it was shown that removing the 34 most frequent tokens before averaging can greatly boost the performance on semantic textual similarity tasks. The authors of [259] conducted a simple experiment on the CNN News corpus with popular WordPiece tokenization and found that 30% of the corpus can be accounted for by using the 13 most frequent tokens, while to cover at least 98% of the corpus, nearly 15,000 tokens are needed. Therefore, rare words are a constituent part of modern NLP pipelines and, unfortunately, they induce deficiencies.

The authors of [255] proposed an explanation for generation models that related observations of the narrow cone form in the representation space and the frequencies of words. They argue that the main culprit is the way models are trained on the language modeling task. During the training process, the ground-truth word embedding will be pushed toward the direction of the hidden state to obtain a higher likelihood. Meanwhile, the embeddings of most words in the vocabulary (with nonappearing and rarely appearing frequencies) will be pushed towards similar directions negatively correlated with most hidden states and thus end up grouped together in the local region of the embedding space. The following work [259] complemented the given explanation with the “common enemies effect” - the effect of the target words producing gradients of the same direction for all of the nontarget words at each step of training with cross-entropy loss, and rare tokens are the most affected by it. The authors also found that the embeddings learned by GPT-2, BERT, and RoBERTa do not degenerate into a narrow cone (they only appear as a cone when projected to a lower-dimensional space) but instead drift in one shared direction. Therefore, the “common enemies effect” is argued to be the main culprit behind the anisotropy of the representation space.

Outliers There is evidence from multiple authors that contextualized representations of Transformer models (in particular, the BERT family) contain undesirable outliers (interestingly, abnormal dimensions were also previously observed for GLoVe vectors

in [263]). These are certain positions in word vectors with unusually high values. The authors of [264] called it outlier dimensions; in [265], it is referred to as outlier neurons, while the authors of [243] call it rogue dimensions. In all of these works, it is agreed that it is a significant contributor to the anisotropy of the representation space.

There is a debate on what causes outliers. The authors of [264] found that outliers are essential for good downstream performance. They regard it as a distinct model property that emerges during training and even encourage one to consider it during weight initialization. In contrast, in [243], it was found that the behavior of the model is not driven by outliers; rather, only a small subset of linguistic abilities is handled there (such as positional information; its relationship with outliers was revealed in [265]). Even more strangely, the authors of [260] found that the embedding space of the multilingual BERT model does not have outliers as the English BERT does, but is still anisotropic.

There are several options to account for outliers. The most obvious solution is to remove them. The authors of [265] showed that this brings improvements for tasks that directly use the geometry of the embeddings, such as semantic textual similarity; however, [264] showed that disabled outliers significantly degrade both Masked Language Modeling loss and downstream task performance. The authors of [243] suggest that the rogue dimensions be accounted for through the standardization transformation. Both options aim to reduce the dominating effect of the highest-value dimensions during the similarity measure, such as cosine similarity or Pearson correlation. On the other hand, one can argue that the similarity measure that is used, and not the representation space, needs improvements.

Criticism Many works try to improve the isotropy of representation spaces, but is that truly the right way to seek performance improvements? Some publications, such as [259], expressed doubts about the role of isotropy in model performance, and the authors of [266] even observed a negative relation between isotropy calibration and downstream performance. The works in [259, 253] ask whether there is truly a “narrow cone” in the embedding space. Other authors of [213] argue that the main problems are various biases that are not related to isotropy. Furthermore, the authors of [253] criticized multiple popular measures of isotropy and warned that they are misleading and inaccurate.

The authors of [267] analyzed the contextual embedding space and found isolated clusters and low-dimensional manifolds. They concluded that, although the embeddings are globally anisotropic, local isotropy exists nevertheless. The authors of [268] continued this observation and applied the known pre-processing techniques of [1], not globally but locally, in clustered regions of representations. They observed that sense-level information is shadowed by structural, syntactic, and tense biases (for verbs), which their method of dominant direction removal helps to reduce. Another work [269] showed that, during the fine-tuning phase, the anisotropy becomes even worse, while, on the contrary, the performance of STS increases. They found that removing

the top dominant directions for fine-tuned representations becomes detrimental, as the most essential information resides there. These findings indicate that the representation space of Transformer models is quite complicated, and that a simple fixation on isotropy can destroy its native features. These also include biases, the usefulness of which may not be obvious to us, but models have found them helpful during the pre-training. In the end, brute-forcing the plain isotropy may harm some intricate semantic details, which are better to be captured.

3.1.2.2 Post-processing methods

Many post-processing methods have been proposed to improve text embeddings. Whether they improve the isotropy of the representation space or help to surface relevant information [270], multiple authors show the benefits of using them. However, these methods generally favor unsupervised tasks, such as semantic textual similarity, with limited improvements for downstream tasks. If enough data on the target task are available, supervised training will inevitably be a better choice as the transformation relevant to the task is learned. However, post-processing may be the only option if the training samples are obscure.

The simplest is the centering operation. It is especially important for the STS task that involves cosine similarities. The authors of [259] found that it restores a nearly perfectly isotropic distribution. Subtracting the mean is also the first step in several other post-processing methods. In some sense, such operations as centering can be viewed as fine-tuning to the target domain, as all target embeddings participate in providing the mean coordinates.

Some operations related to isotropy enhancement were already mentioned in the previous section, such as zeroing target outliers or ABTT. In the following, the remaining important ones will be mentioned.

z-Score Normalization The z-score describes the position of a point in terms of its distance from the mean when measured in units of standard deviation (in the distribution of the target samples). z-score normalization, also called standardization, transforms the distribution of embeddings into a zero mean and a unit standard deviation in all dimensions. The works in [243, 244] recommend that one should consider it as a post-processing step. The authors of [135] advise using normalization, such as the z-score, in supervised settings as a binary hyperparameter, because it can lead to rank changes.

All-But-The-Top Many works are based on the influential ABTT algorithm [1]. It has a hyperparameter that tells the number of dominant directions to be removed. Principal components are either completely removed or left intact. As a result, the main weakness is finding the right number of top components to remove, which can cause either information loss or eliminate an insufficient amount of noise. The authors of [271] proposed post-processing through variance normalization (PVN) to normalize the variance of leading principal components to the same level instead of total compo-

nent disposal. The authors of [272] performed the removal in a more relaxed way by employing matrix conceptors [273] in an unsupervised way. A similar technique that employs conceptors can also target specific components, such as those that incorporate social biases, and diminish them, as in [274]. Other authors of [275] proposed learning weights for each dominant direction removal. All these approaches try to preserve useful information that resides in the main components while narrowing the target noise. The authors of [276] combined ABTT and dimensionality reduction. They found that the best pipeline is to perform ABTT twice, while performing dimensionality reduction in between. In another line of work, the authors of [268, 260], instead of global post-processing, remove local dominant directions in separate clusters in the representation space. In this way, the structure of the representation space is taken into account. Similarly, the authors of [215], in light of the predecessor method of ABTT [16], proposed to eliminate the sentence-dependent principal component, where they rerank the top principal vectors based on the correlation with each sentence. This individual removal of dominant directions was shown to improve the performance on the Semantic Textual Similarity Benchmark (STS-B) task.

Whitening The authors of [98] proposed the whitening approach to alleviate the problem of sentence embedding anisotropy. The whitening operation involves centering embeddings at the origin and making different dimensions have a unit variance and be uncorrelated, turning their covariance matrix into the identity matrix. It was also found to be useful in earlier work for the alignment of bilingual embeddings [277]. As a side effect, the authors of [98] showed that whitening can also be used with a dimension reduction operation. A concurrent work [99] also used a whitening algorithm to improve performance in STS tasks. In that work, the authors combined the first and last layers of the BERT embeddings and then normalized them with whitening. Despite the initial success, subsequent work criticized the whitening operation. The authors of [17] showed that whitened representations greatly improve uniformity, but also suffer degeneration in the alignment property (for alignment and uniformity properties, see [278]). To make matters worse, the authors of [136] called whitening a ‘trick’ that only helps with similarity tasks (by partially overfitting) and harms downstream task performance.

3.1.2.3 Retrofitting

This word vector space specialization method can incorporate semantic knowledge from external resources into word embeddings. The authors of [279] were the first to use the term ‘retrofitting’ for their post-processing step of vector space word representations. Their idea was to incorporate rich relational information from semantic lexicons to word vectors, which are trained in a data-driven fashion using plain texts. The authors showed that the new vectors are similar in both their purely distributional representations and related word types. In addition, they showed improvements in several benchmark tasks.

Retrofitting can be useful for fitting a desired domain that is different from the original one used to pre-train the word vectors. An example is [280] where the authors retrofitted to linkage information in biomedical taxonomy. Another related field is the alignment of cross-lingual word embeddings. The authors of [281] used retrofitting to align the vectors of the source and target languages in the dictionaries. In this way, translation pairs are pulled closer while minimizing deviation from the original embeddings and preserving the existing representation. In all these cases, the retrofitting is based on external relational data.

According to the authors of [282], the weakness of retrofitting is that only the vectors of words that are present in the external constraints (resources) are modified. Therefore, they proposed an explicit retrofitting model in which external knowledge relations are turned into supervised training examples. Here, the distance between synonyms is supposed to be as low as possible and between antonyms as high as possible, while between remaining words not present in the external knowledge base, it should remain the same as in the original representation space. In this way, in addition to the word vectors, a deep feedforward neural network is trained to retrofit all the word embeddings.

Some work also found success in applying retrofitting to contextualized representations. The authors of [247] observed that, for ELMo representations [7], the distance between the shared word in the paraphrases is even greater than the distance between ‘large’ and ‘small’ in random contexts. Therefore, to improve the representation capabilities, they minimized the difference in contextualized representations of the shared word in paraphrased contexts while differentiating between those in other contexts. This retrofit resulted in an improved performance of downstream applications. Other authors of [283] proposed a two-step process: first, to train static vectors from contextualized ones, and then to perform retrofitting. They showed that, compared to baselines, such an approach gives the best results in a range of intrinsic and extrinsic tasks.

3.1.2.4 Other methods

Some methods are proposed to reduce anisotropy during the training process. In particular, the authors of [255] added the specific regularization. According to them, the aperture of the narrow cone of representations can be improved by minimizing the cosine similarities between any two word embeddings. This regularization encourages the vectors to be more evenly spread and expressive. The method is highly related to the now-popular contrastive learning approach. Other authors of [284] improve the isotropy of the output embedding matrix by using the spectrum control method. They guide the singular value distribution of the embedding matrix throughout the training process and control the decay rate of these singular values.

The authors of [258] suggest a learning transformation of the embedding space of the Transformer, sometimes called a flow technique. Their method transforms the BERT sentence embedding distribution into a smooth and isotropic Gaussian distribution. During unsupervised training, only the flow network is optimized, while the BERT parameters remain unchanged. Although BERT-flow showed performance improve-

ments on multiple tasks, the technique is criticized in the literature. First, it needs a specialized implementation and has multiple additional parameters. The authors of [266] report that BERT-flow requires on average 4.2 times more time per training epoch. Furthermore, for the STS benchmark task, the authors of [136] report that BERT-flow with the l_2 similarity metric performs even worse than the baseline of the BERT average. Different post-processing can surface the different information residing in the embeddings. The authors of [270] proposed a simple unsupervised singular value decomposition to reassign feature weights. By changing the similarity order of their transformation, they tailored word embeddings in the semantics/syntax (tasks focusing on sing–chant or sing–singing) and similarity/relatedness (tasks solving car–automobile or car–road relationships) axes. Other authors of [285] used spectral filters to dissect BERT representations at different temporal scales. They showed that a low-pass filter yields the highest probing accuracy in topic classification, a high-pass filter in speech tagging, and a middle-pass filter is best in dialog speech tasks.

3.1.2.5 Similarity measures

Some of the most basic tasks for sentences require measuring the distances between the corresponding embeddings. Usually, cosine similarity is employed. However, if the continuous representation space is curved or a text sequence is treated as a set of words, different metrics or representation space manipulations can be beneficial. The authors of [286] proposed MI, which is well-known in information theory and statistics, as a candidate for a similarity measure. They managed to successfully estimate MI for continuous random variables by the use of Kraskov–Stögbauer–Grassberger (KSG) estimator [287], which is based on elementary nearest-neighbor statistics. The other authors of [288] advised comparing sentence embeddings consisting of word vectors averaged by rank correlation, such as Kendall’s τ . They argued that such measures enable mean pooled representations to rival modern deep ones, used with cosine similarity. The authors of [289] proposed treating sentences as fuzzy sets of words and showed good performance with the specially adapted DynaMax similarity measure. They showed that word vectors alone are sufficient to achieve excellent performance on semantic textual similarity tasks when sentence embeddings and similarity measures are derived by using ideas from the fuzzy set theory. Other authors of [290] suggest processing each sentence with respect to its found k -nearest neighbors. Then, they find an optimal Euclidean subspace of the sentence manifold where cosine similarity would work best. The authors of [291] also project the sentences onto a fixed-dimensional manifold with the objective of preserving local neighborhoods in the original space. All these works mentioned employ various strategies to enhance the comparison of document representations, tailoring their approaches to the given task.

Many similarity measures between sets employing earth mover’s distance were proposed. The authors of [23] were the first to frame the similarity of documents as a transportation problem. The idea is that the distances between similar but different words in two documents should be small. Their Word Mover’s Distance (WMD) is the

cost of transporting a set of word vectors in an embedding space. This approach was effective, as it managed to exploit similarities between different Word2vec word vectors, such as the relation of analogies. However, the main deficiency of the WMD was that the distances were expensive to compute. Moreover, an output, the single number of a distance between two given documents, can only be combined with k -nearest neighbors or k -means, while applications usually require a whole feature vector. The authors of a subsequent work [292] managed to derive vectors for documents using WMD. They constructed a positive-definite word mover's kernel by using a feature map given by the WMD to random documents and then derived document embedding via a Random Features approximation of the kernel. Other authors of [26] proposed the Word Rotator's Distance (WRD). It is designed so that the norm and angle of word vectors correspond to the probability mass and transportation cost in the earth mover's distance, respectively. Therefore, the norm of vectors, which is associated with the vector's significance, does not interfere with the calculation of the transportation cost, as in the WMD. Finally, the authors of [293] attempted to include structural information which is absent from the WMD distance estimation between two sets of words. They represented the sentence vector as a weighted average of substructure vectors at a lower level in a recursive way, while a transport plan at a different level explains how the different substructures align.

3.1.3. Summary of the review

Many methods for token aggregation have been reviewed. The re-iterating factor here is the importance of the token frequency. The more frequent tokens worsen the whole sequence representation. Although there are many ways of weighting it, in the experiments, the simple inverse document frequency and top frequent word filtering is considered. It was also found that characteristic of the Transformer is the prompt-based token extraction, which will also be considered.

Many works have shown that there is room for improvement in representation. The main reason is thought to be the anisotropy of the representation space. Therefore, it was decided to measure the isotropy of the resulting sentence embeddings. Regarding post-processing techniques, it was decided to try out all the technologies implemented in the standard machine learning library of Scikit-learn [294] together with the influential ABTT method.

3.2. Methodology

3.2.1. Problem formulation

Given a text sequence s_0 , which was tokenized into N individual pieces (i.e., tokens)

$$t_1 \ t_2 \ \dots \ t_N \ , \quad (12)$$

and a model representing each token as a d -length vector $v \in \mathbb{R}^d$ in each of its L layers. The static token embeddings denoted as the -1st layer and input embeddings denoted

as the 0th layer were also used. They are sums of token, segment, and position vectors, with layer normalization applied on top. Therefore, the following 3-dimensional representation space $\mathbb{R}^{(L+2) \times N \times d}$ is obtained:

$$\begin{pmatrix} v_1^{-1} & v_2^{-1} & \dots & v_N^{-1} \\ v_1^0 & v_2^0 & \dots & v_N^0 \\ v_1^1 & v_2^1 & \dots & v_N^1 \\ \vdots & \vdots & \ddots & \vdots \\ v_1^L & v_2^L & \dots & v_N^L \end{pmatrix}. \quad (13)$$

It is needed to find such a function f that the $((L+2) \times N \times d)$ -dimensional output from a chosen Transformer model would be aggregated to as meaningful a d -dimensional vector as possible in $\mathbb{R}^d$.

As a baseline for the bert-base-uncased encoder type model with $L = 12$ layers, the average over the N tokens in text and over the first and last layers was considered.

$$f_{\text{first+last}}(s_0) = \sum_{n=1}^N \frac{v_n^1 + v_n^{12}}{2N}. \quad (14)$$

Generally, a corpus has S number of text samples. Therefore, other text sequences can also be used to derive the representation of the target sentence. In this way, the aggregation function f operates on an $(S \times (L+2) \times N \times d)$ -dimensional output from the model. S can also be enlarged by using additional datasets. In particular, the Wikitext-2 from [295] were used. This can be used for various post-processing techniques such as centering, standardization, PCA, and others, where transformations are learned on corpora other than the target corpus.

3.2.2. Transformer model

Text-level representations of the popular BERT Transformer model will be improved. When first introduced in [2], it spectacularly outperformed multiple other models on a wide range of tasks. It is based on the original Transformer architecture of [83]. The main difference is that, instead of sequence-to-sequence workings and the encoder-decoder structure, it is composed only of an encoder side. This allows it to solve multiple classification and regression tasks, maintaining the same pre-trained base while only changing the classifier heads on top.

In this work, the BERT version called bert-base-uncased, a 110 M parameter model containing 12 layers (blocks) and working with lowercase text, was used. The Hugging Face implementation [296] of BERT was employed. It was pre-trained on BooksCorpus (800 M words) [297] and English Wikipedia (2500 M words) datasets, which take 13 GB of combined plain text. The model was trained for 40 epochs (or passes through the corpus). Initially, such a training time, model size and data used were considered very large, yet now they are only a small fraction of what current SOTA models use

[84].

Input to the BERT encoder consists of token-, position-, and token-type embeddings. BERT uses WordPiece tokenization [34]. Splitting less frequent words into subwords (e.g., “transformer” → “transform” and “#er”) rather than splitting everything by word boundaries allows one to reach a manageable vocabulary size of 30,000 tokens. BERT, like other Transformer models, perceives input as a set; therefore, order information must be supplied in addition. To achieve this, additional positional embeddings are used, with a unique value for each position in a token sequence and feature dimension. Token-type embeddings allow the model to distinguish between the two (if there are two) separate sentences (e.g., ⟨ Question, Answer ⟩) in one token sequence. In general, token, positional, and token-type embeddings are added, and then layer normalization [298] and dropout [108] are applied. This results in the input to the BERT model blocks (for simplicity, referred here as layers).

BERT tokenization includes 3 special tokens. [CLS] starts every token sequence, [SEP] ends token sequences and acts as a separator in a pair of sentences, and [MASK] is used during pre-training to mask some percentage of the input tokens at random for the model to predict (this is known as the “Cloze” task [299]). The [CLS] token is also used for the next sentence prediction task – predicting whether the second sentence in a pair actually follows the first one in the training dataset or is a random one – the second unsupervised pre-training task of BERT.

3.2.3. Aggregation methods

Every text contains multiple tokens, each with a corresponding embedding vector. To obtain one vector for the whole text out of the many, usually a simple average is calculated. In fact, it is used as a baseline for multiple methods.

Different methods of token weighting and filtering based on their frequencies were investigated. In classical BoW approaches, it is usually accounted for with the TF-IDF weighting. However, only IDF weighting was used because the same tokens in BERT cannot be simply counted due to different contextualization. Given a dataset with N documents and document frequency df_t , defined to be the number of documents in the given dataset that contain a token t , idf_t for a token t was calculated as

$$idf_t = \log \frac{N}{df_t}. \quad (15)$$

To account for long/short text differences, the idf_t token weights are scaled for each text so that they would sum up to 1. In this way, to calculate the average embedding for a given text, one needs only to sum up the weighted vectors. Two inverse document frequencies were calculated: idf_t^W for the Wikitext-2 dataset from [295] and idf_t^T for all samples from the given target task.

The method in [97, 213] to remove the most frequent tokens was also adopted. 33 tokens were chosen, which are depicted in the Appendix of [213]. Following that article, dropping punctuation and subword tokens was also investigated. It was found

that dropping all three parts – the most frequent, punctuation, and subword tokens – has the best effect, and, following the original work, such token aggregation as with removed biases (“-biases”) is named.

3.2.3.1 Prompting

The classical use of the BERT model involves two steps: 1) a general pre-training on a very large corpus with unsupervised tasks; and 2) a supervised fine-tuning step on a small target task dataset. The first step is expensive, it requires a lot of data, time, and computing resources, but it is carried out only once. Weights that are produced in an unsupervised way contain a lot of useful representations for fine-tuning to only perfect them.

Prompt-based methods allow to leverage the pre-training step without bothering with the fine-tuning stage. The idea is to frame the target task in the original pre-training format, for which, the model is essentially optimized. To extract a general sentence representation, the authors of [213] proposed using “This sentence: “[X]” means [MASK]” template (which here will be called T0), where the target sentence is placed instead of [X] and the final layer representation of [MASK] is used. In this way, the model itself tries to predict the meaning of a given sentence, so the prediction weights can be extracted before they are turned into probabilities over tokens.

During the initial experiments, the manual search for other templates, more complicated than T0, was done, as presented in Table 1 (for more detailed search, see Table 23 in Appendix 1). From these, the T4 template was chosen, which is much longer and has 3 [MASK] tokens that have to be averaged.

Table 1. Manual template search by adding additional text and [MASK] ([M]) tokens. Target sentences are inserted into the place of [X]. Only the average of 12th layer [M] representations are used. Bolded results are the best.

No.	Template	STS	Clust.	Class.
T0	This sentence: “[X]” means [M].	63.4	45.5	78.7
T1	This sentence: “[X]” means [M][M].	63.4	46.6	78.6
T2	This sentence: “[X]” means “[M][M]” and is about [M].	70.4	52.8	78.3
T3	This sentence from the paraphrase dictionary: “[X]” means “[M]”, which is about [M].	69.6	54.2	77.4
T4	This sentence from the dictionary: “[X]” means “[M]” and is about [M], which is a synonym for [M].	69.3	54.2	76.8

3.2.4. Post-processing methods

It is common in machine learning to standardize datasets, as most methods are designed to work best with normally distributed data: Gaussian with zero mean and unit variance. Yet, it is not that trivial, as data may not follow a smooth distribution and

may contain disturbing outliers. In the context of the best practices reviewed in Section 3.1.2, the following embedding processing methods were investigated.

3.2.4.1 z-score normalization

The most basic is the standard score, also called the z-score. It is the number of standard deviations σ by which the value x of a raw score is higher than or below the mean value μ of the raw scores of all samples. Vectors are normalized to have z-score = 1:

$$z = \frac{x - \mu}{\sigma}. \quad (16)$$

3.2.4.2 Quantile normalization to uniform distribution (quantile-u)

This technique works by making two distributions identical in statistical properties, thus reshaping given data values according to some known distribution function. During experiments, it was found that the uniform distribution worked very well as a reference. For a more detailed description of the technique, see [300].

It was also tried to use other methods that are more robust to outliers; however, it was found that their performance is marginally below quantile-u. This includes quantile normalization using a normal distribution and RobustScaler (as it is called in the scikit-learn preprocessing library [294], which was used). The latter method removes the median value instead of the mean, and scales the data according to the selected quantile range.

3.2.4.3 Whitening

It is a transformation that produces uncorrelated components, each with a variance of 1.

3.2.4.4 All-But-The-Top (ABTT)

It is a method introduced in [1]. It starts with the embedding matrix given $A \in \mathbb{R}^{b \times f}$ of b sentences, each with f features. First, it is centered on its mean $\mu \in \mathbb{R}^f$ in $\tilde{A} \in \mathbb{R}^{b \times f}$. By using the centered $\tilde{A}$, and given the number d of the top principal components to remove, the PCA components $U \in \mathbb{R}^{d \times f}$ are calculated. Now, data are projected into these components to acquire $A^{PCA} \in \mathbb{R}^{b \times f}$:

$$A_{bf}^{PCA} = \sum_d \tilde{A}_{bf} U_{df}. \quad (17)$$

The final processed embedding matrix will be:

$$A' = \tilde{A} - A^{PCA}. \quad (18)$$

The only difference of this approach from the original authors of [1] is that, instead of post-processing words, the ABTT is used to post-process documents.

3.2.4.5 Normalization

It was also experimented with normalization: scaling individual sentence vectors to have a unit norm.

3.2.4.6 Learning post-processing

Some target task datasets can also be too scarce to calculate accurate statistics, such as mean, standard deviation, or others, used in post-processing. Therefore, instead, learning these weights on the Wikitext-2 dataset was experimented on. Such techniques are indicated with a superscript ^{W} .

3.2.5. Static models

It is also important to see how improvements in contextualized representations of the BERT model transfer to static ones. The baselines considered are described in detail in the following sections.

3.2.5.1 Averaged BERT (Avg.)

The idea of [206] to average the representations of a token in its different contexts to acquire its static embedding is followed. Yet, in this work, it is expanded even further, and vectors for sentences are constructed by again averaging such static token embeddings over the sentences.

Static tokens were constructed by using the Wikitext-103 [295] dataset and use the same tokenization as the BERT version `bert-base-uncased`. For each of all 28,807 tokens occurring in Wikitext-103, all vectors produced by the BERT model (which differ due to different contexts) were summed and divided by their count. Note that some tokens occur very often, while others are rare (for example, the most frequent, ‘the’, is repeated 6,470,356 times). The process is repeated for every BERT layer as well as combinations of layers, so that the performance dependence on this factor could be observed as well.

Combining Averaged BERT and Regular BERT (BERT + Avg.) It is also important to see how sentence embeddings derived from static averaged BERT tokens can contribute to the original BERT representations. Therefore, sentence embeddings from the two methods mentioned above were averaged.

3.2.5.2 BERT2Static (B2S, B2S-100)

There are more advanced distillation methods than simple BERT token averaging. For example, the authors of [202] trained static word vectors by using BERT’s contextualized representations. They adapted the Sent2Vec [55] model for words and trained it to predict the word given the context element of it by the contextual representation of the BERT model. In this way, the authors obtained vectors for the 750,000 most frequent words. They also showed that this approach results in vectors that perform better than

existing static embeddings trained from scratch, while still enjoying a small memory and computational footprint.

In experiments, the `bert_12layer_para` model, provided in [202], was used, which was trained from the contexts of a paragraph rather than only a sentence. Two versions of BERT2Static were evaluated: a regular one (B2S), and one with the 100 most frequent words filtered out (B2S-100).

3.2.5.3 Random embeddings (RE)

Instead of using a complex model to derive embeddings for words, random vectors as token embeddings were also tested. More specifically, each token was assigned a random vector drawn from the normal (Gaussian) distribution, centered at 0 with 0.1 standard deviation. To facilitate better comparisons with the BERT model, the same tokenizer from the BERT version `bert-base-uncased` is used, and the vectors are made the same 768-size length. The whole sentence representation is then computed as the average of its constituent token vectors.

To further explore RE capabilities, a separate experiment with different tokenizers and vector sizes was constructed. Better aggregation or post-processing methods were not used here, only plain averaging without post-processing. Overall, BERT [2], ByT5[3], Llama [301], mT5 [11] and T5 [22] tokenizers were tested. A word-level baseline that splits text into words by spaces was also included. It was implemented by lower-casing the text and then splitting it by white spaces. In Python, it is simply achieved by using `str.lower().split()`, therefore, this tokenization is named here as ‘lowersplit’. An example tokenization of each tokenizer is shown in Table 2.

Table 2. Properties of tokenizers used for RE model experiment. $\|V\|$ is vocabulary size. Example sentence from STS12 task is “a woman mixes up vegetables .”

Tokenizer	$\ V\ $	Tokenization example
BERT [2]	30 522	[a] [woman] [mixes] [up] [vegetables] [.]
T5 [22]	32 100	[] [a] [_woman] [_mixes] [_up] [_vegetables] [] [.]
mT5 [11]	250 100	[] [a] [_woman] [_mix] [es] [_up] [] [vegetable] [s] [] [.]
ByT5 [3]	384	a woman mixes up vegetables .
Llama [301]	32 000	[_a] [_woman] [_mix] [es] [_up] [_veget] [ables] [_.]
lowersplit	∞	[a] [woman] [mixes] [up] [vegetables] [.]

3.2.6. Text representation evaluation

The investigated methods were evaluated on multiple clustering, semantic textual similarity, and classification tasks.

3.2.6.1 Semantic textual similarity

STS is the most popular metric for the evaluation of sentence representation. The STS evaluation settings from [17] are followed. First, the evaluation is kept unsupervised by not applying any additional regressors; cosine similarity between embeddings in a pair is taken directly as a model score for similarity. To find the degree of correlation between the annotated labels and the models given, Spearman’s rank correlation is used because it measures the rankings instead of the actual scores. Spearman’s correlation ranges from -1 to 1 but in the STS literature it is multiplied by 100 and reported as percentages, due to better readability and the fact that even basic models never yield negative values. Finally, for annual STS challenges, all subsets are concatenated, and the general Spearman correlation is reported for that year (referred to in [17] as ‘all’). As the SentEval toolkit [103] was used, it had to be concatenation option added manually, since only ‘mean’ and ‘wmean’ settings were available. Similarly to [136], the STR [302], a recent semantic relatedness dataset created by comparative annotations, was also added. The statistics of STS datasets used are shown in Table 3.

Table 3. Dataset statistics for STS tasks. N is the number of text samples, L/S is the imbalance number, defined as the size of the largest class divided by that of the smallest class, $\|V\|$ is the vocabulary size, Len is the average number of tokens in each text sample, α is the percent of tokens that are alphabetic. Statistics with plain word tokenization are marked with “W” and with bert-base-uncased model tokenizer as “B”.

Dataset	N	L/S	$\ V\ $		Len		α (%)	
			W	B	W	B	W	B
STS12	3108	5.2	8127	7802	25	28	83	77
STS13	1500	0.7	5152	5141	20	21	88	82
STS14	3750	1.6	9117	8613	21	23	86	80
STS15	3000	0.9	7364	7185	23	24	89	85
STS16	1186	1.2	3971	4175	26	28	87	83
STR	5500	1.0	22,392	12,883	25	32	83	76
SICK-R	4927	3.9/3.9	2271	2291	19	20	99	96
STS-B	1379	1.1	4888	4921	22	24	85	81

3.2.6.2 Transfer tasks

As has already been discussed in Section 2.6, evaluation on STS alone is not enough, as it is easily influenced by post-processing and has a weak correlation with classification tasks. It was noticed that papers submitted to more prestigious conferences always include an additional evaluation on downstream classification tasks. Unlike STS, these tasks are evaluated in a supervised way, given the frozen sentence embeddings. Following the SentEval [103] benchmark suite, the commonly used evaluation protocol is to train a logistic regression or an MLP classifier with a cross-validation

setup on top of the frozen representations, and the testing accuracy is used as a measure of the representation quality. The logistic regression classifier and the 10-fold cross-validation scheme was used. This setting is the most commonly reported in the literature. Detailed statistics of transfer tasks are presented in Table 4.

Table 4. Dataset statistics for classification tasks. N is the number of text samples, C is the number of clusters, L/S is the imbalance number, defined as the size of the largest class divided by that of the smallest class, $\|V\|$ is the vocabulary size, Len is the average number of tokens in each text sample, α is the percent of tokens that are alphabetic. Statistics with plain word tokenization are marked with “W” and with bert-base-uncased model tokenizer as “B”.

Dataset	Split	N	C	L/S	$\ V\ $		Len		α (%)	
					W	B	W	B	W	B
Binary classification										
MR		10,662	2	1.0	20,325	13,802	22	26	84	76
CR		3775	2	1.8	5675	5221	20	22	84	80
SUBJ		10,000	2	1.0	22,636	15,912	25	28	85	78
MPQA		10,606	2	2.2	6239	6248	3	3	97	88
SST2	train	67,349	2	1.3	14,816	11,570	9	11	88	78
	dev	872	2	1.0	4339	4542	20	23	85	76
	test	1821	2	1.0	7053	6824	19	23	85	76
MRPC	train	4076	2	2.1	16,112	12,061	44	50	81	75
	test	1725	2	2.0	10,092	8471	44	50	82	75
Fine-grained classification										
SST5	train	8544	5	2.1	16,579	12,395	19	23	84	75
	dev	1101	5	2.1	5038	5168	19	23	85	76
	test	2210	5	2.3	7929	7478	19	23	85	76
TREC	train	5452	6	14.5	9437	8492	10	11	87	81
	test	500	6	15.3	1100	1247	7	8	85	79
SCICITE	train	7320	3	4.4	27,775	13,603	31	40	100	77
	dev	916	3	4.4	7625	6918	31	40	100	78
	test	1861	3	3.8	12,609	9217	31	41	100	77
SICK-E/R	train	4500	3	3.8/3.6	2258	2277	19	20	99	96
	dev	500	3	3.8/5.0	1122	1172	20	20	99	96
	test	4927	3	3.9/3.9	2271	2291	19	20	99	96
STS-B	train	5749	...	1.3	12,430	10,792	22	25	86	80
	dev	1500	...	0.7	6542	6511	26	28	85	80
	test	1379	...	1.1	4888	4921	22	24	85	81

The following tasks were used:

- MR: sentiment prediction for movie reviews on a five-star scale [303].

- CR: sentiment prediction of customer product reviews [304].
- SUBJ: binary subjectivity status for sentences from movie reviews and plot summaries [303].
- MPQA: phrase-level opinion polarity classification from news articles [305, 306].
- MRPC: Microsoft Research Paraphrase Corpus [307] of parallel news sources for the paraphrase detection task.
- SCICITE: domain-specific classification task that assigns one of three intent labels (“background information”, “method”, “result comparison”) to sentences collected from scientific articles citing other articles [308].
- SICK-E: ternary classification of sentence pairs labeled as “entailed”, “contradiction”, or “neutral” [141].
- SST5: Stanford Sentiment Treebank with 5-level sentiments [149].
- TREC: fine-grained 6 types of questions classification [309].

Regular semantic textual relatedness and similarity tasks SICK-R [141] and STS-B [21] were also evaluated with classification, by splitting the real-valued similarity targets into 5 discrete class labels. For example, a [0,5] score of 3.6 goes to class 3 with weight 0.4 and also to class 4 with weight 0.6.

3.2.6.3 Clustering

Performance on six benchmark datasets for short text clustering was evaluated. Although these are not as popular as STS or Transfer tasks from the SentEval benchmark [103], multiple works [185, 310, 311, 312, 313, 314] were found employing them for sentence representation evaluation. Compared to the usual ones, short datasets pose a challenge due to the weak signal caused by sparsity, which is a big problem for classic count-based approaches such as BoW or TF-IDF. Clustering was performed by running *k*-means with the scikit-learn package [294] and reported the clustering accuracy, calculated using the Hungarian algorithm [315] and averaged over 10 independent runs. A publicly available codebase [316] and clustering datasets [317] were used. Table 5 provides an overview of the main statistics and the details of each dataset are as follows:

- Agnews: a subset of news titles [318], which contains 4 topics selected by the authors of [314].
- Biomedical: a subset of PubMed data distributed by BioASQ [319], where the authors of [320] randomly select 20,000 paper titles from 20 groups.
- GoogleTS: contains titles and snippets of 11,109 news articles related to 152 events [321]. The full version of the dataset was used, which includes both titles and snippets, named GoogleNews-TS in [314].

- Searchsnippets: 12,340 snippets extracted from the web search, associated with 8 groups [322].
- Stackoverflow: a subset of challenge data published by Kaggle [323], where the authors of [320] selected 20,000 question titles associated with 20 different categories.
- Tweet: a total of 2472 tweets with 89 categories [321].

Table 5. Dataset statistics for the short text clustering datasets. N is the number of text samples, C is the number of clusters, L/S is the imbalance number defined as the size of the largest class divided by that of the smallest class, $\|V\|$ is the vocabulary size, Len is the average number of tokens in each text sample, Alpha is the percent of tokens that are alphabetic (all token characters must be defined in the Unicode character database as “Letter”, while tokens with numbers, punctuation, or BERT continuation tokens such as “##ing” are excluded). Statistics with plain word tokenization are marked with “W” and with bert-base-uncased model tokenizer as “B”.

Dataset	N	C	L/S	$\ V\ $		Len		α (%)	
				W	B	W	B	W	B
agnews	8000	4	1	21,062	16,140	23	26	100	86
biomedical	20,000	20	1	18,888	9326	13	20	98	64
googleTS	11,109	152	143	19,508	14,763	28	33	100	85
searchsnippets	12,340	8	7	30,643	16,334	19	24	93	77
stackoverflow	20,000	20	1	22,909	7332	8	12	87	71
tweet	2472	89	249	5098	5091	9	11	100	81

3.2.6.4 Other metrics

In addition to task oriented metrics, it is also important to quantify how well or effective representations are distributed. For this purpose, the IsoScore, Alignment and Uniformity metrics were used.

The IsoScore [253] metric was used to measure the uniformity of the utilization of embedded space. As shown by the authors of [253], IsoScore has stronger properties than other isotropy measures. For each sentence embedding method, the IsoScore for the Wikitext-2 dataset from [295] was calculated. The dataset was split into individual sentences (now samples) and omitted texts with less than 10 characters.

In explaining the success of contrastive methods, the authors of [278] proposed using alignment and uniformity properties to better quantify the quality of representations. Alignment is calculated between semantically related positive pairs and therefore is an

expected distance between embeddings of the paired instances:

$$\mathcal{L}_{\text{align}}(f) = \mathbb{E}_{(x,y) \sim p_{\text{pos}}} [\|f(x) - f(y)\|^2]. \quad (19)$$

Uniformity is computed by using representations of the whole space:

$$\mathcal{L}_{\text{uniform}}(f) = \log \mathbb{E}_{(x,y) \stackrel{\text{i.i.d.}}{\sim} p_{\text{data}}} \left[e^{-2\|f(x) - f(y)\|^2} \right]. \quad (20)$$

Smaller values of both alignment and uniformity indicate better quality of the representations. As a distribution of positive pairs p_{pos} , STS-B task training split pairs with a similarity of 5.0 was used, and for p_{data} , sentences from all pairs were employed.

3.3. Experiments and Results

Here, the results of experiments are presented.

3.3.1. Token aggregation and post-processing techniques

In Table 6, the results of the best-performing token aggregation and embedding post-processing techniques are presented. The results here are averaged for all the different tasks in the class.

It is seen that both the aggregation and post-processing methods have a great impact on STS and clustering performance. For semantic textual similarity tasks, the best model was improved from 62.3 % to 71.6 % average Spearman correlation, while for clustering, the best model was improved from 59.2 % to 64.8 % average accuracy.

These techniques can even improve the dedicated SimCSE model [17], which was fine-tuned with NLI data. Its main strength lies in semantic textual similarity tasks, where it leads with over 10 percentage points difference. However, for clustering tasks, its average accuracy is similar to the other evaluated models at 59.8 % and improves up to 64.0 %, if the best-performing techniques are applied. This showcases a general tendency that the top-performing models are very good only in a narrow subset of tasks and highlights the importance of these more general methods.

For maximum performance, usually, both aggregation and post-processing techniques must be used. As an example, sentences placed within the T4 template with no techniques applied have a 62.3 % average Spearman correlation. With a better aggregation method of idf_t^T , it is pushed up to 69.2 %, while only using quantile-u post-processing gives 67.3 % (see Table 7 with only post-processing techniques applied). However, a combined effort of both idf_t^T and quantile-u gives the average Spearman correlation score of 71.6 %.

Still, the improvements from both aggregation and post-processing do not exactly add up linearly. This indicates that improvements in representations of this type may saturate below perfect scores.

Many post-processing techniques for sentence vectors were considered, and the perfor-

Table 6. Effect of different token aggregation and post-processing methods on multiple text embedding models. Both correlation and accuracy scores range from 0 to 100 (from the worst to the best). Results were also added for SimCSE-BERT model [17], which is fine-tuned on NLI data supervision. The best result for each model is bolded, while underlined results are the best across all the models.

Model	T0	T4	B	B + Avg.	Avg.	B2S	B2S-100	RE
Layer	Last		First + Last			–		
Avg. Spearman correlation (%) of STS tasks								
avg.	61.3	62.3	61.3	61.4	56.4	56.7	60.6	50.3
idf_t^W	67.8	67.4	69.2	69.3	66.9	66.1	65.4	66.4
+ z-score	68.6	68.1	69.8	70.6	69.5	66.4	65.9	66.4
idf_t^T	67.4	69.2	68.7	69.0	67.5	56.7	60.6	64.9
+ quantile-u	69.5	<u>71.6</u>	68.9	69.5	68.4	56.2	61.0	63.0
-biases	64.7	66.0	68.2	69.3	67.3	54.1	57.5	63.7
[MASK]	63.4	69.3						
+ quantile-u	65.6	70.5						
SimCSE performance: 81.5								
Avg. accuracy (%) of clustering tasks								
avg.	53.5	55.0	57.0	59.2	55.2	53.8	55.3	36.3
idf_t^W	53.0	53.0	55.2	57.1	53.7	53.0	53.7	44.3
+ normalize	54.3	54.4	58.0	57.9	54.1	55.8	56.1	49.5
idf_t^T	53.1	57.6	60.7	62.5	58.7	53.8	55.3	42.8
+ quantile-u ^W	57.6	60.0	63.1	64.4	60.2	54.6	56.0	44.4
+ normalize	55.5	57.6	63.4	<u>64.8</u>	59.8	55.5	57.2	47.0
-biases	54.3	56.2	61.1	62.2	61.6	52.6	54.1	40.7
+ normalize	54.4	56.3	62.4	63.4	62.2	54.4	56.8	44.6
[MASK]	45.5	54.2						
+ quantile-u	44.4	54.9						
SimCSE performance: 59.8 and 64.0 (avg. + quantile-u ^W , first + last layers)								
Avg. accuracy (%) of classification tasks								
avg.	80.5	79.9	79.9	79.7	77.3	76.9	75.1	69.5
idf_t^W	73.1	67.1	78.3	77.9	75.3	76.8	74.1	67.6
idf_t^T	80.1	79.7	78.9	78.4	76.1	76.9	75.1	68.1
-biases	80.1	79.9	80.1	79.8	77.2	76.3	74.2	70.3
[MASK]	78.7	76.8						
SimCSE performance: 86.5								

mance of the most popular and best performing ones are shown in Table 7 separately. In fact, almost every method somehow improves the average Spearman correlation for STS tasks. For ABTT, the number of components to remove was varied up to a hundred but settled on removing only the first two, as it was slightly better than the rest. Although highly credited in the literature, ABTT-2 still obtained smaller scores than the others. Plainly averaging the token representations, the highest scores for both STS and clustering tasks were achieved by using quantile-uniform normalization. The simple normalization of vectors to the unit length was beneficial for clustering tasks. Whitening normalization, which was learned on the Wikitext-2 dataset, was favorable for the random embeddings model. The other post-processing methods were trained on Wikitext-2 too, yet they resulted in similar or slightly smaller scores.

Table 7. Performance of post-processing with plain token averaging. abtt-2 is the ABTT method with top 2 principal components removed. The best result for each model is bolded, while underlined results are the best across all the models.

Model	T0	T4	B	B + Avg.	Avg.	B2S	B2S-100	RE
Layer	Last		First + Last			–		
Avg. Spearman correlation (%) of STS tasks								
avg.	61.3	62.3	61.3	61.4	56.4	56.7	60.6	50.3
+ z-score	65.4	67.3	65.1	66.5	64.1	58.5	62.1	55.6
+ quantile-u	65.0	67.3	64.3	65.7	63.0	56.2	61.0	53.0
+ quantile-u ^W	62.8	64.6	63.0	64.7	62.1	55.5	60.5	52.2
+ abtt-2	64.7	66.7	64.3	66.3	64.3	58.3	62.0	54.4
+ normalize	61.3	62.3	61.3	61.4	56.4	56.7	60.6	50.3
+ whiten	63.5	64.2	64.7	65.6	65.5	62.1	61.2	63.2
+ whiten ^W	60.9	63.7	63.1	65.8	67.0	61.4	61.0	64.3
Avg. accuracy (%) of clustering tasks								
avg.	53.5	55.0	57.0	59.2	55.2	53.8	55.3	36.3
+ z-score	54.2	55.7	57.8	59.5	56.1	53.3	55.0	36.3
+ quantile-u	54.6	56.3	58.7	60.7	57.3	54.2	55.8	36.8
+ quantile-u ^W	54.2	56.3	60.1	61.1	58.6	54.5	56.0	36.9
+ abtt-2	53.4	54.8	57.0	59.0	55.8	53.6	55.0	36.5
+ normalize	53.4	55.0	59.7	59.7	56.1	55.5	57.2	38.5
+ whiten	35.4	35.7	33.1	29.8	26.8	25.3	26.0	24.8
+ whiten ^W	49.0	49.8	56.2	58.7	55.0	50.4	50.1	41.5

Unlike unsupervised ones, classification tasks do not benefit from the two aggregation and post-processing techniques. There is only negligible improvement for classification tasks when the biases (punctuation, most frequent, and subword tokens) are eliminated. Otherwise, performance only decreases.

The proposed method did not improve the classification. This result is logical, because similarity or clustering tasks are carried out with the resulting text representations di-

rectly, while the classification is a supervised task learned on top. In the first case, the most informative components of the representations must be present in the largest principal components (i.e., constitute most of the variance in the data) for high scores, while the supervised logistic regression classifier can learn to extract them from the small principal components or ill-shaped representations on its own.

3.3.1.1 Avg. versus B2S and B2S-100

An interesting result, as seen in Table 6, is the comparison of simply averaged BERT tokens from many contexts (the Avg. model) and word vectors from a specially trained Word2vec-style model BERT2Static (the B2S model) using the same BERT contexts, as described in Section 3.2.5.2. If token aggregation and post-processing techniques are not used, B2S performs similar in STS tasks (with a mean Spearman correlation of 56.7 % versus 56.4 %), worse at clustering (with an average clustering accuracy of 53.8 % versus 55.2 %), and is comparable in classification tasks (with an average B2S accuracy of 76.9 % versus 77.3 %) to the Avg. model. As it can be seen, the task performance differences are very small.

Apart from training, the second main difference between the Avg. and B2S models is tokenization. The Avg. model uses sub-word tokens, the same as BERT (tokens of which it averages), while B2S models use a vocabulary of the full set of the most frequent words, 20 times the size of the BERT's. Due to this discrepancy, token aggregations for the BERT and B2S models are applied differently, since inverse document frequency statistics for both tokenizations are different; also, B2S does not have the equivalence of biased tokens such as BERT. However, it is very straightforward for B2S to filter out (not to use during averaging) a portion of the most frequent words. During the experiments, this number was varied, and it was found that removing the 100 most frequent words works best, which improved the average Spearman correlation of STS tasks from 56.7 % to 60.6 % and the average accuracy of clustering tasks from 53.8 % to 55.3 % compared to full B2S, both with plain token averaging and without post-processing applied.

Despite the additional option for B2S to remove the 100 most frequent tokens, using additional token aggregation and post-processing techniques allows a simple Avg. model to surpass both B2S and B2S-100 (see Table 6). Note that the removal of the most frequent words for B2S works in a similar way to weighted token aggregations or post-processing, and the gains are not additive. Now, the best average STS Spearman correlation score becomes 69.5 % for Avg. versus 66.4 % of B2S and the best average clustering accuracy of 62.2 % for Avg. versus 57.2 % of B2S-100. Additional techniques here helped a much simpler Avg. model outperform a much more complex learned B2S.

The success of such a simple Avg. model inspired us to seek further improvement by combining it with the parent BERT model. For the combined BERT + Avg. model results, please see Section 3.3.4.

3.3.1.2 BERT versus random embeddings

The most dramatic increase in performance due to the two techniques is observed for the random embeddings model. For STS tasks, the average Spearman correlation rises from 50.3 % to 66.4 %, while for clustering tasks it increases from 36.3 % to 49.5 %. Although the accuracy for the latter tasks is still very low, for STS, it is just a mere 3.4 % points below what BERT managed to achieve with both applied techniques.

If one were to look at the detailed performance on individual tasks (see Appendix 3.1.), findings show that random embeddings with the help of the two shaping techniques score higher than the also-shaped BERT representations. For STS14 (Table 28), it is 68.8 % versus 68.3 % Spearman correlation, while for the GoogleTS (Table 36), stackoverflow (Table 38), and tweet (Table 39) datasets, the clustering accuracy is, respectively, 69.5 % vs. 68.5 %, 70.6 % vs. 59.9 %, and 58.5 % vs. 55.1 %. This may indicate a smaller complexity of these particular datasets, where the task can be solved largely on the basis of a small set of keywords. Also, if the texts do not contain the natural language of the type the BERT was pre-trained on (e.g., they contain code), the model cannot properly contextualize the tokens, and the random embeddings without contexts work better. Yet, it is important to note that it is achieved only with the help of aggregation and post-processing methods on top of the random embeddings.

The largest clustering accuracy difference between BERT and random embeddings is for the agnews and searchsnippets datasets, with the best scores of 86.8 % versus 43.4 % and 82.9 % versus 36.6 %, respectively. The observed performance gap may be related to the vocabulary and text sizes of the datasets. Stackoverflow and tweet datasets, with random embeddings ahead of BERT, have the smallest vocabulary sizes of 7332 and 8091 (see Table 5), while agnews and searchsnippets have the largest vocabulary sizes of 16,140 and 16,334 unique tokens, respectively. Because random embeddings rely only on the distinctness of tokens, with an increasing number of them, the probability of having exactly the same keywords in two texts drops. BERT embeddings having non-identical but semantically similar tokens and similar representations helps in this case.

Having shorter text lengths (average text sizes of 12 and 11 for stackoverflow and tweet, respectively, versus 26 and 24 for agnews and searchsnippets, respectively) may also help the random embeddings, because they do not average away so easily, and there is less context to consider.

In contrast, other static models have much more comparable scores. For example, the Avg. clustering accuracy is 83.8 % (agnews) and 74.2 % (searchsnippets).

3.3.1.3 Isotropy

Many post-processing methods have previously been proposed to improve the isotropy of representations. It was argued that representations from the BERT model fall into the narrow cone and, therefore, the produced embeddings are anisotropic. Thus, by increasing the isotropy, one can improve the regular task performance.

The IsoScore isotropy metric on the Wikitext-2 dataset was calculated after applying

various post-processing and token aggregation methods for multiple models. Whiten-ing post-processing was excluded, as it always produces representations with the IsoScore at the maximum of 100 %, just due to its working principle. Then the isotropy score with the performance of the STS, clustering, and classification tasks by calculating the Pearson correlation coefficient were compared. For semantic textual similarity tasks, results are shown in Figure 2, and, for clustering tasks, results are shown in Figure 3.

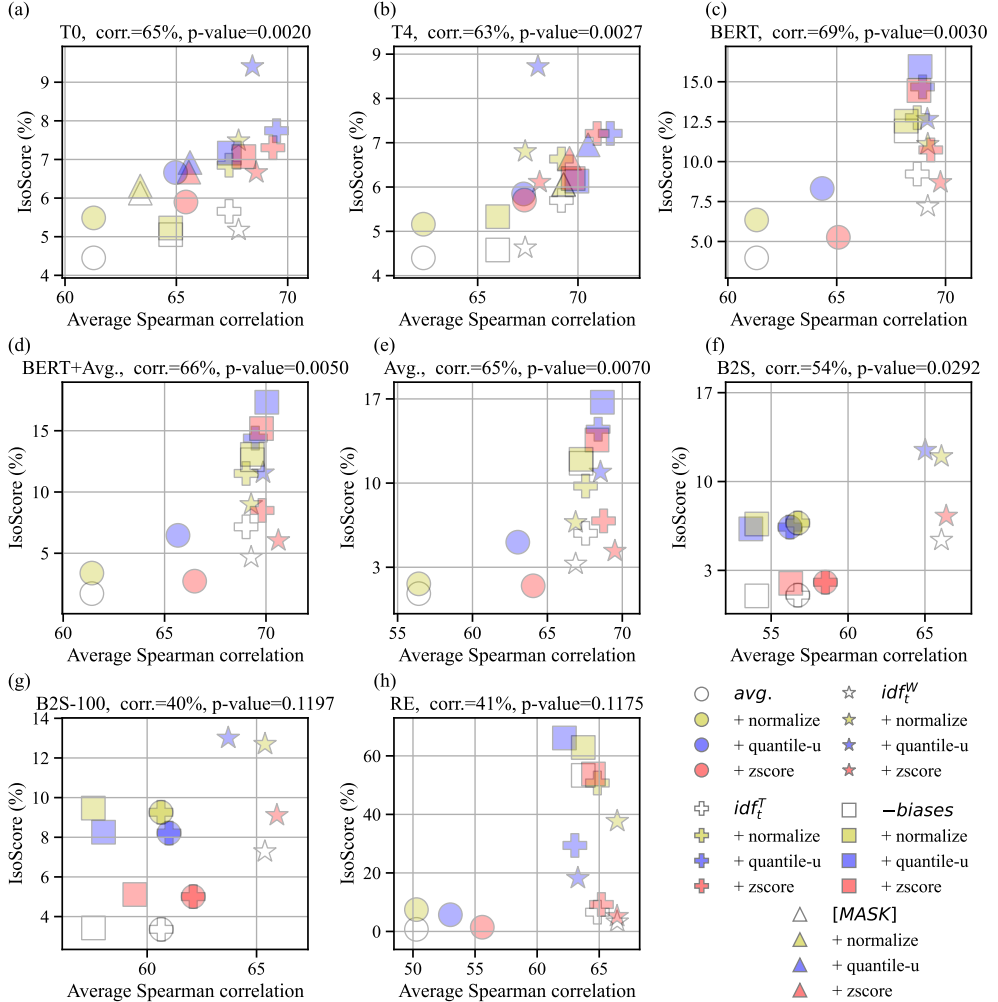


Figure 2. Relation between average Spearman correlation for STS tasks and IsoScore of Wikitext representations for each model. Pearson correlation coefficients are shown.

For each method, the baseline IsoScore is very low: it is always below 5 %. After applying weighted token aggregation techniques or post-processing, it is always in-

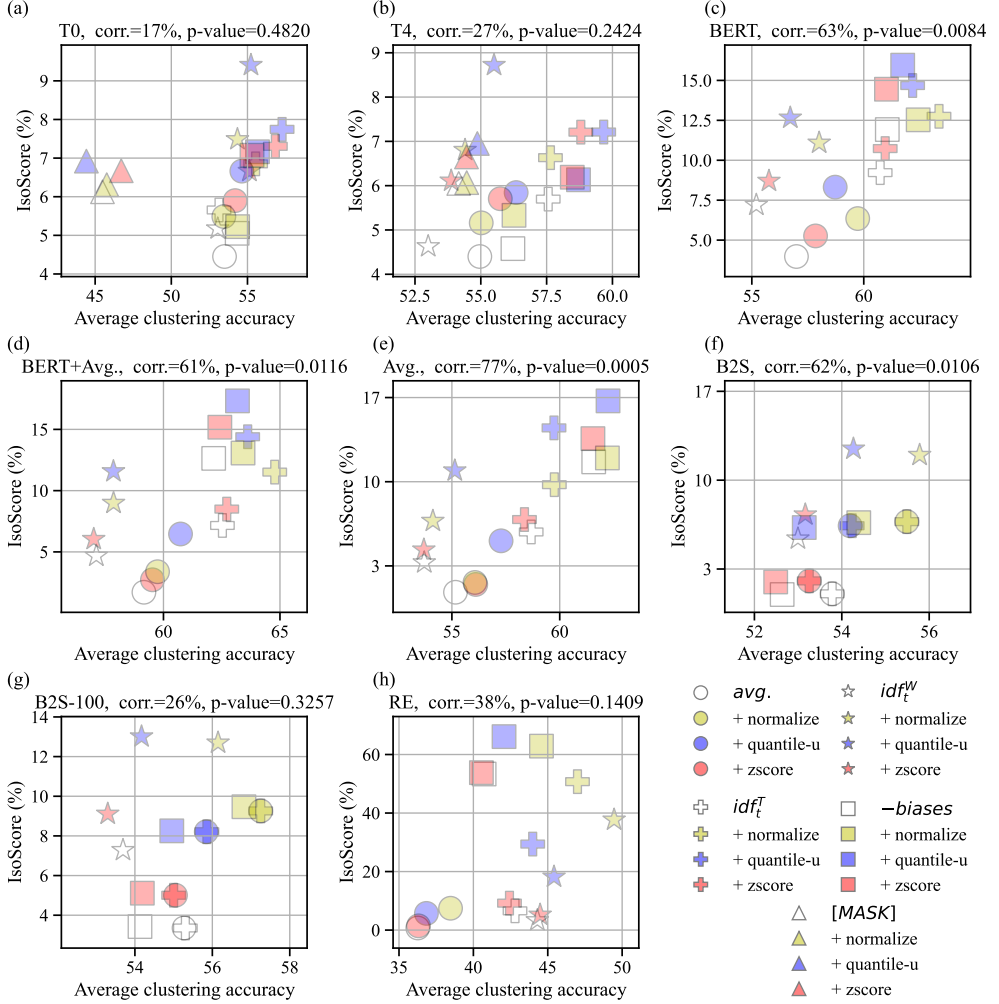


Figure 3. Relation between average clustering accuracy and IsoScore of Wikitext representations for each model. Pearson correlation coefficients are shown.

creased. The smallest improvements of up to 7-8 % score are observed for templated models T0 and T4, while the highest, over 80 %, are for random embeddings. For other models, the IsoScore reaches around 17 %.

Why is the IsoScore of random embeddings improved so much compared to the other models? The answer may be related to the inner workings of RE. Once the random embeddings are generated, these token-level representations have a maximum possible isotropy. The RE model represents text sequences as averages of these vectors; thus, isotropy going from the token level to the document is reduced. However, token aggregation and post-processing techniques mitigate this isotropy loss, allowing document embeddings to recover most of it from the token-level ones. Other models start with already low isotropy for token-level representations and thus have less space to improve it.

Six out of eight models have a moderate correlation (more than 54 %) between Isoscore and the average Spearman correlation of semantic textual similarity tasks (see Figure 2). For clustering (see Figure 3), it is less apparent, with only four models reaching a moderate Pearson correlation. However, for both STS and clustering tasks, the best score for each model is always reached by some improvement of IsoScore, compared to the isotropy score of the plain averaging setup.

In contrast, the classification does not improve with token aggregation and post-processing techniques; therefore, the correlation is not observed (and it is not shown here).

3.3.1.4 Alignment and uniformity

It was observed that token pooling and post-processing techniques do not improve the alignment and uniformity properties of the representations. Let us remind the reader that alignment is calculated with only those STS-B pairs with a similarity score of 5, while uniformity with all pairs and smaller values is better.

It can be observed in Figure 4 that the z-score post-processing always makes the alignment worse, while the normalization of the embeddings almost always does the same for the uniformity of the representations. Excluding the random embeddings model, the best alignment and uniformity properties are almost always with plain averaging and no post-processing applied.

Both alignment and uniformity are sensitive to the scaling of embedding vectors. Depending on the resulting scaling from the post-processing method, either one or the other is increased. However, reducing both alignment and uniformity is found to be difficult. This finding strengthens the reputation of these two metrics that only training the Transformer model, as shown in [17], is capable of improving them.

As presently mentioned, the outlier in its behavior here is the RE model, which improves its uniformity by different weighting of tokens or using quantile-uniform post-processing. Indeed, the latter post-processing is the least harmful for all models considered, and it disturbs alignment and uniformity properties less.

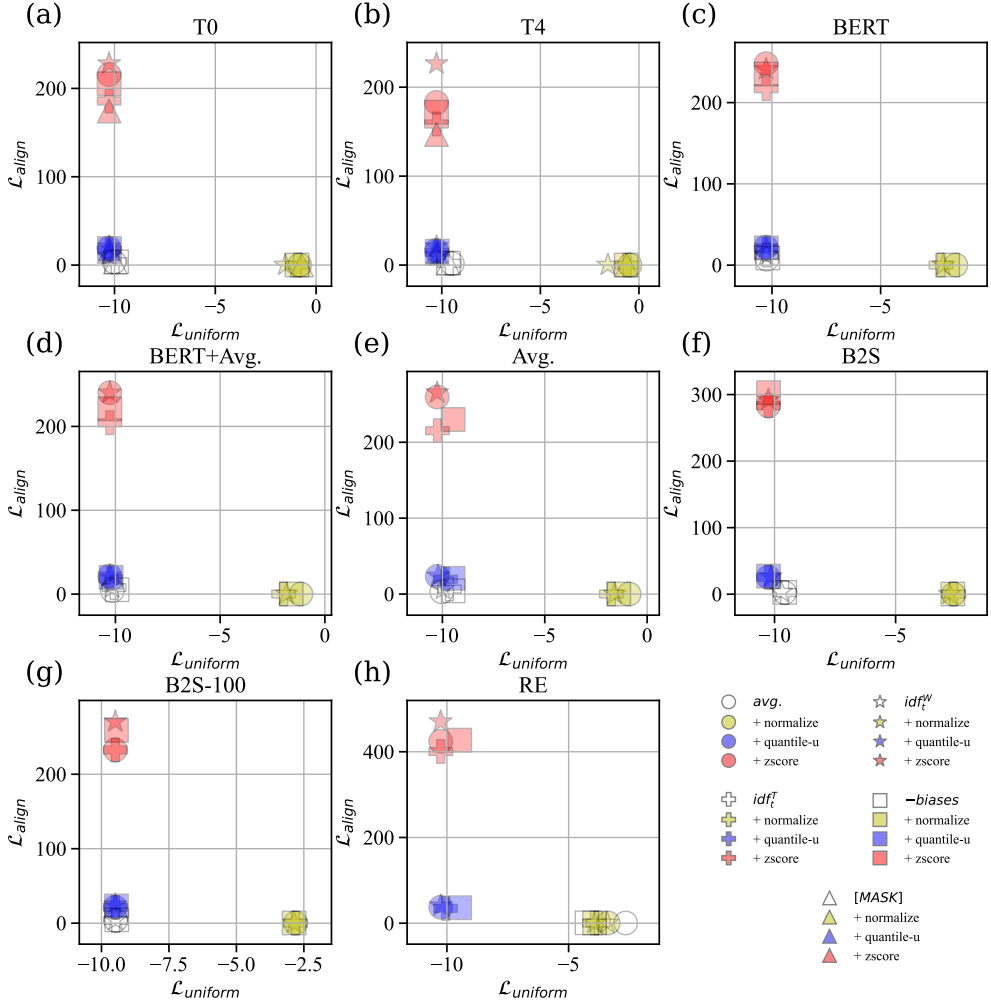


Figure 4. Alignment and uniformity of representations in relation to various token pooling and post-processing techniques. Lower values are better.

3.3.2. Improving RE baseline

As the random embeddings model was quite successful compared to BERT representations, it was separately experimented with different tokenizers and vector sizes for this baseline method. The results of 10 random seed initializations are shown in Table 8.

Table 8. Effect of tokenization and random vector size for 10 random seed initializations. Both correlation and accuracy scores range from 0 to 100 (from the worst to the best). The best result for each vector size is bolded, while the underlined result is the best across all sizes.

Tokenization	Vector size					
	3	48	192	768	3 072	12 288
Avg. Spearman correlation of STS tasks.						
BERT	23.9±1.7	46.4±1.3	49.6±0.7	50.7±0.5	50.8±0.3	50.9±0.1
ByT5	23.9±2.7	44.9±0.8	46.9±0.5	47.1±0.3	47.3±0.1	47.3±0.1
Llama	26.5±2.0	50.9±0.6	54.7±0.7	55.5±0.4	55.7±0.2	<u>55.8±0.1</u>
lowersplit	22.3±2.8	45.5±1.2	47.0±0.7	48.3±0.4	48.4±0.2	48.5±0.1
mT5	21.9±2.6	42.1±2.3	43.2±1.2	43.9±0.4	43.8±0.3	43.9±0.1
T5	23.0±2.8	43.3±2.0	45.4±0.8	46.3±0.5	46.3±0.2	46.4±0.1
Avg. accuracy of clustering tasks.						
BERT	13.6±0.4	31.7±0.5	35.7±0.3	36.1±0.2	36.3±0.4	<u>36.4±0.2</u>
ByT5	14.1±0.3	19.8±0.4	21.2±0.3	21.6±0.3	21.8±0.1	21.8±0.1
Llama	13.7±0.2	31.1±0.4	35.0±0.5	36.3±0.3	36.1±0.2	36.0±0.3
lowersplit	13.2±0.3	28.5±0.3	32.6±0.3	32.8±0.2	33.0±0.4	33.1±0.3
mT5	13.5±0.3	26.5±0.8	28.7±0.5	29.2±0.2	29.4±0.2	29.4±0.2
T5	14.0±0.2	29.1±0.6	31.9±0.5	32.5±0.3	32.7±0.2	32.6±0.2
Avg. accuracy of classification tasks.						
BERT	47.5±0.6	56.7±0.5	63.7±0.2	69.5±0.3	74.0±0.1	75.7±0.1
ByT5	47.7±0.5	53.9±0.2	57.3±0.2	59.9±0.2	60.9±0.1	61.5±0.1
Llama	48.2±0.6	56.6±0.4	63.5±0.2	69.8±0.2	74.1±0.2	<u>75.9±0.2</u>
lowersplit	47.6±0.3	57.0±0.2	63.7±0.4	69.6±0.3	73.6±0.2	75.4±0.2
mT5	47.4±0.2	56.1±0.4	62.8±0.4	69.3±0.2	73.9±0.2	75.8±0.2
T5	47.4±0.4	56.0±0.4	62.9±0.4	69.0±0.2	73.6±0.2	75.5±0.2

The first observation is that increasing the size of the random vector improves all the tasks considered. For STS and clustering, it saturates at about the size of 768, but does not overfit with bigger sizes. For classification tasks, improvements were observed up to the last experiment size of 12 288. Only the classification results of $75.9 \pm 0.2\%$ are comparable to the sentence vectors sourced from unsupervised BERT weights with a classification accuracy of 79.9 % with the difference of just 4 %. This shows that while aggregation and post-processing techniques allow RE to catch up to BERT in STS tasks, vector size scaling does the same for classification tasks.

An interesting result is that Llama tokenization is better than others for semantic textual similarity tasks. From 48 to 12 288 random vector size, its average Spearman correlation is at least 4.9 % higher. Meanwhile, for classification and short text clustering, no more differences were observed.

3.3.3. Using prompts

The normal use of prompts, as presented in [213], is to place the sentence [X] inside a template as “This sentence: "[X]" means [MASK]” and use only the vector of the [MASK] token from the final layer as the full representation of [X]. Experiments show that, in general, using prompts is beneficial; however, it was found some ways to improve performance even more.

First, extracting only the vector of [MASK] is not necessarily the best. It was found that a simple average of all tokens, including the ones from the prompt template, and also the [MASK] token, is still a valid approach. In addition, it outperforms only the [MASK] token approach for clustering and classification tasks, with corresponding 55.0 % and 79.9 % average accuracies versus 54.2 % and 76.8 % for using only the [MASK] token (see T4 model results in Table 6).

The use of the additional text template around the text enriches the representation of the target text. Let us compare the performance of token representations averaged from the 12th layer with and without a template (see Figure 5, T0 avg. and T4 avg. with a template versus BERT avg. without). For all 3 groups of tasks – semantic textual similarity, clustering, and classification – the T0 template achieves 61.3 % average Spearman correlation, 53.5 % clustering accuracy, and 80.5 % classification accuracy; the T4 template reaches 62.3 %, 55.0 %, and 79.9 %, while a regular, non-templated text obtains just 53.2 %, 50.6 %, and 79.0 %, respectively. This shows that the use of the templates allows the model to enrich target text representation. However, this effect peaks at the last, 12th layer, and the achieved performance is similar to the first + last layer combination of the regular non-templated vectors.

The performance of using only the [MASK] token also peaks at the 12th layer, so the influence it has on the enrichment of templated target text representation was investigated. Could it be that this token is the main culprit for better performance of averaged templated text representations? To answer this question, it was attempted to omit the [MASK] token from averaging (see Figure 5 T0 avg. no [MASK] and T4 avg. no [MASK]). For the T4 template and 12th layer representations, averaging all tokens yields 62.3 % average Spearman correlation, 55.0 % clustering accuracy, and 79.9 % classification accuracy; whereas, dropping the [MASK] token from averaging yields 58.7 %, 53.3 %, and 79.8 %; non-templated performance is 53.2 %, 50.6 %, and 79.0 %, respectively. It can be seen that omitting the [MASK] token in averaging indeed hurts the performance. On the other hand, the results show that it is responsible only for approximately half of the improvements, with the other half coming from the other tokens in the templated text.

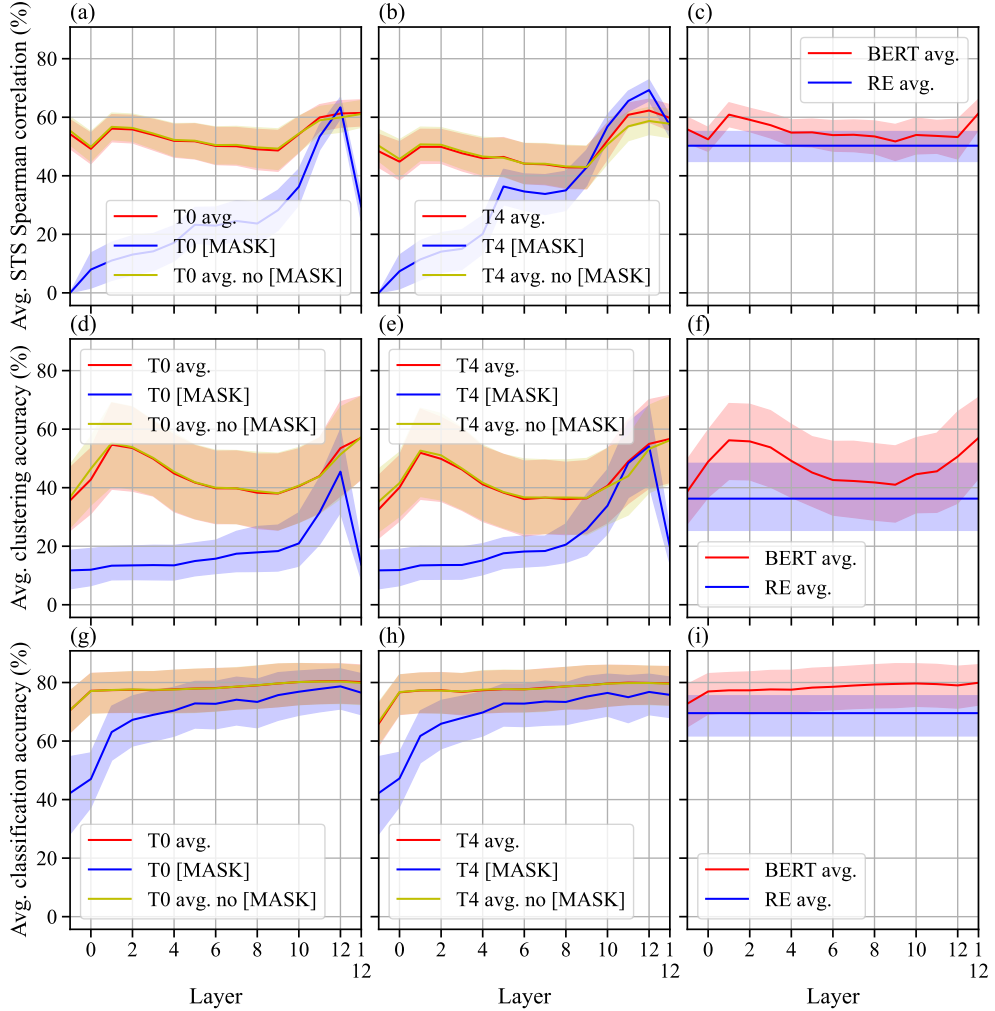


Figure 5. Layer-wise performance of templated models T0 (subfigures a, d, g) and T4 (b, e, h), as well as BERT versus RE with no weighting or post-processing (c, f, i). The layer corresponds to the exact Transformer model layer from which activations are used, the total number of layers and model weights everywhere are the same. The average performance of STS (a, b, c), clustering (d, e, f), and classification tasks (g, h, i) is shown by the lines, while shadowed areas represent 95 % confidence intervals computed via bootstrap resampling (5000 iterations). Also, the first + last aggregation over layers as the last tick $\frac{1}{12}$ on the horizontal axis is shown.

One good reason to use averaging instead of only the [MASK] token is that, then token weighting can also be applied. As it was already shown in Table 6, the T4 template together with idf_t^T token weighting and uniform quantile post-processing allowed us to reach the average Spearman correlation of 71.6 %, which was the best among the tried methods. Also, it was observed that post-processing on text representations from [MASK] token was not as effective as from averaged ones. This also suggests that all tokens in templated texts have richer representations.

3.3.4. BERT + Avg. model

One of the ways to improve BERT Transformer model representations is to combine embeddings of the regular BERT with the ones averaged over multiple contexts. That is, for each token in each BERT layer, many different contexts were collected, and the averaged vector became the vector of the Avg. model. BERT and Avg. models were then combined, according to the parameter w , which shows the fraction of Avg. model representations in the resulting vector v :

$$v = v_{BERT}(1 - w) + v_{Avg}.w. \quad (21)$$

The w parameter was also varied to the negative values in (21) to see if subtracting (instead of adding) the context-average representations from the context-aware ones helps.

The impact of the w parameter and the choice of the layer to source the representations (same layer for both BERT and Avg.) is presented in Figure 6.

One can see that, for clustering and classification tasks, the combination of both models in equal portions of $w = 0.5$ is better than these models alone ($w = 0.0$ corresponds to BERT, $w = 1.0$ is a single Avg. model). If one looks at the best scores with the two techniques applied in Table 6, for the average clustering accuracy, this translates to an improvement of 1.4 % from the 63.4 % of BERT to 64.8 % of BERT + Avg.

Meanwhile, as it can be seen in the same table, the average accuracy of the classification tasks without additional techniques applied to the combination of the first and last layers is similar between only BERT (79.9 %) and BERT + Avg. (79.7 %). However, as is seen in Figure 6, differently from the first + last optimal layer setting for BERT, BERT + Avg. has a sweet spot in the 10th layer with an average classification accuracy of 80.5 %, surpassing that of BERT by 0.6 percentage points.

These results suggest a conclusion that, for clustering and classification tasks, combining a regular BERT token with the same one but averaged over multiple contexts is beneficial.

As it was already mentioned, STS tasks preferred the regular BERT ($w = 0$) to the BERT + Avg. on average. However, for several individual semantic textual similarity datasets, the best weights turned out to be even negative. This is the case for STS15 ($w = -0.25$) and STS16 ($w = -0.75$). The same negative weight of $w = -0.5$ preference was also observed for the searchsnippets clustering dataset. Although the

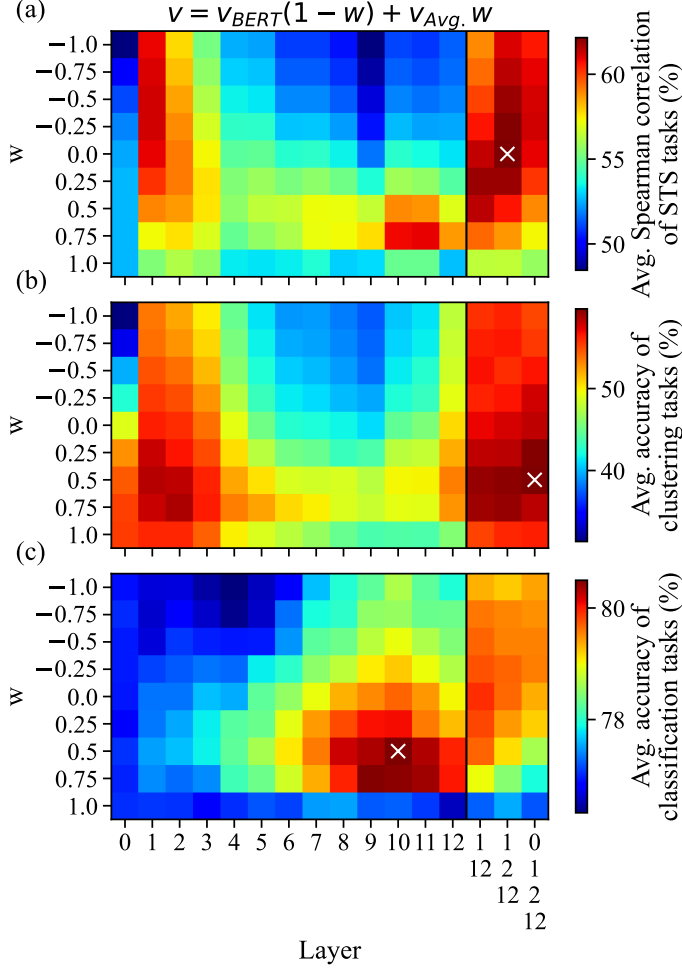


Figure 6. BERT + Avg. model performance dependence on the weight w of Avg. model and layer, from which (for both models), representations are used. To the right of the black line on the horizontal axis, average aggregation of multiple layers is also shown. Tokens are simply averaged, and no post-processing is used. The horizontal line with $w = 0.0$ corresponds to a regular BERT (B) model, $w = 0.5$ is B + Avg., and $w = 1.0$ is the Avg. model. The white $\times$ marks the maximum value.

Spearman correlation for three STS tasks was only up to several percent higher, for searchsnippets, the clustering accuracy increased to a staggering 80.2 % from 72.2 % of $w = 0.0$. This shows that the determined values of $w = 0.0$ for STS and $w = 0.5$ for clustering and classification are not universal, and, for a small percentage of tasks, they can differ. For more BERT + Avg. task-wise details, see Appendix 3.2. figures for STS (Figure 14), clustering (Figure 15), and classification (Figure 16) performance.

3.3.4.1 Layers

Figure 6 also depicts the task performance versus BERT layers. The first observation is that, for STS and clustering tasks, there is a strong preference for the first layers. As a result, the best combination of layers also involves the first ones; for semantic textual similarity tasks, it is the average of representations from one, two, and twelve layers, and for clustering, zero, one, two, and twelve layers. Even individual SICK-E, SICK-R, and STS-B classification tasks, which originate from semantic textual similarity ones, have strong first layers (see Figure 16 in Appendix 3.2.), although, on average, the best layer for classification tasks is the tenth. This shows that a lot of performance for tasks that work on similarities between texts (STS and clustering) depends on the first layers, which, according to [324], have more generalized token representations. However, the last layer, which receives recreated token identities [324], is also important, as the best combinations (such as 1 + 2 + 12) also involve it. This first + last cooperation of layers can be distinctly observed as a U-shape for clustering performance in layers (see Figure 6 and also Figure 5).

For most classification tasks (excluding those originating from STS), the U-shape is inverse. This is very clearly seen for binary classification tasks (see Figure 16 in Appendix 3.2.), where the highest classification accuracy is concentrated between the ninth and the eleventh layers. This aligns with the explanations of the previous work [241], which argues that the last layer is over-specialized for the training objective.

3.4. Conclusions

This work investigated how sentence representations derived from pre-trained Transformer models can be substantially improved without additional task-specific training. A systematic study of token aggregation, post-processing, layer selection, prompting strategies, random embeddings, and hybrid modeling approaches was performed. All methods were tested on eight STS, six short text clustering, and 12 classification tasks. Several important conclusions can be drawn.

Token aggregation and post-processing techniques are highly impactful. The experiments demonstrate that simple yet principled token aggregation and embedding post-processing techniques significantly improve performance in STS and clustering tasks. For STS, the average Spearman correlation increased by more than 9 percentage points, while clustering accuracy improved by more than 5 percentage points. The improvements are consistent across models, including pretrained BERT and even Sim-

CSE. The best results are obtained when aggregation and post-processing are applied jointly, although gains are not additive, which indicates representational saturation effects.

Improvements primarily benefit similarity-based tasks. The proposed techniques consistently improve tasks that directly operate on embedding geometry (STS and clustering), but do not improve supervised classification. This suggests that aggregation and post-processing reshape the global geometry of the embedding space rather than inject new task-specific information. In supervised classification, downstream models can already compensate for anisotropy or suboptimal variance structure, limiting the benefits of additional shaping.

Isotropy correlates with similarity performance, but not universally. Most post-processing and weighting methods substantially increase isotropy, and moderate correlations between IsoScore and STS performance are observed for most models. Importantly, the best STS and clustering results are always achieved after some isotropy improvement. However, alignment and uniformity metrics do not improve correspondingly, and in many cases, degrade. This indicates that simple geometric reshaping can improve practical performance without necessarily improving theoretically motivated alignment–uniformity properties, which appear to require contrastive training objectives.

Simple averaging is competitive. A key finding is that a simple proposed context-averaged static representation (Avg.) can match or surpass more complex alternatives such as BERT2Static. When combined with token weighting and post-processing, Avg. outperforms the learned B2S models. This highlights that much of the performance gain is not stemmed from additional training but from how token representations are aggregated and normalized.

Random embeddings as a strong baseline method. One of the most striking results is the strong performance of random embeddings when combined with aggregation and post-processing. In STS tasks, shaped random embeddings approach the performance of pre-trained BERT, and on some datasets even surpass it. This effect is most pronounced in short-text or small-vocabulary settings, where lexical distinctness is sufficient. Scaling the dimensionality further narrows the gap in classification tasks. These findings emphasize that embedding geometry and variance structure alone can account for a substantial portion of performance.

Prompting enriches representations beyond the [MASK] token. Prompt-based templating improves performance across all types of tasks. Averaging all tokens within the template — including but not limited to the [MASK] token — yields stronger and more stable representations than using the [MASK] vector alone. The improvement is partly attributable to the [MASK] token, but also to contextual enrichment from the

full template. In addition, the weighting of token and post-processing remain effective in templated settings, leading to the overall best STS results.

Combining contextual and context-averaged representations is beneficial. The hybrid BERT + Avg. model demonstrates that contextualized and context-averaged representations capture complementary information. For clustering and classification tasks, equal-weight combinations outperform either component alone. Interestingly, optimal weights vary by task and may even become negative, suggesting that subtracting context-average components can improve discrimination in certain datasets. This indicates that contextual embeddings contain both beneficial and redundant global components.

Layer selection is task-dependent. Similarity-based tasks strongly prefer the early Transformer layers, often in combination with the final layer, producing a characteristic U-shaped performance curve. In contrast, classification tasks favor higher middle layers. These findings reinforce previous evidence that the lower layers encode more general lexical-semantic information, while the higher layers become increasingly specialized toward the objectives pre-training.

Overall conclusion. This dissertation chapter demonstrates that substantial gains in the sentence representation quality can be achieved through principled manipulation of token aggregation, representation geometry, and layer selection. The results challenge the assumption that improvements in embedding performance must stem primarily from more sophisticated training objectives. Instead, a considerable part of performance is governed by geometric properties of the embedding space and the way token-level information is aggregated. These findings provide both theoretical information about the representation structure and practical guidelines for improving sentences embeddings in resource-efficient settings.

4. TEXT-LEVEL TRANSFORMATIONS FOR LITHUANIAN LANGUAGE

The experiments and results on the previous chapter were done on the English language. It has a more favorable position as it has abundant resources of datasets and models. For mid- or low-resource languages, the availability of resources plays an important role. The smaller set of available models and datasets puts an additional constraints of what can be accomplished. Therefore, in this chapter, the text representation experiments will be employed on a mid-resource language, specifically, Lithuanian.

Lithuanian has been chosen for several reasons. First, it is the mother tongue of the author of this dissertation, thus making the search and acquisition of resources more manageable. Second, Lithuanian lacks the abundance of models and resources available for more widely used languages, making NLP applications significantly more constrained and challenging, compared to English. Third, the development of Transformer-based models contributes to the linguistic and cultural preservation of Lithuanian, ensuring that the language remains relevant in digital spaces. Lastly, advancing NLP for Lithuanian can have a broader impact by providing insights and methodologies that benefit other mid- and low-resource and morphologically rich languages, fostering more inclusive and diverse development.

The previous chapter covered the work regarding the extraction of numerical text representations from the pre-trained Transformer model. Various aggregation and post-processing techniques that greatly improve the performance of such text embeddings on text-level tasks were proposed. In this chapter the text representation pipeline will be complimented with text-level transformations. In particular, Transformer-based models will be fine-tuned on the Lithuanian language to produce a condensed and normalized versions of text. The first Transformer model for abstractive Lithuanian news summarization using encoder-decoder T5 [22] Transformer model is presented, followed by the byte-level tokenization based Transformer model ByT5 [3] for GEC task. This chapter is concluded with the effects of these text-level transformations on both text and resulting numerical representations.

4.1. Lithuanian NLP Resources

The key factors regarding Transformer models are the availability of the datasets and the pre-trained models themselves, who then could be fine-tuned. These two aspects will be reviewed below.

4.1.1. Datasets

The core strength of the current SOTA Transformer model is the large amount of training data. The well-known BERT [2] in 2018 was pre-trained on the text corpus of 3.3 billion tokens. Now, at the year of 2025, the current trending DeepSeek-V3 [84] is trained on 14.8 trillions of tokens. During these years, both data, model size, and compute resources used scaled in several orders of magnitude, driving the performance of

the top models. While the number of floating point operations and hardware properties depend on financial capabilities, the size of the training dataset becomes the bottleneck. The Lithuanian language has an orders of magnitude fewer speakers and associated resources than current top-used languages such as English, Mandarin, and Spanish.

The main source of text data is the Internet itself. A dedicated repository of Common Crawl [325] maintains a publicly available scrape of the Internet, adding up to 5 billion new pages each month. Multiple works used it to select smaller, higher quality subsets. The authors of OSCAR [326] created a pipeline to filter, classify by language and shuffle (due to copyright issues) the Common Crawl data. Similarly, the authors of [11] filtered out a mC4 corpus from Common Crawl for their multilingual mT5 model pre-training. In a recent work of [327], the authors updated the OSCAR [326] and merged it with mC4 [11], following meticulous deduplication and cleaning to create a CulturaX dataset.

Apart from the amount, the quality of texts also plays an important role in the final pre-trained model performance. Early works such as BERT used decent quality datasets such as Wikipedia and BookCorpus [297], while avoiding webtexts. However, as was noticed by the authors of [328], the Wikipedia and similarly sized datasets are too small, and Common Crawl filtering, particularly for low-resource languages, can increase the amount of data by several orders of magnitude. Web-texts just have to be accordingly pre-processed. The authors of the RefinedWeb corpus [329] showed that properly filtered and duplicated web data alone can lead to powerful models, while the authors of [330] reported that just using different text extraction from HTML libraries can provide several percent improvement of the accuracy of the trained model. Significant model improvements were also shown to be influenced by the diversity of subsets by the authors of the Pile corpus [331]. However, adding new sources does not scale as well as simply filtering new web data.

The available text resources for the Lithuanian language are targeted at older rule-based methods. From freely available ones, it is worth to mention the ALKSNIS dataset, consisting of 3 643 syntactically annotated sentences [332], which is included in Universal Dependencies [333], and a morphologically annotated corpus MATAS [334], with 144 047 sentences. The latter also has information on the different genres of documents (14 %), fiction (19 %), periodics (36 %), scientific texts (24 %), and transcripts (7 %). These two are very high-quality datasets. However, they are too small to be used in unsupervised pre-training. For use as evaluation, only tasks of a limited domain, such as part of speech tagging or dependency parsing, are possible.

As there are no evaluation benchmarks, testing of new methods in the Lithuanian language is usually done on private, individually scraped web data [335, 336, 30, 337, 338, 339]. Due to the legal issues, these datasets are not shared, thus making the comparison of research contributions from different works very difficult. A positive direction is the recent work of [340], where the authors created an open Question/Answer dataset from the ChatGPT [27] summarizations of a set of Lithuanian Wikipedia pages.

4.1.2. Models

From the resources tailored specifically for the Lithuanian language, only classical methods are of great abundance [341], including GLoVe-type word vectors for Lithuanian [342]. 7 years have already passed since the BERT [2] was introduced in 2018, and there is still no specifically Lithuanian BERT available of comparable quality as the English one. Currently, the closest to such one is the trilingual Lithuanian, Latvian and English BERT of [343]. At present, generative models capture the main research attention, and recently, the authors of [340] fine-tuned Llama 2 [344] LLMs (7 and 13 billion parameter versions) for the Lithuanian language, by using over 14 billion tokens CulturaX dataset [327].

An alternative to a monolingual model is multilingual ones. Since the main English version of BERT [2] was released, the authors also published the multilingual variant. Soon, other authors followed with the multilingual XLM-R [328], which was pre-trained on an order of more multilingual data. There are also seq2seq multilingual T5 family [22] models mT5 [11] and ByT5 [3] available. Although such language-agnostic models offer a solution for rarer languages, they come with several drawbacks. First, the vocabulary has to be enlarged to contain more combinations of word pieces, adding more parameters to already huge models. Secondly, due to smaller word pieces, text is tokenized into many smaller chunks than it would be if the monolingual tokenizer was used. These two factors considerably slow down model training and inference.

Usually, models are targeted at a few languages comprising the majority of the speaker base, but their training data also incorporate texts from many other languages and can still be useful. In the recent work of [345], the authors analyzed to what extent the popular open source LLMs support Lithuanian, Latvian, and Estonian languages. They found that, unless specifically fine-tuned (like in [340]), these models are prone to lexical hallucinations. Therefore, the monolingual model is preferable not only because of its speed but also because of its quality.

4.2. Lithuanian News Clustering

In this section, results are presented of one of the first Transformer model applications for the Lithuanian language. Due to the unsupervised nature of the clustering task, it is straightforward to evaluate the numerical representations produced by the model. Here, encoder-type multilingual pre-trained Transformer models are evaluated and compared with previous methods. These early results regarding Lithuanian news clustering using Transformer models were published in [346].

4.2.1. Methodology

Data Lithuanian news articles text were scraped together with their category information. Only articles with more than 200 characters were selected. Categories were aggregated into 12 distinct ones, resulting to a total of 259 996 texts with the following category distribution:

- World news (68 635 articles);
- Lithuanian news (60 158 articles);
- Crime (30 967 articles);
- Other (30 643 articles that do not fall into clear categories).
- Business (19 964 articles);
- Sports (14 910 articles);
- Opinions (9 728 articles);
- Culture (7 967 articles);
- Cars (6 313 articles);
- Lifestyle (3 811 articles);
- Technologies (4 438 articles);
- Entertainment (2 462 articles).

To account for different sizes of the categories, 50 balanced sets by randomly sampling 125 news articles from each of the 12 categories 50 times were prepared.

Classic models For these models, each text was lower-cased and tokenized. PV-DBOW [6] was used with the hyperparameters as in [347] and trained on full dataset. Word level GLoVe [263] type vectors trained by the authors of [342] for the Lithuanian language were also used. As these have to be aggregated, several simple approaches were explored: 1) simple average; 2) average of top 20 word vectors by TF-IDF weight ranking; 3) weighting by *Softmax* of TF-IDF weights.

Pre-trained Transformer models Multilingual BERT [2] and a substantially larger XLM-R [328] were examined. Experiments were performed on their contextualized token aggregation options. Also, a multilingual BERT uncased model was fine-tuned on a subset of 27743 news articles with the batch size of 4 for 5 epochs on the original masked language modeling task.

Clustering K-means clustering was used. It is fast but requires the number of target clusters in advance, which were used as the actual number of labels. The clustering was repeated 50 times with the same initialization, but with a different sampling of input documents. K-means implementation from the Python scikit-learn library [294] was used.

Table 9. Confusion matrix for types of pair counts after clustering for MCC calculation

	Predicted same cluster	Predicted different cluster
Same label	TP (True Positive)	FN (False Negative)
Different label	FP (False Positive)	TN (True Negative)

Evaluation To evaluate the quality of clustering, target pair labels and their clustered assignment relationships must be distinguished. There are 4 types of outcomes, as depicted in Table 9.

For all possible pairings, the counts of TP , FP , FN and TN were collected. Then, the clustering was evaluated by using Matthews Correlation Coefficient (MCC) [348] score (22)

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}. \quad (22)$$

A score of +1 represents a perfect prediction, 0 an average random prediction, and -1 an inverse prediction.

4.2.2. Results

The results of the classic and the Transformer method groups are shown in Figure 7. The simple averaged GLoVe vectors only reach the MCC score of 0.203 ± 0.016 . Selecting the top tokens by TF-IDF or using weighted averaging improves the average MCC score to 0.264. Surprisingly, this is already higher than averaged contextualized BERT vectors (score 0.244 ± 0.018) or a special [CLS] BERT token (score 0.2 ± 0.013). Even averaging much larger XLM-R tokens resulted in just 0.251 ± 0.016 of MCC. Several ways to improve pre-trained Transformer model representations were tried out. First, the number of tokens to feed into the model and the number of output vectors to average were examined. It was found that feeding all 512 is not optimal. For multilingual BERT, it was decided to feed only 144 tokens and average all of them, while for XLM-R to feed 512 and average only the first 288. Such an effect was not noticed in the experiments in the previous section primarily due to shorter-length tasks. Here, on the other hand, Lithuanian news items are much longer, and selecting only the most relevant information, usually contained in the beginning of the article, is better. The second explored improvement is fine-tuning the BERT model on the masked language modeling task. It improved the averaging of tokens from 0.244 ± 0.018 to 0.322 ± 0.020 MCC score, finally surpassing GLoVe vectors. [CLS] token representations, on the other hand, only worsened, perhaps due to not using the next sentence prediction objective involving it.

Overall, it was found that BERT representations can be improved by fine-tuning the model, yet it was insufficient to outperform the older PV-DBOW model with the MCC score of 0.442 ± 0.028 . The main weakness of such training is a relatively small dataset

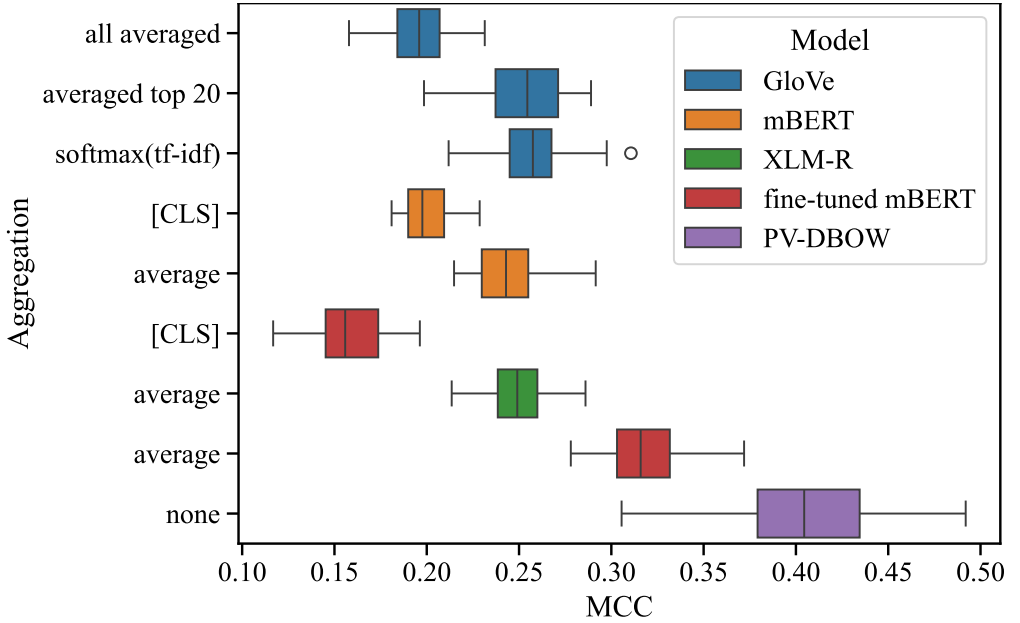


Figure 7. Lithuanian news clustering results. The higher the MCC score, the better the clustering is. Each configuration was tested with 50 clustering experiments. XLM-R model received first 512 tokens as inputs and only the first 288 output vectors were averaged, while multilingual BERT models received first 144 tokens, and all 144 outputs were averaged.

which could have contributed to the poor performance of the Transformer model. Although this fine-tuning was comparably short, it still required dedicated GPU hardware. This highlights that Transformer models are not plug-and-play replacements of the older methods, and they require data along with computation-intensive adaptation. The findings of these early experiments showed that it is unlikely to successfully fine-tune a Transformer model for text vectorization task directly. Huge compute and specific task data, such as NLI, requirements are not satisfied for the Lithuanian language. Instead, the focus of this dissertation was turned to improving representations in an unsupervised way. More specifically, we utilized the self-supervised nature of abstractive summarization and GEC tasks to produce improved text transformations for a regular text representation pipeline.

4.3. Abstractive Summarization of Lithuanian News

In this section, a model will be investigated, which produces a condensed text representation. This is the goal of the abstractive summarization task. By optimizing the textual (rather than numerical) representation, the advantage of the current boom in the generative artificial intelligence will be taken. Various factors will be considered: input tokenization, fine-tuning setting and training setting, and duration. Also, unlike English, the Lithuanian language does not have any benchmark for text level representation evaluation. This further encourages us to seek other, automated ways,

to evaluate capabilities of learnable text representation models. Partial results of this work on abstractive summarization were published in [349].

4.3.1. Methodology

Data The previous data collection pipeline was improved by also incorporating structure information (description/summary, main part) of the texts. The dataset was enlarged by considering more websites. The following filtering conditions were applied on the resulting dataset: 1) summary length is greater than 10 characters; 2) main text length is greater than 100 characters; 3) main text length in characters was at least twice the summary length; 4) the longest matching string overlap between the summary and the main text is less than 20 %. The final size of the dataset was 2 031 514 news articles, with 4 096 articles set aside for validation.

Decoding algorithms Generative models are generally trained to generate one token at a time. Given a dataset $\mathbf{x} = \{x^1, \dots, x^{|\mathbf{x}|}\}$ with $|\mathbf{x}|$ text samples, a generative neural network with parameters θ is trained by minimizing the negative log-likelihood

$$\mathcal{L}(\mathbf{x}) = - \sum_{k=1}^{|\mathbf{x}|} \log \prod_{i=1}^{|x^k|} p_{\theta}(x_i^k | x_{<i}^k), \quad (23)$$

where x_i^k is the i -th token of the k -th text in the dataset D . The output of such a network is softmax probabilities for each token in the vocabulary. Although training cares only about the target token getting the maximum probability, there are multiple ways to use the generated probability distribution during inference. The beam search decoding keeps the number of best sequences found at each token generation step. This way, one high probability token will not supersede a high overall probability sequence. However, the authors of [350] noted that human language does not follow a distribution of high-probability next words. Therefore, sampling-based methods are used, such as top-k [351], sampling from the most probable top-of-the-top k tokens, or top-p (nucleus) [350], which samples only from the smallest set of tokens whose cumulative probability just exceed p .

Training In the previous clustering experiments, it was noticed that, usually, Lithuanian news articles contain some structure: title, summary, main text. In an abstractive summarization task, such a setting is an advantage. English pre-trained seq2seq T5 [22] was fine-tuned to generate summaries, given as input only the main parts of the news articles. Yet its English tokenizer was replaced with the custom to better account for the Lithuanian language. SentencePiece [37] was employed to encode text as unigram [36] subwords. A total of 10^6 samples of the collected text corpus was used to train the tokenizer, and the vocabulary size of 32 000 was used, the same as in the original T5.

The training of the Transformer model was performed by using the Python implemen-

tation of [296]. The model was trained with batches of 128 text-summary pairs, both truncated to 512 tokens for 350 000 steps totaling 22 epochs. Although the mixed precision and GeForce RTX 2080 Ti GPU was used, it still took approximately 500 hours. To pass through batches due to a limited GPU memory, a gradient was sequentially accumulated in 32 steps. It was one of the longest and most compute-intensive trainings performed. The learning rate started with 10 000 warm-up steps and followed by the inverse square root schedule. Similar to [22], the Adafactor [352] optimizer was employed.

Alternatively, to a limited extent, starting the training from multilingual mT5 [11], using its original multilingual tokenizer, was also explored.

Evaluation An automatic word-overlap-based evaluation metrics Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [14] was used. In particular, we used the ROUGE-2 metric, which is a ratio between the common 2-grams between the reference and the generated summary, divided by the total count of 2-grams of the reference [353].

4.3.2. Results

The tokenizer trained specifically for the Lithuanian language allowed one to train the T5 model much more efficiently. The resulting token counts were tested in the main text parts of the validation set. As shown in Figure 8, the English tokenizer produces 1448 tokens on average, multilingual – 789, and the trained one – 530. Tokens are the critical part in Transformer models, especially because their computation complexity has quadratic dependency on the sequence length. Indeed, the training mT5 process with its original tokenizer ended up being 4 times slower. This is not only due to the longer sequences and a resulting increase of computations, but also more weight updates needed. The mT5 model has 582 million parameters, from which, vocabulary and softmax output matrices alone amount to 256 million.

It was decided to replace only the tokenizer in T5 to train on Lithuanian texts. As is visible in Figure 9, this way the validation loss profile is lower than from the corresponding model full random weight initialization. It is also clearly seen that the model started to overfit the training data after 250 000 training steps or 15 epochs.

Many decoding methods were experimented with, including greedy, stochastic and combinations of both types. Overall, 10 beams search was the best-performing generation method. After 250 000 training steps, the mean ROUGE-2 reached 14.8 % ROUGE-2 for greedy, 16.3 % for 10 beams search, and 16.1 % for top-50 10 beams search decoding. Increasing the beam size only increased the generation time instead of additional improvements.

Automatic ROUGE-2 evaluation was found to be problematic. First, its use creates a slight contradiction here. On the one hand, it is important to get good abstractive summaries (as opposed to extractive) that are not excerpts of the main text. Even training

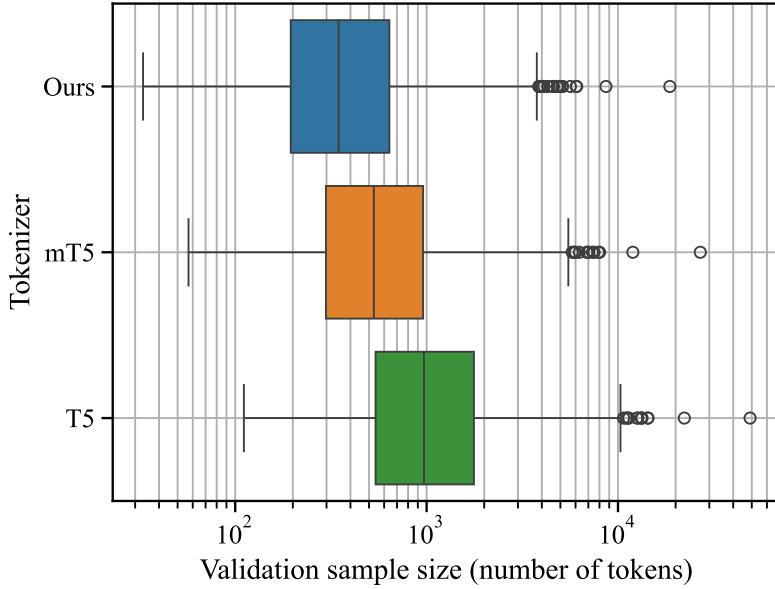


Figure 8. Number of tokens in validation samples for different tokenizers.

samples, where the target summaries do not have this quality, were eliminated. Despite this, ROUGE metrics were used as the evaluation of summaries, which highly rewards the words and phrases taken from the original text. This accentuates again the lack of good automatic evaluation metrics that would mimic how humans evaluate the quality of summaries better. Another observation was that a deviation of results was very high, $\approx 14\%$. This may be due to the differences of how the same meaning is expressed by the different tokens which do not match the ones from the target summary. A limited manual inspection of the generated summaries revealed a lot of high-ROUGE samples being just combinations of multiple excerpts from the main text, while there were very good summaries with ROUGE scores just above zero. In addition, there were a lot of instances of generated text samples with minor factual mistakes. This limits the application of the model as human supervision is the only viable way to precisely assess the generation quality.

A generated summary of examples is provided in Appendix 4.

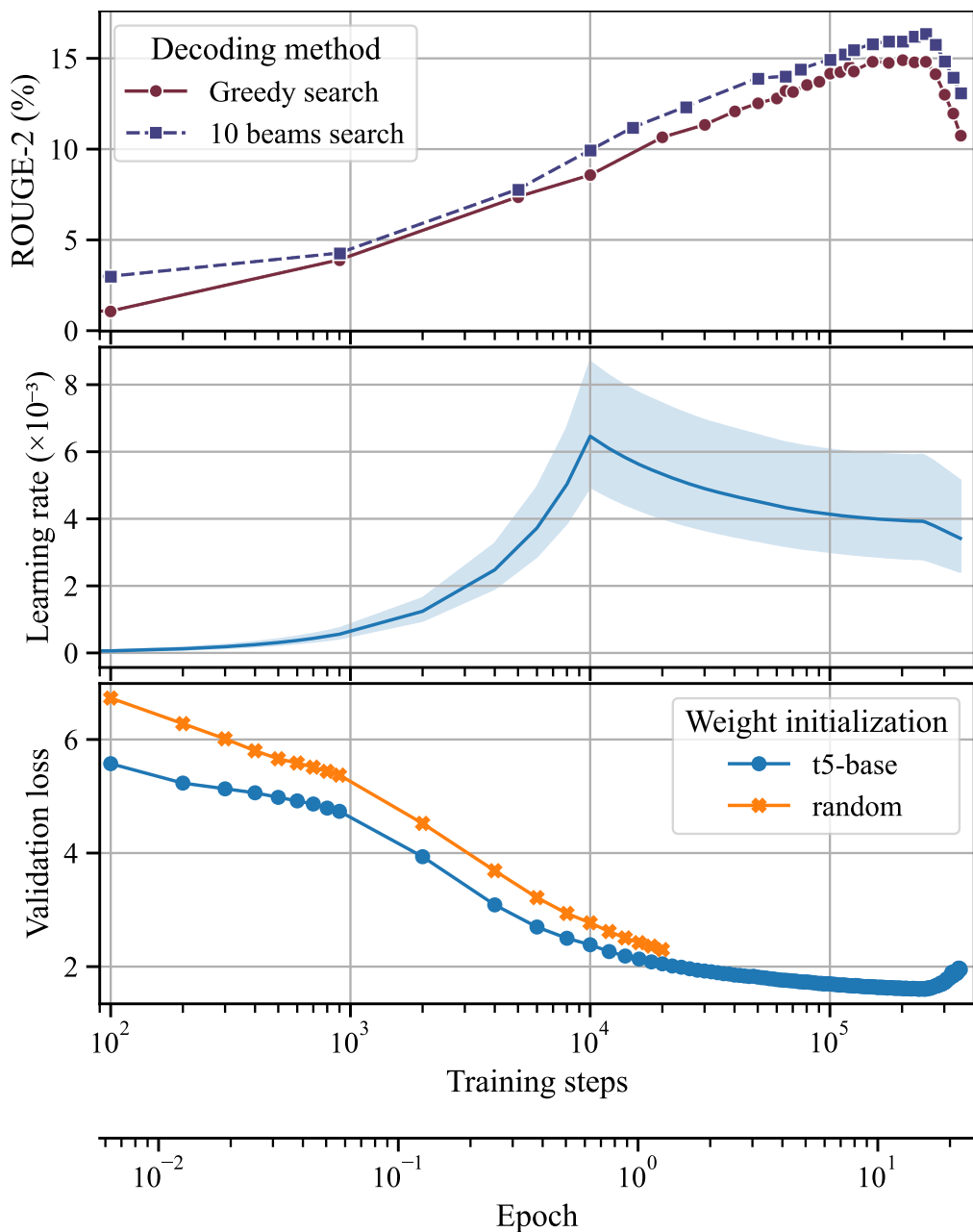


Figure 9. Results of training Lithuanian abstractive news summarization model. ROUGE-2 mean F-score for greedy and 10 beams searches (left vertical axis), learning rate, and validation loss (right vertical axis) dependency on model training steps. Shaded areas represent 95 % confidence intervals computed via bootstrap resampling (5000 iterations). The learning rate is shown for a model trained with *t5-base* initialization.

4.4. Grammatical Error Correction

One of the most influential French linguists, Antoine Meillet (1886-1936), once stated: “Anyone wishing to hear how Indo-Europeans spoke should come and listen to a Lithuanian peasant”. The Lithuanian language retained most of the features of the Indo-European Protolanguage and thus can be characterized by a very ancient linguistic structure: declensions (of nouns, adjectives, and pronouns), short and long vowels, diphthongs, etc. Lithuanian is also known to have similarities to Sanskrit, the ancient classical language of India. Such roots distance Lithuanian from English, the language most widely used in the NLP field. It also makes Lithuanian complex: relations in a sentence are expressed by word endings; verbs can be made from any onomatopoeia; there are 13 different participial forms of the verb [354]; 47 % of Lithuanian word forms are morphologically ambiguous [355]. All these and other properties make Lithuanian a perfect candidate for the automatic GEC task.

Let us consider a Lithuanian example sentence, which has to be checked for correctness: “Oras gera, sunelis vejas drugelius ant kelio, arbata dar karšta?”. Although only diacritics were removed, there are already multiple ambiguities: 1) “suneis” can be “šuneis” – a little dog (diminutive of “šuo”) or “sūneis” – little son (diminutive of “sūnus”); 2) vejas can be “vejas” – to chase or “vėjas” – wind; 3) “karsta” can be “karšta” – hot or “karsta” – coffin. The correct target meaning depends on the context, and the traditional rule-based methods cannot distinguish the correct option. To successfully correct the sentence, the method has to first understand the correct meaning of each word. Therefore, the Transformer model, which uses contextualized token representations to capture the sequence ‘meaning’ is a promising approach for this task. In this section, seq2seq type Transformer models for the Lithuanian GEC task will be applied. It will be looked further at how different granularity of tokenization impacts the representation learning required to remove artificially augmented noise. First, simpler diacritics and typos restoration tasks will be covered, and the results are compared for multiple languages on the established benchmark datasets. Then, the focus will move on the Lithuanian language only, expanding to more language-specific error groups. Some of the results of this work have been published in [356, 357].

4.4.1. Methodology

4.4.1.1 Multilingual text data

For diacritics and typos restoration experiments, the same 12-language (Croatian, Czech, French, Hungarian, Irish, Latvian, Polish, Romanian, Slovak, Spanish, Turkish, and Vietnamese) Wikipedia dataset, as by the authors of [358], was used. This set did not include Lithuanian, so it was added to the list, by using the tools publicly provided by the authors of [358].

The multilingual dataset consists of training, development, and testing sets. All three are lowercased, tokenized to words, and are split into sentences. The split between sets is performed on the Wikipedia article level. Statistics of the training set are shown in

Table 10. The testing sets do not differ much, except that each language has exactly 30 000 sentences allocated to it, and, thus, has a similar amount of words. The percentages of alpha-words, diacritic words, and diacritic letters in the testing sets do not deviate by more than 10 %, compared to their training counterparts.

Table 10. Languages and the training dataset statistics. Diacritic percentages are calculated among alphabetical words or letters. Alphabetical words (alpha-words) range from 72% to 86% of all the total words, including numbers.

Language			Dataset			
Name	Diacritic Letters	Keyboard Family	Sentences	Alpha- Words	Diacritic %	
					Words	Letters
Croatian	5	QWERTZ	802,610	12,914,186	14.55	2.78
Czech	19	QWERTY	952,909	14,730,260	48.69	12.90
French	15	AZERTY	1,818,618	37,612,736	16.49	3.72
Hungarian	9	QWERTZ	1,294,605	17,587,448	50.05	11.48
Irish	5	QWERTY	50,825	1,005,620	29.52	7.04
Latvian	15	QWERTY	315,807	4,244,914	48.57	10.27
Lithuanian	9	QWERTY	612,724	7,096,677	38.75	7.00
Polish	9	QWERTY	1,069,841	16,178,130	32.71	6.42
Romanian	6	QWERTY	837,647	16,050,136	27.04	5.87
Slovak	25	QWERTZ	613,727	9,180,800	42.38	9.32
Spanish	7	QWERTY	1,735,516	42,863,263	11.50	2.33
Turkish	11	QWERTY	875,781	10,785,516	31.35	6.30
Vietnamese	67	QWERTY	819,918	20,241,117	81.18	25.94

4.4.1.2 Lithuanian text data

For more general GEC experiments for the Lithuanian language, the dataset gathered during the clustering (Section 4.2) and summarization (see Section 4.3) applications from the previous sections was further expanded. In addition to crawling more texts, the structure at each website was also manually checked so that only the relevant parts: title, summary (optionally), and the main text paragraphs would be scrapped. The following types of web pages were crawled: news, literature, blogs, encyclopedias, and others. Titles and summaries were added to the main text paragraphs as additional paragraphs. Finally, the data were split into paragraphs. As a result, a single paragraph became a single sample of the dataset. Although a well-curated data scraping was performed, there were still some artifacts in the data that had to be corrected or removed. Firstly, some minor visual inspections of acquired texts was performed. A relatively high rate of spelling errors was noticed in one website, and it turned out that it contained various essays written by students. Another website had similar problems due to the lack of editing in blog posts. All texts were removed from such websites. Fol-

lowing the common mistakes in Lithuanian web texts [359], the following automatic corrections were performed:

1. Normalized quotation marks into „ABC“ from “ABC”, ”ABC” or others.
2. Added omitted spaces, i.e., “1918m. vasario 16d.”, “ir t.t.” and “A.Sabonis” were corrected into “1918 m. vasario 16 d.”, “ir t. t.” and “A. Sabonis”.
3. Deleted unnecessary spaces, correcting texts such as “tik darbuī , visiškai pamirštant poilsį ,” into “tik darbuī , visiškai pamirštant poilsį.”.

The above corrections were skipped for text from the high grammatical correctness websites *vlkk.lt* and *vle.lt*. The text samples based on several statistics were also filtered:

1. The sample text length should be at least 20 characters.
2. The fraction of Lithuanian letters in a sample should be at least 0.98. This filters out text from other languages and with miscellaneous characters. The characters “€£\$%wx” were included as Lithuanian since they are used quite often.
3. The fraction of spaces to non-spaces should be at most 0.02. This filtered out samples dominated by URL addresses.

After the final deduplication step, the dataset consisted of 4 194 304 samples.

4.4.1.3 Model of typographical errors

Diacritic removal is performed simply by replacing all diacritic letters with the non-diacritic equivalents. However, for typographical error inductions, a dedicated realistic corruption model is required. The approach from the authors of [360, 361] is followed, where a smaller dataset is used to calculate each error group probabilities and is then applied on the target one. According to the authors of [362, 363], more than 80 % of errors can be attributed to substitution, deletion, insertion, or transposition errors. Therefore, these error groups were used in the typos model.

It is assumed that typographical errors are induced only by the human typing text on a physical keyboard. A single keypress instruction consists of information of which the hand, finger, and key row are required to select. The authors of [364] argue that the confusion of these instructions is the main culprit of substitution errors, while mixed instruction timing between the two hands (operating on different parts of a keyboard) is the main culprit of transposition errors. There are also known visual and phonological causes of errors [365], however, the setting is restricted to the physical keyboard layout influence. This allows us to model typographical errors for all languages, given the distribution of keyboard errors for a single language. Overall, the typographical model moves through each symbol and induces errors in a stochastic manner by evaluating probabilities of various error types for each character, which is similar to a process of text writing.

The chance for a letter to participate in a particular error type is determined according to the frequency of errors in the reference dataset. The largest known original typo dataset, the GitHub Typo Corpus [366], was used. The dataset was filtered for only English language typos, and the characters were selected with a count of at least 1 000. Given the final character set C , the total number of times $f(c)$ the character $c \in C$ or a specific typo pattern appeared in the selected corpus, the following probabilities for each character are considered:

$$P(\text{deletion} \mid c) = \frac{f(c \rightarrow \emptyset)}{f(c)}, \quad (24)$$

$$P(\text{substitution} \mid c) = \frac{\sum_{\underline{c} \in C} f(c \rightarrow \underline{c})}{f(c)}, \quad (25)$$

$$P(\text{insertion after} \mid c) = \frac{\sum_{\underline{c} \in C} f(c \rightarrow c\underline{c})}{2f(c)}, \quad (26)$$

$$P(\text{insertion before} \mid c) = \frac{\sum_{\underline{c} \in C} f(c \rightarrow \underline{c}c)}{2f(c)}, \quad (27)$$

$$P(\text{transposition} \mid cc') = \frac{f(cc' \rightarrow c'c)}{f(cc')}. \quad (28)$$

Note that insertion errors were divided into two distinct categories, whether the character is inserted after the one in question, or before. Both insertion probabilities are collected from the same samples, so they were divide by two. An alternative way would be to collect triplets of characters before the one in question and after, but the probabilities would then be sparse. Nevertheless, the chosen approach covers the so-called ‘fat-finger’ errors.

Typographical error induction experiments were run on the original GitHub Corpus, and it was confirmed that this generation method aligns with the original error type distribution. Initially, only about 1 % of characters were corrupted, so the probabilities were scaled by a factor of three to be close to the low error rate, as defined in [360]. The final error type distribution and percentage of the corrupted characters for each language are depicted in Figure 10. The amount of generated errors for each language slightly varies because the letter frequencies derived from English differ in other languages.

Insertion and substitution errors can result in many different outcomes. The probabilities for specific letters to emerge, given that this type of error occurs at a specific place,

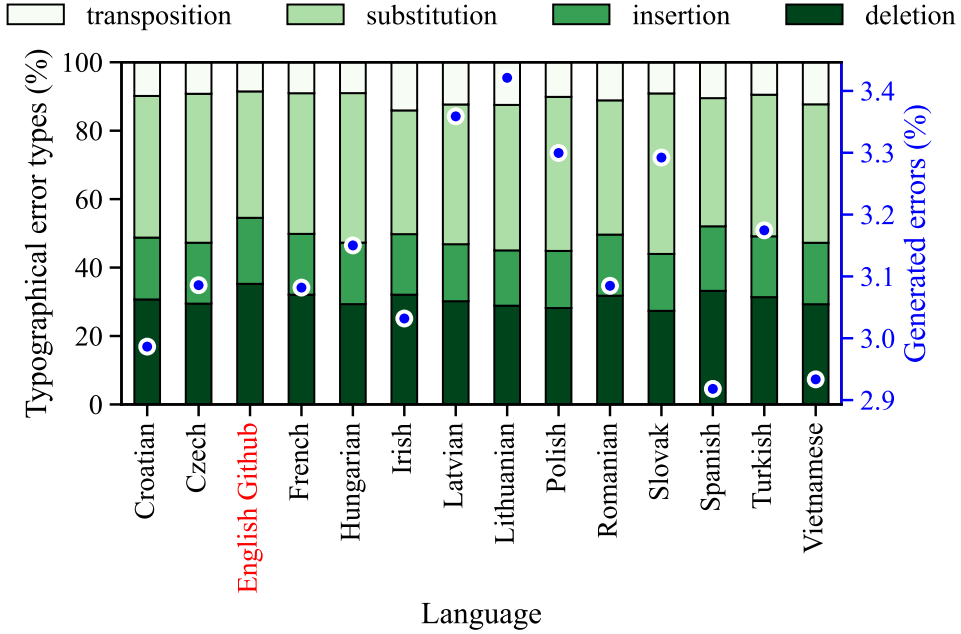


Figure 10. Distribution of generated typographical errors by category (the left vertical axis and stacked bars). Proportions for the English part of the GitHub Corpus (used to derive generation probabilities) are also depicted for reference. The total percentage of induced corruptions are included (the right vertical axis and corresponding blue dots).

are computed by the following equations:

$$P(c \rightarrow c' \mid c, \text{substitution}) = \frac{f(c \rightarrow c')}{\sum_{\underline{c} \in C} f(c \rightarrow \underline{c})}, \quad (29)$$

$$P(c \rightarrow cc' \mid c, \text{insertion after}) = \frac{f(c \rightarrow cc')}{\sum_{\underline{c} \in C} f(c \rightarrow c\underline{c})}, \quad (30)$$

$$P(c \rightarrow c'c \mid c, \text{insertion before}) = \frac{f(c \rightarrow c'c)}{\sum_{\underline{c} \in C} f(c \rightarrow \underline{c}c)}. \quad (31)$$

The typo statistics from the English dataset were used, and the assumption was followed that typos are based purely on the layout of the keyboard (the proximity of keys, etc.), so the same typo statistics will be in all the other languages using the QWERTY layout. The extensions of the character sets and keyboard layouts for different languages were excluded, as typos were only introduced to the undiacritized versions of the texts, irrespective of the case. Other possible minor variations in the keyboard layouts were disregarded as insignificant.

Croatian, French, Hungarian, and Slovak languages have different keyboard layouts

(see Table 10). Therefore, the original English QWERTY dataset was remapped before inferring typo probabilities for these languages. For example, for Croatian, which has a QWERTZ layout, the letters “z” and “y” had to be swapped when calculating probabilities. No significant model performance differences were observed between the QWERTY and remapped typo generation versions.

4.4.1.4 Augmenting other grammatical errors

For Lithuanian-only experiments, more language-specific errors were added, as described below.

1. Letters that sound similarly. These mistakes were modeled by defining sets of characters that sound alike. Take a set of letters “iijy” as an example. Then, probabilities for each of the set letters occurring were collected. Following the example, it is way more common to mistakenly write “i” instead of “i̇”, rather than “i̇” instead of “i”, as “i” is much more common. Finally, a fraction of all letters in the texts, belonging to the particular set with other set items, according to their probabilities, was randomly substituted. Detailed sets of letters and their member probabilities are derived from the raw (no pre-processing) subset of 2 909 403 samples, and are presented in Appendix 5.
2. Gemination are doubled consonant letters that sound like a single one and thus are prone to be typed only once. This also applies to any consecutive letters from “ċċṡṡżż” group. For example, the words “puṡseṡeṙė už̇ṡimeṙk̇ė̇” may be mistakenly written as “puṡeṡeṙė uṡimeṙk̇ė̇” as they sound similar due to the gemination.
3. Assimilation to an adjacent letter. This is specific to any letter of “ṗṫkṡṧ” being before any of the “ḃḋġżż” or vice versa. For example, the words “diṙbti̇, liḃḋavȯ” may be mistakenly written as “diṙpṫi̇, liḃḋavȯ” as this is how they sound due to the assimilation.
4. Uppercasing or lowercasing the first letter in a word. For example, the word “ȧžuolas” can start both with the lower or upper case depending on whether it is a tree (oak) or person’s name. The first words in a sentence were excluded as these always have to start the upper case.
5. Delete and add space. All occurrences of spaces and all empty strings not at a word boundary were separately matched.

The typos for Lithuanian-only experiment texts were also induced by the typographical error model. A total of 2 % characters were corrupted, from which, we determined: substitution, 36.1 %; deletion, 31.7 %; insertion, 17.8 %; transposition, 14.4 %.

4.4.1.5 Models

To better account for fine-grained differences in erroneous text, a ByT5 [3] model was employed, that uses byte-level inputs. It is a general-purpose pre-trained multilingual

seq2seq model. The authors of [3] addressed many issues of the previous mT5 [11]. They got rid of sub-word multilingual vocabulary embeddings and softmax activations comprising a large part of model weights. Instead, in ByT5 model’s case, the total vocabulary size is just 384, consisting of: 1) *<pad>* for padding; 2) *</s>* for the end of the sequence; 3) *<unk>* for an unknown token; 4) $256 = 2^8$ values of the main eight-bit byte; and 5) 125 extra sentinel tokens used only in the pre-training task. In the small ByT5 version, such a smaller vocabulary accounts only for 0.3 % of the total parameters. Also, the authors concentrated 3/4 weights into the encoder part, to better account for finer-grade tokens, resulting in 12 encoder layers and four decoder layers. For Lithuanian, the GEC task fine-tuned T5 [22] model was used, which was previously adapted from English by training on a summarization task (see Section 4.3).

Finer tokenization has a drawback in producing more tokens, therefore, increasing the model computation time. However, the authors of ByT5 showed that, for a short-to-medium length text, this increase is not significant. For multilingual typos and diacritics restoration experiments, the input dataset consists of single sentences, thus it is also a not an issue. However, for Lithuanian GEC, the input is made up of whole paragraphs and can cause a significant slowdown. However, as is shown in results, the benefits of a finer-grade input text ingestion far outweigh the slower training time, compared to the token-level alternative T5 model.

For simpler diacritics-only and diacritics-typos restoration tasks, the results were compared to the simple dictionary method. The same multilingual typo-free diacritization-only training dataset was used for training ByT5 models to collect mappings of corrupted words to the corresponding corrections. This method works best if there is only one translation word. In case of more correction options, the more frequent one was selected. It is a good baseline indicating the model’s performance in restoring only less ambiguous corruptions. Also, a combination of dictionary and ByT5 was tried, by replacing single mappings by the dictionary, and others, with more than one candidate, by ByT5 generation.

The results were also compared to the open-source Hunspell [367] spellchecker. The words that it found to be incorrectly spelled were replaced with its first suggestion.

4.4.1.6 Training

Following the other authors of [22, 11, 3, 368], the Adafactor optimizer [352] and a learning rate of 0.001 were used. The learning rate was not changed during the training to make it simpler, and to better compare results across different setups.

A GeForce RTX 2080 Ti GPU was used. Due to the modest memory size, for the ByT5 Transformer model, only a single sequence of 1024 tokens could fit. Multiple accumulation steps had to be used to process a larger batch sequentially by smaller 1 sequence parts. Although it may look inefficient, there is no other way to train the model with given hardware. On the other hand, feeding only a single sample at a time allowed us to avoid padding.

For multilingual experiments, the batch size of 256 was chosen. Such a configuration

matches the total maximum number of tokens ($256 \times 1024 = 2048 \times 128$) with the best previous system for diacritics restoration [369]. Each model was trained for 2048 steps, each consisting of 256 sentences/samples, with a total of $2048 \times 256 = 524\,288$ sentences, and this took up to 10 h for a single model. For example, for the Lithuanian language, this corresponds to a 0.86 epoch over the total 612,724 sentences in its dataset (Table 10). In these results, the basic training is the one that is trained for $\times 1$ the number of sentences (#samples). This training length was fixed, irrespective of the available dataset, for each language (Table 10) to make the training comparable among languages. In experiments where models were trained for longer (e.g., $\times 8$), the whole dataset was passed as many times as needed, e.g., for Lithuanian, $\times 8$ corresponds to 6.8 epochs.

For the diacritics restoration task with each language, three different models were trained. Each model has a different weight initialization, and data sampling is performed differently, according to a given random seed. The results are reported as a mean and a standard deviation over these three runs. In addition, models were trained for simultaneous diacritics and typographical error corrections for each language.

Several models were also trained for a much longer time. First, the basic fine-tuning setup with a batch size of 258 to 6 000 steps (all other basic setups are up to 2,048) was continued. At this stage, the loss became noisy (although it was low), so the batch size was increased to 8 192 and continued training further. Due to the change in the batch size, the model training steps are reported by how much training data, compared to the basic setup, it consumed. In results, models are reported as trained for $\times 8$ and $\times 19$ the number of samples in the basic setup. Ceiling-rounded numbers were chosen as a means of convenience in the setup. As long training is very time-consuming, only a few of long-ones were performed. It should still sufficiently indicate the scaling effects.

For Lithuanian-only experiments the total batch size of 128 was used. For the T5 model, it was realized by 64 gradient accumulation steps of batch size 2. It took us approximately 100 hours for ByT5 and 30 hours for T5 fine-tuning on 4 194 304 text samples. ByT5 took substantially longer due to the longer sequences produced by the finer byte-level tokenization .

The training script and the Pytorch model implementation were used from the Hugging Face library [296]. If not stated otherwise, all default parameters as they are in this library version 4.12.0 were used.

4.4.2. Evaluation

For diacritics and typos restoration, the alpha-word accuracy metric [358] was used. It considers only the alpha words, i.e., for all word characters, the general Unicode category property is one of “Lm”, “Lt”, “Lu”, “LI”, or “Lo”; and the word has at least one letter. Then, the alpha-word accuracy is the ratio of a number of correctly predicted words T_s and the number of target (gold) words T_g :

$$\text{alpha-word accuracy} = \frac{T_s}{T_g} \cdot 100 \%. \quad (32)$$

This metric accounts both occasions of necessary and unnecessary accent generation and makes sure to omit words that cannot have accents (e.g., numbers).

To evaluate more error types induced for the Lithuanian language, ERRANT [370], a popular GEC evaluation metric, was adopted. It applies a set of rules operating over a set of linguistic annotations to construct the alignment and extract individual edits between corrupted, corrected, and gold-standard texts. This way, precision, recall, and *F*-score can be calculated. The original ERRANT was customized by using Hunspell dictionaries [371], stemmer [372], spaCy version 3.2 pipeline `lt_core_news_lg` [373], and corresponding part-of-speech tags for the Lithuanian language.

During all experiments, texts were generated by using a simple greedy decoding with a beam size of 1. That is, for each subsequent token, the selected one is with the highest model-assigned probability.

4.4.3. Experiments and results

Models were trained to: 1) restore diacritics; 2) both diacritics and typos; and, for Lithuanian, 3) multiple language-specific error types. The results of these experiments are presented below.

4.4.3.1 Diacritics restoration

The results of the diacritics restoration are presented in Table 11. The column “Raw” indicates the alpha-word accuracy of the uncorrected text for comparison. Naturally, the more diacritic-heavy the language, the lower the number. The results of the ByT5 model, trained for 524 288 sentences ($\times 1$), are between the Dictionary, the column “Dict” (a simple statistical Unigram model) and the previous SOTA model [369]. The highest alpha-word accuracy is for French, Spanish and Croatian, with results that were only 0.34 %, 0.29 %, and 0.56 % behind the state of the art, respectively. These languages have the smallest percentage of diacritic words (see Table 10). The lowest scores are recorded for Vietnamese and Latvian at 94.25 % and 96.33 %, respectively. It can also be noted that the Irish language, with the smallest dataset, has the highest standard deviation of 0.32 %.

Words were grouped by how many translation targets in the dictionary they have, and the ratio of ByT5 to dictionary error rates is shown in Table 12. The resulting values that are higher than 1 indicate that the Dictionary outperforms the ByT5 model. This is the case for all languages for a single dictionary candidate word and ByT5 model training up to $\times 1$ data. The first observation is that ByT5 scores improve for words, having more than one dictionary candidate. It even outperforms the dictionary method, except for the Croatian, Turkish, and Vietnamese languages. This shows that the Transformer model indeed leverages context understanding capabilities to disambiguate the correction choice. The second observation is that the Transformer model improves for all word groups and all languages with longer training. With eight times more training samples, it even surpasses the Dictionary model in a word group of one translation

Table 11. Alpha-word accuracy results (%) for the diacritics restoration task. Means and standard deviations are reported for three separate training runs with different initial model weights and dataset samplings trained for 524 288 sentences ($\times 1$) and a single run for eight times more ($\times 8$), cycling through the available training data (Table 10) as needed.

Language	Raw	Dict.	[369]	ByT5		Dict.+ByT5	
				$\times 1$	$\times 8$	$\times 1$	$\times 8$
Croatian	85.01	99.11	99.73	99.17 ± 0.06		99.42 ± 0.03	
Czech	49.71	95.67	99.22	98.01 ± 0.03		98.38 ± 0.04	
French	83.11	97.98	99.71	99.37 ± 0.04		99.49 ± 0.03	
Hungarian	50.34	96.22	99.41	98.42 ± 0.02	99.20	98.78 ± 0.01	99.25
Irish	69.97	96.65	98.88	98.14 ± 0.32		98.40 ± 0.16	
Latvian	50.14	90.59	98.63	96.33 ± 0.12	97.78	96.62 ± 0.09	97.66
Lithuanian	60.76	93.83	—	97.94 ± 0.19	99.07	98.18 ± 0.13	98.95
Polish	66.73	97.00	99.66	99.00 ± 0.03		99.16 ± 0.02	
Romanian	70.37	96.09	98.64	97.99 ± 0.03		98.17 ± 0.04	
Slovak	56.34	96.88	99.32	98.43 ± 0.06		98.77 ± 0.02	
Spanish	87.97	99.11	99.62	99.33 ± 0.04		99.43 ± 0.02	
Turkish	68.39	98.41	98.95	98.86 ± 0.04		99.03 ± 0.02	
Vietnamese	15.88	73.53	98.53	94.25 ± 0.07	97.53	94.29 ± 0.07	97.54
Average	62.67	94.70	99.19	98.10		98.32	

candidate for the Lithuanian and Latvian languages. The detailed alpha-word accuracy improvement during the diacritics restoration training is depicted in Figure 11.

The hybrid approach, the combination of Dictionary and the ByT5 model, improved the single ByT5 results by up to 0.37 %, on average, and allowed us to achieve the SOTA results for the Turkish language. The combination of dictionary and fine-tuned ByT5 show improvements for all languages over ByT5 trained for half a million sentences. This classic and Transformer approach combines the strengths of both methods. The dictionary method works best with less ambiguous words, which, for a large part, are rare. Although the Transformer model is less effective for rarer and less seen words during training, it can take advantage of its capabilities for frequent and ambiguous words, for which the dictionary fails. However, it can be observed that, with longer training, the pure ByT5 model can catch up to, or even surpass, the hybrid approach. Note that, at half the training, the standard deviation of the Turkish ratio is abnormally high. The reason is that one of the three ByT5 training runs temporarily failed. However, with further training, the run recovered to the same accuracy level as the other two. This is a good example of how different training dynamics can depend on different initial conditions and different data sampling.

Naturally, word frequencies follow a Zipf distribution. It is a heavy-tailed distribution, where there are a large number of words with low frequencies. This is a challenge for

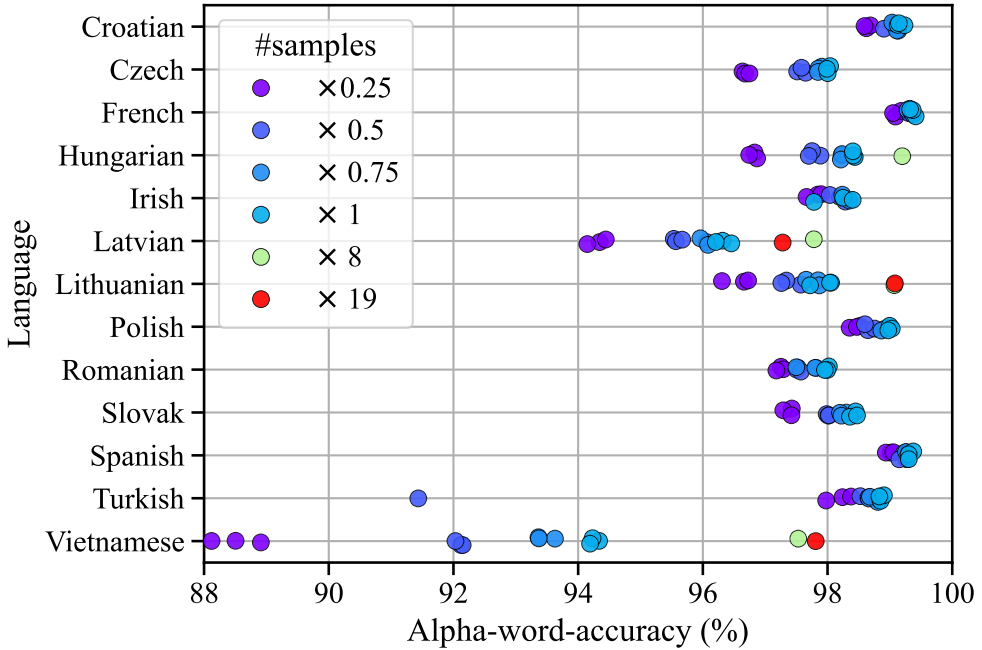


Figure 11. Alpha-word accuracy improvement during diacritics restoration training. Training steps of $\times 1$ corresponds to 2048×256 sentences for a given language. There is a visible outlier for the Turkish language at $\times 0.5$ training steps, but that model regained the accuracy later in training.

learning-based systems, which favor equal class distributions in training data. Words were grouped that were in the test set by their frequencies in the training set. The ByT5 results are shown in Table 13 for three such groups:

- Unseen: present in the test but not in train data;
- $[1, 100]$: words appearing in the training set from 1 to 100 times;
- $[101, 10\,000]$: words appearing in training set from 101 to 10 000 times.

A substantial part of the errors come from the words that are unseen during the training. Not including Irish and Vietnamese, this ranges from 13 % (French, Spanish) to 36 % for Slovak. The Vietnamese outlier of 1 % may be due to its linguistic nature, while the Irish outlier of 46 % is due to its very small dataset. In general, the smaller the dataset (Table 10), the more unseen or rare words and the associated errors are seen. The longer training of $\times 8$ pushes the error distribution to unseen words, as the more frequent words receive the greatest error reductions.

Similarly to the Dictionary method and the other classical methods, unseen data are also a significant source of errors for the Transformer model. However, different from the classical approaches is the Transformer model, which is based on neural networks,

Table 12. Alpha-word error ratio between the ByT5 and Dictionary methods for two word groups and models in different training stages. The values higher than 1 indicate that the Dictionary method restores diacritics better. The first word group corresponds to words with exactly one possible translation target, and the second word group corresponds to words with two translation targets. Groups are determined by the training set statistics, while results are reported on the testing set. $\times \{0.5, 1, 8\}$ shows results with training on the corresponding times more samples.

Language	One Dictionary Candidate			Two Dictionary Candidates		
	$\times 0.5$	$\times 1$	$\times 8$	$\times 0.5$	$\times 1$	$\times 8$
Croatian	6.37 ± 0.98	4.98 ± 0.52		1.18 ± 0.03	1.01 ± 0.06	
Czech	4.74 ± 0.19	3.53 ± 0.03		0.45 ± 0.02	0.37 ± 0.01	
French	5.29 ± 0.17	4.98 ± 0.48		0.31 ± 0.02	0.27 ± 0.01	
Hungarian	7.35 ± 0.48	4.37 ± 0.10	1.42	0.84 ± 0.03	0.62 ± 0.00	0.34
Irish	2.13 ± 0.21	2.27 ± 0.84		0.56 ± 0.03	0.57 ± 0.08	
Latvian	2.43 ± 0.17	1.77 ± 0.11	0.69	0.43 ± 0.00	0.37 ± 0.02	0.23
Lithuanian	2.61 ± 0.18	2.00 ± 0.36	0.58	0.34 ± 0.02	0.27 ± 0.01	0.12
Polish	4.06 ± 0.24	2.56 ± 0.15		0.30 ± 0.03	0.24 ± 0.01	
Romanian	3.66 ± 0.44	2.63 ± 0.12		0.82 ± 0.02	0.63 ± 0.02	
Slovak	4.29 ± 0.03	3.00 ± 0.21		0.54 ± 0.02	0.44 ± 0.01	
Spanish	5.4 ± 0.54	4.18 ± 0.57		0.95 ± 0.04	0.82 ± 0.03	
Turkish	10.5 ± 10.84	2.70 ± 0.24		3.83 ± 4.68	1.01 ± 0.03	
Vietnamese	2.6 ± 0.10	2.38 ± 0.20	1.31	1.52 ± 0.16	1.21 ± 0.04	0.32

and it can be generalized to unseen data. To investigate this generalization, words that were in the testing set and not in the training were filtered. The calculated percentages are shown in Table 14. If the correct model decisions for restoring words that should have diacritics and decisions for leaving the words as they are, because they should be without diacritics, are summed, ByT5 reaches more than 76 % alpha-word accuracy score for unseen words for all languages.

Transformer models depend on the amount of training data, and small sizes can hinder performance. Hungarian and Latvian languages, with a very similar percentage of diacritics (and, hence, the task difficulty), had a difference of four times between their dataset sizes. As a result, the achieved restoration score for Latvian was almost 2 % lower. On the other hand, the alpha-word accuracy of over 96 % and 98 % can still be reached for the Latvian and Irish languages, with dataset sizes of 5.5 M and 1.2 M words, respectively. This indicates a correlation between the difficulty of the task and the size of the dataset needed.

Overall, a 98.3 % average alpha-word accuracy (within 1 % of the state of the art of the time) was reached on the diacritic restoration task over 13 benchmark languages with a ByT5 universal byte-level Transformer model approach, a smaller training dataset (Wikipedia), and a much-reduced training time (Table 11). When the training time is limited, the model is slightly improved by the assistance of a simple statistical Unigram

Table 13. The distribution of diacritics restoration errors for the different frequencies of the words in the training dataset are shown in the first three columns. The intervals indicate the bounds of how many times the words in this group were encountered in the training dataset. $\times 8$ shows results with training on 8 times more samples. The last two columns indicate the percentages of how much of the training dataset was constituted by these word groups.

Language	Percentage of All Errors			% in Training	
	Unseen	[1, 100]	[101, 10 ⁴]	[1, 100]	[101, 10 ⁴]
Croatian	31.42 $\pm$ 1.65	54.23 $\pm$ 1.27	12.67 $\pm$ 1.45	11.79	51.02
Czech	20.42 $\pm$ 0.24	51.07 $\pm$ 1.31	26.43 $\pm$ 0.80	10.62	54.54
French	13.55 $\pm$ 0.88	42.29 $\pm$ 3.26	28.53 $\pm$ 1.66	2.06	37.53
Hungarian	26.78 $\pm$ 0.33	49.96 $\pm$ 0.59	19.67 $\pm$ 0.69	9.25	52.82
$\times 8$	35.97	43.45	17.64		
Irish	46.51 $\pm$ 6.04	35.37 $\pm$ 0.65	13.64 $\pm$ 3.17	23.53	45.03
Latvian	25.94 $\pm$ 0.95	49.87 $\pm$ 0.16	21.35 $\pm$ 0.77	22.20	56.23
$\times 8$	33.71	43.23	20.49		
Lithuanian	26.13 $\pm$ 0.60	47.93 $\pm$ 1.03	24.83 $\pm$ 1.11	16.41	63.61
$\times 8$	40.28	39.60	19.55		
Polish	18.31 $\pm$ 0.52	48.49 $\pm$ 0.89	29.78 $\pm$ 0.46	10.43	56.29
Romanian	16.06 $\pm$ 0.29	43.46 $\pm$ 0.96	27.83 $\pm$ 1.08	6.83	48.39
Slovak	36.68 $\pm$ 1.00	52.96 $\pm$ 0.29	10.01 $\pm$ 0.70	14.98	52.18
Spanish	13.22 $\pm$ 0.24	39.29 $\pm$ 1.58	28.74 $\pm$ 0.86	2.19	36.81
Turkish	24.11 $\pm$ 0.48	54.03 $\pm$ 1.13	21.07 $\pm$ 0.71	11.26	59.66
Vietnamese	1.48 $\pm$ 0.01	7.85 $\pm$ 0.19	58.40 $\pm$ 0.46	0.96	26.15
$\times 8$	3.37	10.69	55.43		

model (Dict.+ByT5). There is a solid indication, however, that longer training gets very close to the SOTA model, even without this assistance, and with the smaller dataset (Figure 11).

4.4.3.2 Diacritics and typos restoration

The increase in task complexity from diacritics-only restoration to simultaneous diacritic and typographic error corrections results in a decrease in alpha-word accuracy for all methods. As depicted in Table 15, the average raw score of uncorrected words lowered from 62.67 % to 46.91 %, and the average score of the Dictionary method from 94.70 % to 71.89 %. Correction results were also added, that were obtained with the open source Hunspell spellchecker [367] by replacing the words that it found to be incorrectly spelled with its first suggestion. The results indicate that it is barely better than the raw uncorrected sentences. It is also significantly worse than the Dictionary approach, which is specialized in restoring diacritics. Unlike classical baselines, Trans-

Table 14. The confusion matrix of the unseen word diacritics restoration performance by the ByT5 model. $\times 8$ shows results with training on 8 times more samples. Unseen words with and without diacritics are presented separately. The last column depicts the total number of unseen words for each language.

Language	With Diacritics, %		Without Diacritics, %		#Unseen
	Failed	Restored	Failed	Left Correct	
Croatian	6.8 ± 0.2	15.9 ± 0.2	4.3 ± 0.4	73.0 ± 0.4	12,147
Czech	16.4 ± 0.2	37.7 ± 0.2	5.0 ± 0.4	40.9 ± 0.4	9,398
French	10.1 ± 0.1	9.8 ± 0.1	4.6 ± 0.3	75.6 ± 0.3	3,794
Hungarian	10.1 ± 0.2	58.2 ± 0.2	2.3 ± 0.1	29.5 ± 0.1	16,350
$\times 8$	7.1	61.2	1.2	30.5	
Irish	12.1 ± 0.5	25.6 ± 0.5	5.3 ± 1.0	57.0 ± 1.0	29,470
Latvian	16.8 ± 0.7	39.9 ± 0.7	5.5 ± 0.6	37.9 ± 0.6	17,449
$\times 8$	13.6	43.1	3.9	39.4	
Lithuanian	8.1 ± 0.4	29.7 ± 0.4	4.4 ± 1.4	57.7 ± 1.4	15,547
$\times 8$	6.0	31.9	2.8	59.3	
Polish	7.0 ± 0.1	20.4 ± 0.1	2.3 ± 0.2	70.3 ± 0.2	9,461
Romanian	15.6 ± 0.5	15.1 ± 0.5	6.9 ± 0.9	62.4 ± 0.9	8,493
Slovak	14.4 ± 0.3	33.7 ± 0.3	4.9 ± 1.6	46.9 ± 1.6	13,357
Spanish	8.1 ± 0.2	11.6 ± 0.2	5.4 ± 1.0	75.0 ± 1.0	5,115
Turkish	7.0 ± 0.1	25.3 ± 0.1	3.7 ± 0.2	63.9 ± 0.2	9,594
Vietnamese	14.4 ± 0.3	1.7 ± 0.3	1.9 ± 0.4	82.1 ± 0.4	4,260
$\times 8$	13.1	2.9	2.7	81.2	

former-based methods had lower average score decreases, from 98.10 % to 90.26 % for ByT5, and from 98.32 % to 94.60 % for a combination of Dictionary and ByT5. Training models up to 19 times longer substantially improves performance. This reduces the gap between ByT5 and hybrid models for the Hungarian, Lithuanian, and Vietnamese languages from 6 %, 5.02 % and 0.52 % to 2.89 %, 2.54 % and 0.37 %, accordingly. The remaining gap indicates that there are still some rarer words that the Transformer model did not learn, but the Dictionary method can efficiently map. Languages with fewer diacritics, such as French and Spanish, have diminishing gains from longer training. Overall, longer training is a must for the more difficult tasks. Overall, a 94.6 % average alpha-word accuracy on the simultaneous diacritics restoration and typo correction tasks was achieved. This is a much harder task than a simple diacritics restoration, and it is problematic for the specialized systems. There is also a strong indication that longer training can significantly improve these results (Figure 12).

Table 15. Alpha-word accuracies for the simultaneous diacritics and typographic error corrections.

Language	Raw	Hunspell	Dict.	ByT5		Dict.+ByT5	
				#samples: ×1	×19	×1	×19
Croatian	64.05	66.43	74.06	90.27		96.71	
Czech	38.68	40.15	71.37	89.88		94.52	
French	60.81	64.94	70.87	93.45		96.52	
Hungarian	38.16	46.35	69.84	88.31	93.96	94.31	96.85
Irish	53.49	56.01	73.16	89.48		94.49	
Latvian	37.69	44.21	66.29	88.88		93.01	
Lithuanian	44.78	44.87	68.44	89.68	94.19	94.70	96.73
Polish	49.10	56.61	70.02	91.38		96.76	
Romanian	51.93	54.54	70.29	90.50		94.14	
Slovak	43.92	48.59	72.89	91.05		95.56	
Spanish	64.07	68.03	71.58	93.12		95.98	
Turkish	51.18	51.69	72.58	90.00		95.29	
Vietnamese	11.92	11.84	56.19	87.34	93.40	87.86	93.77
Average	46.91	50.33	71.89	90.26		94.60	

4.4.3.3 Grammatical error correction for Lithuanian

Below, the results for even more complex task of GEC for Lithuanian, which include many more error types, are presented. Two different models were trained – one with byte-level, and other – with subword level tokenization. Training dynamics are depicted in Figure 13. Only after 6 % of training, the ByT5 score $F_{0.5} = 0.85$ is already higher than the $F_{0.5} = 0.80$ T5 managed to reach after the full epoch. This shows that the ByT5 model is a much better choice for GEC. Although T5 with its learned tokenization is better aware of the possible character combinations, the finer input granularity is the leading factor. It helps the model to better grasp the input representation and thus perform the better output token classification during the generation.

The longer training is also beneficial. The performance is seen as steadily increasing during the fine-tuning and is expected to continue doing so for both models. The same results are indicated by the training loss. It is much lower for the ByT5, hence the model is better. It is also visible that it is much noisier, therefore it could also benefit from larger batch sizes (here, the batch size of 128 was used, while, for previous multilingual experiments, it was 256).

Synthesized errors were divided into several groups and corrupted the test set with each group separately from the others. The evaluation results for each group are presented in Table 16. It can be seen that the easiest task for both models was adding or deleting spaces, while the hardest task is correcting assimilation and gemination mistakes. This group may lag in performance due to the smaller abundance (only 2 % of samples) in the training data. The lack of samples for the specific type of error is harder to mitigate.

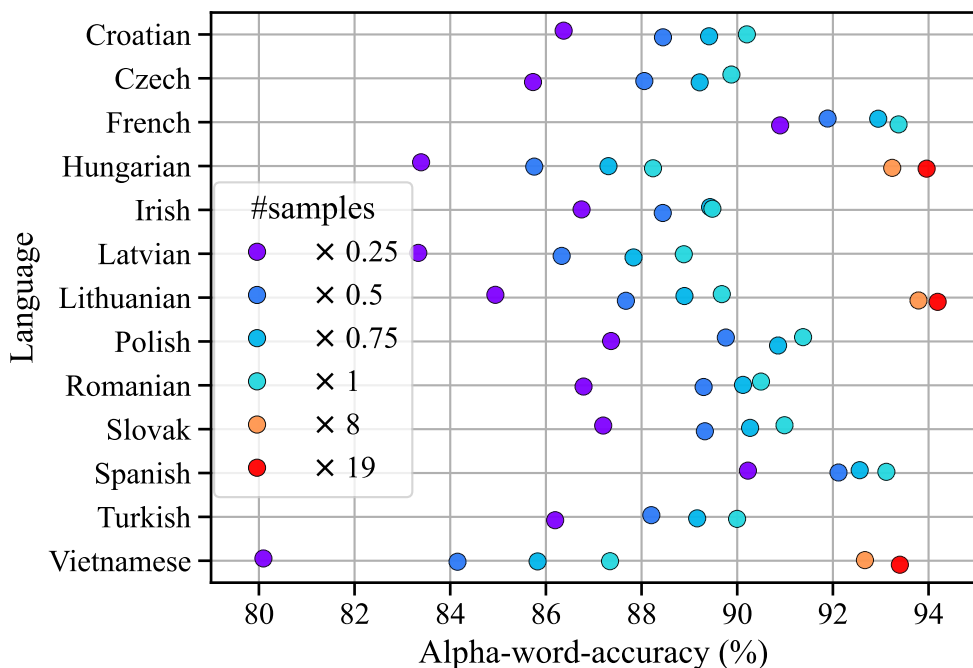


Figure 12. Alpha-word accuracy improvement during the simultaneous diacritics and typographical errors correction training. Training data of $\times 1$ corresponds to 2048×256 sentences for a given language.

Meanwhile, the other types of typographical, similarly sounding letters, upper/lower casing, and punctuation errors are abundant and have a large potential to improve with longer training.

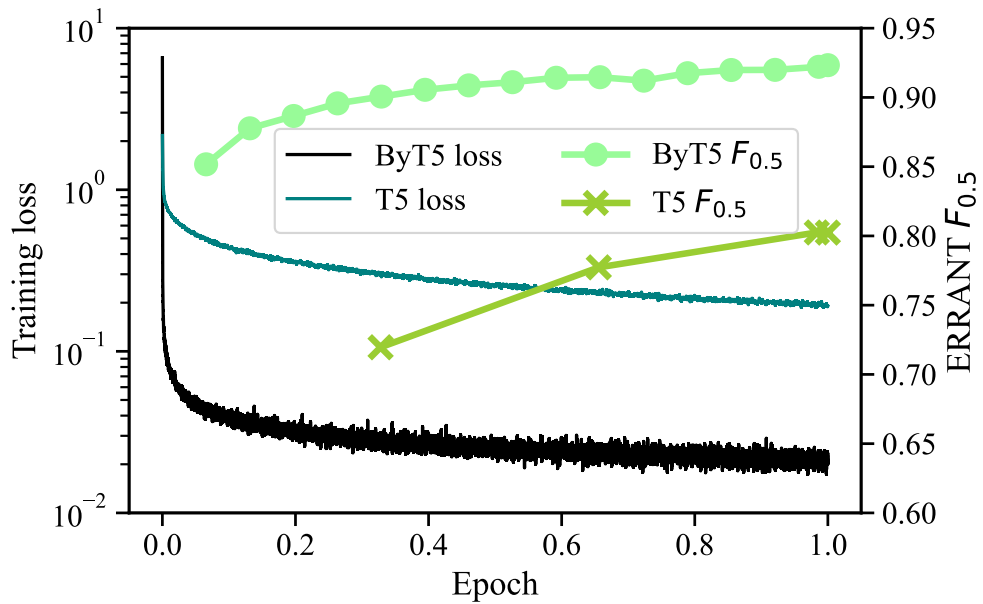


Figure 13. Training loss and F_{0.5} score for both T5 and ByT5 on the Lithuanian GEC task.

Table 16. Evaluation for the separate error categories with models trained for one epoch. Synthetic corruption was applied for the test set of ByT5 (a total of 2 155 samples) and T5 (a total of 2 099 samples) with each error group separately. Both ERRANT F_{0.5} score and the number of samples (#samples) affected and evaluated on, are shown.

Error group	ByT5		T5	
	F _{0.5}	#samples	F _{0.5}	#samples
Typographical	0.87	1 916	0.72	1 868
Punctuation	0.81	489	0.36	460
Similar sounding letters	0.88	1 115	0.55	1 143
Add/delete spaces	0.96	1 873	0.74	1 832
Assimilation/Gemination	0.79	56	0.30	43
Upper/Lower casing	0.86	785	0.47	781

4.5. Summarization and Grammatical Error Correction as Text Transformation

In previous sections, methods for abstractive Lithuanian news summarization and GEC were developed. In this section, they will be used as a first step in the text representation pipeline to improve the input text.

In early clustering experiments of Lithuanian news articles, it was noticed that using less tokens to average the final text representation led to a greater clustering performance. This could be explained by the more important words being used at the beginning of articles. Similarly to the beginning part of the text, the generated summary should capture the essential information for the whole text. Thus, the hypothesis of this dissertation is that using a summary instead of raw text should improve both textual and later numerical representation. Moreover, a shorter text is faster to process, takes less memory to store, and thus is more efficient.

One of the hardest domains for NLP is informal language. It is densely populated by OOV tokens, and many of those are just various typing errors. The experiments regarding GEC in the previous section showed that unseen tokens during the fine-tuning of the Transformer model are the main source of errors. This is even more obvious for other more traditional NLP methods. Therefore, the other hypothesis of this dissertation is that the use of GEC transformation of informal texts should improve both textual and later numerical representation. To validate this claim, an informal text task of Lithuanian hate speech detection by the authors of [374] was used.

4.5.1. Methodology

Clustering experiments mostly follow the methodology already described in Section 4.2. The main difference here is that different categories were used: world news, science, law, politics, weather, crime, and, education. For each experiment run, 125 articles were randomly sampled from each category.

The hate speech detection dataset was provided by the authors of [374]. It contains 21885 comments in train, 2736 in validation, and 2736 in test set splits. There are 3 classification labels: 56% non-hate, 29% offensive, and 15% hate. Classifier was trained only by using the train split, and the result are reported on the test split. Classification of Lithuanian hate speech was performed by using logistic regression classifier from the Python scikit-learn library [294].

Overall, 3 types of text vectorization methods were included: 1) average of 10th layer activations from BERT-type Transformer models; 2) FastText Lithuanian model; and 3) random embedding baseline, proposed in this dissertation. Similarly to the authors of [374], the cased version of multilingual BERT was used. For the random embeddings method, vectors of 4096 elements were generated following the same methodology as in Chapter 3.

4.5.1.1 Evaluation

Clustering was evaluated with the MCC score. Averages of 10 clustering experiments were reported for each setting, with different K-Means initializations and different input text sampling for each category. The hate speech classification task was evaluated by using weighted F1-score, following the authors of the dataset [374]. This metric calculates the F1-score for each class independently, and then averages them, weighting each class by its number of true instances (support). We also run a sanity check with random document-level vectors. It showed the lowest possible clustering MCC score of 0 and the lowest possible weighted F-score for hate speech detection of 0.38 (corresponding to 13 chance to guess the label).

4.5.2. Results

The text transformation impact for numerical representations results is shown in Table 17 for clustering and in Table 18 for hate speech classification. There are several findings in these results.

Table 17. Effect of summarization for Lithuanian news clustering.

Encoder	Avg. MCC	
	Raw text	Summary
LitLAT BERT	0.22	0.20
+ z-score	0.30	0.23
BERT	0.16	0.13
+ z-score	0.17	0.14
FastText	0.24	0.16
+ z-score	0.26	0.20
RE _{LitLAT BERT}	0.05	0.04
+ z-score	0.06	0.04
RE _{T5 LT}	0.05	0.03
+ z-score	0.06	0.03
RE _{BERT}	0.08	0.03
+ z-score	0.08	0.03

The conclusions of the previous Chapter 3 are repeated. First, the post-processing is shown to be beneficial for the unsupervised clustering task, with the z-score post-processing achieving the highest MCC score of clustering for LitLat BERT of 0.30, thus surpassing the non post-processing configuration score of 0.22. Second, the random embedding baseline shows strong results against advanced pre-trained BERT model

Table 18. Effect of GEC for Lithuanian hate speech detection task.

Encoder	Weighted F1-score	
	Raw text	After GEC
Random document vectors	0.38	0.38
FastText	0.65	0.65
RE _{LitLAT BERT}	0.65	0.64
RE _{T5 LT}	0.65	0.64
RE _{BERT}	0.65	0.65
LitLAT BERT	0.72	0.71
fine-tuned [374]	0.76	
BERT	0.65	0.65
fine-tuned [374]	0.73	

representations, both achieving a weighted F1-score for hate speech detection of 0.65. Training and inference times for Lithuanian text transformation models are shown in Table 19. The training times are very long, covering hundreds of hours. The inference time of the summarization model in clustering task texts was a fraction of an hour, while the GEC model took multiple hours to process hate speech detection task texts. The longer time of ByT5 inference is attributed to the finer tokenization producing longer sequences to process.

Table 19. Training and inference times for Lithuanian text transformation models. Training was performed on GeForce RTX 2080 Ti, while inference was processed on RTX 4090 GPU.

Model	#parameters	Task	Training	Inference	
				Speed	Memory
T5	296.93M	Summarization	500 h	38 samples/s	5.87 GB
ByT5	300.77M	GEC	100 h	7 samples/s	6 GB

Both research hypotheses on text transformations that improve numerical representation must be rejected. For all encoders in the Lithuanian news clustering task, the MCC scores even decreased after text summarization, by 0.02 to 0.05 points. The weighted F1-scores for Lithuanian hate speech detections were also not improved, with the score decreasing by up to 0.01. This indicates that improvements in the text quality do not necessarily transfer into the numerical representation. Either such text transformations are not helpful, or this particular use of Transformer models is not optimized towards

the given task goals.

4.6. Conclusions

This chapter extended the text representation pipeline to a mid-resource and morphologically rich language – Lithuanian – and investigated how two text-level Transformer-based transformations affect both the textual quality and downstream numerical representations. Two Transformer models were fine-tuned – T5 for abstractive Lithuanian news summarization and ByT5 for GEC. The results allow for several key conclusions.

Transformer models are not plug-and-play in mid-resource settings. Early Lithuanian news clustering experiments demonstrated that multilingual pre-trained Transformer models do not automatically outperform classical approaches. The average multilingual BERT and XLM-R representations were inferior to the GLoVe vectors weighted by TF-IDF and substantially worse than PV-DBOW. Only after additional masked language modeling fine-tuning did BERT surpass static embeddings, yet it still failed to outperform term:pvd bow. These findings highlight that, in mid-resource settings, data availability, and compute constraints critically limit the effectiveness of direct Transformer adaptation.

Input length and information locality matter more in long-document settings. Unlike shorter English benchmark tasks, Lithuanian news clustering revealed that feeding the full 512 tokens is suboptimal. Averaging only the initial portion of contextualized representations yielded better clustering results, thus indicating that the informative content is concentrated in the beginning of news articles. This demonstrated that the document structure must be considered when designing representation extraction strategies for longer texts.

Language-specific tokenization is crucial for efficient generative modeling. In abstractive summarization, replacing the English tokenizer of T5 with a Lithuanian SentencePiece tokenizer drastically reduced sequence lengths (from 1448 to 530 tokens on average), significantly improving computational efficiency. This confirms that sub-word vocabularies trained on morphologically rich languages are essential for effective Transformer adaptation. The multilingual mT5 model, despite the same size, was considerably slower to train due to longer token sequences and vocabulary overhead, which demonstrates the importance of tokenization.

Abstractive summarization is feasible but difficult to evaluate automatically. The Lithuanian T5 summarization model achieved up to 16.3% ROUGE-2 with beam search decoding. However, automatic ROUGE evaluation proved problematic: extractive summaries were often rewarded, while semantically accurate but lexically different summaries were penalized. High variance and factual inconsistencies in the generated outputs further emphasized the limitations of n-gram overlap metrics for morpho-

logically rich languages. This highlights the need for improved automatic evaluation frameworks beyond lexical overlap.

Byte-level modeling is highly effective for grammatical error correction. The byte-level ByT5 model achieved near SOTA diacritics restoration performance (98.3% average alpha-word accuracy across 13 languages) by using a smaller dataset and a reduced training time. Importantly, it demonstrated strong contextual disambiguation capabilities for words with multiple dictionary candidates, outperforming unigram-based dictionary methods in ambiguous cases. This confirms that contextual Transformer representations provide a significant advantage over purely statistical lexical approaches.

Hybrid classical–neural approaches are complementary. Combining dictionary-based correction with ByT5 yielded consistent improvements under limited training regimes. The dictionary method handled rare, unambiguous words effectively, while the Transformer excelled in frequent and context-dependent cases. However, with sufficient training, the pure Transformer model closed or eliminated this gap. This suggests that hybrid systems are especially beneficial when computational or data resources are constrained.

Data scale and Zipfian distribution strongly influence performance. A substantial portion of remaining correction errors originated from unseen or rare words, thus reflecting the heavy-tailed Zipfian nature of language. Longer training disproportionately improved frequent word corrections, shifting the error mass toward unseen vocabulary. Nevertheless, even for unseen words, ByT5 achieved over 76% alpha-word accuracy across languages, demonstrating meaningful generalization beyond memorization.

Byte-level tokenization is particularly suitable for morphologically rich languages. Lithuanian-specific grammatical error experiments showed that finer-grained byte-level modeling is better suited for handling diacritics, assimilation, gemination, spacing, and casing errors than subword tokenization. Although computationally more expensive, byte-level modeling substantially improved robustness against diverse error types, which are especially prevalent in morphologically complex languages.

Text-level transformations did not improve representation pipelines indirectly. Unlike the first chapter, where improvements were achieved directly at the embedding level, this chapter demonstrated attempts to enhance the textual input itself, in particular, by using trained Lithuanian abstractive news summarization and GEC models. Unlike the text quality, numerical representations for the news clustering and Lithuanian hate speech detection task were not improved.

Overall conclusion. This chapter demonstrates that, in mid-resource and morphologically rich language settings, Transformer-based methods require careful adaptation and cannot be assumed to outperform classical approaches by default. While language-specific tokenization, byte-level modeling, and targeted fine-tuning enable effective generative tasks such as summarization and GEC, these improvements in the textual quality do not automatically translate into gains in downstream numerical representation tasks. The results show that the tokenization strategy, data scale, and document structure play a decisive role in model effectiveness, and that hybrid classical–neural approaches remain competitive under constrained resources. Most importantly, the findings indicate that enhancing the textual surface quality alone is insufficient to reshape embedding geometry for tasks such as clustering or hate speech detection. Therefore, in mid-resource environments, improvements in the representation quality require direct optimization of embedding-level properties rather than relying solely on generative text transformations.

5. CONCLUSIONS

A comprehensive literature review has been performed. Current SOTA models for the universal text representation are the pre-trained deep neural networks of Transformer architecture fine-tuned by using contrastive learning setting on NLI type datasets, use subword level tokenization, and their output representations are usually evaluated by semantic textual similarity and transfer learning tasks. There are no competitive methods of using only the pre-trained model features. Also, since such models require a lot of data and compute resources, their application in less popular languages, such as Lithuanian, is limited.

Several methods have been developed and investigated for text representation extraction from pre-trained Transformer models on multiple STS, short text clustering, and transfer learning tasks:

1. Token-layer aggregation and geometric post-processing combinations were found that substantially improve the embedding quality, particularly, for similarity-based tasks. These techniques improved the features of the BERT model for STS on average by 8.5 % of Spearman correlation and short text clustering by 6.4 % average accuracy, while having minimal effect for supervised classification tasks.
2. The proposed static Avg text representation method of simply averaged BERT tokens outperforms other more sophisticated Word2vec-style models. With representation shaping techniques, it reached an average of 69.5 % Spearman correlation for STS tasks, 62.2 % accuracy for short text clustering, and 77.3 % accuracy for classification.
3. The proposed adequately processed random embeddings model is a good baseline for text-level tasks. For STS tasks, it scores an average of 66.4 % Spearman correlation, for short text clustering – 49.5 % accuracy and is able to reach 75.9 % classification accuracy. The introduction of a shaped random embedding baseline revealed that embedding geometry alone explains a considerable share of performance traditionally attributed to pretrained contextual models. This finding provides new insights into the structural properties of Transformer embedding spaces and helps disentangle training-induced knowledge from geometric effects.

For text-level transformation methods, the first Lithuanian abstractive summarization and universal GEC models have been created. Investigation of these methods has revealed the following:

1. Byte-level Transformer is particularly well suited for morphologically complex languages. Extensive multilingual GEC experiments demonstrated that vocabulary-free architectures can achieve near SOTA diacritics restoration performance with

limited resources. A 98.3 % average alpha-word accuracy (within 1 % of the state of the art) on the diacritic restoration task was achieved over 13 benchmark languages (including Lithuanian) with a ByT5 universal byte-level Transformer model approach, a smaller training dataset (Wikipedia), and a significantly reduced training time.

2. Transformer-based text-level transformation of abstractive Lithuanian news articles summarization reached 16.3 % ROUGE-2 and, for Lithuanian GEC, a score of 0.92 ERRANT $F_{0.5}$ was achieved.
3. In mid-resource and morphologically rich language settings, Transformer models are not inherently superior to classical approaches. Their effectiveness depends critically on the tokenization strategy, data scale, and the adaptation procedure.
4. Text-level transformations such as summarization and GEC improved the textual surface quality but did not automatically enhance downstream numerical representations for Lithuanian news clustering and hate speech detection tasks. Therefore, improvement of the embedding quality requires direct manipulation of the representation geometry or contrastive fine-tuning on dedicated datasets.

SANTRAUKA

Įvadas

Mes gyvename informacijos amžiuje. Skaitmeninės tekstinės informacijos kiekiai auga nepaliojiamu greičiu. Milijardai svetainių, naujienų ir socialinės medijos pranešimų, laiškų, mokslinių straipsnių ir dar daugybė kitų skaitmeninio žodžio formų yra sugeneruojama kiekvieną dieną. Vienas žmogus yra pajėgus aprėpti tik mažą dalelytę šio milžiniško informacijos kiekio. Dabar kaip niekada svarbu pasitelkti į pagalbą automatinius tokios informacijos supratimo ir apdorojimo būdus. Tai ypač svarbu paieškos, automatinio santraukų rašymo, melagienų ir žalingo turinio atpažinimo sistemoms. Deja, tai nėra taip paprasta.

Mūsų kasdienybėje naudojami kompiuteriai nėra suprojektuoti suprasti tekstinę informaciją. Už teksto simbolio informacijos kompiuterio atmintyje slepiasi tik to simbolio pavadinimas be jokios prasmės informacijos. Dėl to naudojami specialūs metodai, kurie atpažintų ryšius tarp tekstų. Klasikiniai, tokie kaip TF-IDF, n-grams, sugeba nustatyti tam tikrus paviršutinius ryšius. Deja, jie nėra tinkami sudėtingesnėms užduotims, ir tai riboja jų pritaikomumą. Proveržį šioje srityje galima sieti su Word2vec metodu [24, 25], kurio mokymas leido atlikti aritmetines operacijas žodžių prasmės lygmeniu: $\text{king} - \text{man} + \text{woman} \approx \text{queen}$. Daugybė tolimesnių tyrimų tobulino šias žodžių reprezentacijas, pradedant panašiais besluoksniais, rekurentiniais ir galiausiai, dabar, transformerio architektūros dirbtiniais neuroniniais tinklais.

Būtent pastarieji, transformeriai, ir yra nagrinėjami šiame darbe. Disertacijoje siekiama išgauti tokias reprezentacijas, kurios pasižymėtų panašiomis savybėmis, kaip ir Word2vec, tačiau būtų ne žodžių, o visų sakinių, pastraipų ar net dar ilgesnio teksto lygio. Po pradinio mokymo (angl. *pre-training*) transformerių modeliai negali sukurti kokybiškų viso teksto reprezentacijų, nors mokant naudoti milžiniški duomenų ir skaičiavimų resursai. Tik po koregavimo (angl. *fine-tuning*) transformeriai tapo pranašesni už kitus metodus. Šioje disertacijoje pasiūlyti metodai, kurie įgalina išgauti geresnes transformerių reprezentacijas net ir po pradinio mokymo. Tai aktualu turint mažai ar visai neturint tikslinės srities duomenų ar nenorint jų naudoti mokant dėl jų jautrumo. Šiame darbe taip pat tiriamas transformerių modelių koregavimas lietuvių kalbai. Dėl gerokai mažesnio populiarumo nei anglų kalba tenka susidurti su ribotais teksto, egzistuojančių modelių bei įvertinimo užduočių ištekliais. Nepaisant to, lietuvių kalbos tekstams atlikti skirtingi mokymai ir parinkti automatiniai įvertinimai leido palyginti skirtingai išgautų reprezentacijų kokybę lemiančius faktorius.

Mokslinio darbo tikslai ir uždaviniai

Disertacijos tikslas yra ištirti ir pagerinti anglų ir lietuvių kalbų teksto reprezentacijas, išgaunamas iš pradinio mokymo transformerių modelių. Šis tikslas bus pasiektas įvykdant šiuos uždavinius:

1. Apžvelgti išmokstamų teksto reprezentacijų metodus ir jų įvertinimo būdus.
2. Ištirti ir sukurti metodus pagerintoms reprezentacijoms išgauti iš pradinio mokymo transformerių modelių, įtraukiant žymenų ir modelio sluoksnių agregacijas bei vėlesnio reprezentacijų apdorojimo būdus.
3. Įgyvendinti ir ištirti papildomas teksto transformacijas teksto vektorizavimo užduočiai lietuvių kalbai: transformerių modeliais gaunamas abstrakčias teksto santraukas bei gramatinių klaidų taisymus.

Pagrindiniai tyrimo klausimai, į kuriuos bus siekiama atsakyti šiame darbe, yra šie:

1. Kaip ir kiek pirminio mokymo transformerių modelių reprezentacijos gali būti pagerintos?
2. Kaip efektyviai transformerių modeliai gali atlikti teksto lygio transformacijas ir pagerinti tiek tekstines, tiek skaitines reprezentacijas?

Mokslinis naujumas

1. Sukurtas metodas, grįstas žymenų (angl. *token*) ir transformerio sluoksnių agregavimo ir reprezentacijų vėlesnio apdorojimo (angl. *post-processing*) technikomis. Nustatytos minėtų parametrų kombinacijos, kurios leidžia panaudoti transformerio modelio aktyvacijas be papildomo mokymo išgaunant geresnį neprižiūrimų užduočių atlikimą.
2. Šių technikų ir atsitiktinių vektorių metodo pagrindu sukurtas palyginamasis bazinis modelis. Jį taikant galima atskirti papildomų technikų ir vektorių kontekstualizacijos įtaką teksto reprezentacijų kokybei.
3. Sukurti pirmi transformerių modelių taikymai lietuvių kalbai naujienų straipsnių abstrakčioms santraukoms sudaryti ir rašybos klaidoms taisyti.
4. Diakritiniams ženklams atkurti ir bendram diakritinių ženklų ir teksto įvedimo klaidų taisymui panaudotas sekos-į-seką (angl. *sequence-to-sequence*) baido lygio transformerio modelis ByT5 [3] 13-ai skirtingų kalbų su aukštu atkūrimo tikslumu ir daug mažesniu mokymo tekstynu bei trumpesniu mokymo laiku.

Literatūros apžvalga

Teksto reprezentacijos modeliai

Klasikiniu teksto reprezentacijos metodu galima laikyti žodžių maišo (angl. BoW) modelį [31]. Teksto reprezentaciją BoW sudaro tiesiog tame tekste esančių žodžių statistika, paprasčiausiu atveju – pasikartojimų skaičiai. Nors tokį modelį labai lengva interpretuoti, jis neišsaugo žodžių eiliškumo informacijos, dėl paprastai didelio žodyno jo reprezentacijos yra labai išsisklaidžiusios – didžiąją dalį vektorių sudaro nulinės

reikšmės. Dėl to naudojama daugybė išankstinio apdorojimo būdų: žodžiai kamienizuojami, lemuojami, išmetami ypač dažni ir ypač reti. Dėl žodžių dažnio atvirkščio proporcingumo jų eilei pagal dažnumą [32] reprezentacijos atitinkamai normalizuojamos (TF-IDF metodas), įvertinamas dokumento ilgio ir terminų svorio įsisotinimas (BM25 [33] metodas). Taikomi dimensiškumo sumažinimo metodai iš žodžių požymių į bendresnius, temų požymius, naudojant tokius metodus, kaip PCA [38, 39], LSA [41], NMF [42, 43], PLSA [44] ar LDA [45, 46]. Visi šie papildomi metodai prideda sudėtingumo, reikalauja individualių sprendimų ir su jais susijusių parametrų parinkimo ir galiausiai vis tiek tinka tik paprastoms ir nesudėtingoms teksto užduotims spręsti. Word2vec modelis [24, 25] žymi perėjimą nuo statistinių prie išmokstamų teksto reprezentacijų. Šis modelis parodė, kad galimos apytikslės aritmetinės žodžių vektorių operacijos, atitinkančios semantinius šių žodžių ryšius. Pavyzdžiui, iš žodžio „karalius“ vektoriiaus atėmus „vyras“ ir pridėjus „moteris“ vektorių, gaunama reprezentacija, kuriai artimiausias vektorius yra žodžio „karalienė“ [47]. Vėlesnis GLoVe modelis [8] sujungė globalią klasikinę žodžių pasikartojimo informaciją, o fastText [50] modelis žymenims (angl. *token*) naudojo trumpesnes simbolių dalis, taip geriau atsižvelgiant į morfologiją ir sukuriant reprezentacijas nežinomiems žodžiams. Doc2vec [6] buvo pasiūlyta traktuoti visą tekstą kaip atskirą žodį, taip sugeneruojant jam analogišką reprezentaciją. Po to vykę patobulinimai, Doc2VecC [54], Sent2Vec [55] bei FastSent [56] pasiūlė efektyvesnius sprendimus, kaip iš pavienių žymenų reprezentacijų išgauti viso dokumento vektorius. Word2vec priskiriami metodai išsprendė BoW metodo labai ilgų ir beveik tuščių vektorių problemą. Vis tik išryškėjo kiti trūkumai. Vienas svarbesnių – tie patys žodžiai, vartojami skirtinguose kontekstuose skirtingomis prasmėmis, vis tiek turėjo tą patį vektorius.

Gan praktiškas sprendimas teksto reprezentacijoms yra rekurentiniai neuroniniai tinklai RNN [57]. Kaip ir natūrali kalba, jų būsenos atnaujinamos viena po kitos, priklauso nuo prieš tai buvusiųjų, taip sukoncentruojant viso teksto reprezentaciją į paskutinę RNN būseną. Deja, pakankamas informacijos išlikimas iki paskutinio žingsnio reikalauja papildomų metodų: LSTM tipo RNN su atminties galimybėmis [59], dvikrypčio informacijos apdorojimo [61] ar dėmesio mechanizmo [62], kuris vietoj prieš tai buvusios būsenos naudoja svertinę visų praėjusių žingsnių būsenų sumą. Žymesni RNN tipo teksto reprezentacijų modeliai vis tik rėmėsi informacijos sukoncentravimu į paskutinę būseną, su kuria buvo atliekamas šalia esančių sakinių generavimas [18], klasifikacija [64, 67], rikiavimas [65], pirminio teksto atkūrimas [56] ar vertimas į kitą kalbą [70]. Vienas sėkmingesnių, InferSent [68] modelis, buvo mokytas su SNLI tekstyno [19] duomenimis, modeliuojančiais tekstų panašumą ar skirtumą. Rekurentiniai tinklai taip pat yra vienas iš būdų, kaip agreguoti daugelio žodžių vektorius į kontekstinę informaciją, pridedant ją prie Word2vec tipo vektorių [4, 71] ar visiškai juos pakeičiant [7]. Dabartinis, moderniausias transformerio modelis [83] įgalino didžiulį kokybinį teksto užduočių atlikimo šuolį. Jis perėmė dėmesio mechanizmą [62] iš rekurentinių tinklų, tačiau visai pašalino jų nuoseklumą ir skaičiavimų priklausymą nuo praėjusios būsenos. Kitaip nei RNN, skaičiavimų apimtis kvadratiškai priklauso ne nuo žodyno, o nuo

teksto ilgio. Transformerio generuojami žymenų vektoriai jau po pirmojo sluoksnio yra įgavę visų kitų žymenų kontekstinę informaciją. Galiausiai, transformerių modeliai paprasčiau didinami ir lengviau mokomi su didesniais tekstynais dėl skaičiavimų lygiagretumo. Visos šios savybės leido transformerių modelius efektyviai pritaikyti sudėtingesnėms užduotims, išnaudojant didelius prieinamų duomenų ir skaičiavimų resursų kiekius. Pirmas žymus transformerio modelis, kurio pradinio mokymo reprezentacijos sulaukė plataus pritaikymo, yra BERT [2]. Tai buvo vienas pirmųjų sėkmingų perkeliamų mokymų (angl. *transfer learning*) teksto sričiai. Pradinio mokymo metu BERT modelis mokomas universalios teksto supratimo užduotimi – paslėptų žymenų atspėjimu naudojant didelės apimties tekstynus ir skaičiavimo resursus. Vėliau, taikant modelį konkrečiai užduočiai, mokoma jau nuo šių svorių, pritaikant juos trumpo ir mažai resursų reikalaujančio koregavimo mokymo metu. Dėl BERT ir transformerio architektūros atsirado daugybė naujų galimybių taikyti šiuos sudėtingesnius modelius, labai išpopuliarėjo dirbtinis intelektas.

Pradinio mokymo BERT teksto reprezentacijos yra gana žemos kokybės [15, 98, 99, 100]. Tačiau po nesudėtingo koregavimo (pradedant nuo pradinio mokymo svorių) BERT modelis pirmavo daugelyje teksto užduočių rinkinių, aplenkdamas visus ankstesnius metodus. Šioje disertacijoje ir koncentruojamasi kaip tik į šią problemą – kaip iš jau pradinio transformerio modelio išgauti kokybišką teksto reprezentaciją, išvengiant specifinio koregavimo mokymo žingsnio.

Skirtingai nei šiame darbe, dauguma kitų autorių taiko antrąjį žingsnį – koregavimo mokymą. Vienas toks pirmųjų taikymų buvo SentenceBERT [15] modelis, analogiškai kaip [68], mokytas su teksto panašumo modeliavimo tekstynų rinkiniais. Vėliau išpopuliarėjo kontrastinis mokymas, kuris šią užduotį iš klasifikavimo perkėlė tiesiai į reprezentacijų formavimą mokymo metu – tos pačios semantinės prasmės reprezentacijų vektoriai artinami, o skirtingų – tolinami. Priklausomai nuo to, koku būdu parenkami panašių ir priešingų tekstų pavyzdžiai, priklauso mokyto modelio reprezentacijų kokybė.

Skaitinių reprezentacijų pagerinimas

Jau nuo Word2vec pasirodymo laikų buvo tiriama, kokie būdai yra optimalūs suagreguoti visus sakinio žodžių vektorius į vieną maksimaliai reprezentatyvų. Dažnai gan efektyviai naudotas paprastas vidurkio metodas tiek reprezentacijai išgauti [190, 191, 192, 15], tiek kaip tarpinis veiksmas besimokančių modelių viduje [198, 199, 200, 201, 55, 202, 203, 204]. Tačiau, kaip ir žodžių maišo metodas, jis turėjo trūkumų. Dėl žodžių eiliškumo nepaisymo prastesni rezultatai pastebėti tada, kai užduotys reikalauja sintaksės informacijos [198], dažnesni žodžiai yra labiau išreikšti nei reikšmingi [197]. Ne visi žodžiai tekste yra vienodai svarbūs. Netikėtas įvykis, pavyzdžiui, žodis sakinyje, turi didesnį kiekį informacijos nei tas, kuris buvo tikėtinas [209]. Taigi tokių žodžių vektoriai turėtų turėti didesnę indėlį vidurkinant bendrą teksto reprezentaciją. Ankstesniuose darbuose svoriai tam tikriems žymenims priskirti taisyklėmis grįžtais metodais [189], taikant TF-IDF [211]. Kiek žymesnis yra SIF metodas [16], kuriuo

dažniausių žodžių svoriai sumažinami tolygiai. Vienu metu jis net buvo tapęs geriausiu metodu panašumo užduotims [197] bei patobulintas į uSIF [212], nereikalaujanti papildomų parametrų nustatymo. Rerezentacijos taip pat gali būti pagerintos išmetant neinformatyvius žodžius [191, 97, 213] ar apsiribojant trumpesnėmis teksto iškarpomis su svarbiausia informacija [214]. Kituose darbuose išvedami gan sudėtingi svoriai, grįsti dėmesio mechanizmu [200], kiekvieno žodžio naujumu ar priešingumu visiems kitiems [215, 214, 216], ar pagal tai, kaip sparčiai rerezentacijos keičiasi per modelio sluoksnius [216]. Taip pat pastebėta, kad žodžių vektorių ilgiai turi didelį išsibarsčymą [217, 218], todėl vidurkinant tai gali veikti kaip automatinis žodžio svarbumo svoris [26]. Galiausiai, vidurkis gali būti traktuojamas kaip tam tikras apibendrintojo vidurkio atvejis su pirmuoju laipsniu [375, 250], taip išnaudojant ir kitų tipų vidurkių agregavimą [251].

Rerezentacijoms ištraukti iš tranformerio modelio galima panaudoti ir tam tikras tik šiam modeliui būdingas ypatybes. Štai BERT modelis turi specialius [CLS], [MASK] ir [SEP] žymenis, naudotas atitinkamai pradinio mokymo užduotyje norint klasifikuoti sakinį, identifikuoti paslėptą žymenį bei atskirti du sakinius. Nors [CLS] žymuo turėtų apimti visą pradinio teksto rerezentaciją, kaip buvo manyta jo sumanytojų [2], vėlesni darbai parodė, kad jis nėra veiksmingas [233, 99, 234]. Tačiau [MASK] žymuo buvo sėkmingai panaudotas kartu su užklausų formulavimo (angl. *prompting*) technika [95]. Vietoj papildomo koregavimo mokymo tekstas įterpiamas į kitą tekstą, kurio paskirtis – modelį priversti sukoncentruoti pirmojo teksto rerezentaciją į specifinį vektorių [213]. Toks sprendimas gerokai pagerino semantinio teksto panašumo užduočių atlikimą.

Kita rerezentacijų ašis yra skirtingi modelio sluoksniai, iš kurių gali būti naudojamos teksto sužadintos aktyvacijos. Manoma, kad, informacijai keliaujant per sluoksnius, maži, bet svarbūs požymiai yra vis labiau išryškunami [198]. Taigi pirmieji sluoksniai labiau susiję su fundamentine ir žemesnio lygio informacija [235]. Viduriniai sluoksniai yra labiausiai universalūs pakartotiniam panaudojimui [173] ir talpina pačią svarbiausią informaciją [171]. Paskutiniai yra sukoncentruoti į kontekstą [240] ir mokymo užduotį [235, 242], todėl jų rerezentacijos yra gana prastos kokybės [122], tačiau gali būti pagerintos naudojant vėlesnio apdorojimo (angl. *post-processing*) metodus [243, 244]. Neprižiūrimoms teksto rerezentacijų užduotims pasiteisino pirmų ir paskutinių BERT sluoksnių rerezentacijų vidurkis [99].

Daugelis darbų, siekdami pagerinti teksto rerezentacijų kokybę, matuoja jų izotropiškumą ir siekia kuo didesnių jo verčių. Žodžių vektorių anizotropiškumas buvo žinomas jau prieš tranformerių modelių išpopuliarėjimą [1]. Buvo pastebėta, kad visų vektorių suma nelygi nuliui, o pirmosios principinės komponentės išsaugo žodžių dažnio informaciją. ABTT metodas (liet. viskas, išskyrus viršus) pašalina šias komponentes ir leido pagerinti žodžių vektorius [1]. Taip pat pastebėtas ir tranformerių modelių vektorių anizotropiškumas [255, 83, 240]. Visos atsitiktinai parinktų žodžių rerezentacijos turėjo didelį panašumą, nors kosinuso panašumas turėtų būti lygus 0. Taip pat pastebėta, kad visų žymenų rerezentacijos užima siaurą kūgio formos erdvę. Pastebėta, kad anizotropiškumas glaudžiai susijęs su žymenų dažniu [258, 259, 260], o siauro

kūgio pavidalo priežastis yra mokymo metu skirtingai dažnai veikiamos dažnų ir re-
tų žodžių reprezentacijos [255, 259]. Nors patogus, tačiau tiesioginis izotropiškumo
optimizavimas yra kritikuojamas [259], nes jis neigiamai koreliuoja su klasifikavimo
užduočių įverčiais [266] bei yra dažnai netiksliai išmatuojamas [253]. Galiausiai, nors
globaliai reprezentacijos ir yra anizotropinės, tam tikruose klasteriuose jos vis tiek yra
izotropinės [267], taigi svarbiau yra optimizuoti konkrečių užduočių atlikimą nei aklaui
sekti erdvės izotropiškumą.

Dėl reprezentacijų netolygaus ar tiesiog nepalankaus išsidėstymo erdvėje joms taiko-
mi vėlesnio apdorojimo metodai. Iš paprastesnių galima paminėti centravimą [259],
standartizavimą [243, 244, 135], principinių komponentių dekoreliavimą [98, 99, 17].
Sudėtingesni metodai šalina pirmąsias principines komponentes globaliai [16, 1, 271,
272] ar lokaliai [268, 260]. Tokie metodai leidžia pagerinti semantinio tekstinio pana-
šumo užduočių atlikimą.

Lietuvių kalbos resursai

Svarbi transformerių modelių sėkmės priežastis yra dideli mokymo duomenų kiekiai.
Populiarusis BERT modelis [2] buvo mokytas naudojant 3,3 mlrd. žymenų tekstyną,
sudarytą vien iš Vikipedijos ir knygų [297]. Šių laikų DeepSeek-V3 modelis [84] jau
naudoja 14,8 trilijonus žymenų. Pagrindinis pastarojo duomenų šaltinis yra internetas.
Common Crawl [325] duomenų talpykla saugo viešai prieinamus iš interneto surink-
tus (angl. *web-scraping*) puslapius, kiekvieną mėnesį pasipildanti po 5-iais mlrd. naujų.
Keletas iš šio šaltinio pagal kalbą atrinktų ir nufiltruotų mažų rinkinių: OSCAR [326],
mC4 [11] ir CulturaX [327] gali būti naudojami įvairių kalbų modeliams mokytį. Ta-
čiau iš interneto paimti tekstai nėra geros kokybės ir reikalauja atitinkamo filtravimo
[329, 330], skirtingų šaltinių parinkimo [331]. Kokybiški laisvai prieinami tekstai lie-
tuvų kalbai, tokie kaip ALKSNIS [332] ir MATAS [334], yra labai mažos apimties,
nes daugiausia buvo kurti taisyklėmis grįstiems metodams, centruojantis į kokybę.
Žiūrint iš praktinės pusės, didžioji dauguma darbų [335, 336, 30, 337, 338, 339] lie-
tuvų kalbai surenka privačius duomenų rinkinius, tiek mokymui, tiek testavimui, kas
labai apsunkina skirtingų darbų metodų palyginimą. Taigi išteklių tiek mokytį, tiek
įvertinti reprezentacijas lietuvių kalbai yra labai riboti.

Kokybiškų vienakalbių transformerių modelių, tokių kaip BERT [2] anglų kalbai, lietu-
vų kalbai nėra. Viena iš galimybių yra naudoti daugiakalbius transformerių modelius,
pritaikytus iškart daugybei kalbų. Tokie yra mBERT [2], XLM-R [328], mT5 [11] ir
ByT5 [3], taip pat trikalbis lietuvių, latvių ir anglų kalbų BERT modelis [343]. Tačiau
tokie universalūs modeliai turi papildomų trūkumų: kad būtų padengta daugiau žodžių
kombinacijų, naudojami didesni, trumpesnių žymenų žodynai. Dėl to žymenų sekos
gerokai pailgėja, be to, didelis žodynas reikalauja daugiau modelio parametrų. Tai ga-
liausiai stipriai sulėtina transformerio modelio veikimą. Taigi, tiek efektyvumo, tiek
kokybės aspektais geriau yra turėti vienakalbį modelį.

Pradinio mokymo transformerių reprezentacijų pagerinimas

Tyrimų metodika

Sprendžiama teksto reprezentacijų išgavimo iš pradinio mokymo transformerio modelio problema. Į šį paduodamas tekstas s_0 , kuris yra išskaidytas į N atskirų vienas po kito einančių žymenų:

$$t_1 \quad t_2 \quad \dots \quad t_N . \quad (33)$$

Kiekviename iš L sluoksnių transformerio modelis reprezentuoja kiekvieną žymenį kaip d -ilgio vektorių $v \in \mathbb{R}^d$. Prieš patekdami į modelio sluoksnius, žymenys turi savo unikalų vektorių (galime vadinti -1 -uoju sluoksniu) bei vektorių, kuris yra pastarojo bei segmentų ir vietos vektorių suma (laikysime 0 -iu sluoksniu). Taigi, iš viso nagrinėjama 3 -matė erdvė $\mathbb{R}^{(L+2) \times N \times d}$:

$$\begin{array}{cccc} v_1^{-1} & v_2^{-1} & \dots & v_N^{-1} \\ v_1^0 & v_2^0 & \dots & v_N^0 \\ v_1^1 & v_2^1 & \dots & v_N^1 \\ \vdots & \vdots & \ddots & \vdots \\ v_1^L & v_2^L & \dots & v_N^L \end{array} . \quad (34)$$

Užduotis yra rasti tokią agregacijos funkciją f , kad $((L+2) \times N \times d)$ dydžio išvestis iš transformerio modelio būtų sumažinta į kuo įmanoma geresnį d dydžio reprezentacijos vektorių $\mathbb{R}^d$.

Kontekstualizuotoms reprezentacijoms išgauti pasirinkti 3-jų tipų BERT modeliai. Pirmojo tipo (20 lentelėje žymime tiesiog B) nagrinėjamos paskutinio sluoksnio aktyvacijos tokio teksto, koks ir buvo paduotas. Kiti du tipai naudoja skirtingų užklausų formuluotę. 20 lentelėje žymimi T0 ir T4 turi atitinkamas formuluotes: „This sentence: ”[X]” means [M].“ ir „This sentence from the dictionary: ”[X]” means ”[M]” and is about [M], which is a synonym for [M].“. Čia [X] žymi pradinio teksto įterpimo vietą, o [M] – užmaskuotąjį žymenį, į kurį tikimasi, bus sukoncentruota pradinio teksto reprezentacija.

Statinėms reprezentacijoms išgauti pasirinkti modeliai, išvardinti žemiau.

- Kontekstualizuotų BERT reprezentacijų vidurkis (Avg.). Kiekvienam žymeniui iš Wikitext-103 [295] tekstyno suvidurkinami jo visi kontekstualizuoti vektoriai, gauti naudojant BERT modelį. Tuomet teksto reprezentacija randama kaip šių sakinių sudarančių statinių vektorių vidurkis.
- BERT2Static (B2S) [202] modelis, mokytas Word2vec stiliumi iš kontekstualizuotų BERT reprezentacijų. Taip pat eksperimentuojama su šiuo modeliu išmetant 100 dažniausių jo žodžių (B2S-100).
- Atsitiktinių vektorių (RE) bazinis modelis. Kiekvieno žymens vektorius užpildomas atsitiktinėmis reikšmėmis iš Gauso pasiskirstymo, centruoto ties reikšme 0 su 0,1 standartiniu nuokrypiu.

Skaitinėms reprezentacijoms agreguoti naudojami įvairūs metodai. Paprasčiausias – visų teksto vektorių vidurkis (avg.). Sudėtingesnis – IDF svertinis vidurkis. Turint tekstų rinkinį su N tekstų ir tekstų dažnį kiekvienam žymeniui df_t , apibrėžtą kaip tekstų skaičių, kuriuose yra žymuo t , idf_t randamas:

$$idf_t = \log \frac{N}{df_t}. \quad (35)$$

Nagrinėjami du IDF svoriai: idf_t^W , skaičiuojamas su Wikitext-2 [295] tekstynu, bei idf_t^T , skaičiuojamas su visais sprendžiamos užduoties teksta. Remiantis [213], naudojamas metodas (“-biases”) išmetant 33 dažniausius žymenis. Kita, specifinė agregacija, taikyta T0 ir T4 užklausų formavimo modeliams, yra tik [M] tipo žymenų vidurkio skaičiavimas.

Reprezentacijos apdorotos su dekoreliavimo (angl. *whitening*) (whiten), standartizavimo (z-score), ABTT [1], modifikuojant pasiskirstymą į tolygųjį (quantile-u) bei vektorių ilgių suvienodinimo iki vieneto (normalize) metodais. Taip pat eksperimentuota reprezentacijų apdorojimo metodą mokant ne su užduoties tekstynu, o su Wikitext-2, pažymint tokį metodą $\cdot^W$.

Skaitinėms reprezentacijoms įvertinti naudojami trijų užduočių rinkiniai, išvardinti žemiau.

1. Semantinis tekstinis panašumas (žr. angl. sutrumpinimą STS). Kiekviename pavyzdyje yra du sakiniai ir jų panašumo įvertis. Vertinama skaitinių tekstinių reprezentacijų panašumo koreliacija su minėtu įverčiu. Naudoti 2012-2016 metinių konkursų STS [130, 131, 132, 133, 134, 21], STR [302], SICK-Relatedness [141] bei STS-B [21] duomenų rinkiniai.
2. Trumpų tekstų klasterizavimas. Naudojamas k -means klasterizavimo algoritmas iš scikit-learn [294] Python bibliotekos. Klasterizavimo tikslumas suskaičiuojamas Hungarian algoritmu [315] ir vidutinis tikslumas randamas iš 10 kartų pakartotų nepriklausomų klasterizavimų. Naudoti tekstai yra 4 sričių: 4 temų naujienos [318, 314], biomedicininį straipsnių pavadinimai iš 20 grupių [319, 320], 152 naujienų pavadinimai su teksto fragmentais [321, 314], 8 grupių žiniatinklio paieškos frazės [322], 20-ies kategorijų klausimų pavadinimai, susiję su programavimu [323], bei 89 kategorijų socialinio tinklo žinutės [321].
3. Tekstų klasifikavimas. Po to, kai modeliai paverčia tekstą į skaitines reprezentacijas, su šiomis, jų nekeičiant, mokomas logistinės regresijos klasifikatorius ir įvertinamas su 10-ies dalių kryžmine validacija. Naudotos įvairios užduotys iš SentEval [103] bazinio užduočių rinkinio: MR [303], CR [304], SUBJ [303], MPQA [307], SST5 [149], TREC [309], MRPC [307]; bei kitos: SICK-Entailment [141], SCICITE [308], SICK-Relatedness [141] ir STS-B [21].

Teksto reprezentacijų erdvės panaudojamumui įvertinti naudota IsoScore [253] metrika Wikitext-2 tekstynui iš [295]. Taip pat naudotos tolygumo bei lygiavimosi metrikos

[278] reprezentacijų kokybei nusakyti.

Rezultatai

Kaip pateikta 20 lentelėje, tiek agregacijos, tiek apdorojimo technikos stipriai pagerina reprezentacijas STS bei klasterizavimo užduotims. STS užduotyse geriausias modelis buvo pagerintas nuo 62,3 % iki 71,6 % Spearmano koreliacijos, o klasterizavime – nuo 59,2 % iki 64,8 % tikslumo. Didžiausi pokyčiai gaunami taikant abu metodus – tiek agregavimą, tiek apdorojimą. Štai T4 modelis be jokių technikų STS užduotyse turi 62,3 % koreliaciją. Vien tik su geresne agregacija idf_t^T – 69,2 %, o vien tik su geresniu quantile-u apdorojimu – 67,3 %. O abiejų technikų vienu metu taikymas leidžia pasiekti 71,6 % Spearmano koreliaciją. Klasifikavimo užduotims šie metodai nėra naudingi, nes jų metu dar vykdomas mokymas, kurio metu logistinis klasifikatorius pats išmoksta atsirinkti konkrečiai užduočiai reikalingus informacijos komponentus. Neprižiūrimoms užduotims, tokioms kaip semantinio tekstinio panašumo bei klasterizavimo, svarbu, kad informacija būtų pirmoje principinėje komponentėje.

Ankstesniuose darbuose buvo parodyta, kad reprezentacijas galima gerokai pagerinti, naudojant užklausų formulavimus, kaip kad T0 ir T4 modeliai, ir ištraukiant tik vieną [M] žymens reprezentaciją. Tačiau šio darbo rezultatai rodo, kad tai nėra geriausias būdas. Teksto klasterizavimo ir klasifikavimo užduotims paprastas visų žymenų, taip pat ir esančių užklausos formuluotėje, vidurkis pralenkė vien tik [MASK] žymens naudojimą. Pavyzdžiui, T0 vidurkio rezultatas klasterizavimui yra 53,5 %, o vieno [M] žymens – tik 45,5 % (žr. 20 lentelę). Nėgana to, dar papildomas pagerinimas galimas naudojant jau minėtas technikas, tarp jų – žymenų agregavimą. T0 šablono klasterizavimo užduočiai taip pasiekiant net 57,6 % tikslumą, naudojant idf_t^T žymenų agregavimą bei tolimesnį quantile-u^W apdorojimą. Taigi, šis darbas parodė, kad užklausos formuluotė pagerina visų dalyvaujančių žymenų vektorių reprezentacijas.

Didžiausias agregavimo ir apdorojimo metodų poveikis pastebėtas atsitiktinių vektorių (RE) metodui. STS užduotims nuo 50,9 % iki 66,4 %, o klasterizavimo – nuo 36,3 % iki 49,5 %. Nors toks tikslumas yra gana žemas, tačiau STS užduotims jis yra vos 3,4 % žemiau to, ką pavyko gauti su BERT modeliu, nenaudojant specifinių užklausų formuluočių. Tai paskatino toliau patyrinėti šį paprastą ir gana vidutiniškai veikiantį metodą. Eksperimentuojant su atsitiktinio vektoriaus ilgiu ir skirtingais žyvenavimais (angl. *tokenization*), gautas didesnis tikslumas ir klasifikavimo užduočių. Naudojant Llama [301] žyvenavimą bei pailginus atsitiktinio vektoriaus ilgį iki 12 288 elementų, vidutinis klasifikavimo užduočių tikslumas išauga net iki $75,9 \pm 0,2$ %, vos 3,9 % mažiau nei BERT modelio rezultatas.

Šiame darbe pasiūlytas Avg. metodas sėkmingai aplenkė B2S, kuris statines reprezentacijas iš BERT modelio surinko naudodamas Word2vec tipo mokymą. Lyginant tik paprastai suvidurkintas reprezentacijas (avg.), abu modeliai yra panašūs tiek semantinio tekstinio panašumo užduotyse – 56,4 % ir 56,7 %, tiek vertinant klasterizavimą – 55,5 % ir 53,8 %, tiek klasifikavimą – 77,3 % ir 76,9 %. Tačiau Avg. metodo priva-

Modelis	T0	T4	B	B + Avg.	Avg.	B2S	B2S-100	RE
Sluoksnius	Paskutinis		Pirmas + paskutinis			–		
STS užduočių vid. Spearmano koreliacija (%)								
avg.	61,3	62,3	61,3	61,4	56,4	56,7	60,6	50,3
idf_t^W	67,8	67,4	69,2	69,3	66,9	66,1	65,4	66,4
+ z-score	68,6	68,1	69,8	70,6	69,5	66,4	65,9	66,4
idf_t^T	67,4	69,2	68,7	69,0	67,5	56,7	60,6	64,9
+ quantile-u	69,5	71,6	68,9	69,5	68,4	56,2	61,0	63,0
-biases	64,7	66,0	68,2	69,3	67,3	54,1	57,5	63,7
[MASK]	63,4	69,3						
+ quantile-u	65,6	70,5						
Klasterizavimo užduočių vid. tikslumas (%)								
avg.	53,5	55,0	57,0	59,2	55,2	53,8	55,3	36,3
idf_t^W	53,0	53,0	55,2	57,1	53,7	53,0	53,7	44,3
+ normalize	54,3	54,4	58,0	57,9	54,1	55,8	56,1	49,5
idf_t^T	53,1	57,6	60,7	62,5	58,7	53,8	55,3	42,8
+ quantile-u ^W	57,6	60,0	63,1	64,4	60,2	54,6	56,0	44,4
+ normalize	55,5	57,6	63,4	64,8	59,8	55,5	57,2	47,0
-biases	54,3	56,2	61,1	62,2	61,6	52,6	54,1	40,7
+ normalize	54,4	56,3	62,4	63,4	62,2	54,4	56,8	44,6
[MASK]	45,5	54,2						
+ quantile-u	44,4	54,9						
Klasifikavimo užduočių vid. tikslumas (%)								
avg.	80,5	79,9	79,9	79,7	77,3	76,9	75,1	69,5
idf_t^W	73,1	67,1	78,3	77,9	75,3	76,8	74,1	67,6
idf_t^T	80,1	79,7	78,9	78,4	76,1	76,9	75,1	68,1
-biases	80,1	79,9	80,1	79,8	77,2	76,3	74,2	70,3
[MASK]	78,7	76,8						

20 lentelė. Tokenų agregavimo ir reprezentacijų apdorojimo įtaka įvairiems teksto modeliams. Abu koreliacijos ir tikslumo įverčiai yra nuo 0% (blogiausia) iki 100% (geriausia). Geriausi kiekvieno modelio rezultatai yra paryškinti, o iš šių geriausias pabrauktas.

lumai išryškėja pritaikius agregavimo ir reprezentacijų apdorojimo metodus. Taip geriausi įvertiniai Avg. ir B2S/B2S-100 tampa 69,5 % ir 66,4 % STS užduočių bei 62,2 % ir 57,2 % klasterizavimo. Tokia statinio Avg. metodo sėkmė paskatino išbandyti jį kartu su eilinio BERT modelio reprezentacijomis, jas suvidurkinant. Toks dvigubas B+Avg. metodas pasiekė geriausią klasterizavimo užduočių įvertį – 64,8 % iš visų tirtų modelių, įskaitant ir patį BERT, o agreguojant 10-ojo sluoksnio reprezentacijas – ir aukščiausią 80,5 % klasifikavimo tikslumą.

Teksto lygio transformacijos lietuvių kalbai

Praėjusio skyriaus eksperimentai atlikti anglų kalbai. Ši turi daugiausiai resursų: modelių bei duomenų rinkinių. Retesnėms kalboms resursų prieinamumas vaidina labai svarbų vaidmenį. Kuo mažiau galimų modelių ir duomenų rinkinių – tuo mažiau yra galimybių. Taigi, šiame skyriuje eksperimentai bus atlikti vidutinių resursų lietuvių kalbai. Tai leis prisidėti prie šios kalbos išsaugojimo skaitmeninėje erdvėje bei prisidės prie kitų, retesnių kalbų tyrimų.

Praėjusiame skyriuje buvo pristatyti teksto reprezentacijų, išgautų iš pradinio mokymo BERT modelio, pagerinimo metodai. Buvo pasiūlytos įvairios agregacijos ir vėlesnio apdorojimo technikos, smarkiai pagerinančios teksto reprezentacijas. Šiame skyriuje teksto reprezentacijų išgavimas bus papildytas išankstinio apdorojimo būdais. transformerių modeliai bus koreguojami atlikti lietuviško teksto santraukas bei taisymą. Tai yra tokie pirmi transformerių modeliai lietuvių kalbai. Šiame skyriuje bus tiriama ne tik kaip gerai gali būti atliekamos santraukos ir teksto taisymas, bet ir kaip gali pagerėti tokio teksto skaitinė reprezentacija, naudojant jau ankstesniame skyriuje pristatytus metodus.

Tyrimų metodika

Iš viso atlikti 4 skirtingų užduočių koregavimo mokymai (žr. 21 lentelę).

Užduotis	Modelis	Tekstų skaičius	
		Epocha	Žingsnis
Abstraktus lietuviškų naujienų apibendrinimas	T5 T5 ₀	2 036 363	128
Diakritų atkūrimas	ByT5	524 288	256
Diakritų ir įvedimo klaidų pašalinimas			
Lietuvių rašybos klaidų taisymas	ByT5 T5 _{LT}	4 194 304	128

21 lentelė. Generatyvinių transformerių modelių mokymo užduotys. T5 modeliai yra su išmokytu lietuvišku tokenizatoriumi. T5₀ – su atsitiktiniais pradiniais svoriais, o T5_{LT} – po lietuviškų naujienų apibendrinimo užduoties mokymo.

Abstraktaus lietuviškų naujienų apibendrinimo užduočiai buvo surinkti naujienų teks-

tai su pagrindinėmis ir santraukos dalimis iš įvairių lietuviškų naujienų portalų. Mokymui išfiltruoti tik tie tekstai, kurių 1) santrauka ilgesnė nei 10 simbolių; 2) pagrindinis tekstas ilgesnis nei 100 simbolių ir 3) bent dvigubai ilgesnis už santrauką bei 4) abiejų dalių persidengimas yra mažesnis nei 20 %. Mokymas atliktas su seka-į-seka tipo anglų kalbos T5 [22] pradiniais svoriais, taip pat ir atsitiktiniais iki 20 000 žingsnių (T5₀, žr. 21 lentelę). T5 modeliui buvo mokytas naujas lietuviškas žymenavimas. Naudota SentencePiece [37] biblioteka ir *unigram* žymenų modelis [36]. Žymenavimas mokytas su 10⁶ teksto pavyzdžių, o žymenų skaičius parinktas 32 000.

Diakritinių ženklų (diakritų) atkūrimo ir klaviatūros įvedimo užduotims naudotas ByT5 [3] modelis. Šis naudoja baitų lygio žymenis, taigi turi labai mažą, vos 384 žymenų žodyną su $256 = 2^8$ pagrindiniais žymenimis simbolių baitams. Tokia konfigūracija pritaikyta daugiakalbėms įvestims. Palyginti su T5 ir mT5 modeliais, ByT5 [3] modelio autoriai taip pat perdislokavo didelę parametrų dalį iš dekoderio pusės į enkoderį, kad daugiau pajėgumų būtų skirta teksto iš baitų įvesties supratimui. Pasinaudojant ByT5 įvairių kalbų palaikymu, diakritų atkūrimo bei diakritų atkūrimo ir klaviatūros įvedimo klaidų taisymo vienu metu užduotys buvo mokytos iš viso 13-ai kalbų, kiekvienai atskirai. Naudoti Vikipedijos tekstynai, kaip ir [358], taip pat pridodant ir lietuvių kalbą su [358] autorių viešai pasidalintais įrankiais. Kad būtų lengviau palyginti, tarp kalbų visi mokymai atlikti su tuo pačiu tekstų skaičiumi – 524 288. Kai kurie modeliai buvo mokyti ilgiau, tęsiant mokymą nuo 2 048 iki 6 000 žingsnių. Šioje mokymo stadijoje mokymo užduoties įvykdymo rodiklis tapo gan triukšmingas, todėl nuspręsta padidinti duomenų partijos dydį iki 8 192 tekstų per vieną žingsnį. Diakritų atkūrimo užduočiai mokyta po 3 modelius su skirtinga tvarka naudotais duomenimis modelių rezultatų patikimumui patikrinti.

Lietuvių rašybos klaidoms taisyti apibendrinimo užduoties tekstynas buvo praplėstas surenkant daugiau tekstų, iš viso iki 4 194 304 paragrafų. Šie gauti pavadinimą ir santrauką pridodant kaip papildomus paragrafus prie pagrindinio teksto. Atliktas tik vienos epochos mokymas tiek ByT5 modelio, tiek T5_{LT}, kuris buvo gautas lietuviškų naujienų apibendrinimo užduoties mokymo metu.

Remiantis ankstesniais T5 transformerio darbais [22, 11, 3, 368], naudotas Adafactor optimizatorius [352] ir 0,001 mokymosi greitis. Pirmiesiems abstraktaus apibendrinimo eksperimentams šis greitis buvo keistas pirma atliekant apšilimą per pirmus 10 000 žingsnių, po to mažinamas pagal atvirkštinės kvadratinės šaknies grafiką. Vėlesniuose klaidų taisymo eksperimentuose naudotas tik pastovus mokymosi greitis, kad skirtingų kalbų rezultatai tarpusavyje būtų lengviau palyginami.

Mokymo eksperimentams naudota GeForce RTX 2080 Ti grafikos plokštė ir Python biblioteka *transformers 4.12.0* [296]. Tam, kad prieš kiekvieną modelio svorių reikšmių atnaujinimą būtų praleisti didesni duomenų partijų kiekiai, naudotas akumuliacinis režimas, kai duomenys modeliu leidžiami ne lygiagrečiai, o nuosekliai.

Rašymo klaviatūra klaidų modelis

Diakritinių ženklų klaidas galima paprastai sumodeliuoti priskiriant analogiškas raides be diakritų. Tam, kad būtų sumodeliuotos rašymo klaviatūra klaidos, panašiai kaip [360, 361] darbuose, buvo panaudotas mažesnis tekstynas, šiuo atveju GitHub Typo Corpus [366], klaidų tikimybėms skaičiuoti ir generuoti dideliame mokymo duomenų tekstynui. Remiantis [362, 363], klaidos buvo išskaidytos į 4 grupes: 1) pakeitimo; 2) ištrynimo; 3) įterpimo; bei 4) susikeitimo.

Lietuvių gramatikos klaidų modeliavimas

Kitaip nei rašymo klaviatūra, čia pagrindinės klaidos modeliuotos pagal tai, kad skiriasi tai, kaip garsai tariaami ir kaip jie rašomi. Pirmiausia tai yra panašios raidės, pavyzdžiui, „iį“. Kitos klaidos yra dėl priebalsių supanašėjimo ir suskardėjimo, kai žodžiai „dirbti, lipdavo“ gali būti klaidingai užrašyti kaip „dirpti, libdavo“. Taip pat įtraukti visi atvejai, kai rašomos dvi tokios pačios raidės iš eilės, nors girdisi tik viena, pavyzdžiui, „pussesserė“. Galiausiai, įtrauktos ir skyrybos klaidos. Dauguma šių klaidų modeliuotos statistiškai randant jų tikimybes tam tikrose grupėse ir pagal šias tikimybes generuojant atitinkamą klaidą tos grupės rastame atvejuje. Detalios naudotos tikimybės yra aprašytos 5 priede.

Įvertinimas

Tekstų santraukų generavimui įvertinti naudota metrika ROUGE-2 [14], tikrinanti 2-gram žodžių persidengimus ir lygi santykiui bendrų (tarp sugeneruotų ir turimų sugeneruoti) 2-gram skaičiaus iš visų turimų sugeneruoti 2-gram skaičiaus [353].

Diakritinių ženklų ir teksto įvedimo klaidų taisymui įvertinti naudotas alfa-žodžių tikslumas [358]:

$$\text{alfa-žodžių tikslumas} = \frac{T_s}{T_g} \cdot 100 \%, \quad (36)$$

čia T_s yra teisingai atkurtų žodžių skaičius, o T_g – visų žodžių skaičius. Vertinami tik tie žodžiai, kurie turi bent vieną raidę ir yra priskiriami *alfa* kategorijai.

Bendresniam gramatinių klaidų taisymo vertinimui naudota ERRANT [370] metrika, pritaikyta lietuvių kalbai naudojant lietuvišką Hunspell žodyną [371], kamienizavimą [372] bei žodžių tipų atpažinimą [373].

Teksto transformavimas

Koreguoti transformerių modeliai buvo panaudoti kaip pirminis teksto apdorojimas geresnėms skaitinėms reprezentacijoms išgauti. Tuo tikslu atlikti eksperimentai su lietuviškų naujienų klasterizavimu, išbandant teksto transformavimo į santraukas metodą, bei bandymai su neapykantos kalbos klasifikavimu, išbandant gramatinių klaidų taisymo įtaką. Klasterizavimui atrinktos 7 lietuviškų naujienų straipsnių kategorijos: pasaulio naujienos, mokslas, teisė, politika, orai, nusikaltimai, mokslas. Iš šių atsitiktiniu būdu buvo paimama po 125 straipsnius ir atliekamas K-means klasterizavimas.

Neapykantos kalbai klasifikuoti naudotas duomenų rinkinys [374] su trimis komentarų klasėmis: neapykanta (15 %), agresyvus (29 %) bei neutralus (56 %).

Rezultatai

Lietuviškų naujienų santraukų generavimo modelis po 250 000 mokymo žingsnių pasiekė maksimalų 16,3 % ROUGE-2 įvertinimą ir toliau pradėjo persimokyti. Iš pirmo įspūdžio sugeneruotos santraukos atrodo sklandžios ir kokybiškos. Tačiau detalesnės analizės parodė, kad dažnai pasitaiko haliucinacijų – faktų iškraipymų ar neatitikimų. ROUGE-2 metrika pasirodė gan problemiška, nes turėjo gana didelį išsibarstymą, $\approx 14\%$, be to, buvo pastebėta daug klaidingų jos įvertinimų – dažnai kokybiškos santraukos buvo įvertinamos menku įverčiu, nes turėjo mažai persidengiančių žodžių porų.

Naujo lietuviško žymenavimo mokymas leido pagreitinti paties modelio mokymo procesą. Angliškasis tekstą suskaidydavo vidutiniškai į 1448 dalis, daugiakalbis mT5 [11] – į 789, o mokytas lietuvių kalbai – į 530. Jei būtų naudojamas daugiakalbis mT5 modelis, mokymas būtų 4 kartus lėtesnis, tiek dėl daugiau modelio parametrų, susijusių su daugiau žymenų, tiek dėl ilgesnių sekų dėl smulkesnio jų skaidymo. Taip pat pasiteisino sprendimas mokytis ne nuo pradinių atsitiktinių, bet nuo angliškojo modelio parametrų, nes mokymo užduoties įgyvendinimo profilis taip gerėjo daug sparčiau.

Diakritų atkūrimo užduotimi mokytis modeliai pasiekė vidutinį 98,10 % diakritų atkūrimo tikslumą (žr. 22 lentelę). Tai yra tik šiek tiek mažiau nei geriausio modelio rezultatas 99,19 % [369]. Vis dėlto pastarajam mokytis naudota ne tik Vikipedija, kaip šiame darbe, bet prieš tai ir dideli internetinių tekstynų kiekiai, tad mokymo trukmė kiekvienai kalbai užtruko po dvi savaites. Šiame darbe ByT5 modelio mokymas kiekvienai kalbai užtruko tik 10 valandų. Tęsiant mokymą ilgiau (žr. 22 lentelę, $\times 8$), geriausias buvęs rezultatas būtų neabejotinai viršytas.

Didžioji dalis transformerio ByT5 klaidų susijusios su retai ar visai nematytais žodžiais mokymo metu. Daugumai kalbų nematyti žodžiai buvo susiję su nuo 13 % iki 36 % klaidingų ištaisymų. Tęsiant mokymą ilgiau, šis procentas tik didėja. Atsižvelgiant į šią silpnę, pasiūlytas kombinuotas klasikinio Žodyno ir ByT5 metodas, kai Žodynas naudojamas ištaisyti klaidoms, turinčioms tik vieną teisingą alternatyvą, o jei daugiau – naudojant ByT5. Kaip matyti 22 lentelėje, toks kombinuotas metodas yra geresnis, tačiau jo privalumai, ilginant ByT5 mokymo laiką, mažėja. Svarbu pažymėti, kad transformerio modelis apskritai gan gerai susitvarko su nematytais žodžiais, nors tai ir yra pagrindinis klaidų šaltinis. Vertinant žodžių be diakritų palikimą tokių, kokie ir yra, bei su diakritais ištaisymą, ByT5 pasiekia daugiau nei 76 % alfa-žodžių tikslumą visoms kalboms mokant nematytiems žodžiams. Papildomos klaviatūros įvesties klaidų rūšies pridėjimas gerokai pablogina vidutinį atkūrimo tikslumą iki 90,26 %, o kombinuotas Žodyno metodo lenkimas (94,60 %) rodo, kad reti žodžiai čia yra didesnė problema. Vis tik atliekant tokią sudėtingesnę užduotį tikslumas labiau pagerėja vykdant mokymą ilgiau.

Kalba	Diakritai				Diakritai ir įvedimas			
	ByT5		Žod.+ByT5		ByT5		Žod.+ByT5	
	×1	×8	×1	×8	×1	×19	×1	×19
Kroatų	99,17 ± 0,06		99,42 ± 0,03		90,27		96,71	
Čekų	98,01 ± 0,03		98,38 ± 0,04		89,88		94,52	
Prancūzų	99,37 ± 0,04		99,49 ± 0,03		93,45		96,52	
Vengrų	98,42 ± 0,02	99,20	98,78 ± 0,01	99,25	88,31	93,96	94,31	96,85
Airių	98,14 ± 0,32		98,40 ± 0,16		89,48		94,49	
Latvių	96,33 ± 0,12	97,78	96,62 ± 0,09	97,66	88,88		93,01	
Lietuvių	97,94 ± 0,19	99,07	98,18 ± 0,13	98,95	89,68	94,19	94,70	96,73
Lenkų	99,00 ± 0,03		99,16 ± 0,02		91,38		96,76	
Rumunų	97,99 ± 0,03		98,17 ± 0,04		90,50		94,14	
Slovakų	98,43 ± 0,06		98,77 ± 0,02		91,05		95,56	
Ispanų	99,33 ± 0,04		99,43 ± 0,02		93,12		95,98	
Turkų	98,86 ± 0,04		99,03 ± 0,02		90,00		95,29	
Vietnamiečių	94,25 ± 0,07	97,53	94,29 ± 0,07	97,54	87,34	93,40	87,86	93,77
Vidurkis	98,10		98,32		90,26		94,60	

22 lentelė. Alfa-žodžių tikslumo rezultatai (%) diakritų atkūrimo bei bendrai diakritų ir klaviatūros įvedimo klaidų ištaisymo užduočiai. Modeliai išmokyti su 524 288 sakinių, žymimi ×1, o su 8 ir 19 kartų daugiau, kartojant turimus tekstus iš naujo – ×8 ir ×19.

Lietuvių kalbos rašybos taisymo mokymo užduotis buvo dar sudėtingesnė, nes čia dar papildomai panaudotos ir lietuvių kalbai specifinės klaidos. Vis dėlto ByT5 modelis pasiekė aukštą ($F_{0,5} = 0,92$) įvertį, aplenkdamas T5 ($F_{0,5} = 0,80$), mokyta su tokiu pačiu tekstų kiekiu. Šis eksperimentas parodė, kad smulkesnės baitų lygio reprezentacijos yra naudingesnės rašybos klaidų užduočiai. Dėl smulkesnio skaidymo modelis gali įvertinti ir išmokti daugiau susijusių detalių, kurios ypač svarbios tokio tipo užduočiai. Įvertinus lietuvių kalbos rašybos taisymo modelio gebą taisyti kiekvienos rūšies klaidas atskirai, buvo patvirtinti ankstesnių eksperimentų rezultatai – dažniau mokymo duomenyse pasitaikančios klaidos yra ištaisomos geriau.

Išmokytų transformerių modelių taikymas geresnėms teksto transformacijoms išgauti nepasiteisino. Santraukų sudarymas šiek tiek pablogino naujienų klasterizavimą, o gramatinių klaidų ištaisymas neapykantos kalbos klasifikavimo užduočiai praktiškai neturėjo jokios įtakos.

Išvados

Atlikta išsami teksto reprezentacijos modelių ir jų įvertinimo literatūros apžvalga. Dabartiniai geriausi modeliai yra pradinio mokymo transformerio architektūros dirbtiniai neuroniniai tinklai, naudojantys žodžio dalių žymenavimą ir koreguoti kontrastinio mokymo užduotimi naudojant NLI tipo tekstynus. Universalioms tekstinėms reprezentacijoms įvertinti naudojamos semantinio tekstinio panašumo ir klasifikavimo užduotys iš statinių reprezentacijų požymių. Metodai, naudojantys tik pradinio mokymo

reprezentacijas, gerokai atsilieka nuo naudojančių ir koregavimo mokymą. Lietuvių kalbai Transformacinių modelių pritaikymas yra ribotas, nes reikalauja daug mokymo duomenų ir skaičiavimų resursų; taip pat nėra įvertinimo užduočių rinkinių.

Ištirti ir pasiūlyti metodai teksto reprezentacijoms išgauti iš pirminio mokymo transformerių modelių semantinio teksto panašumo, trumpo teksto klasifikavimo ir klasifikavimo užduotims:

1. Rastos tokios žymenų ir sluoksnių agregavimo bei vėlesnio apdorojimo kombinacijos, kurios pagerina skaitines reprezentacijas, ypač tekstų panašumu grįstoms užduotims. Šios technikos pagerino BERT modelio pradinio mokymo reprezentacijas semantinio teksto panašumo užduotims vidutiniškai 8,5 % ir trumpo teksto klasterizavimo užduotims – 6,4 %, o prižiūrimoms klasifikavimo užduotims įtakos praktiškai neturėjo.
2. Pasiūlytas statinis Avg. teksto reprezentacijų metodas iš paprastai suvidurkintų BERT žymenų pralenkia kitus sudėtingesnius Word2Vec tipo modelius. Kartu taikant reprezentacijų erdvės pakeitimus šis metodas pasiekia 69,5 % tikslumą semantinio teksto panašumo užduotims, 62,2 % trumpo teksto klasterizavimo bei 77,3% tikslumą klasifikavimo užduotyse.
3. Pasiūlytas atsitiktinių vektorių metodas, naudojantis agregavimo ir reprezentacijų apdorojimo technikas, yra puikus palyginamasis modelis. Taikant reprezentacijų erdvės modifikacijas, jis vidutiniškai pasiekia 66,4 % tikslumą semantinio teksto panašumo užduotims, trumpo teksto klasterizavimo – 49,5 %, o klasifikavimo užduotyse gali pasiekti net 75,9 % vidutinį tikslumą. Šis bazinis modelis parodė, kad reprezentacijų geometrija atsakinga už reikšmingą dalį pagerėjimų, kurie paprastai priskirtini sudėtingų kontekstualizuotų modelių įtakai.

Sukurtos pirmosios lietuvių kalbai transformerių modeliais grįstos teksto transformacijos: abstrakčių santraukų generavimas bei gramatinių klaidų taisymas. Šių metodų tyrimai atskleidė:

1. Baitų lygio transformerio modelis yra labai tinkamas morfologiškai sudėtingoms kalboms. Įvairių kalbų gramatinių klaidų eksperimentuose gauti ištaisymo tikslumai artimi geriausiems žinomiems, naudojant mažiau resursų. Pasiektas vidutinis 98,3 % alfa-žodžių tikslumas (vos 1 % žemiau geriausio modelio) diakritinių ženklų atkūrimo užduočiai 13-ai kalbų (tarp jų ir lietuvių) su daug mažesniu mokymo tekstyne bei trumpesniu mokymo laiku.
2. Transformerio modeliu grįstas naujienų straipsnių abstrakčių santraukų generavimas pasiekė 16,3 % ROUGE-2 įvertį, o vertinant lietuvių rašybos klaidų taisymą gautas 0,92 ERRANT $F_{0,5}$ tikslumo įvertis.
3. Morfologiškai turtingoms ir vidutinių resursų kalboms transformerių modeliai aplenkia klasikinius metodus. Jų efektyvumas priklauso nuo žymenavimo strategijos, duomenų gausos ir pritaikymo procedūrų.

4. Santraukų generavimo ir klaidų taisymo transformacijos pagerino teksto kokybę, tačiau nepagerino skaitinių lietuviškų naujienų klasterizavimo ir neapykantos kalbos klasifikavimo užduočių reprezentacijų. Šių užduočių atlikimui pagerinti reikalingas kontrastinio koregavimo mokymas naudojant anotuotus panašumo užduočių tekstynus.

LIST OF REFERENCES

1. MU, J. and VISWANATH, P. All-but-the-Top: Simple and Effective Postprocessing for Word Representations. In: *International Conference on Learning Representations*. 2018. Available also from: <https://openreview.net/forum?id=HkuGJ3kCb>.
2. DEVLIN, J.; CHANG, M.-W.; LEE, K. and TOUTANOVA, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: BURSTEIN, J.; DORAN, C. and SOLORIO, T. (eds.). *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, 2019, p. 4171–4186. Available from DOI: 10.18653/v1/N19-1423.
3. XUE, L.; BARUA, A.; CONSTANT, N.; AL-RFOU, R.; NARANG, S.; KALE, M.; ROBERTS, A. and RAFFEL, C. ByT5: Towards a Token-Free Future with Pre-trained Byte-to-Byte Models. *Transactions of the Association for Computational Linguistics* Vol. 10, 2022, p. 291–306. Available from DOI: 10.1162/tac1_a_00461.
4. MCCANN, B.; BRADBURY, J.; XIONG, C. and SOCHER, R. Learned in translation: Contextualized word vectors. In: *Advances in Neural Information Processing Systems*. 2017, p. 6294–6305.
5. AHMED, N.; NATARAJAN, T. and RAO, K. Discrete Cosine Transform. *IEEE Transactions on Computers* Vol. C-23, 1974, no. 1, p. 90–93. Available from DOI: 10.1109/T-C.1974.223784.
6. LE, Q. and MIKOLOV, T. Distributed Representations of Sentences and Documents. In: XING, E. P. and JEBARA, T. (eds.). *Proceedings of the 31st International Conference on Machine Learning*. Beijing, China: PMLR, 2014, vol. 32, p. 1188–1196. *Proceedings of Machine Learning Research*, no. 2. Available also from: <https://proceedings.mlr.press/v32/le14.html>.
7. PETERS, M. E.; NEUMANN, M.; IYYER, M.; GARDNER, M.; CLARK, C.; LEE, K. and ZETTLEMOYER, L. Deep Contextualized Word Representations. In: WALKER, M.; JI, H. and STENT, A. (eds.). *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, 2018, p. 2227–2237. Available from DOI: 10.18653/v1/N18-1202.
8. PENNINGTON, J.; SOCHER, R. and MANNING, C. Glove: Global vectors for word representation. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, p. 1532–1543.
9. CHO, K.; MERRIËNBOER, B. van; GULCEHRE, C.; BAHDANAU, D.; BOUGARES, F.; SCHWENK, H. and BENGIO, Y. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In: MOSCHITTI, A.; PANG, B. and DAELEMANS, W. (eds.). *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014, p. 1724–1734. Available from DOI: 10.3115/v1/D14-1179.

10. ENEVOLDSEN, K.; CHUNG, I.; KERBOUA, I.; KARDOS, M.; MATHUR, A.; STAP, D.; GALA, J.; SIBLINI, W.; KRZEMIŃSKI, D.; WINATA, G. I.; STURUA, S.; UTPALA, S.; CIANCONE, M.; SCHAEFFER, M.; MISRA, D.; DHAKAL, S.; RYSTRØM, J.; SOLOMATIN, R.; ÇAĞATAN, Ö. V.; KUNDU, A.; BERNSTORFF, M.; XIAO, S.; SUKHLECHA, A.; PAHWA, B.; POŚWIATA, R.; GV, K. K.; ASHRAF, S.; AURAS, D.; PLÜSTER, B.; HARRIES, J. P.; MAGNE, L.; MOHR, I.; ZHU, D.; GISSEROT-BOUKHLEF, H.; AARSEN, T.; KOSTKAN, J.; WOJTASIK, K.; LEE, T.; SUPPA, M.; ZHANG, C.; ROCCA, R.; HAMDY, M.; MICHAIL, A.; YANG, J.; FAYASSE, M.; VATOLIN, A.; THAKUR, N.; DEY, M.; VASANI, D.; CHITALE, P. A.; TEDESCHI, S.; TAI, N.; SNEGIREV, A.; HENDRIKSEN, M.; GÜNTHER, M.; XIA, M.; SHI, W.; LÜ, X. H.; CLIVE, J.; K, G.; ANNA, M.; WEHRLI, S.; TIKHONOVA, M.; PANCHAL, H. S.; ABRAMOV, A.; OSTENDORFF, M.; LIU, Z.; CLEMATIDE, S.; MIRANDA, L. J. V.; FENOGENOVA, A.; SONG, G.; SAFI, R. B.; LI, W.-D.; BORGHINI, A.; CASSANO, F.; HANSEN, L.; HOOKER, S.; XIAO, C.; ADLAKHA, V.; WELLER, O.; REDDY, S. and MUENNIGHOFF, N. MMTEB: Massive Multilingual Text Embedding Benchmark. In: *The Thirteenth International Conference on Learning Representations*. 2025. Available also from: <https://openreview.net/forum?id=z13pfz4VCV>.
11. XUE, L.; CONSTANT, N.; ROBERTS, A.; KALE, M.; AL-RFOU, R.; SIDDHANT, A.; BARUA, A. and RAFFEL, C. mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer. In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, 2021, p. 483–498. Available from DOI: 10.18653/v1/2021.naacl-main.41.
12. MUENNIGHOFF, N.; TAZI, N.; MAGNE, L. and REIMERS, N. MTEB: Massive Text Embedding Benchmark. In: VLACHOS, A. and AUGENSTEIN, I. (eds.). *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*. Dubrovnik, Croatia: Association for Computational Linguistics, 2023, p. 2014–2037. Available from DOI: 10.18653/v1/2023.eacl-main.148.
13. WILLIAMS, A.; NANGIA, N. and BOWMAN, S. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, 2018, p. 1112–1122. Available from DOI: 10.18653/v1/N18-1101.
14. LIN, C.-Y. ROUGE: A Package for Automatic Evaluation of Summaries. In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, 2004, p. 74–81. Available also from: <https://aclanthology.org/W04-1013/>.
15. REIMERS, N. and GUREVYCH, I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In: INUI, K.; JIANG, J.; NG, V. and WAN, X. (eds.). *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, 2019, p. 3982–3992. Available from DOI: 10.18653/v1/D19-1410.

16. ARORA, S.; LIANG, Y. and MA, T. A simple but tough-to-beat baseline for sentence embeddings. In: *5th International Conference on Learning Representations, ICLR 2017*. 2017.
17. GAO, T.; YAO, X. and CHEN, D. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021, p. 6894–6910. Available from DOI: 10.18653/v1/2021.emnlp-main.552.
18. KIROS, R.; ZHU, Y.; SALAKHUTDINOV, R. R.; ZEMEL, R.; URTASUN, R.; TORRALBA, A. and FIDLER, S. Skip-Thought Vectors. In: CORTES, C.; LAWRENCE, N.; LEE, D.; SUGIYAMA, M. and GARNETT, R. (eds.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2015, vol. 28. Available also from: <https://proceedings.neurips.cc/paper/2015/file/f442d33fa06832082290ad8544a8da27-Paper.pdf>.
19. BOWMAN, S. R.; ANGELI, G.; POTTS, C. and MANNING, C. D. A large annotated corpus for learning natural language inference. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, 2015, p. 632–642. Available from DOI: 10.18653/v1/D15-1075.
20. HOWARD, J. and RUDER, S. Universal Language Model Fine-tuning for Text Classification. In: GUREVYCH, I. and MIYAO, Y. (eds.). *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, 2018, p. 328–339. Available from DOI: 10.18653/v1/P18-1031.
21. CER, D.; DIAB, M.; AGIRRE, E.; LOPEZ-GAZPIO, I. and SPECIA, L. SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation. In: *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Vancouver, Canada: Association for Computational Linguistics, 2017, p. 1–14. Available from DOI: 10.18653/v1/S17-2001.
22. RAFFEL, C.; SHAZEER, N.; ROBERTS, A.; LEE, K.; NARANG, S.; MATENA, M.; ZHOU, Y.; LI, W. and LIU, P. J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* Vol. 21, 2020, no. 140, p. 1–67. Available also from: <http://jmlr.org/papers/v21/20-074.html>.
23. KUSNER, M.; SUN, Y.; KOLKIN, N. and WEINBERGER, K. From Word Embeddings To Document Distances. In: BACH, F. and BLEI, D. (eds.). *Proceedings of the 32nd International Conference on Machine Learning*. Lille, France: PMLR, 2015, vol. 37, p. 957–966. *Proceedings of Machine Learning Research*. Available also from: <https://proceedings.mlr.press/v37/kusnerb15.html>.
24. MIKOLOV, T.; CHEN, K.; CORRADO, G. and DEAN, J. *Efficient Estimation of Word Representations in Vector Space*. arXiv, 2013. Available from DOI: 10.48550/ARXIV.1301.3781. [Last accessed on 17 December 2025].

25. MIKOLOV, T.; SUTSKEVER, I.; CHEN, K.; CORRADO, G. S. and DEAN, J. Distributed Representations of Words and Phrases and their Compositionality. In: BURGESS, C. J.; BOTTOU, L.; WELLING, M.; GHAHRAMANI, Z. and WEINBERGER, K. Q. (eds.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2013, vol. 26. Available also from: <https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf>.
26. YOKOI, S.; TAKAHASHI, R.; AKAMA, R.; SUZUKI, J. and INUI, K. Word Rotator's Distance. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, 2020, p. 2944–2960. Available from DOI: 10.18653/v1/2020.emnlp-main.236.
27. OPENAI; ACHIAM, J.; ADLER, S.; AGARWAL, S.; AHMAD, L. et al. *GPT-4 Technical Report*. 2024. Available from arXiv: 2303.08774 [cs.CL]. [Last accessed on 2025-03-07].
28. MALLIN, A.; ASAI, A.; ZHONG, V.; DAS, R.; KHASHABI, D. and HAJISHIRZI, H. When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories. In: ROGERS, A.; BOYD-GRABER, J. and OKAZAKI, N. (eds.). *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Toronto, Canada: Association for Computational Linguistics, 2023, p. 9802–9822. Available from DOI: 10.18653/v1/2023.acl-long.546.
29. IZACARD, G.; LEWIS, P.; LOMELI, M.; HOSSEINI, L.; PETRONI, F.; SCHICK, T.; DWIVEDI-YU, J.; JOULIN, A.; RIEDEL, S. and GRAVE, E. Atlas: few-shot learning with retrieval augmented language models. *J. Mach. Learn. Res.* Vol. 24, 2023, no. 1. ISSN 1532-4435.
30. VILEIKYTĖ, B.; LUKOŠEVIČIUS, M. and STANKEVIČIUS, L. Sentiment Analysis of Lithuanian Online Reviews Using Large Language Models. In: *CEUR workshop proceedings of the 29th international conference on information society and university studies (IVUS 2023) Kaunas, Lithuania, May 17, 2024* [arXiv]. CEUR-WS, 2024, vol. 3885, p. 249–258 [visited on 2024-07-29]. Available from DOI: 10.48550/arXiv.2407.19914.
31. HARRIS, Z. S. Distributional Structure. *WORD* Vol. 10, 1954, no. 2-3, p. 146–162. Available from DOI: 10.1080/00437956.1954.11659520.
32. ZIPF, G. K. Human behavior and the principle of least effort. Vol. 1949.
33. ROBERTSON, S. and ZARAGOZA, H. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends® in Information Retrieval* Vol. 3, 2009, no. 4, p. 333–389. ISSN 1554-0669. Available from DOI: 10.1561/15000000019.
34. WU, Y.; SCHUSTER, M.; CHEN, Z.; LE, Q. V.; NOROUZI, M.; MACHEREY, W.; KRIKUN, M.; CAO, Y.; GAO, Q.; MACHEREY, K.; KLINGNER, J.; SHAH, A.; JOHNSON, M.; LIU, X.; KAISER, Ł.; GOUWS, S.; KATO, Y.; KUDO, T.; KAZAWA, H.; STEVENS, K.; KURIAN, G.; PATIL, N.; WANG, W.; YOUNG, C.; SMITH, J.; RIESA, J.; RUDNICK, A.; VINYALS, O.; CORRADO, G.; HUGHES, M. and DEAN, J. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. 2016. Available from arXiv: 1609.08144 [cs.CL]. [Last accessed on 17 December 2025].

35. SENNRICH, R.; HADDOW, B. and BIRCH, A. Neural Machine Translation of Rare Words with Subword Units. In: ERK, K. and SMITH, N. A. (eds.). *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, 2016, p. 1715–1725. Available from DOI: 10.18653/v1/P16-1162.
36. KUDO, T. Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates. In: GUREVYCH, I. and MIYAO, Y. (eds.). *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, 2018, p. 66–75. Available from DOI: 10.18653/v1/P18-1007.
37. KUDO, T. and RICHARDSON, J. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In: BLANCO, E. and LU, W. (eds.). *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Brussels, Belgium: Association for Computational Linguistics, 2018, p. 66–71. Available from DOI: 10.18653/v1/D18-2012.
38. PEARSON, K. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* Vol. 2, 1901, no. 11, p. 559–572. Available from DOI: 10.1080/14786440109462720.
39. HOTELLING, H. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology* Vol. 24, 1933, no. 6, p. 417.
40. AGGARWAL, C. C. and ZHAI, C. *Mining text data*. Springer Science & Business Media, 2012.
41. FURNAS, G. W.; DEERWESTER, S.; DUMAIS, S. T.; LANDAUER, T. K.; HARSHMAN, R. A.; STREETER, L. A. and LOCHBAUM, K. E. Information retrieval using a singular value decomposition model of latent semantic structure. In: *Proceedings of the 11th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 1988, p. 465–480.
42. PAATERO, P. and TAPPER, U. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics* Vol. 5, 1994, no. 2, p. 111–126.
43. LEE, D. D. and SEUNG, H. S. Learning the parts of objects by non-negative matrix factorization. *Nature* Vol. 401, 1999, no. 6755, p. 788.
44. HOFMANN, T. Learning the Similarity of Documents: An Information-Geometric Approach to Document Retrieval and Categorization. In: SOLLA, S.; LEEN, T. and MÜLLER, K. (eds.). *Advances in Neural Information Processing Systems*. MIT Press, 1999, vol. 12. Available also from: https://proceedings.neurips.cc/paper_files/paper/1999/file/9d2682367c3935defcb1f9e247a97c0d-Paper.pdf.
45. BLEI, D. M.; NG, A. Y. and JORDAN, M. I. Latent dirichlet allocation. *Journal of machine Learning research* Vol. 3, 2003, no. Jan, p. 993–1022.
46. BLEI, D. M. Probabilistic Topic Models. *Commun. ACM* Vol. 55, 2012, no. 4, p. 77–84. ISSN 0001-0782. Available from DOI: 10.1145/2133806.2133826.

47. MIKOLOV, T.; YIH, W.-t. and ZWEIG, G. Linguistic Regularities in Continuous Space Word Representations. In: VANDERWENDE, L.; DAUMÉ III, H. and KIRCHHOFF, K. (eds.). *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, 2013, p. 746–751. Available also from: <https://aclanthology.org/N13-1090/>.
48. MORIN, F. and BENGIO, Y. Hierarchical Probabilistic Neural Network Language Model. In: COWELL, R. G. and GHAHRAMANI, Z. (eds.). *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*. PMLR, 2005, vol. R5, p. 246–252. *Proceedings of Machine Learning Research*. Available also from: <https://proceedings.mlr.press/r5/morin05a.html>. Reissued by PMLR on 30 March 2021.
49. GUTMANN, M. U. and HYVÄRINEN, A. Noise-Contrastive Estimation of Unnormalized Statistical Models, with Applications to Natural Image Statistics. *Journal of Machine Learning Research* Vol. 13, 2012, no. 11, p. 307–361. Available also from: <http://jmlr.org/papers/v13/gutmann12a.html>.
50. BOJANOWSKI, P.; GRAVE, E.; JOULIN, A. and MIKOLOV, T. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* Vol. 5, 2017, p. 135–146.
51. MIKOLOV, T.; GRAVE, E.; BOJANOWSKI, P.; PUHRSCHE, C. and JOULIN, A. Advances in Pre-Training Distributed Word Representations. In: CALZOLARI, N.; CHOUKRI, K.; CIERI, C.; DECLERCK, T.; GOGGI, S.; HASIDA, K.; ISAHARA, H.; MAEGAARD, B.; MARIANI, J.; MAZO, H.; MORENO, A.; ODIJK, J.; PIPERIDIS, S. and TOKUNAGA, T. (eds.). *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA), 2018. Available also from: <https://aclanthology.org/L18-1008/>.
52. LUONG, T.; SOCHER, R. and MANNING, C. Better Word Representations with Recursive Neural Networks for Morphology. In: HOCKENMAIER, J. and RIEDEL, S. (eds.). *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. Sofia, Bulgaria: Association for Computational Linguistics, 2013, p. 104–113. Available also from: <https://aclanthology.org/W13-3512/>.
53. YOUNG, T.; HAZARIKA, D.; PORIA, S. and CAMBRIA, E. Recent Trends in Deep Learning Based Natural Language Processing [Review Article]. *IEEE Computational Intelligence Magazine* Vol. 13, 2018, no. 3, p. 55–75. ISSN 1556-6048. Available from DOI: 10.1109/MCI.2018.2840738.
54. CHEN, M. Efficient Vector Representation for Documents through Corruption. In: *International Conference on Learning Representations*. 2017. Available also from: <https://openreview.net/forum?id=B1Igu2ogg>.
55. PAGLIARDINI, M.; GUPTA, P. and JAGGI, M. Unsupervised Learning of Sentence Embeddings Using Compositional n-Gram Features. In: WALKER, M.; JI, H. and STENT, A. (eds.). *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, 2018, p. 528–540. Available from DOI: 10.18653/v1/N18-1049.

56. HILL, F.; CHO, K. and KORHONEN, A. Learning Distributed Representations of Sentences from Unlabelled Data. In: KNIGHT, K.; NENKOVA, A. and RAMBOW, O. (eds.). *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, 2016, p. 1367–1377. Available from DOI: 10.18653/v1/N16-1162.
57. ELMAN, J. L. Finding structure in time. *Cognitive Science* Vol. 14, 1990, no. 2, p. 179–211. ISSN 0364-0213. Available from DOI: [https://doi.org/10.1016/0364-0213\(90\)90002-E](https://doi.org/10.1016/0364-0213(90)90002-E).
58. BENGIO, Y.; SIMARD, P. and FRASCONI, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks* Vol. 5, 1994, no. 2, p. 157–166. Available from DOI: 10.1109/72.279181.
59. HOCHREITER, S. and SCHMIDHUBER, J. Long Short-Term Memory. *Neural Computation* Vol. 9, 1997, no. 8, p. 1735–1780. Available from DOI: 10.1162/neco.1997.9.8.1735.
60. HE, K.; ZHANG, X.; REN, S. and SUN, J. Deep Residual Learning for Image Recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.
61. SCHUSTER, M. and PALIWAL, K. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* Vol. 45, 1997, no. 11, p. 2673–2681. Available from DOI: 10.1109/78.650093.
62. BAHDANAU, D.; CHO, K. and BENGIO, Y. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016. Available from arXiv: 1409.0473 [cs.CL]. [Last accessed on 17 December 2025].
63. SUTSKEVER, I.; VINYALS, O. and LE, Q. V. Sequence to Sequence Learning with Neural Networks. In: GHAHRAMANI, Z.; WELLING, M.; CORTES, C.; LAWRENCE, N. and WEINBERGER, K. (eds.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2014, vol. 27. Available also from: https://proceedings.neurips.cc/paper_files/paper/2014/file/5a18e133cbf9f257297f410bb7eca942-Paper.pdf.
64. LOGESWARAN, L. and LEE, H. An efficient framework for learning sentence representations. In: *International Conference on Learning Representations*. 2018. Available also from: <https://openreview.net/forum?id=rJvJXZb0W>.
65. LOGESWARAN, L.; LEE, H. and RADEV, D. *Sentence Ordering using Recurrent Neural Networks*. 2017. Available also from: <https://openreview.net/forum?id=S1AG8zYeg>. [Last accessed on 17 December 2025].
66. JERNITE, Y.; BOWMAN, S. R. and SONTAG, D. *Discourse-Based Objectives for Fast Unsupervised Sentence Representation Learning*. 2017. Available from arXiv: 1705.00557 [cs.CL]. [Last accessed on 17 December 2025].
67. NIE, A.; BENNETT, E. and GOODMAN, N. DisSent: Learning Sentence Representations from Explicit Discourse Relations. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, 2019, p. 4497–4510. Available from DOI: 10.18653/v1/P19-1442.

68. CONNEAU, A.; KIELA, D.; SCHWENK, H.; BARRAULT, L. and BORDES, A. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, 2017, p. 670–680. Available from DOI: 10.18653/v1/D17-1070.
69. SUBRAMANIAN, S.; TRISCHLER, A.; BENGIO, Y. and PAL, C. J. Learning General Purpose Distributed Sentence Representations via Large Scale Multi-task Learning. In: *International Conference on Learning Representations*. 2018. Available also from: <https://openreview.net/forum?id=B18WgG-CZ>.
70. ARTETXE, M. and SCHWENK, H. Massively Multilingual Sentence Embeddings for Zero-Shot Cross-Lingual Transfer and Beyond. *Transactions of the Association for Computational Linguistics* Vol. 7, 2019, p. 597–610. Available from DOI: 10.1162/tac1_a_00288.
71. PETERS, M. E.; AMMAR, W.; BHAGAVATULA, C. and POWER, R. Semi-supervised sequence tagging with bidirectional language models. In: BARZILAY, R. and KAN, M.-Y. (eds.). *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, 2017, p. 1756–1765. Available from DOI: 10.18653/v1/P17-1161.
72. KIM, Y.; JERNITE, Y.; SONTAG, D. and RUSH, A. M. Character-aware neural language models. In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. Phoenix, Arizona: AAAI Press, 2016, p. 2741–2749. AAAI’16.
73. JOZEFOWICZ, R.; VINYALS, O.; SCHUSTER, M.; SHAZEER, N. and WU, Y. *Exploring the Limits of Language Modeling*. 2016. Available from arXiv: 1602.02410 [cs.CL]. [Last accessed on 2025-03-07].
74. SRIVASTAVA, R. K.; GREFF, K. and SCHMIDHUBER, J. Training Very Deep Networks. In: CORTES, C.; LAWRENCE, N.; LEE, D.; SUGIYAMA, M. and GARNETT, R. (eds.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2015, vol. 28. Available also from: https://proceedings.neurips.cc/paper_files/paper/2015/file/215a71a12769b056c3c32e7299f1c5ed-Paper.pdf.
75. ERHAN, D.; BENGIO, Y.; COURVILLE, A.; MANZAGOL, P.-A.; VINCENT, P. and BENGIO, S. Why Does Unsupervised Pre-training Help Deep Learning? *Journal of Machine Learning Research* Vol. 11, 2010, no. 19, p. 625–660. Available also from: <http://jmlr.org/papers/v11/erhan10a.html>.
76. YOSINSKI, J.; CLUNE, J.; BENGIO, Y. and LIPSON, H. How transferable are features in deep neural networks? In: *Advances in neural information processing systems*. 2014, p. 3320–3328.
77. MERITY, S.; KESKAR, N. S. and SOCHER, R. Regularizing and Optimizing LSTM Language Models. In: *International Conference on Learning Representations*. 2018. Available also from: <https://openreview.net/forum?id=SyyGPP0TZ>.

78. EISENSCHLOS, J.; RUDER, S.; CZAPLA, P.; KADRAS, M.; GUGGER, S. and HOWARD, J. MultiFiT: Efficient Multi-lingual Language Model Fine-tuning. In: INUI, K.; JIANG, J.; NG, V. and WAN, X. (eds.). *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, 2019, p. 5702–5707. Available from DOI: 10.18653/v1/D19-1572.
79. BRADBURY, J.; MERITY, S.; XIONG, C. and SOCHER, R. Quasi-Recurrent Neural Networks. In: *International Conference on Learning Representations*. 2017. Available also from: <https://openreview.net/forum?id=H1zJ-v5x1>.
80. MERITY, S.; KESKAR, N. S. and SOCHER, R. *An Analysis of Neural Language Modeling at Multiple Scales*. 2018. Available from arXiv: 1803.08240 [cs.CL]. [Last accessed on 2025-03-07].
81. SMITH, L. N. and TOPIN, N. Super-convergence: very fast training of neural networks using large learning rates. In: PHAM, T. (ed.). *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*. SPIE, International Society for Optics and Photonics, 2019, vol. 11006, p. 1100612. Available from DOI: 10.1117/12.2520589.
82. SZEGEDY, C.; VANHOUCKE, V.; IOFFE, S.; SHLENS, J. and WOJNA, Z. Rethinking the Inception Architecture for Computer Vision. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.
83. VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L. and POLOSUKHIN, I. Attention is All you Need. In: GUYON, I.; LUXBURG, U. V.; BENGIO, S.; WALLACH, H.; FERGUS, R.; VISHWANATHAN, S. and GARNETT, R. (eds.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017, vol. 30. Available also from: <https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf>.
84. DEEPSEEK-AI. *DeepSeek-V3 Technical Report*. 2024. Available from arXiv: 2412.19437 [cs.CL]. [Last accessed on 2025-03-07].
85. TAY, Y.; DEGHANI, M.; BAHRI, D. and METZLER, D. Efficient Transformers: A Survey. *ACM Comput. Surv.* Vol. 55, 2022, no. 6. ISSN 0360-0300. Available from DOI: 10.1145/3530811.
86. NARANG, S.; CHUNG, H. W.; TAY, Y.; FEDUS, L.; FEVRY, T.; MATENA, M.; MALKAN, K.; FIEDEL, N.; SHAZEER, N.; LAN, Z.; ZHOU, Y.; LI, W.; DING, N.; MARCUS, J.; ROBERTS, A. and RAFFEL, C. Do Transformer Modifications Transfer Across Implementations and Applications? In: MOENS, M.-F.; HUANG, X.; SPECIA, L. and YIH, S. W.-t. (eds.). *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021, p. 5758–5773. Available from DOI: 10.18653/v1/2021.emnlp-main.465.

87. SHAW, P.; USZKOREIT, J. and VASWANI, A. Self-Attention with Relative Position Representations. In: WALKER, M.; JI, H. and STENT, A. (eds.). *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, 2018, p. 464–468. Available from DOI: 10.18653/v1/N18-2074.
88. HUANG, C.-Z. A.; VASWANI, A.; USZKOREIT, J.; SIMON, I.; HAWTHORNE, C.; SHAZEER, N.; DAI, A. M.; HOFFMAN, M. D.; DINCULESCU, M. and ECK, D. Music Transformer. In: *International Conference on Learning Representations*. 2019. Available also from: <https://openreview.net/forum?id=rJe4ShAcF7>.
89. WANG, A.; SINGH, A.; MICHAEL, J.; HILL, F.; LEVY, O. and BOWMAN, S. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In: LINZEN, T.; CHRUPAŁA, G. and ALISHAHI, A. (eds.). *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Brussels, Belgium: Association for Computational Linguistics, 2018, p. 353–355. Available from DOI: 10.18653/v1/W18-5446.
90. YANG, Z.; DAI, Z.; YANG, Y.; CARBONELL, J.; SALAKHUTDINOV, R. R. and LE, Q. V. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In: WALLACH, H.; LAROCHELLE, H.; BEYGELZIMER, A.; D’ALCHÉ-BUC, F.; FOX, E. and GARNETT, R. (eds.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019, vol. 32. Available also from: https://proceedings.neurips.cc/paper_files/paper/2019/file/dc6a7e655d7e5840e66733e9ee67cc69-Paper.pdf.
91. DAI, Z.; YANG, Z.; YANG, Y.; CARBONELL, J.; LE, Q. and SALAKHUTDINOV, R. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In: KORHONEN, A.; TRAUM, D. and MÁRQUEZ, L. (eds.). *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, 2019, p. 2978–2988. Available from DOI: 10.18653/v1/P19-1285.
92. LIU, Y.; OTT, M.; GOYAL, N.; DU, J.; JOSHI, M.; CHEN, D.; LEVY, O.; LEWIS, M.; ZETTLEMOYER, L. and STOYANOV, V. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. Available from arXiv: 1907.11692 [cs.CL]. [Last accessed on 2025-03-07].
93. LAN, Z.; CHEN, M.; GOODMAN, S.; GIMPEL, K.; SHARMA, P. and SORICUT, R. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In: *International Conference on Learning Representations*. 2020. Available also from: <https://openreview.net/forum?id=H1eA7AEtvS>.
94. CLARK, K.; LUONG, M.-T.; LE, Q. V. and MANNING, C. D. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In: *International Conference on Learning Representations*. 2020. Available also from: <https://openreview.net/forum?id=r1xMH1BtvB>.
95. LIU, P.; YUAN, W.; FU, J.; JIANG, Z.; HAYASHI, H. and NEUBIG, G. Pre-Train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Computing Surveys* Vol. 55, 2023, no. 9. ISSN 0360-0300. Available from DOI: 10.1145/3560815.

96. WEI, J.; BOSMA, M.; ZHAO, V.; GUU, K.; YU, A. W.; LESTER, B.; DU, N.; DAI, A. M. and LE, Q. V. Finetuned Language Models are Zero-Shot Learners. In: *International Conference on Learning Representations*. 2022. Available also from: <https://openreview.net/forum?id=gEZrGCozdqR>.
97. YAN, Y.; LI, R.; WANG, S.; ZHANG, F.; WU, W. and XU, W. ConSERT: A Contrastive Framework for Self-Supervised Sentence Representation Transfer. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, 2021, p. 5065–5075. Available from DOI: 10.18653/v1/2021.acl-long.393.
98. SU, J.; CAO, J.; LIU, W. and OU, Y. *Whitening Sentence Representations for Better Semantics and Faster Retrieval*. arXiv, 2021. Available from DOI: 10.48550/ARXIV.2103.15316. [Last accessed on 2025-03-07].
99. HUANG, J.; TANG, D.; ZHONG, W.; LU, S.; SHOU, L.; GONG, M.; JIANG, D. and DUAN, N. WhiteningBERT: An Easy Unsupervised Sentence Embedding Approach. In: *Findings of the Association for Computational Linguistics: EMNLP 2021*. Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021, p. 238–244. Available from DOI: 10.18653/v1/2021.findings-emnlp.23.
100. XU, L.; XIE, H.; LI, Z.; WANG, F. L.; WANG, W. and LI, Q. Contrastive Learning Models for Sentence Representations. *ACM Trans. Intell. Syst. Technol.* Vol. 14, 2023, no. 4. ISSN 2157-6904. Available from DOI: 10.1145/3593590.
101. CER, D.; YANG, Y.; KONG, S.-y.; HUA, N.; LIMTIACO, N.; ST. JOHN, R.; CONSTANT, N.; GUAJARDO-CESPEDES, M.; YUAN, S.; TAR, C.; STROPE, B. and KURZWEIL, R. Universal Sentence Encoder for English. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Brussels, Belgium: Association for Computational Linguistics, 2018, p. 169–174. Available from DOI: 10.18653/v1/D18-2029.
102. YANG, Y.; YUAN, S.; CER, D.; KONG, S.-y.; CONSTANT, N.; PILAR, P.; GE, H.; SUNG, Y.-H.; STROPE, B. and KURZWEIL, R. Learning Semantic Textual Similarity from Conversations. In: AUGENSTEIN, I.; CAO, K.; HE, H.; HILL, F.; GELLA, S.; KIROS, J.; MEI, H. and MISRA, D. (eds.). *Proceedings of the Third Workshop on Representation Learning for NLP*. Melbourne, Australia: Association for Computational Linguistics, 2018, p. 164–174. Available from DOI: 10.18653/v1/W18-3022.
103. CONNEAU, A. and KIELA, D. SentEval: An Evaluation Toolkit for Universal Sentence Representations. In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA), 2018. Available also from: <https://aclanthology.org/L18-1269>.
104. WANG, K.; REIMERS, N. and GUREVYCH, I. TSDAE: Using Transformer-based Sequential Denoising Auto-Encoder for Unsupervised Sentence Embedding Learning. In: MOENS, M.-F.; HUANG, X.; SPECIA, L. and YIH, S. W.-t. (eds.). *Findings of the Association for Computational Linguistics: EMNLP 2021*. Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021, p. 671–688. Available from DOI: 10.18653/v1/2021.findings-emnlp.59.

105. WU, B. and ZHAO, H. Sentence Representation Learning with Generative Objective rather than Contrastive Objective. In: GOLDBERG, Y.; KOZAREVA, Z. and ZHANG, Y. (eds.). *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, 2022, p. 3356–3368. Available from DOI: 10.18653/v1/2022.emnlp-main.221.
106. ROSE, S.; ENGEL, D.; CRAMER, N. and COWLEY, W. Automatic Keyword Extraction from Individual Documents. In: *Text Mining*. John Wiley & Sons, Ltd, 2010, chap. 1, p. 1–20. ISBN 9780470689646. Available from DOI: <https://doi.org/10.1002/9780470689646.ch1>.
107. HADSELL, R.; CHOPRA, S. and LECUN, Y. Dimensionality Reduction by Learning an Invariant Mapping. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. 2006, vol. 2, p. 1735–1742. Available from DOI: 10.1109/CVPR.2006.100.
108. SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I. and SALAKHUTDINOV, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* Vol. 15, 2014, no. 56, p. 1929–1958. Available also from: <http://jmlr.org/papers/v15/srivastava14a.html>.
109. WU, X.; GAO, C.; ZANG, L.; HAN, J.; WANG, Z. and HU, S. ESIMCSE: Enhanced Sample Building Method for Contrastive Learning of Unsupervised Sentence Embedding. In: CALZOLARI, N.; HUANG, C.-R.; KIM, H.; PUSTEJOVSKY, J.; WANNER, L.; CHOI, K.-S.; RYU, P.-M.; CHEN, H.-H.; DONATELLI, L.; JI, H.; KUROHASHI, S.; PAGGIO, P.; XUE, N.; KIM, S.; HAHM, Y.; HE, Z.; LEE, T. K.; SANTUS, E.; BOND, F. and NA, S.-H. (eds.). *Proceedings of the 29th International Conference on Computational Linguistics*. Gyeongju, Republic of Korea: International Committee on Computational Linguistics, 2022, p. 3898–3907. Available also from: <https://aclanthology.org/2022.coling-1.342/>.
110. KLEIN, T. and NABI, M. SCD: Self-Contrastive Decorrelation of Sentence Embeddings. In: MURESAN, S.; NAKOV, P. and VILLAVICENCIO, A. (eds.). *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Dublin, Ireland: Association for Computational Linguistics, 2022, p. 394–400. Available from DOI: 10.18653/v1/2022.acl-short.44.
111. JIANG, Y.; ZHANG, L. and WANG, W. Improved Universal Sentence Embeddings with Prompt-based Contrastive Learning and Energy-based Learning. In: GOLDBERG, Y.; KOZAREVA, Z. and ZHANG, Y. (eds.). *Findings of the Association for Computational Linguistics: EMNLP 2022*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, 2022, p. 3021–3035. Available from DOI: 10.18653/v1/2022.findings-emnlp.220.
112. ZHUO, W.; SUN, Y.; WANG, X.; ZHU, L. and YANG, Y. WhitenedCSE: Whitening-based Contrastive Learning of Sentence Embeddings. In: ROGERS, A.; BOYD-GRABER, J. and OKAZAKI, N. (eds.). *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Toronto, Canada: Association for Computational Linguistics, 2023, p. 12135–12148. Available from DOI: 10.18653/v1/2023.acl-long.677.

113. CHUANG, Y.-S.; DANGOVSKI, R.; LUO, H.; ZHANG, Y.; CHANG, S.; SOLJACIC, M.; LI, S.-W.; YIH, S.; KIM, Y. and GLASS, J. DiffCSE: Difference-based Contrastive Learning for Sentence Embeddings. In: CARPUAT, M.; MARNEFFE, M.-C. de and MEZA RUIZ, I. V. (eds.). *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Seattle, United States: Association for Computational Linguistics, 2022, p. 4207–4218. Available from DOI: 10.18653/v1/2022.naacl-main.311.
114. LIU, J.; LIU, J.; WANG, Q.; WANG, J.; WU, W.; XIAN, Y.; ZHAO, D.; CHEN, K. and YAN, R. RankCSE: Unsupervised Sentence Representations Learning via Learning to Rank. In: ROGERS, A.; BOYD-GRABER, J. and OKAZAKI, N. (eds.). *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Toronto, Canada: Association for Computational Linguistics, 2023, p. 13785–13802. Available from DOI: 10.18653/v1/2023.acl-long.771.
115. LI, X. and LI, J. AoE: Angle-optimized Embeddings for Semantic Textual Similarity. In: KU, L.-W.; MARTINS, A. and SRIKUMAR, V. (eds.). *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Bangkok, Thailand: Association for Computational Linguistics, 2024, p. 1825–1839. Available from DOI: 10.18653/v1/2024.acl-long.101.
116. ZHANG, Y.; HE, R.; LIU, Z.; BING, L. and LI, H. Bootstrapped Unsupervised Sentence Representation Learning. In: ZONG, C.; XIA, F.; LI, W. and NAVIGLI, R. (eds.). *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, 2021, p. 5168–5180. Available from DOI: 10.18653/v1/2021.acl-long.402.
117. FANG, H.; WANG, S.; ZHOU, M.; DING, J. and XIE, P. *CERT: Contrastive Self-supervised Learning for Language Understanding*. arXiv, 2020. Available from DOI: 10.48550/ARXIV.2005.12766. [Last accessed on 17 December 2025].
118. ROZSA, A.; RUDD, E. M. and BOULT, T. E. Adversarial diversity and hard positive generation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2016, p. 25–32.
119. CAO, R.; WANG, Y.; LIANG, Y.; GAO, L.; ZHENG, J.; REN, J. and WANG, Z. Exploring the Impact of Negative Samples of Contrastive Learning: A Case Study of Sentence Embedding. In: MURESAN, S.; NAKOV, P. and VILLAVICENCIO, A. (eds.). *Findings of the Association for Computational Linguistics: ACL 2022*. Dublin, Ireland: Association for Computational Linguistics, 2022, p. 3138–3152. Available from DOI: 10.18653/v1/2022.findings-acl.248.
120. SU, P.; PENG, Y. and VIJAY-SHANKER, K. Improving BERT Model Using Contrastive Learning for Biomedical Relation Extraction. In: *Proceedings of the 20th Workshop on Biomedical Language Processing*. Online: Association for Computational Linguistics, 2021, p. 1–10. Available from DOI: 10.18653/v1/2021.bionlp-1.1.
121. LIU, F.; VULIĆ, I.; KORHONEN, A. and COLLIER, N. Fast, Effective, and Self-Supervised: Transforming Masked Language Models into Universal Lexical and Sentence Encoders. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021, p. 1442–1459. Available from DOI: 10.18653/v1/2021.emnlp-main.109.

122. CARLSSON, F.; GYLLENSTEN, A. C.; GOGOULOU, E.; HELLQVIST, E. Y. and SAHLGREN, M. Semantic re-tuning with contrastive tension. In: *International Conference on Learning Representations*. 2020.
123. KIM, T.; YOO, K. M. and LEE, S.-g. Self-Guided Contrastive Learning for BERT Sentence Representations. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, 2021, p. 2528–2540. Available from DOI: 10.18653/v1/2021.acl-long.197.
124. WU, X.; GAO, C.; WANG, J.; ZANG, L.; WANG, Z. and HU, S. *DisCo: Effective Knowledge Distillation For Contrastive Learning of Sentence Embeddings*. arXiv, 2021. Available from DOI: 10.48550/ARXIV.2112.05638. [Last accessed on 17 December 2025].
125. WANG, H. and DOU, Y. SNCSE: Contrastive Learning for Unsupervised Sentence Embedding with Soft Negative Samples. In: HUANG, D.-S.; PREMARATNE, P.; JIN, B.; QU, B.; JO, K.-H. and HUSSAIN, A. (eds.). *Advanced Intelligent Computing Technology and Applications*. Singapore: Springer Nature Singapore, 2023, p. 419–431. ISBN 978-981-99-4752-2.
126. WU, Q.; TAO, C.; SHEN, T.; XU, C.; GENG, X. and JIANG, D. PCL: Peer-Contrastive Learning with Diverse Augmentations for Unsupervised Sentence Embeddings. In: GOLDBERG, Y.; KOZAREVA, Z. and ZHANG, Y. (eds.). *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, 2022, p. 12052–12066. Available from DOI: 10.18653/v1/2022.emnlp-main.826.
127. GIORGI, J.; NITSKI, O.; WANG, B. and BADER, G. DeCLUTR: Deep Contrastive Learning for Unsupervised Textual Representations. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, 2021, p. 879–895. Available from DOI: 10.18653/v1/2021.acl-long.72.
128. NI, J.; HERNANDEZ ABREGO, G.; CONSTANT, N.; MA, J.; HALL, K.; CER, D. and YANG, Y. Sentence-T5: Scalable Sentence Encoders from Pre-trained Text-to-Text Models. In: MURESAN, S.; NAKOV, P. and VILLAVICENCIO, A. (eds.). *Findings of the Association for Computational Linguistics: ACL 2022*. Dublin, Ireland: Association for Computational Linguistics, 2022, p. 1864–1874. Available from DOI: 10.18653/v1/2022.findings-acl.146.
129. RUDER, S. *Challenges and Opportunities in NLP Benchmarking* [<http://ruder.io/nlp-benchmarking>]. 2021. [Last accessed on 17 December 2025].
130. AGIRRE, E.; CER, D.; DIAB, M. and GONZALEZ-AGIRRE, A. SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity. In: **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*. Montréal, Canada: Association for Computational Linguistics, 2012, p. 385–393. Available also from: <https://aclanthology.org/S12-1051>.

131. AGIRRE, E.; CER, D.; DIAB, M.; GONZALEZ-AGIRRE, A. and GUO, W. *SEM 2013 shared task: Semantic Textual Similarity. In: *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*. Atlanta, Georgia, USA: Association for Computational Linguistics, 2013, p. 32–43. Available also from: <https://aclanthology.org/S13-1004>.
132. AGIRRE, E.; BANE, C.; CARDIE, C.; CER, D.; DIAB, M.; GONZALEZ-AGIRRE, A.; GUO, W.; MIHALCEA, R.; RIGAU, G. and WIEBE, J. SemEval-2014 Task 10: Multilingual Semantic Textual Similarity. In: *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Dublin, Ireland: Association for Computational Linguistics, 2014, p. 81–91. Available from DOI: 10.3115/v1/S14-2010.
133. AGIRRE, E.; BANE, C.; CARDIE, C.; CER, D.; DIAB, M.; GONZALEZ-AGIRRE, A.; GUO, W.; LOPEZ-GAZPIO, I.; MARITXALAR, M.; MIHALCEA, R.; RIGAU, G.; URI, L. and WIEBE, J. SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability. In: *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Denver, Colorado: Association for Computational Linguistics, 2015, p. 252–263. Available from DOI: 10.18653/v1/S15-2045.
134. AGIRRE, E.; BANE, C.; CER, D.; DIAB, M.; GONZALEZ-AGIRRE, A.; MIHALCEA, R.; RIGAU, G. and WIEBE, J. SemEval-2016 Task 1: Semantic Textual Similarity, Monolingual and Cross-Lingual Evaluation. In: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. San Diego, California: Association for Computational Linguistics, 2016, p. 497–511. Available from DOI: 10.18653/v1/S16-1081.
135. EGER, S.; RÜCKLÉ, A. and GUREVYCH, I. Pitfalls in the Evaluation of Sentence Embeddings. In: *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*. Florence, Italy: Association for Computational Linguistics, 2019, p. 55–60. Available from DOI: 10.18653/v1/W19-4308.
136. WANG, B.; KUO, C.-C. J. and LI, H. Just Rank: Rethinking Evaluation with Word and Sentence Similarities. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, 2022, p. 6060–6077. Available from DOI: 10.18653/v1/2022.acl-long.419.
137. BENTIVOGLI, L.; DAGAN, I.; DANG, H. T.; GIAMPICCOLO, D. and MAGNINI, B. The Fifth PASCAL Recognizing Textual Entailment Challenge. In: *In Proc Text Analysis Conference (TAC'09)*. 2009.
138. POLIAK, A.; HALDAR, A.; RUDINGER, R.; HU, J. E.; PAVLICK, E.; WHITE, A. S. and VAN DURME, B. Collecting Diverse Natural Language Inference Problems for Sentence Representation Evaluation. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, 2018, p. 67–81. Available from DOI: 10.18653/v1/D18-1007.

139. NIE, Y.; WILLIAMS, A.; DINAN, E.; BANSAL, M.; WESTON, J. and KIELA, D. Adversarial NLI: A New Benchmark for Natural Language Understanding. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, 2020, p. 4885–4901. Available from DOI: 10.18653/v1/2020.acl-main.441.
140. BUDANITSKY, A. and HIRST, G. Evaluating WordNet-based Measures of Lexical Semantic Relatedness. *Computational Linguistics* Vol. 32, 2006, no. 1, p. 13–47. Available from DOI: 10.1162/coli.2006.32.1.13.
141. MARELLI, M.; MENINI, S.; BARONI, M.; BENTIVOGLI, L.; BERNARDI, R. and ZAMPARELLI, R. A SICK cure for the evaluation of compositional distributional semantic models. In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. Reykjavik, Iceland: European Language Resources Association (ELRA), 2014, p. 216–223. Available also from: http://www.lrec-conf.org/proceedings/lrec2014/pdf/363_Paper.pdf.
142. ZELLERS, R.; BISK, Y.; SCHWARTZ, R. and CHOI, Y. SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, 2018, p. 93–104. Available from DOI: 10.18653/v1/D18-1009.
143. ZELLERS, R.; HOLTZMAN, A.; BISK, Y.; FARHADI, A. and CHOI, Y. HellaSwag: Can a Machine Really Finish Your Sentence? In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, 2019, p. 4791–4800. Available from DOI: 10.18653/v1/P19-1472.
144. RAJPURKAR, P.; ZHANG, J.; LOPYREV, K. and LIANG, P. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, 2016, p. 2383–2392. Available from DOI: 10.18653/v1/D16-1264.
145. CHEN, Z.; ZHANG, H.; ZHANG, X. and ZHAO, L. *Quora question pairs*. 2018. [Last accessed on 17 December 2025].
146. DOLAN, W. B. and BROCKETT, C. Automatically Constructing a Corpus of Sentential Paraphrases. In: *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*. 2005. Available also from: <https://aclanthology.org/I05-5002>.
147. WIETING, J. and KIELA, D. No Training Required: Exploring Random Encoders for Sentence Classification. In: *International Conference on Learning Representations*. 2019. Available also from: <https://openreview.net/forum?id=BkgPajAcY7>.
148. WARSTADT, A.; SINGH, A. and BOWMAN, S. R. Neural Network Acceptability Judgments. *Transactions of the Association for Computational Linguistics* Vol. 7, 2019, p. 625–641. Available from DOI: 10.1162/tac1_a_00290.

149. SOCHER, R.; PERELYGIN, A.; WU, J.; CHUANG, J.; MANNING, C. D.; NG, A. and POTTS, C. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, 2013, p. 1631–1642. Available also from: <https://aclanthology.org/D13-1170>.
150. WANG, A.; PRUKSACHATKUN, Y.; NANGIA, N.; SINGH, A.; MICHAEL, J.; HILL, F.; LEVY, O. and BOWMAN, S. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. In: WALLACH, H.; LAROCHELLE, H.; BEYGEZLIMER, A.; ALCHÉ-BUC, F. d'; FOX, E. and GARNETT, R. (eds.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019, vol. 32. Available also from: <https://proceedings.neurips.cc/paper/2019/file/4496bf24afe7fab6f046bf4923da8de6-Paper.pdf>.
151. KIELA, D.; BARTOLO, M.; NIE, Y.; KAUSHIK, D.; GEIGER, A.; WU, Z.; VIDGEN, B.; PRASAD, G.; SINGH, A.; RINGSHIA, P.; MA, Z.; THRUSH, T.; RIEDEL, S.; WASEEM, Z.; STENETORP, P.; JIA, R.; BANSAL, M.; POTTS, C. and WILLIAMS, A. Dynabench: Rethinking Benchmarking in NLP. In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, 2021, p. 4110–4124. Available from DOI: 10.18653/v1/2021.naacl-main.324.
152. GEHRMANN, S.; ADEWUMI, T.; AGGARWAL, K.; AMMANAMANCHI, P. S.; AREMU, A.; BOSSELUT, A.; CHANDU, K. R.; CLINCIU, M.-A.; DAS, D.; DHOLE, K.; DU, W.; DURMUS, E.; DUŠEK, O.; EMEZUE, C. C.; GANGAL, V.; GARBACEA, C.; HASHIMOTO, T.; HOU, Y.; JERNITE, Y.; JHAMTANI, H.; JI, Y.; JOLLY, S.; KALE, M.; KUMAR, D.; LADHAK, F.; MADAAAN, A.; MADDELA, M.; MAHAJAN, K.; MAHAMOOD, S.; MAJUMDER, B. P.; MARTINS, P. H.; MCMILLAN-MAJOR, A.; MILLE, S.; MILTENBURG, E. van; NADEEM, M.; NARAYAN, S.; NIKOLAEV, V.; NIYONGABO RUBUNGO, A.; OSEI, S.; PARIKH, A.; PEREZ-BELTRACHINI, L.; RAO, N. R.; RAUNAK, V.; RODRIGUEZ, J. D.; SANTHANAM, S.; SEDOC, J.; SELLAM, T.; SHAIKH, S.; SHIMORINA, A.; SOBREVILLA CABEZUDO, M. A.; STROBELT, H.; SUBRAMANI, N.; XU, W.; YANG, D.; YERUKOLA, A. and ZHOU, J. The GEM Benchmark: Natural Language Generation, its Evaluation and Metrics. In: *Proceedings of the 1st Workshop on Natural Language Generation, Evaluation, and Metrics (GEM 2021)*. Online: Association for Computational Linguistics, 2021, p. 96–120. Available from DOI: 10.18653/v1/2021.gem-1.10.
153. GAO, L.; TOW, J.; BIDERMAN, S.; BLACK, S.; DIPOFI, A.; FOSTER, C.; GOLDING, L.; HSU, J.; MCDONELL, K.; MUENNIGHOFF, N.; PHANG, J.; REYNOLDS, L.; TANG, E.; THITE, A.; WANG, B.; WANG, K. and ZOU, A. *A framework for few-shot language model evaluation*. Zenodo, 2021. Version v0.0.1. Available from DOI: 10.5281/zenodo.5371628. [Last accessed on 17 December 2025].
154. HENDRYCKS, D.; BURNS, C.; BASART, S.; ZOU, A.; MAZEIKA, M.; SONG, D. and STEINHARDT, J. Measuring Massive Multitask Language Understanding. In: *International Conference on Learning Representations*. 2021. Available also from: <https://openreview.net/forum?id=d7KBjmI3GmQ>.

155. MCCANN, B.; KESKAR, N. S.; XIONG, C. and SOCHER, R. *The Natural Language Decathlon: Multitask Learning as Question Answering*. arXiv, 2018. Available from DOI: 10.48550/ARXIV.1806.08730. [Last accessed on 1 April 2026].
156. HOFFMANN, J.; BORGEAUD, S.; MENSCH, A.; BUCHATSKAYA, E.; CAI, T.; RUTHERFORD, E.; LAS CASAS, D. de; HENDRICKS, L. A.; WELBL, J.; CLARK, A.; HENNIGAN, T.; NOLAND, E.; MILLICAN, K.; DRIESSCHE, G. van den; DAMOC, B.; GUY, A.; OSINDERO, S.; SIMONYAN, K.; ELSEN, E.; VINYALS, O.; RAE, J. and SIFRE, L. An empirical analysis of compute-optimal large language model training. In: KOYEJO, S.; MOHAMED, S.; AGARWAL, A.; BELGRAVE, D.; CHO, K. and OH, A. (eds.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2022, vol. 35, p. 30016–30030. Available also from: https://proceedings.neurips.cc/paper_files/paper/2022/file/c1e2faff6f588870935f114ebe04a3e5-Paper-Conference.pdf.
157. WANG, Y.; MA, X.; ZHANG, G.; NI, Y.; CHANDRA, A.; GUO, S.; REN, W.; ARULRAJ, A.; HE, X.; JIANG, Z.; LI, T.; KU, M.; WANG, K.; ZHUANG, A.; FAN, R.; YUE, X. and CHEN, W. MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark. In: *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. 2024. Available also from: <https://openreview.net/forum?id=y10DM6R2r3>.
158. HU, J.; RUDER, S.; SIDDHANT, A.; NEUBIG, G.; FIRAT, O. and JOHNSON, M. XTREME: A Massively Multilingual Multi-task Benchmark for Evaluating Cross-lingual Generalisation. In: III, H. D. and SINGH, A. (eds.). *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 2020, vol. 119, p. 4411–4421. Proceedings of Machine Learning Research. Available also from: <https://proceedings.mlr.press/v119/hu20b.html>.
159. LIANG, Y.; DUAN, N.; GONG, Y.; WU, N.; GUO, F.; QI, W.; GONG, M.; SHOU, L.; JIANG, D.; CAO, G.; FAN, X.; ZHANG, R.; AGRAWAL, R.; CUI, E.; WEI, S.; BHARTI, T.; QIAO, Y.; CHEN, J.-H.; WU, W.; LIU, S.; YANG, F.; CAMPOS, D.; MAJUMDER, R. and ZHOU, M. XGLUE: A New Benchmark Dataset for Cross-lingual Pre-training, Understanding and Generation. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, 2020, p. 6008–6018. Available from DOI: 10.18653/v1/2020.emnlp-main.484.
160. BELINKOV, Y. Probing Classifiers: Promises, Shortcomings, and Advances. *Computational Linguistics* Vol. 48, 2022, no. 1, p. 207–219. ISSN 0891-2017. Available from DOI: 10.1162/coli_a_00422.
161. CONNEAU, A.; KRUSZEWSKI, G.; LAMPLE, G.; BARRAULT, L. and BARONI, M. What you can cram into a single $\mathbb{R}^d$ vector: Probing sentence embeddings for linguistic properties. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, 2018, p. 2126–2136. Available from DOI: 10.18653/v1/P18-1198.
162. BELINKOV, Y. and GLASS, J. Analysis Methods in Neural Language Processing: A Survey. *Transactions of the Association for Computational Linguistics* Vol. 7, 2019, p. 49–72. ISSN 2307-387X. Available from DOI: 10.1162/tac1_a_00254.

163. FRANKLAND, S. M. and GREENE, J. D. An architecture for encoding sentence meaning in left mid-superior temporal cortex. *Proceedings of the National Academy of Sciences of the United States of America* Vol. 112, 2015, no. 37, p. 11732–11737. Available from DOI: 10.1073/pnas.1421236112.
164. ETTINGER, A.; ELGOHARY, A. and RESNIK, P. Probing for semantic evidence of composition by means of simple classification tasks. In: *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*. Berlin, Germany: Association for Computational Linguistics, 2016, p. 134–139. Available from DOI: 10.18653/v1/W16-2524.
165. ADI, Y.; KERMANY, E.; BELINKOV, Y.; LAVI, O. and GOLDBERG, Y. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In: *International Conference on Learning Representations*. 2017. Available also from: <https://openreview.net/forum?id=BJh6Ztuxl>.
166. ALAIN, G. and BENGIO, Y. Understanding intermediate layers using linear classifier probes. In: *5th International Conference on Learning Representations*. 2017. Available also from: <https://openreview.net/forum?id=HJ4-rAVt1>.
167. VELDHOFEN, S.; HUPKES, D. and ZUIDEMA, W. H. Diagnostic classifiers revealing how neural networks process hierarchical structure. In: *CoCo@ NIPS*. 2016.
168. ETTINGER, A.; ELGOHARY, A.; PHILLIPS, C. and RESNIK, P. Assessing Composition in Sentence Vector Representations. In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, 2018, p. 1790–1801. Available also from: <https://aclanthology.org/C18-1152>.
169. SHI, X.; PADHI, I. and KNIGHT, K. Does String-Based Neural MT Learn Source Syntax? In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, 2016, p. 1526–1534. Available from DOI: 10.18653/v1/D16-1159.
170. ZHANG, K. and BOWMAN, S. Language Modeling Teaches You More than Translation Does: Lessons Learned Through Auxiliary Syntactic Task Analysis. In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Brussels, Belgium: Association for Computational Linguistics, 2018, p. 359–361. Available from DOI: 10.18653/v1/W18-5448.
171. TENNEY, I.; XIA, P.; CHEN, B.; WANG, A.; POLIAK, A.; MCCOY, R. T.; KIM, N.; DURME, B. V.; BOWMAN, S.; DAS, D. and PAVLICK, E. What do you learn from context? Probing for sentence structure in contextualized word representations. In: *International Conference on Learning Representations*. 2019. Available also from: <https://openreview.net/forum?id=SJzSgnRcKX>.
172. ALMARWANI, N.; ALDARMAKI, H. and DIAB, M. Efficient Sentence Embedding using Discrete Cosine Transform. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, 2019, p. 3672–3678. Available from DOI: 10.18653/v1/D19-1380.

173. LIU, N. F.; GARDNER, M.; BELINKOV, Y.; PETERS, M. E. and SMITH, N. A. Linguistic Knowledge and Transferability of Contextual Representations. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, 2019, p. 1073–1094. Available from DOI: 10.18653/v1/N19-1112.
174. HEWITT, J. and LIANG, P. Designing and Interpreting Probes with Control Tasks. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, 2019, p. 2733–2743. Available from DOI: 10.18653/v1/D19-1275.
175. PIMENTEL, T.; VALVODA, J.; HALL MAUDSLAY, R.; ZMIGROD, R.; WILLIAMS, A. and COTTERELL, R. Information-Theoretic Probing for Linguistic Structure. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, 2020, p. 4609–4622. Available from DOI: 10.18653/v1/2020.acl-main.420.
176. VOITA, E. and TITOV, I. Information-Theoretic Probing with Minimum Description Length. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, 2020, p. 183–196. Available from DOI: 10.18653/v1/2020.emnlp-main.14.
177. ZHU, Z. and RUDZICZ, F. An information theoretic view on selecting linguistic probes. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, 2020, p. 9251–9262. Available from DOI: 10.18653/v1/2020.emnlp-main.744.
178. HEWITT, J.; ETHAYARAJH, K.; LIANG, P. and MANNING, C. Conditional probing: measuring usable information beyond a baseline. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021, p. 1626–1639. Available from DOI: 10.18653/v1/2021.emnlp-main.122.
179. PERONE, C. S.; SILVEIRA, R. and PAULA, T. S. *Evaluation of sentence embeddings in downstream and linguistic probing tasks*. arXiv, 2018. Available from DOI: 10.48550/ARXIV.1806.06259. [Last accessed on 17 December 2025].
180. RAVICHANDER, A.; BELINKOV, Y. and HOVY, E. Probing the Probing Paradigm: Does Probing Accuracy Entail Task Relevance? In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguistics, 2021, p. 3363–3377. Available from DOI: 10.18653/v1/2021.eacl-main.295.
181. WU, Z.; CHEN, Y.; KAO, B. and LIU, Q. Perturbed Masking: Parameter-free Probing for Analyzing and Interpreting BERT. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, 2020, p. 4166–4176. Available from DOI: 10.18653/v1/2020.acl-main.383.

182. CAO, S.; SANH, V. and RUSH, A. Low-Complexity Probing via Finding Subnetworks. In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, 2021, p. 960–966. Available from DOI: 10.18653/v1/2021.naacl-main.74.
183. MICHAEL, J.; BOTHA, J. A. and TENNEY, I. Asking without Telling: Exploring Latent Ontologies in Contextual Representations. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, 2020, p. 6792–6812. Available from DOI: 10.18653/v1/2020.emnlp-main.552.
184. ZHOU, Y. and SRIKUMAR, V. DirectProbe: Studying Representations without Classifiers. In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, 2021, p. 5070–5083. Available from DOI: 10.18653/v1/2021.naacl-main.401.
185. ZHANG, D.; LI, S.-W.; XIAO, W.; ZHU, H.; NALLAPATI, R.; ARNOLD, A. O. and XIANG, B. Pairwise Supervised Contrastive Learning of Sentence Representations. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021, p. 5786–5798. Available from DOI: 10.18653/v1/2021.emnlp-main.467.
186. STANKEVIČIUS, L. and LUKOŠEVIČIUS, M. Extracting Sentence Embeddings from Pretrained Transformer Models. *Applied Sciences* Vol. 14, 2024, no. 19. ISSN 2076-3417. Available from DOI: 10.3390/app14198887.
187. PELLETIER, F. J. The principle of semantic compositionality. *Topoi*. Vol. 13, 1994, no. 1, p. 11–24.
188. MILAJEVS, D.; KARTSAKLIS, D.; SADRZADEH, M. and PURVER, M. Evaluating Neural Word Representations in Tensor-Based Compositional Settings. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014, p. 708–719. Available from DOI: 10.3115/v1/D14-1079.
189. MITCHELL, J. and LAPATA, M. Composition in Distributional Models of Semantics. *Cognitive Science* Vol. 34, 2010, no. 8, p. 1388–1429. Available from DOI: <https://doi.org/10.1111/j.1551-6709.2010.01106.x>.
190. XING, C.; WANG, D.; ZHANG, X. and LIU, C. Document classification with distributions of word vectors. In: *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific*. 2014, p. 1–5. Available from DOI: 10.1109/APSIPA.2014.7041633.
191. YU, L.; HERMANN, K. M.; BLUNSOM, P. and PULMAN, S. *Deep Learning for Answer Sentence Selection*. arXiv, 2014. Available from DOI: 10.48550/ARXIV.1412.1632. [Last accessed on 17 December 2025].
192. LEV, G.; KLEIN, B. and WOLF, L. In Defense of Word Embedding for Generic Text Representation. In: BIEMANN, C.; HANDSCHUH, S.; FREITAS, A.; MEZIANE, F. and MÉTAIS, E. (eds.). *Natural Language Processing and Information Systems*. Cham: Springer International Publishing, 2015, p. 35–50. ISBN 978-3-319-19581-0.

193. RITTER, S.; LONG, C.; PAPERNO, D.; BARONI, M.; BOTVINICK, M. and GOLDBERG, A. Leveraging Preposition Ambiguity to Assess Compositional Distributional Models of Semantics. In: *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*. Denver, Colorado: Association for Computational Linguistics, 2015, p. 199–204. Available from DOI: 10.18653/v1/S15-1023.
194. WHITE, L.; TOGNERI, R.; LIU, W. and BENNAMOUN, M. How Well Sentence Embeddings Capture Meaning. In: *Proceedings of the 20th Australasian Document Computing Symposium*. Parramatta, NSW, Australia: Association for Computing Machinery, 2015. ADCS '15. ISBN 9781450340403. Available from DOI: 10.1145/2838931.2838932.
195. SHEN, D.; WANG, G.; WANG, W.; MIN, M. R.; SU, Q.; ZHANG, Y.; LI, C.; HENAO, R. and CARIN, L. Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, 2018, p. 440–450. Available from DOI: 10.18653/v1/P18-1041.
196. WIETING, J.; BANSAL, M.; GIMPEL, K. and LIVESCU, K. *Towards Universal Paraphrastic Sentence Embeddings*. arXiv, 2015. Available from DOI: 10.48550/ARXIV.1511.08198. [Last accessed on 17 December 2025].
197. ALDARMAKI, H. and DIAB, M. Evaluation of Unsupervised Compositional Representations. In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, 2018, p. 2666–2677. Available also from: <https://aclanthology.org/C18-1226>.
198. IYYER, M.; MANJUNATHA, V.; BOYD-GRABER, J. and DAUMÉ III, H. Deep Unordered Composition Rivals Syntactic Methods for Text Classification. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, 2015, p. 1681–1691. Available from DOI: 10.3115/v1/P15-1162.
199. PHAM, N. T.; KRUSZEWSKI, G.; LAZARIDOU, A. and BARONI, M. Jointly optimizing word representations for lexical and sentential tasks with the C-PHRASE model. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, 2015, p. 971–981. Available from DOI: 10.3115/v1/P15-1094.
200. WIETING, J. and GIMPEL, K. Revisiting Recurrent Networks for Paraphrastic Sentence Embeddings. In: BARZILAY, R. and KAN, M.-Y. (eds.). *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, 2017, p. 2078–2088. Available from DOI: 10.18653/v1/P17-1190.
201. KENTER, T.; BORISOV, A. and RIJKE, M. de. Siamese CBOW: Optimizing Word Embeddings for Sentence Representations. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, 2016, p. 941–951. Available from DOI: 10.18653/v1/P16-1089.

202. GUPTA, P. and JAGGI, M. Obtaining Better Static Word Embeddings Using Contextual Embedding Models. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, 2021, p. 5241–5253. Available from DOI: 10.18653/v1/2021.acl-long.408.
203. HILL, F.; CHO, K.; KORHONEN, A. and BENGIO, Y. Learning to Understand Phrases by Embedding the Dictionary. *Transactions of the Association for Computational Linguistics* Vol. 4, 2016, p. 17–30. Available from DOI: 10.1162/tac1_a_00080.
204. JOULIN, A.; GRAVE, E.; BOJANOWSKI, P. and MIKOLOV, T. Bag of Tricks for Efficient Text Classification. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain: Association for Computational Linguistics, 2017, p. 427–431. Available also from: <https://aclanthology.org/E17-2068>.
205. CHOI, H.; KIM, J.; JOE, S. and GWON, Y. Evaluation of BERT and ALBERT Sentence Embedding Performance on Downstream NLP Tasks. In: *2020 25th International Conference on Pattern Recognition (ICPR)*. 2021, p. 5482–5487. Available from DOI: 10.1109/ICPR48806.2021.9412102.
206. BOMMASANI, R.; DAVIS, K. and CARDIE, C. Interpreting Pretrained Contextualized Representations via Reductions to Static Embeddings. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, 2020, p. 4758–4781. Available from DOI: 10.18653/v1/2020.acl-main.431.
207. YIN, W. and SCHÜTZE, H. Learning Word Meta-Embeddings. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, 2016, p. 1351–1360. Available from DOI: 10.18653/v1/P16-1128.
208. COATES, J. and BOLLEGALA, D. Frustratingly Easy Meta-Embedding – Computing Meta-Embeddings by Averaging Source Word Embeddings. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, 2018, p. 194–198. Available from DOI: 10.18653/v1/N18-2031.
209. SHANNON, C. E. A mathematical theory of communication. *The Bell System Technical Journal* Vol. 27, 1948, no. 3, p. 379–423. Available from DOI: 10.1002/j.1538-7305.1948.tb01338.x.
210. RIEDEL, B.; AUGENSTEIN, I.; SPITHOURAKIS, G. P. and RIEDEL, S. *A simple but tough-to-beat baseline for the Fake News Challenge stance detection task*. arXiv, 2017. Available from DOI: 10.48550/ARXIV.1707.03264. [Last accessed on 17 December 2025].
211. SINGH, P. and MUKERJEE, A. Words are not Equal: Graded Weighting Model for Building Composite Document Vectors. In: *Proceedings of the 12th International Conference on Natural Language Processing*. Trivandrum, India: NLP Association of India, 2015, p. 11–19. Available also from: <https://aclanthology.org/W15-5903>.

212. ETHAYARAJH, K. Unsupervised Random Walk Sentence Embeddings: A Strong but Simple Baseline. In: *Proceedings of The Third Workshop on Representation Learning for NLP*. Melbourne, Australia: Association for Computational Linguistics, 2018, p. 91–100. Available from DOI: 10.18653/v1/W18-3012.
213. JIANG, T.; JIAO, J.; HUANG, S.; ZHANG, Z.; WANG, D.; ZHUANG, F.; WEI, F.; HUANG, H.; DENG, D. and ZHANG, Q. PromptBERT: Improving BERT Sentence Embeddings with Prompts. In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, 2022, p. 8826–8837. Available also from: <https://aclanthology.org/2022.emnlp-main.603>.
214. AN, Y.; KALINOWSKI, A. and GREENBERG, J. Clustering and Network Analysis for the Embedding Spaces of Sentences and Sub-Sentences. In: *2021 Second International Conference on Intelligent Data Science Technologies and Applications (IDSTA)*. 2021, p. 138–145. Available from DOI: 10.1109/IDSTA53674.2021.9660801.
215. YANG, Z.; ZHU, C. and CHEN, W. Parameter-free Sentence Embedding via Orthogonal Basis. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, 2019, p. 638–648. Available from DOI: 10.18653/v1/D19-1059.
216. WANG, B. and KUO, C.-C. J. SBERT-WK: A Sentence Embedding Method by Dissecting BERT-Based Word Models. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.* Vol. 28, 2020, p. 2146–2157. ISSN 2329-9290. Available from DOI: 10.1109/TASLP.2020.3008390.
217. SCHAKEL, A. M. J. and WILSON, B. J. *Measuring Word Significance using Distributed Representations of Words*. arXiv, 2015. Available from DOI: 10.48550/ARXIV.1508.02297. [Last accessed on 17 December 2025].
218. AREFYEV, N.; ERMOLAEV, P. and PANCHENKO, A. *How much does a word weigh? Weighting word embeddings for word sense induction*. arXiv, 2018. Available from DOI: 10.48550/ARXIV.1805.09209. [Last accessed on 17 December 2025].
219. LUHN, H. P. The Automatic Creation of Literature Abstracts. *IBM Journal of Research and Development* Vol. 2, 1958, no. 2, p. 159–165. Available from DOI: 10.1147/rd.22.0159.
220. AMIRI, H. and MOHTARAMI, M. Vector of Locally Aggregated Embeddings for Text Representation. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, 2019, p. 1408–1414. Available from DOI: 10.18653/v1/N19-1143.
221. GUPTA, V.; KARNICK, H.; BANSAL, A. and JHALA, P. Product Classification in E-Commerce using Distributional Semantics. In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan: The COLING 2016 Organizing Committee, 2016, p. 536–546. Available also from: <https://aclanthology.org/C16-1052>.

222. KIM, H. K.; KIM, H. and CHO, S. Bag-of-concepts: Comprehending document representation through clustering words in distributed representation. *Neurocomputing* Vol. 266, 2017, p. 336–352. ISSN 0925-2312. Available from DOI: <https://doi.org/10.1016/j.neucom.2017.05.046>.
223. GUO, S. and YAO, N. Document Vector Extension for Documents Classification. *IEEE Transactions on Knowledge and Data Engineering* Vol. 33, 2021, no. 8, p. 3062–3074. Available from DOI: 10.1109/TKDE.2019.2961343.
224. LI, M.; BAI, H.; TAN, L.; XIONG, K.; LI, M. and LIN, J. *Latte-Mix: Measuring Sentence Semantic Similarity with Latent Categorical Mixtures*. arXiv, 2020. Available from DOI: 10.48550/ARXIV.2010.11351. [Last accessed on 17 December 2025].
225. MEKALA, D.; GUPTA, V.; PARANJAPPE, B. and KARNICK, H. SCDV : Sparse Composite Document Vectors using soft clustering over distributional representations. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, 2017, p. 659–669. Available from DOI: 10.18653/v1/D17-1069.
226. GUPTA, V.; SAW, A.; NOKHIZ, P.; NETRAPALLI, P.; RAI, P. and TALUKDAR, P. P-sif: Document embeddings using partition averaging. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2020, vol. 34, p. 7863–7870. No. 05.
227. AHARON, M.; ELAD, M. and BRUCKSTEIN, A. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing* Vol. 54, 2006, no. 11, p. 4311–4322. Available from DOI: 10.1109/TSP.2006.881199.
228. GUPTA, V.; SAW, A.; NOKHIZ, P.; GUPTA, H. and TALUKDAR, P. Improving Document Classification with Multi-Sense Embeddings. In: *Proceedings of the European Conference on Artificial Intelligence*. 2020.
229. GUPTA, A. and GUPTA, V. Unsupervised Contextualized Document Representation. In: *Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing*. Virtual: Association for Computational Linguistics, 2021, p. 166–173. Available from DOI: 10.18653/v1/2021.sustainlp-1.17.
230. MEKALA, D. and SHANG, J. Contextualized Weak Supervision for Text Classification. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, 2020, p. 323–333. Available from DOI: 10.18653/v1/2020.acl-main.30.
231. ALMARWANI, N. and DIAB, M. Discrete Cosine Transform as Universal Sentence Encoder. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Online: Association for Computational Linguistics, 2021, p. 419–426. Available from DOI: 10.18653/v1/2021.acl-short.53.
232. KAYAL, S. and TSATSARONIS, G. EigenSent: Spectral sentence embeddings using higher-order Dynamic Mode Decomposition. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, 2019, p. 4536–4546. Available from DOI: 10.18653/v1/P19-1445.

233. TANAKA, H.; SHINNOU, H.; CAO, R.; BAI, J. and MA, W. Document Classification by Word Embeddings of BERT. In: NGUYEN, L.-M.; PHAN, X.-H.; HASIDA, K. and TOJO, S. (eds.). *Computational Linguistics*. Singapore: Springer Singapore, 2020, p. 145–154. ISBN 978-981-15-6168-9.
234. MA, X.; WANG, Z.; NG, P.; NALLAPATI, R. and XIANG, B. *Universal Text Representation from BERT: An Empirical Study*. arXiv, 2019. Available from DOI: 10 . 48550/ARXIV.1910.07973. [Last accessed on 17 December 2025].
235. KOVALEVA, O.; ROMANOV, A.; ROGERS, A. and RUMSHISKY, A. Revealing the Dark Secrets of BERT. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, 2019, p. 4365–4374. Available from DOI: 10 . 18653/v1/D19-1445.
236. ZHONG, Z.; FRIEDMAN, D. and CHEN, D. Factual Probing Is [MASK]: Learning vs. Learning to Recall. In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, 2021, p. 5017–5033. Available from DOI: 10 . 18653/v1/2021.naacl-main.398.
237. DALVI, F.; KHAN, A. R.; ALAM, F.; DURRANI, N.; XU, J. and SAJJAD, H. Discovering Latent Concepts Learned in BERT. In: *International Conference on Learning Representations*. 2022. Available also from: <https://openreview.net/forum?id=P0TMtpYI1xH>.
238. MULLER, B.; ELAZAR, Y.; SAGOT, B. and SEDDAH, D. First Align, then Predict: Understanding the Cross-Lingual Ability of Multilingual BERT. In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguistics, 2021, p. 2214–2231. Available from DOI: 10 . 18653/v1/2021.eacl-main.189.
239. HÄMMERL, K.; LIBOVICKÝ, J. and FRASER, A. Combining Static and Contextualised Multilingual Embeddings. In: *Findings of the Association for Computational Linguistics: ACL 2022*. Dublin, Ireland: Association for Computational Linguistics, 2022, p. 2316–2329. Available from DOI: 10 . 18653/v1/2022.findings-acl.182.
240. ETHAYARAJH, K. How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, 2019, p. 55–65. Available from DOI: 10 . 18653/v1/D19-1006.
241. CHUNG, H. W.; FEVRY, T.; TSAI, H.; JOHNSON, M. and RUDER, S. Rethinking Embedding Coupling in Pre-trained Language Models. In: *International Conference on Learning Representations*. 2021. Available also from: https://openreview.net/forum?id=xpFFI_NtgpW.
242. ROGERS, A.; KOVALEVA, O. and RUMSHISKY, A. A Primer in BERTology: What We Know About How BERT Works. *Transactions of the Association for Computational Linguistics* Vol. 8, 2020, p. 842–866. Available from DOI: 10 . 1162/tac1_a_00349.

243. TIMKEY, W. and SCHIJNDEL, M. van. All Bark and No Bite: Rogue Dimensions in Transformer Language Models Obscure Representational Quality. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021, p. 4527–4546. Available from DOI: 10.18653/v1/2021.emnlp-main.372.
244. SAJJAD, H.; ALAM, F.; DALVI, F. and DURRANI, N. Effect of Post-processing on Contextualized Word Representations. In: CALZOLARI, N.; HUANG, C.-R.; KIM, H.; PUSTEJOVSKY, J.; WANNER, L.; CHOI, K.-S.; RYU, P.-M.; CHEN, H.-H.; DONATELLI, L.; JI, H.; KUROHASHI, S.; PAGGIO, P.; XUE, N.; KIM, S.; HAHM, Y.; HE, Z.; LEE, T. K.; SANTUS, E.; BOND, F. and NA, S.-H. (eds.). *Proceedings of the 29th International Conference on Computational Linguistics*. Gyeongju, Republic of Korea: International Committee on Computational Linguistics, 2022, p. 3127–3142. Available also from: <https://aclanthology.org/2022.coling-1.277/>.
245. ZHAO, H.; LU, Z. and POUPART, P. Self-Adaptive Hierarchical Sentence Model. In: *Proceedings of the 24th International Conference on Artificial Intelligence*. Buenos Aires, Argentina: AAAI Press, 2015, p. 4069–4076. IJCAI’15. ISBN 9781577357384.
246. YANG, J. and ZHAO, H. *Deepening Hidden Representations from Pre-trained Language Models*. 2021. Available also from: <https://openreview.net/forum?id=a9nIW5-0rh>. [Last accessed on 17 December 2025].
247. SHI, W.; CHEN, M.; ZHOU, P. and CHANG, K.-W. Retrofitting Contextualized Word Embeddings with Paraphrases. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, 2019, p. 1198–1203. Available from DOI: 10.18653/v1/D19-1113.
248. TOSHNIWAL, S.; SHI, H.; SHI, B.; GAO, L.; LIVESCU, K. and GIMPEL, K. A Cross-Task Analysis of Text Span Representations. In: *Proceedings of the 5th Workshop on Representation Learning for NLP*. Online: Association for Computational Linguistics, 2020, p. 166–176. Available from DOI: 10.18653/v1/2020.repl4nlp-1.20.
249. SOCHER, R.; PENNINGTON, J.; HUANG, E. H.; NG, A. Y. and MANNING, C. D. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, UK: Association for Computational Linguistics, 2011, p. 151–161. Available also from: <https://aclanthology.org/D11-1014>.
250. SALTON, G.; FOX, E. A. and WU, H. Extended Boolean Information Retrieval. *Commun. ACM* Vol. 26, 1983, no. 11, p. 1022–1036. ISSN 0001-0782. Available from DOI: 10.1145/182.358466.
251. RÜCKLÉ, A.; EGER, S.; PEYRARD, M. and GUREVYCH, I. *Concatenated Power Mean Word Embeddings as Universal Cross-Lingual Sentence Representations*. 2018. Available from arXiv: 1803.01400 [cs.CL]. [Last accessed on 17 December 2025].
252. LIN, Z.; FENG, M.; SANTOS, C. N. dos; YU, M.; XIANG, B.; ZHOU, B. and BENGIO, Y. A STRUCTURED SELF-ATTENTIVE SENTENCE EMBEDDING. In: *International Conference on Learning Representations*. 2017. Available also from: https://openreview.net/forum?id=BJC_jUqxe.

253. RUDMAN, W.; GILLMAN, N.; RAYNE, T. and EICKHOFF, C. IsoScore: Measuring the Uniformity of Embedding Space Utilization. In: *Findings of the Association for Computational Linguistics: ACL 2022*. Dublin, Ireland: Association for Computational Linguistics, 2022, p. 3325–3339. Available from DOI: 10.18653/v1/2022.findings-acl.262.
254. ZHOU, T.; SEDOC, J. and RODU, J. Getting in Shape: Word Embedding SubSpaces. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 2019, p. 5478–5484. Available from DOI: 10.24963/ijcai.2019/761.
255. GAO, J.; HE, D.; TAN, X.; QIN, T.; WANG, L. and LIU, T. Representation Degeneration Problem in Training Natural Language Generation Models. In: *International Conference on Learning Representations*. 2019. Available also from: <https://openreview.net/forum?id=SkEYojRqtm>.
256. RADFORD, A.; WU, J.; CHILD, R.; LUAN, D.; AMODEI, D. and SUTSKEVER, I. *Language Models are Unsupervised Multitask Learners*. 2019. Available also from: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf. [Last accessed on 1 April 2026].
257. SCHNABEL, T.; LABUTOV, I.; MIMNO, D. and JOACHIMS, T. Evaluation methods for unsupervised word embeddings. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, 2015, p. 298–307. Available from DOI: 10.18653/v1/D15-1036.
258. LI, B.; ZHOU, H.; HE, J.; WANG, M.; YANG, Y. and LI, L. On the Sentence Embeddings from Pre-trained Language Models. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, 2020, p. 9119–9130. Available from DOI: 10.18653/v1/2020.emnlp-main.733.
259. BIŠ, D.; PODKORYTOV, M. and LIU, X. Too Much in Common: Shifting of Embeddings in Transformer Language Models and its Implications. In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, 2021, p. 5117–5130. Available from DOI: 10.18653/v1/2021.naacl-main.403.
260. RAJAEI, S. and PILEHVAR, M. T. An Isotropy Analysis in the Multilingual BERT Embedding Space. In: *Findings of the Association for Computational Linguistics: ACL 2022*. Dublin, Ireland: Association for Computational Linguistics, 2022, p. 1309–1316. Available from DOI: 10.18653/v1/2022.findings-acl.103.
261. SCHICK, T. and SCHÜTZE, H. Rare Words: A Major Problem for Contextualized Embeddings and How to Fix it by Attentive Mimicking. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2020, vol. 34, p. 8766–8774. No. 05. Available from DOI: 10.1609/aaai.v34i05.6403.
262. MILLER, G. A. WordNet: A Lexical Database for English. *Commun. ACM* Vol. 38, 1995, no. 11, p. 39–41. ISSN 0001-0782. Available from DOI: 10.1145/219717.219748.

263. LEE, Y.-Y.; KE, H.; HUANG, H.-H. and CHEN, H.-H. Less is More: Filtering Abnormal Dimensions in GloVe. In: *Proceedings of the 25th International Conference Companion on World Wide Web*. Montréal, Québec, Canada: International World Wide Web Conferences Steering Committee, 2016, p. 71–72. WWW’16 Companion. ISBN 9781450341448. Available from DOI: 10.1145/2872518.2889381.
264. KOVALEVA, O.; KULSHRESHTHA, S.; ROGERS, A. and RUMSHISKY, A. BERT Busters: Outlier Dimensions that Disrupt Transformers. In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Online: Association for Computational Linguistics, 2021, p. 3392–3405. Available from DOI: 10.18653/v1/2021.findings-acl.300.
265. LUO, Z.; KULMIZEV, A. and MAO, X. Positional Artefacts Propagate Through Masked Language Model Embeddings. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, 2021, p. 5312–5327. Available from DOI: 10.18653/v1/2021.acl-long.413.
266. DING, Y.; MARTINKUS, K.; PASCUAL, D.; CLEMATIDE, S. and WATTENHOFER, R. On Isotropy Calibration of Transformer Models. In: *Proceedings of the Third Workshop on Insights from Negative Results in NLP*. Dublin, Ireland: Association for Computational Linguistics, 2022, p. 1–9. Available from DOI: 10.18653/v1/2022.insights-1.1.
267. CAI, X.; HUANG, J.; BIAN, Y. and CHURCH, K. Isotropy in the Contextual Embedding Space: Clusters and Manifolds. In: *International Conference on Learning Representations*. 2021. Available also from: <https://openreview.net/forum?id=xYGN0860WDH>.
268. RAJAEI, S. and PILEHVAR, M. T. A Cluster-based Approach for Improving Isotropy in Contextual Embedding Space. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Online: Association for Computational Linguistics, 2021, p. 575–584. Available from DOI: 10.18653/v1/2021.acl-short.73.
269. RAJAEI, S. and PILEHVAR, M. T. How Does Fine-tuning Affect the Geometry of Embedding Space: A Case Study on Isotropy. In: *Findings of the Association for Computational Linguistics: EMNLP 2021*. Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021, p. 3042–3049. Available from DOI: 10.18653/v1/2021.findings-emnlp.261.
270. ARTETXE, M.; LABAKA, G.; LOPEZ-GAZPIO, I. and AGIRRE, E. Uncovering Divergent Linguistic Information in Word Embeddings with Lessons for Intrinsic and Extrinsic Evaluation. In: *Proceedings of the 22nd Conference on Computational Natural Language Learning*. Brussels, Belgium: Association for Computational Linguistics, 2018, p. 282–291. Available from DOI: 10.18653/v1/K18-1028.
271. WANG, B.; CHEN, F.; WANG, A. and KUO, C.-C. J. Post-Processing of Word Representations via Variance Normalization and Dynamic Embedding. In: *2019 IEEE International Conference on Multimedia and Expo (ICME)*. 2019, p. 718–723. Available from DOI: 10.1109/ICME.2019.00129.

272. LIU, T.; UNGAR, L. and SEDOC, J. Unsupervised Post-Processing of Word Vectors via Conceptor Negation. In: *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*. Honolulu, Hawaii, USA: AAAI Press, 2019. AAAI'19/IAAI'19/EAAI'19. ISBN 978-1-57735-809-1. Available from DOI: 10.1609/aaai.v33i01.33016778.
273. JAEGER, H. *Controlling Recurrent Neural Networks by Conceptors*. 2024. Available from arXiv: 1403.3369 [cs.NE]. [Last accessed on 17 December 2025].
274. KARVE, S.; UNGAR, L. and SEDOC, J. Conceptor Debiasing of Word Representations Evaluated on WEAT. In: *Proceedings of the First Workshop on Gender Bias in Natural Language Processing*. Florence, Italy: Association for Computational Linguistics, 2019, p. 40–48. Available from DOI: 10.18653/v1/W19-3806.
275. LIANG, Y.; CAO, R.; ZHENG, J.; REN, J. and GAO, L. Learning to Remove: Towards Isotropic Pre-Trained BERT Embedding. In: *Artificial Neural Networks and Machine Learning – ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part V*. Bratislava, Slovakia: Springer-Verlag, 2021, p. 448–459. ISBN 978-3-030-86382-1. Available from DOI: 10.1007/978-3-030-86383-8_36.
276. RAUNAK, V.; GUPTA, V. and METZE, F. Effective Dimensionality Reduction for Word Embeddings. In: *Proceedings of the 4th Workshop on Representation Learning for NLP (Repl4NLP-2019)*. Florence, Italy: Association for Computational Linguistics, 2019, p. 235–243. Available from DOI: 10.18653/v1/W19-4328.
277. ARTETXE, M.; LABAKA, G. and AGIRRE, E. Generalizing and Improving Bilingual Word Embedding Mappings with a Multi-Step Framework of Linear Transformations. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2018, vol. 32. No. 1. Available from DOI: 10.1609/aaai.v32i1.11992.
278. WANG, T. and ISOLA, P. Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere. In: III, H. D. and SINGH, A. (eds.). *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 2020, vol. 119, p. 9929–9939. Proceedings of Machine Learning Research. Available also from: <https://proceedings.mlr.press/v119/wang20k.html>.
279. FARUQUI, M.; DODGE, J.; JAUHAR, S. K.; DYER, C.; HOVY, E. and SMITH, N. A. Retrofitting Word Vectors to Semantic Lexicons. In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, Colorado: Association for Computational Linguistics, 2015, p. 1606–1615. Available from DOI: 10.3115/v1/N15-1184.
280. YU, Z.; COHEN, T.; WALLACE, B.; BERNSTAM, E. and JOHNSON, T. Retrofitting Word Vectors of MeSH Terms to Improve Semantic Similarity Measures. In: *Proceedings of the Seventh International Workshop on Health Text Mining and Information Analysis*. Austin, TX: Association for Computational Linguistics, 2016, p. 43–51. Available from DOI: 10.18653/v1/W16-6106.

281. ZHANG, M.; FUJINUMA, Y.; PAUL, M. J. and BOYD-GRABER, J. Why Overfitting Isn't Always Bad: Retrofitting Cross-Lingual Word Embeddings to Dictionaries. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, 2020, p. 2214–2220. Available from DOI: 10.18653/v1/2020.acl-main.201.
282. GLAVAŠ, G. and VULIĆ, I. Explicit Retrofitting of Distributional Word Vectors. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, 2018, p. 34–45. Available from DOI: 10.18653/v1/P18-1004.
283. ZHENG, J.; WANG, Y.; WANG, G.; XIA, J.; HUANG, Y.; ZHAO, G.; ZHANG, Y. and LI, S. Using Context-to-Vector with Graph Retrofitting to Improve Word Embeddings. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, 2022, p. 8154–8163. Available from DOI: 10.18653/v1/2022.acl-long.561.
284. WANG, L.; HUANG, J.; HUANG, K.; HU, Z.; WANG, G. and GU, Q. Improving Neural Language Generation with Spectrum Control. In: *International Conference on Learning Representations*. 2020. Available also from: <https://openreview.net/forum?id=ByxY8CNtvr>.
285. TAMKIN, A.; JURAFSKY, D. and GOODMAN, N. Language Through a Prism: A Spectral Approach for Multiscale Language Representations. In: LAROCHELLE, H.; RANZATO, M.; HADSELL, R.; BALCAN, M. and LIN, H. (eds.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2020, vol. 33, p. 5492–5504. Available also from: <https://proceedings.neurips.cc/paper/2020/file/3acb2a202ae4bea8840224e6fce16fd0-Paper.pdf>.
286. ZHELEZNIAK, V.; SAVKOV, A. and HAMMERLA, N. Estimating Mutual Information Between Dense Word Embeddings. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, 2020, p. 8361–8371. Available from DOI: 10.18653/v1/2020.acl-main.741.
287. KRASKOV, A.; STÖGBAUER, H. and GRASSBERGER, P. Estimating mutual information. *Phys. Rev. E* Vol. 69, 2004, p. 066138. Available from DOI: 10.1103/PhysRevE.69.066138.
288. ZHELEZNIAK, V.; SAVKOV, A.; SHEN, A. and HAMMERLA, N. Correlation Coefficients and Semantic Textual Similarity. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, 2019, p. 951–962. Available from DOI: 10.18653/v1/N19-1100.
289. ZHELEZNIAK, V.; SAVKOV, A.; SHEN, A.; MORAMARCO, F.; FLANN, J. and HAMMERLA, N. Y. Don't Settle for Average, Go for the Max: Fuzzy Sets and Max-Pooled Word Vectors. In: *International Conference on Learning Representations*. 2019. Available also from: <https://openreview.net/forum?id=SkxXg2C5FX>.

290. MIN, C.; CHU, Y.; YANG, L.; XU, B. and LIN, H. Locality Preserving Sentence Encoding. In: *Findings of the Association for Computational Linguistics: EMNLP 2021*. Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021, p. 3050–3060. Available from DOI: 10.18653/v1/2021.findings-emnlp.262.
291. KAYAL, S. Unsupervised Sentence-embeddings by Manifold Approximation and Projection. In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguistics, 2021, p. 1–11. Available from DOI: 10.18653/v1/2021.eacl-main.1.
292. WU, L.; YEN, I. E.-H.; XU, K.; XU, F.; BALAKRISHNAN, A.; CHEN, P.-Y.; RAVIKUMAR, P. and WITBROCK, M. J. Word Mover’s Embedding: From Word2Vec to Document Embedding. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, 2018, p. 4524–4534. Available from DOI: 10.18653/v1/D18-1482.
293. WANG, Z.; ZHANG, Y. and WU, H. *Structural-Aware Sentence Similarity with Recursive Optimal Transport*. arXiv, 2020. Available from DOI: 10.48550/ARXIV.2002.00745. [Last accessed on 1 April 2026].
294. PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M. and DUCHESNAY, É. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* Vol. 12, 2011, no. 85, p. 2825–2830. Available also from: <http://jmlr.org/papers/v12/pedregosa11a.html>.
295. MERITY, S.; XIONG, C.; BRADBURY, J. and SOCHER, R. Pointer Sentinel Mixture Models. In: *International Conference on Learning Representations*. 2017. Available also from: <https://openreview.net/forum?id=Byj72udxe>.
296. WOLF, T.; DEBUT, L.; SANH, V.; CHAUMOND, J.; DELANGUE, C.; MOI, A.; CISTAC, P.; RAULT, T.; LOUF, R.; FUNTOWICZ, M.; DAVISON, J.; SHLEIFER, S.; PLATEN, P. von; MA, C.; JERNITE, Y.; PLU, J.; XU, C.; LE SCAO, T.; GUGGER, S.; DRAME, M.; LHOEST, Q. and RUSH, A. Transformers: State-of-the-Art Natural Language Processing. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, 2020, p. 38–45. Available from DOI: 10.18653/v1/2020.emnlp-demos.6.
297. ZHU, Y.; KIROS, R.; ZEMEL, R.; SALAKHUTDINOV, R.; URTASUN, R.; TORRALBA, A. and FIDLER, S. Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society, 2015, p. 19–27. ISSN 2380-7504. Available from DOI: 10.1109/ICCV.2015.11.
298. BA, J. L.; KIROS, J. R. and HINTON, G. E. *Layer Normalization*. 2016. Available from arXiv: 1607.06450 [stat.ML]. [Last accessed on 1 April 2026].
299. TAYLOR, W. L. “Cloze Procedure”: A New Tool for Measuring Readability. *Journalism Quarterly* Vol. 30, 1953, no. 4, p. 415–433. Available from DOI: 10.1177/107769905303000401.

300. BOLSTAD, B.; IRIZARRY, R.; ÅSTRAND, M. and SPEED, T. A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics* Vol. 19, 2003, no. 2, p. 185–193. ISSN 1367-4803. Available from DOI: 10.1093/bioinformatics/19.2.185.
301. TOUVRON, H.; LAVRIL, T.; IZACARD, G.; MARTINET, X.; LACHAUX, M.-A.; LACROIX, T.; ROZIÈRE, B.; GOYAL, N.; HAMBRO, E.; AZHAR, F.; RODRIGUEZ, A.; JOULIN, A.; GRAVE, E. and LAMPLE, G. *LLaMA: Open and Efficient Foundation Language Models*. 2023. Available from arXiv: 2302.13971 [cs.CL]. [Last accessed on 17 December 2025].
302. ABDALLA, M.; VISHNUBHOTLA, K. and MOHAMMAD, S. M. What Makes Sentences Semantically Related: A Textual Relatedness Dataset and Empirical Study. In: *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Dubrovnik, Croatia: Association for Computational Linguistics, 2023.
303. PANG, B. and LEE, L. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. In: *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Ann Arbor, Michigan: Association for Computational Linguistics, 2005, p. 115–124. ACL ’05. Available from DOI: 10.3115/1219840.1219855.
304. HU, M. and LIU, B. Mining and Summarizing Customer Reviews. In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Seattle, WA, USA: Association for Computing Machinery, 2004, p. 168–177. KDD ’04. ISBN 1581138881. Available from DOI: 10.1145/1014052.1014073.
305. WIEBE, J.; WILSON, T. and CARDIE, C. Annotating expressions of opinions and emotions in language. *Language resources and evaluation* Vol. 39, 2005, p. 165–210. Available from DOI: 10.1007/s10579-005-7880-9.
306. DENG, L. and WIEBE, J. MPQA 3.0: An entity/event-level sentiment corpus. In: *Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: human language technologies*. 2015, p. 1323–1328. Available also from: <https://aclanthology.org/N15-1146.pdf>.
307. DOLAN, B.; QUIRK, C. and BROCKETT, C. Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources. In: *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*. Geneva, Switzerland: COLING, 2004, p. 350–356. Available also from: <https://aclanthology.org/C04-1051>.
308. COHAN, A.; AMMAR, W.; ZUYLEN, M. van and CADY, F. Structural Scaffolds for Citation Intent Classification in Scientific Publications. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, 2019, p. 3586–3596. Available from DOI: 10.18653/v1/N19-1361.
309. LI, X. and ROTH, D. Learning Question Classifiers. In: *COLING 2002: The 19th International Conference on Computational Linguistics*. 2002. Available also from: <https://aclanthology.org/C02-1150>.

310. HADIFAR, A.; STERCKX, L.; DEMEESTER, T. and DEVELDER, C. A Self-Training Approach for Short Text Clustering. In: AUGENSTEIN, I.; GELLA, S.; RUDER, S.; KANN, K.; CAN, B.; WELBL, J.; CONNEAU, A.; REN, X. and REI, M. (eds.). *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*. Florence, Italy: Association for Computational Linguistics, 2019, p. 194–199. Available from DOI: 10.18653/v1/W19-4322.
311. DENG, J.; WAN, F.; YANG, T.; QUAN, X. and WANG, R. Clustering-Aware Negative Sampling for Unsupervised Sentence Representation. In: ROGERS, A.; BOYD-GRABER, J. and OKAZAKI, N. (eds.). *Findings of the Association for Computational Linguistics: ACL 2023*. Toronto, Canada: Association for Computational Linguistics, 2023, p. 8713–8729. Available from DOI: 10.18653/v1/2023.findings-acl.555.
312. ZHANG, D.; NAN, F.; WEI, X.; LI, S.-W.; ZHU, H.; MCKEOWN, K.; NALLAPATI, R.; ARNOLD, A. O. and XIANG, B. Supporting Clustering with Contrastive Learning. In: TOUTANOVA, K.; RUMSHISKY, A.; ZETTLEMOYER, L.; HAKKANI-TUR, D.; BELTAGY, I.; BETHARD, S.; COTTERELL, R.; CHAKRABORTY, T. and ZHOU, Y. (eds.). *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, 2021, p. 5419–5430. Available from DOI: 10.18653/v1/2021.naacl-main.427.
313. ZHANG, D.; XIAO, W.; ZHU, H.; MA, X. and ARNOLD, A. Virtual Augmentation Supported Contrastive Learning of Sentence Representations. In: MURESAN, S.; NAKOV, P. and VILLAVICENCIO, A. (eds.). *Findings of the Association for Computational Linguistics: ACL 2022*. Dublin, Ireland: Association for Computational Linguistics, 2022, p. 864–876. Available from DOI: 10.18653/v1/2022.findings-acl.70.
314. RAKIB, M. R. H.; ZEH, N.; JANKOWSKA, M. and MILIOS, E. Enhancement of Short Text Clustering by Iterative Classification. In: MÉTAIS, E.; MEZIANE, F.; HORACEK, H. and CIMIANO, P. (eds.). *Natural Language Processing and Information Systems*. Cham: Springer International Publishing, 2020, p. 105–117. ISBN 978-3-030-51310-8.
315. MUNKRES, J. Algorithms for the Assignment and Transportation Problems. *Journal of the Society for Industrial and Applied Mathematics* [online] Vol. 5, 1957, no. 1, p. 32–38 [visited on 2023-02-28]. ISSN 03684245. Available from: <http://www.jstor.org/stable/2098689>.
316. ZHANG, D. *Sentence Representations Learning with Transformers* [<https://github.com/amazon-science/sentence-representations>]. [N.d.]. [Last accessed on 11 April 2025].
317. RAKIB, M. R. H. *Short text clustering datasets* [GitHub]. [N.d.]. Available also from: <https://github.com/rashadulrakib/short-text-clustering-enhancement/tree/master/data>. [Last accessed on 11 April 2025].
318. ZHANG, X. and LECUN, Y. *Text Understanding from Scratch*. arXiv, 2015. Available from DOI: 10.48550/ARXIV.1502.01710. [Last accessed on 17 December 2025].
319. *BioASQ Challenge* [<https://participants-area.bioasq.org/>]. [N.d.]. [Last accessed on 11 April 2025].

320. XU, J.; XU, B.; WANG, P.; ZHENG, S.; TIAN, G.; ZHAO, J. and XU, B. Self-Taught convolutional neural networks for short text clustering. *Neural Networks* Vol. 88, 2017, p. 22–31. ISSN 0893-6080. Available from DOI: <https://doi.org/10.1016/j.neunet.2016.12.008>.
321. YIN, J. and WANG, J. A model-based approach for text clustering with outlier detection. In: *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. 2016, p. 625–636. Available from DOI: 10.1109/ICDE.2016.7498276.
322. PHAN, X.-H.; NGUYEN, L.-M. and HORIGUCHI, S. Learning to Classify Short and Sparse Text & Web with Hidden Topics from Large-Scale Data Collections. In: *Proceedings of the 17th International Conference on World Wide Web*. Beijing, China: Association for Computing Machinery, 2008, p. 91–100. WWW’08. ISBN 9781605580852. Available from DOI: 10.1145/1367497.1367510.
323. *Predict Closed Questions on Stack Overflow* [<https://www.kaggle.com/competitions/predict-closed-questions-on-stack-overflow/data>]. [N.d.]. [Last accessed on 11 April 2025].
324. VOITA, E.; SENNRICH, R. and TITOV, I. The Bottom-up Evolution of Representations in the Transformer: A Study with Machine Translation and Language Modeling Objectives. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, 2019, p. 4396–4406. Available from DOI: 10.18653/v1/D19-1448.
325. *Common Crawl* [<https://commoncrawl.org/>]. [N.d.]. [Last accessed on 11 April 2025].
326. ORTIZ SUÁREZ, P. J.; SAGOT, B. and ROMARY, L. Asynchronous pipelines for processing huge corpora on medium to low resource infrastructures. In: BAŃSKI, P.; BARBARESI, A.; BIBER, H.; BREITENEDER, E.; CLEMATIDE, S.; KUPIETZ, M.; L’UNGEN, H. and ILIADI, C. (eds.). Mannheim: Leibniz-Institut für Deutsche Sprache, 2019, p. 9–16. Proceedings of the Workshop on Challenges in the Management of Large Corpora (CMLC-7) 2019. Cardiff, 22nd July 2019. Available from DOI: 10.14618/ids-pub-9021.
327. NGUYEN, T.; NGUYEN, C. V.; LAI, V. D.; MAN, H.; NGO, N. T.; DERNONCOURT, F.; ROSSI, R. A. and NGUYEN, T. H. CulturaX: A Cleaned, Enormous, and Multilingual Dataset for Large Language Models in 167 Languages. In: CALZOLARI, N.; KAN, M.-Y.; HOSTE, V.; LENCI, A.; SAKTI, S. and XUE, N. (eds.). *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*. Torino, Italia: ELRA and ICCL, 2024, p. 4226–4237. Available also from: <https://aclanthology.org/2024.lrec-main.377/>.
328. CONNEAU, A.; KHANDELWAL, K.; GOYAL, N.; CHAUDHARY, V.; WENZKE, G.; GUZMÁN, F.; GRAVE, E.; OTT, M.; ZETTLEMOYER, L. and STOYANOV, V. Unsupervised Cross-lingual Representation Learning at Scale. In: JURAFSKY, D.; CHAI, J.; SCHLUTER, N. and TETREAULT, J. (eds.). *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, 2020, p. 8440–8451. Available from DOI: 10.18653/v1/2020.acl-main.747.

329. PENEDO, G.; MALARTIC, Q.; HESSLOW, D.; COJOCARU, R.; ALOBEIDLI, H.; CAPPELLI, A.; PANNIER, B.; ALMAZROUEI, E. and LAUNAY, J. The Refined-Web Dataset for Falcon LLM: Outperforming Curated Corpora with Web Data Only. In: *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. 2023. Available also from: <https://openreview.net/forum?id=kM5eGcdCzq>.
330. PENEDO, G.; KYDLÍČEK, H.; ALLAL, L. B.; LOZHKOVA, A.; MITCHELL, M.; RAFFEL, C.; WERRA, L. V. and WOLF, T. The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale. In: *The Thirty-eighth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. 2024. Available also from: <https://openreview.net/forum?id=n6SCKn2QaG>.
331. GAO, L.; BIDERMAN, S.; BLACK, S.; GOLDING, L.; HOPPE, T.; FOSTER, C.; PHANG, J.; HE, H.; THITE, A.; NABESHIMA, N.; PRESSER, S. and LEAHY, C. *The Pile: An 800GB Dataset of Diverse Text for Language Modeling*. 2020. Available from arXiv: 2101.00027 [cs.CL]. [Last accessed on 2025-03-07].
332. RIMKUTĖ, E.; BIELINSKIENĖ, A.; KOVALEVSKAITĖ, J.; BOIZOU, L.; ALEKSANDRAVIČIŪTĖ, G.; BROKAITĖ, K. and UTKA, A. *Lithuanian Treebank ALKSNIS (2019-10-24)*. 2019. Available also from: <http://hdl.handle.net/20.500.11821/21>. [Last accessed on 2025-03-07].
333. DE MARNEFFE, M.-C.; MANNING, C. D.; NIVRE, J. and ZEMAN, D. Universal dependencies. *Computational linguistics* Vol. 47, 2021, no. 2, p. 255–308.
334. RIMKUTĖ, E.; BIELINSKIENĖ, A.; DADURKEVIČIUS, V.; KOVALEVSKAITĖ, J.; UTKA, A. and BOIZOU, L. *Lithuanian morphologically annotated corpus - MATAS v3.0*. 2024. Available also from: <http://hdl.handle.net/20.500.11821/61>. [Last accessed on 2025-03-07].
335. KAPOČIŪTĖ-DZIKIENĖ, J. and SALIMBAJEVS, A. Comparison of deep learning approaches for Lithuanian sentiment analysis. *Baltic journal of modern computing* Vol. 10, 2022, no. 3. ISSN 2255-8950. Available from DOI: 10.22364/bjmc.2022.10.3.02.
336. PRANCKAITIS, V. and LUKOŠEVIČIUS, M. Clustering of Lithuanian news articles. In: *Proceedings of the IVUS 2017*. Kaunas, Lithuania, 2017, p. 27–32. Available also from: <http://ceur-ws.org/Vol-1856/p05.pdf>.
337. ŠTRIMAITIS, R.; STEFANOVIČ, P.; RAMANAUSKAITĖ, S. and SLOTKIENĖ, A. A Combined Approach for Multi-Label Text Data Classification. *Computational Intelligence and Neuroscience* Vol. 2022, 2022, no. 1, p. 3369703. Available from DOI: <https://doi.org/10.1155/2022/3369703>.
338. MANDRAVICKAITĖ, J.; RIMKIENĖ, E.; KAPKAN, D. K.; KALINAUSKAITĖ, D.; ČENYS, A. and KRILAVIČIUS, T. Automatic Text Simplification for Lithuanian: Transforming Administrative Texts into Plain Language. *Mathematics* Vol. 13, 2025, no. 3. ISSN 2227-7390. Available from DOI: 10.3390/math13030465.

339. SONGAILAITĖ, M.; RIMKIENĖ, E.; MANDRAVICKAITĖ, J.; PETKEVIČIUS, M.; ZARANKA, E. and KRILAVIČIUS, T. Hate speech detection for Lithuanian language. In: *DAMSS: 15th conference on data analysis methods for software systems, Druskininkai, Lithuania, November 28-30, 2024*. Vilnius: Vilniaus universiteto leidykla, 2024, vol. 52. ISBN 9786090711125. ISSN 2669-0233. Available from DOI: 10.15388/DAMSS.15.2024.
340. NAKVOSAS, A.; DANIUŠIS, P. and MULEVIČIUS, V. *Open Llama2 Model for the Lithuanian Language*. 2024. Available from arXiv: 2408.12963 [cs.CL]. [Last accessed on 2025-03-07].
341. VITKUTĖ-ADŽGAUSKIENĖ, D.; UTKA, A.; AMILEVIČIUS, D. and KRILAVIČIUS, T. NLP Infrastructure for the Lithuanian Language. In: CALZOLARI, N.; CHOUKRI, K.; DECLERCK, T.; GOGGI, S.; GROBELNIK, M.; MAEGAARD, B.; MARIANI, J.; MAZO, H.; MORENO, A.; ODIJK, J. and PIPERIDIS, S. (eds.). *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. Portorož, Slovenia: European Language Resources Association (ELRA), 2016, p. 2539–2542. Available also from: <https://aclanthology.org/L16-1403/>.
342. BIELINSKIENĖ, A.; BOIZOU, L.; BUMBULIENĖ, I.; KOVALEVSKAITĖ, J.; KRILAVIČIUS, T.; MANDRAVICKAITĖ, J.; RIMKUTĖ, E. and VILKAITĖ-LOZDIENĖ, L. *Lithuanian Word embeddings*. 2019. Available also from: <http://hdl.handle.net/20.500.11821/26>. [Last accessed on 2025-03-07].
343. ULČAR, M. and ROBNIK-ŠIKONJA, M. Training Dataset and Dictionary Sizes Matter in BERT Models: The Case of Baltic Languages. In: BURNAEV, E.; IGNATOV, D. I.; IVANOV, S.; KHACHAY, M.; KOLTSOVA, O.; KUTUZOV, A.; KUZNETSOV, S. O.; LOUKACHEVITCH, N.; NAPOLI, A.; PANCHENKO, A.; PARDALOS, P. M.; SARAMÄKI, J.; SAVCHENKO, A. V.; TSYMBALOV, E. and TUTUBALINA, E. (eds.). *Analysis of Images, Social Networks and Texts*. Cham: Springer International Publishing, 2022, p. 162–172. ISBN 978-3-031-16500-9.
344. TOUVRON, H.; MARTIN, L.; STONE, K.; ALBERT, P.; ALMAHAIRI, A.; BABAEI, Y.; BASHLYKOV, N.; BATRA, S.; BHARGAVA, P.; BHOSALE, S.; BIKEL, D.; BLECHER, L.; FERRER, C. C.; CHEN, M.; CUCURULL, G.; ESIÖBU, D.; FERNANDES, J.; FU, J.; FU, W.; FULLER, B.; GAO, C.; GOSWAMI, V.; GOYAL, N.; HARTSHORN, A.; HOSSEINI, S.; HOU, R.; INAN, H.; KARDAS, M.; KERKEZ, V.; KHABSA, M.; KLOUMANN, I.; KORENEV, A.; KOURA, P. S.; LACHAUX, M.-A.; LAVRIL, T.; LEE, J.; LISKOVICH, D.; LU, Y.; MAO, Y.; MARTINET, X.; MIHAYLOV, T.; MISHRA, P.; MOLYBOG, I.; NIE, Y.; POULTON, A.; REIZENSTEIN, J.; RUNGTA, R.; SALADI, K.; SCHELLEN, A.; SILVA, R.; SMITH, E. M.; SUBRAMANIAN, R.; TAN, X. E.; TANG, B.; TAYLOR, R.; WILLIAMS, A.; KUAN, J. X.; XU, P.; YAN, Z.; ZAROV, I.; ZHANG, Y.; FAN, A.; KAMBADUR, M.; NARANG, S.; RODRIGUEZ, A.; STOJNIC, R.; EDUNOV, S. and SCIALOM, T. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. 2023. Available from arXiv: 2307.09288 [cs.CL]. [Last accessed on 2025-03-07].
345. KAPOČIŪTĖ-DZIKIENĖ, J.; BERGMANIS, T. and PINNIS, M. Localizing AI: evaluating open-weight language models for languages of Baltic States. In: *Tartu: University of Tartu Library*, 2025, vol. 25. ISSN 1736-8197. Available also from: <https://hdl.handle.net/20.500.12259/275478>.

346. STANKEVIČIUS, L. and LUKOŠEVIČIUS, M. Testing pre-trained Transformer models for Lithuanian news clustering. In: *Proceedings of the IVUS 2020* [online]. Kaunas, Lithuania: CEUR, 2020, vol. 2698, p. 46–53 [visited on 2020-04-23]. ISSN 1613-0073. Available from: <http://ceur-ws.org/Vol-2698/p08.pdf>.
347. STANKEVIČIUS, L. and LUKOŠEVIČIUS, M. Lithuanian news clustering using document embeddings. In: *Proceedings of the International Conference on Information Technologies*. Kaunas, 2019, p. 104–109. Available also from: <http://ceur-ws.org/Vol-2470/>.
348. CHICCO, D. and JURMAN, G. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC genomics* Vol. 21, 2020, no. 1, p. 6.
349. STANKEVIČIUS, L. and LUKOŠEVIČIUS, M. Generating Abstractive Summaries of Lithuanian News Articles Using a Transformer Model. In: LOPATA, A.; GUDONIENĖ, D. and BUTKIENĖ, R. (eds.). *Information and Software Technologies*. Cham: Springer International Publishing, 2021, p. 341–352. ISBN 978-3-030-88304-1.
350. HOLTZMAN, A.; BUYS, J.; DU, L.; FORBES, M. and CHOI, Y. The Curious Case of Neural Text Degeneration. In: *International Conference on Learning Representations*. 2020. Available also from: <https://openreview.net/forum?id=rygGQyrFvH>.
351. FAN, A.; LEWIS, M. and DAUPHIN, Y. Hierarchical Neural Story Generation. In: GUREVYCH, I. and MIYAO, Y. (eds.). *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, 2018, p. 889–898. Available from DOI: 10.18653/v1/P18-1082.
352. SHAZEER, N. and STERN, M. Adafactor: Adaptive learning rates with sublinear memory cost. In: *International Conference on Machine Learning*. PMLR, 2018, p. 4596–4604.
353. ALLAHYARI, M.; POURIYEH, S.; ASSEFI, M.; SAFAEI, S.; TRIPPE, E. D.; GUTIERREZ, J. B. and KOCHUT, K. Text Summarization Techniques: A Brief Survey. *International Journal of Advanced Computer Science and Applications (ijacsa)* Vol. 8, 2017, no. 10, p. 397.
354. KLIMAS, A. Some Unique Features of Lithuanian. *Lituanus* Vol. 30, 1984, no. 3, p. 51–64.
355. RIMKUTĖ, E. *Morfologinio daugiareikšmiškumo ribojimas kompiuteriniame tekstyne [Morphological Disambiguation of the Corpus of Lithuanian Language]*. Kaunas, Lithuania, 2006. PhD thesis. Vytautas Magnus University. (Accessed on 3 January 2022).
356. STANKEVIČIUS, L.; LUKOŠEVIČIUS, M.; KAPOČIŪTĖ-DZIKIENĖ, J.; BRIEDIENĖ, M. and KRILAVIČIUS, T. Correcting Diacritics and Typos with a ByT5 Transformer Model. *Applied Sciences* Vol. 12, 2022, no. 5. ISSN 2076-3417. Available from DOI: 10.3390/app12052636.
357. STANKEVIČIUS, L. and LUKOŠEVIČIUS, M. Towards Lithuanian Grammatical Error Correction. In: SILHAVY, R. (ed.). *Artificial Intelligence Trends in Systems*. Cham: Springer International Publishing, 2022, p. 490–503. ISBN 978-3-031-09076-9.

358. NÁPLAVA, J.; STRAKA, M.; STRAŇÁK, P. and HAJIČ, J. Diacritics Restoration Using Neural Networks. In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA), 2018. Available also from: <https://aclanthology.org/L18-1247>.
359. TAMULIONIENĖ, A. et al. Būdingiausios rašybos klaidos mokinių rašiniuose ir tinklaraščiuose. *Bendrinė kalba (iki 2014 metų–Kalbos kultūra)* Vol. 2015, no. 88, p. 1–24.
360. SHAH, K. and MELO, G. de. Correcting the Autocorrect: Context-Aware Typographical Error Correction via Training Data Augmentation. In: *Proceedings of The 12th Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, 2020, p. 6930–6936. Available also from: <https://www.aclweb.org/anthology/2020.lrec-1.856>.
361. KUZNETSOV, A. and URDIALES, H. *Spelling Correction with Denoising Transformer*. 2021. Available from arXiv: 2105.05977 [cs.CL]. [Last accessed on 2022-03-07].
362. DAMERAU, F. J. A Technique for Computer Detection and Correction of Spelling Errors. *Commun. ACM* Vol. 7, 1964, no. 3, p. 171–176. ISSN 0001-0782. Available from DOI: 10.1145/363958.363994.
363. POLLOCK, J. J. and ZAMORA, A. Automatic Spelling Correction in Scientific and Scholarly Text. *Commun. ACM* Vol. 27, 1984, no. 4, p. 358–368. ISSN 0001-0782. Available from DOI: 10.1145/358027.358048.
364. MITTON, R. *English spelling and the computer*. Harlow, Essex, UK: Longman Group, 1996.
365. BABA, Y. and SUZUKI, H. How Are Spelling Errors Generated and Corrected? A Study of Corrected and Uncorrected Spelling Errors Using Keystroke Logs. In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Jeju Island, Korea: Association for Computational Linguistics, 2012, p. 373–377. Available also from: <https://aclanthology.org/P12-2073>.
366. HAGIWARA, M. and MITA, M. GitHub Typo Corpus: A Large-Scale Multilingual Dataset of Misspellings and Grammatical Errors. In: *Proceedings of the 12th Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, 2020, p. 6761–6768. ISBN 979-10-95546-34-4. Available also from: <https://aclanthology.org/2020.lrec-1.835>.
367. NÉMETH, L. *Hunspell* [<http://hunspell.github.io/>]. [N.d.]. [Last accessed on 17 December 2021].
368. ROTHE, S.; MALLINSON, J.; MALMI, E.; KRAUSE, S. and SEVERYN, A. A Simple Recipe for Multilingual Grammatical Error Correction. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*. Online: Association for Computational Linguistics, 2021, p. 702–707.
369. NÁPLAVA, J.; STRAKA, M. and STRAKOVÁ, J. Diacritics Restoration using BERT with Analysis on Czech language. *The Prague Bulletin of Mathematical Linguistics* Vol. 116, 2021, p. 27–42. ISSN 0032-6585. Available from DOI: 10.14712/00326585.013.

370. BRYANT, C.; FELICE, M. and BRISCOE, T. Automatic Annotation and Evaluation of Error Types for Grammatical Error Correction. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, 2017, p. 793–805. Available from DOI: 10.18653/v1/P17-1074.
371. DADURKEVIČIUS, V. *Lithuanian Hunspell dictionary*. 2025. Available also from: <http://hdl.handle.net/20.500.11821/64>. [Last accessed on 2025-03-07].
372. BOULTON, R. *PyStemmer* [<https://pypi.org/project/PyStemmer/>]. [N.d.]. [Last accessed on 11 April 2025].
373. EXPLOSION. *spaCy pipelines for Lithuanian* [<https://spacy.io/models/lt>]. 2016. [Last accessed on 11 April 2025].
374. MANDRAVICKAITĖ, J.; RIMKIENĖ, E.; PETKEVIČIUS, M.; SONGAILAITĖ, M.; ZARANKA, E. and KRILAVIČIUS, T. Exploring Hate Speech Detection Models for Lithuanian Language. In: CALABRESE, A.; KOCK, C. de; NOZZA, D.; PLAZA-DEL-ARCO, F. M.; TALAT, Z. and VARGAS, F. (eds.). *Proceedings of the The 9th Workshop on Online Abuse and Harms (WOAH)*. Vienna, Austria: Association for Computational Linguistics, 2025, p. 206–218. ISBN 979-8-89176-105-6. Available also from: <https://aclanthology.org/2025.woah-1.18/>.
375. ŠYKORA, S. *Mathematical means and averages: Basic properties*. 2009. Available from DOI: 10.3247/SL3Math09.001. [Last accessed on 17 December 2025].

CURRICULUM VITAE AND DESCRIPTION OF CREATIVE ACTIVITIES (CV)

Lukas Stankevičius

Kaunas, Lithuania | lukas.stankevicius@ktu.lt

I specialize in developing and applying machine learning models for NLP tasks. My research focuses on training and utilizing transformer models, particularly in GEC.

Education

- 2017–2019 Master of Computing at Faculty of Informatics, Kaunas University of Technology
- 2013–2017 Bachelor of Physics and Teachers Training at Kaunas University of Technology

Work Experience

- 2024–Current Data analyst at *Energy Advice*
- 2020–2024 Assistant at Kaunas University of Technology
- 2018–2019 Machine learning engineer at *Nextquestion LTD*

Publications

1. L. Stankevičius, M. Lukoševičius // Extracting sentence embeddings from pre-trained transformer models // *Applied Sciences* 14, no. 19: 8887, 2024. [IF 2.500 (2023)]
2. L. Stankevičius, M. Lukoševičius, J. Kapočiūtė-Dzikienė; M. Briedienė; T. Kriavičius // Correcting Diacritics and Typos with a ByT5 Transformer Model // *Applied Sciences* 12, no. 5: 2636, 2022. [IF 2.679 (2021)]
3. L. Stankevičius, M. Lukoševičius // Towards Lithuanian GEC // Artificial intelligence trends in systems: proceedings of 11th computer science on-line conference (CSOC 2022), Zlín, Czech Republic, p. 490-503. Apr 27, 2022.
4. L. Stankevičius, M. Lukoševičius // Generating abstractive summaries of Lithuanian news articles using a transformer model // *International Conference on Information and Software Technologies (ICIST 2021)*, Kaunas, Lithuania, p. 341-352. Oct 14, 2021.
5. L. Stankevičius, M. Lukoševičius // Testing pre-trained transformer models for Lithuanian news clustering // *Proceedings of the information society and university studies (IVUS 2020)*, Kaunas, Lithuania, p. 46-53. Apr 23, 2020.

LIST OF ARTICLES AND CONFERENCE PROCEEDINGS

The main results of this dissertation have been published in the following journal articles and conference proceedings.

Journal articles

1. [S1; CH; OA] Stankevičius, Lukas; Lukoševičius, Mantas. Extracting sentence embeddings from pretrained transformer models // *Applied Sciences*. Basel : MDPI. ISSN 2076-3417. 2024, vol. 14, iss. 19, art. no. 8887, p. 1-66. DOI: 10.3390/app14198887. [Science Citation Index Expanded (Web of Science); Scopus] [IF: 2.500; AIF: 5.000; IF/AIF: 0.500; Q1 (2023, InCites JCR SCIE)] [Field: N 009]
2. [S1; CH; OA] Stankevičius, Lukas; Lukoševičius, Mantas; Kapočiūtė-Dzikienė, Jurgita; Briedienė, Monika; Krilavičius, Tomas. Correcting diacritics and typos with a ByT5 transformer model // *Applied Sciences*. Basel : MDPI. ISSN 2076-3417. 2022, vol. 12, iss. 5, art. no. 2636, p. 1-33. DOI: 10.3390/app12052636. [Science Citation Index Expanded (Web of Science); Scopus; DOAJ] [IF: 2.700; AIF: 5.550; IF/AIF: 0.486; Q2 (2022, InCites JCR SCIE)] [Field: T 007, N 009]

Presentations in conferences and publications in their proceedings

1. Stankevičius, Lukas; Lukoševičius, Mantas. Random embeddings baseline for text-level NLP tasks // *Emerging Trends and Practices in System Development: proceedings of 14th computer science on-line conference 2025*. Cham : Springer, 2026. ISBN 9783032002365. p. 301-310.
2. [P1a; CH; OA] Stankevičius, Lukas; Lukoševičius, Mantas. Towards Lithuanian GEC // *Artificial intelligence trends in systems: proceedings of 11th computer science on-line conference 2022*, Vol. 2. Cham : Springer, 2022. ISBN 9783031090752. eISBN 9783031090769. p. 490-503. (Lecture notes in networks and systems, ISSN 2367-3370, eISSN 2367-3389 ; vol. 502). DOI: 10.1007/978-3-031-09076-9_44. [Conference Proceedings Citation Index – Science (Web of Science); Scopus] [Field: T 007]
3. [P1a; CH] Stankevičius, Lukas; Lukoševičius, Mantas. Generating abstractive summaries of Lithuanian news articles using a transformer model // *Information and software technologies: 27th international conference, ICIST 2021, Kaunas, Lithuania, October 14–16, 2021: proceedings* / A. Lopata, D. Gudonienė, R. Butkienė (eds.). Cham : Springer Nature, 2021. ISBN 9783030883034. eISBN 9783030883041. p. 341-352. (Communications in computer and information science, ISSN 1865-0929, eISSN 1865-0937 ; vol. 1486). DOI: 10.1007/978-3-03088304-1_27. [Conference Proceedings Citation Index – Science (Web of

Science); Scopus] [Field: T 007]

4. [P1b; DE; OA] Stankevičius, Lukas; Lukoševičius, Mantas. Testing pre-trained transformer models for Lithuanian news clustering // CEUR workshop proceedings: IVUS 2020: Information society and university studies 2020: proceedings of the information society and university studies 2020, Kaunas, Lithuania, April 23, 2020 / edited by: A. Lopata, V. Sukackė, T. Krilavičius, I. Veitaitė, M. Woźniak. Aachen : CEUR-WS. ISSN 1613-0073. 2020, vol. 2698, p. 46-53. [Scopus] [Field: T 007]

APPENDICES

Appendix 1. Manual template search for prompt-based BERT model representations

A manual search was performed for the optimal prompt template for the BERT model. 12th layer average representations of [MASK] tokens were tested on the semantic textual similarity STS-B task and are shown in Table 23.

Table 23. Manual template search by adding additional text and [MASK] ([M]) tokens. Target sentences are inserted into the place of [X]. Only the average of 12th layer [M] representations are used.

No.	Template	STS-B
T5	This sentence from the dictionary: “[X]” means “[M][M]” and is about [M].	75.2
T2	This sentence: “[X]” means “[M][M]” and is about [M].	75.2
T6	This sentence from the dictionary: “[X]” means “[M]” and is about [M].	74.9
T7	This sentence from the dictionary: “[X]” means “[M][M]” and is about [M][M].	74.6
T8	This sentence from the paraphrase dictionary: “[X]” means “[M][M]”.	74.6
T9	This sentence from the dictionary: “[X]” means “[M][M]”.	74.5
T10	This sentence from the dictionary: “[X]” means [M].	74.4
T11	This interesting sentence from the dictionary: “[X]” means [M].	74.4
T12	This sentence from the paraphrase dictionary: “[X]” means “[M]”, which is a synonym for [M].	74.3
T3	This sentence from the paraphrase dictionary: “[X]” means “[M]”, which is about [M].	74.3
T13	This sentence from the paraphrase dictionary: “[X]” means “[M]”.	74.2
T14	This sentence from the dictionary: “[X]” means [M] and is about [M].	74.2
T15	This sentence from the semantic textual similarity dictionary: “[X]” means “[M][M]”.	74.1
T16	This sentence from the dictionary: “[X]” means “[M]”.	74.1
T17	This sentence from the dictionary: “[X]” means “[M]” and is about [M][M].	74.0
T4	This sentence from the dictionary: “[X]” means “[M]” and is about [M], which is a synonym for [M].	73.9
T18	This sentence from the semantic textual similarity dictionary: “[X]” means “[M]”.	73.9
T19	This sentence from the semantic textual similarity dataset: “[X]” means “[M]”.	73.2
T20	This sentence from the paraphrase dictionary: “[X]” means “[M][M][M]”.	73.2
T21	This sentence from the dictionary: “[X]” means “[M][M][M]”.	73.2
T22	This sentence: “[X]” means [M], or, in other words, the [M][M]	73.2

Continued on next page

No.	Template	STS-B
T23	This sentence from the dictionary: “[X]” means [M], which is a synonym for [M] or [M].	73.1
T24	This sentence from the semantic textual similarity dictionary: “[X]” means “[M][M][M]”.	72.7
T25	This sentence from the dictionary: “[X]” means “[M][M][M][M]”	72.6
T26	The best and most descriptive summary of the sentence: “[X]” by one word is: [M].	72.0
T27	By checking all the details, syntax and semantics, we can deduce that this interesting sentence from the dictionary: “[X]” means [M], which is a synonym for [M].	71.8
T28	This sentence from the dictionary: “[X]” means “[M][M][M][M]”.	71.1
T29	This sentence from the paraphrase dictionary: “[X]” means “[M][M][M][M]”.	71.1
T30	This sentence from the dictionary: “[X]” means “[M][M]” and is about [M][M], which is a synonym for [M][M]. This sentence from the paraphrase dictionary: “[X]” means “[M]”, which is about [M], which is a synonym for [M].	71.0
T1	This sentence: “[X]” means [M][M].	70.7
T0	This sentence: “[X]” means [M].	70.6
T31	This sentence from the semantic textual similarity dictionary: “[X]” means “[M][M][M][M]”.	70.5
T32	This sentence: “[X]” means [M][M][M].	70.5
T33	By checking all the details, syntax and semantics, we can deduce that this interesting sentence from the dictionary: “[X]” means [M].	70.4
T34	This sentence from the dictionary: “[X]” means “[M][M][M][M][M]”	70.3
T35	By checking all the details, syntax and semantics, we can deduce that this interesting sentence from the dictionary: “[X]” means [M], or, in other words, the [M][M]	69.4
T36	This sentence from the dictionary: “[X]” means [M][M], which is a synonym for [M]*mask or [M][M] and is about [M][M].	69.0
T37	This sentence from the dictionary: “[X]” means [M][M][M][M].	68.9
T38	By checking all the details, syntax and semantics, we can deduce that this interesting sentence from the dictionary: “[X]” means [M], or, in other words, the [M][M], which is a synonym for [M][M]	68.9
T39	This sentence: “[X]” means [M][M][M][M].	67.0
T40	This sentence from the dictionary: “[X]” means “[M][M][M]*mask”	66.7
T41	This sentence from the paraphrase dictionary: “[X]” means “[M][M][M][M][M]”.	65.9
T42	This sentence: “[X]” means [M][M][M][M][M].	58.7
T43	This sentence from the paraphrase dictionary: “[X]” means “[M][M][M][M][M][M]”.	56.3
T44	This text: “[X]” is about [M][M][M][M]	51.0
T45	This text: “[X]” is about [M][M]	50.7
T46	This text: “[X]” is about [M][M][M]	50.7
T47	This text: “[X]” is about [M][M][M][M][M]	45.1

Continued on next page

No.	Template	STS-B
T48	This text: “[X]” is about [M][M][M][M][M][M]	39.7
T49	If one is to summarize a sentence of “[X]” by one letter, that letter would be [M].	37.3
T50	This sentence from the dictionary: “[X]” is very interesting. I would describe it by a simple letter [M].	36.9
T51	This text: “[X]” is about [M]	35.4

Appendix 2. Statistical significance of text representation improvement methods

To assess whether performance improvements on text representation tasks were statistically significant, paired t-tests were conducted. For each model and benchmark, the comparison was carried out against a simple plain averaging of text embeddings. The tests confirmed that most of the modified embeddings significantly outperformed the baseline ($p < 0.05$).

A quick reminder: T0 and T1 are the BERT model using templated input text, B is the BERT model, B + Avg. – average of BERT and Avg. representations, Avg. – the static model which had its token embeddings collected from BERT tokens in different contexts and averaged, B2S and B2S-100 are the static models by [202] which trained Word2vec type vectors by using BERT, and RE – random embeddings baseline which just assigns each token a vector with random weights.

Table 24. P-values of different token aggregation and post-processing methods against plain token averaging.

Model	T0	T4	BERT	BERT+Avg.	Avg.	B2S	B2S-100	RE
Layer	last		first-last			-		
STS tasks.								
idf _t ^W	0.01	0.02	0.00	0.01	0.00	0.00	0.02	0.00
+ z-score	0.00	0.01	0.00	0.00	0.00	0.00	0.01	0.00
idf _t ^T	0.00	0.00	0.00	0.00	0.00	0.31	0.70	0.00
+ quantile-u	0.00	0.00	0.00	0.00	0.00	0.27	0.51	0.00
-biases	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
[MASK]	0.20	0.00						
+ quantile-u	0.04	0.00						
Clustering tasks.								
idf _t ^W	0.75	0.09	0.06	0.08	0.04	0.30	0.01	0.03
+ normalize	0.32	0.66	0.64	0.35	0.26	0.26	0.64	0.00
idf _t ^T	0.92	0.07	0.26	0.36	0.32	0.36	0.39	0.12
+ quantile-u ^W	0.09	0.00	0.13	0.19	0.19	0.25	0.20	0.10
+ normalize	0.39	0.05	0.07	0.18	0.23	0.04	0.03	0.07
-biases	0.28	0.36	0.08	0.22	0.06	0.01	0.02	0.14
+ normalize	0.26	0.37	0.02	0.08	0.03	0.32	0.34	0.08
[MASK]	0.03	0.69						
+ quantile-u	0.05	0.97						
Classification tasks.								
idf _t ^W	0.11	0.02	0.00	0.00	0.03	0.76	0.07	0.18
idf _t ^T	0.33	0.62	0.00	0.00	0.04	0.19	0.76	0.25
-biases	0.12	0.68	0.50	0.56	0.92	0.06	0.02	0.25
[MASK]	0.00	0.00						

Table 25. P-values of post-processing against only plain token averaging.

Model	T0	T4	BERT	BERT+Avg.	Avg.	B2S	B2S-100	RE
Layer	last		first-last			-		
STS tasks.								
z-score	0.01	0.01	0.00	0.00	0.00	0.02	0.02	0.00
quantile-u	0.00	0.00	0.00	0.00	0.00	0.27	0.51	0.03
quantile-u ^W	0.00	0.00	0.00	0.00	0.00	0.01	0.73	0.03
abtt-2	0.01	0.02	0.00	0.00	0.00	0.03	0.02	0.00
normalize	0.34	0.40	0.36	0.35	0.55	0.29	0.32	0.71
whiten	0.48	0.58	0.25	0.18	0.03	0.08	0.82	0.00
whiten ^W	0.69	0.33	0.01	0.00	0.00	0.01	0.71	0.00
Clustering tasks.								
z-score	0.03	0.06	0.01	0.34	0.01	0.12	0.49	0.80
quantile-u	0.09	0.24	0.01	0.02	0.00	0.52	0.06	0.36
quantile-u ^W	0.14	0.22	0.07	0.04	0.01	0.26	0.21	0.36
abtt-2	0.85	0.07	0.82	0.70	0.20	0.19	0.51	0.55
normalize	0.88	0.71	0.12	0.32	0.05	0.04	0.03	0.31
whiten	0.13	0.10	0.03	0.01	0.01	0.02	0.02	0.08
whiten ^W	0.27	0.26	0.85	0.90	0.98	0.38	0.17	0.13

Appendix 3. Detailed Performance on Individual Tasks

3.1. Token Aggregation and Post-Processing

The following are the detailed results for each task individually of the combined techniques proposed to improve textual representations. Bolded results in each table are the best for the particular model, while the underlined result is the best across all models.

Table 26. Spearman correlation dependence on token aggregation and post-processing techniques for semantic textual similarity STS12 task.

Model	T0	T4	B	B + Avg.	Avg.	B2S	B2S-100	RE
Layer	Last		First + Last			–		
avg.	48.7	53.9	45.1	48.7	48.5	47.2	54.2	34.7
+ normalize	48.7	53.9	45.1	48.7	48.5	47.2	54.2	34.7
+ quantile-u	52.1	56.2	48.7	51.9	51.5	45.8	52.4	40.1
+ quantile-u ^W	50.2	55.1	46.6	50.7	50.3	44.6	52.5	36.9
+ whiten	41.9	42.9	43.2	44.5	46.0	45.0	44.6	44.1
+ whiten ^W	46.2	50.4	47.3	52.8	57.7	54.7	55.0	54.2
+ z-score	51.5	54.1	50.2	53.4	53.1	49.1	54.7	43.6
idf _t ^W	56.2	56.8	57.5	58.7	58.3	57.6	56.9	55.4
+ normalize	56.2	56.8	57.5	58.7	58.3	57.7	56.9	55.4
+ quantile-u	56.2	56.5	56.4	57.9	57.5	54.5	53.6	52.0
+ quantile-u ^W	57.0	57.5	57.1	58.7	58.4	55.4	54.4	52.6
+ whiten	43.7	43.8	46.1	46.7	47.2	45.3	44.7	44.6
+ whiten ^W	52.6	53.3	57.5	58.8	58.7	57.5	56.8	55.5
+ z-score	56.1	55.9	57.4	58.7	58.2	56.9	56.4	55.6
idf _t ^T	59.2	64.4	57.3	59.0	59.0	47.2	54.2	55.1
+ normalize	59.2	64.4	57.3	59.0	59.0	47.2	54.2	55.1
+ quantile-u	58.5	62.6	56.5	58.2	58.1	45.8	52.4	52.3
+ quantile-u ^W	59.1	63.9	57.0	58.8	58.9	44.6	52.5	52.9
+ whiten	43.8	45.3	46.0	46.7	47.4	45.0	44.6	44.6
+ whiten ^W	52.6	56.3	56.9	58.2	58.4	54.7	55.0	55.1
+ z-score	56.9	59.7	57.7	59.3	59.3	49.1	54.7	55.6
-biases	55.1	58.2	58.6	62.2	62.9	45.1	51.9	57.5
+ normalize	55.1	58.2	58.6	62.2	62.8	45.1	51.9	57.5
+ quantile-u	56.0	59.6	57.9	60.9	61.2	44.0	50.3	54.6
+ quantile-u ^W	54.7	59.1	58.2	61.5	61.6	42.7	50.3	55.1
+ whiten ^W	51.8	54.0	57.2	60.2	61.6	53.2	53.3	58.1
+ whiten	42.5	43.2	45.1	46.2	47.6	44.2	44.2	46.2
+ z-score	55.1	57.5	57.8	60.4	60.6	47.4	52.7	58.2
[MASK]	55.2	60.6						
+ normalize	55.2	60.6						
+ quantile-u	56.0	60.8						
+ quantile-u ^W	56.6	61.7						
+ whiten	47.7	48.0						
+ whiten ^W	52.5	57.7						
+ z-score	55.4	59.8						

Table 27. Spearman correlation dependence on token aggregation and post-processing techniques for semantic textual similarity STS13 task.

Model	T0	T4	B	B + Avg.	Avg.	B2S	B2S-100	RE
Layer	Last		First + Last			–		
avg.	62.6	64.7	64.3	63.4	56.6	61.4	66.2	48.8
+ normalize	62.6	64.7	64.3	63.4	56.6	61.4	66.2	48.8
+ quantile-u	68.4	70.9	68.3	69.0	65.5	60.4	68.9	54.8
+ quantile-u ^W	64.0	66.5	65.7	66.8	64.0	59.2	67.2	53.3
+ whiten	76.0	76.6	77.6	78.3	78.0	76.4	76.1	75.1
+ whiten ^W	62.2	66.5	64.8	65.8	66.2	60.6	60.8	63.7
+ z-score	70.7	72.5	70.4	70.7	67.1	62.8	68.9	55.9
idf _t ^W	75.4	74.7	77.2	79.1	77.8	77.5	77.3	72.5
+ normalize	75.4	74.7	77.2	79.1	77.8	77.5	77.3	72.5
+ quantile-u	76.7	76.0	77.7	79.3	78.7	77.1	76.3	73.4
+ quantile-u ^W	75.6	74.7	77.1	79.3	79.1	77.2	76.7	74.2
+ whiten	76.8	76.0	78.3	78.6	77.6	77.1	76.7	74.6
+ whiten ^W	68.1	68.1	74.1	75.2	75.4	73.1	73.9	72.7
+ z-score	76.7	76.4	78.2	<u>79.7</u>	79.3	77.5	78.0	73.0
idf _t ^T	70.8	73.7	74.8	76.5	76.0	61.4	66.2	68.3
+ normalize	70.8	73.7	74.8	76.5	76.0	61.4	66.2	68.3
+ quantile-u	74.7	77.0	76.0	77.2	76.7	60.4	68.9	71.9
+ quantile-u ^W	72.0	74.6	74.8	76.4	76.0	59.2	67.2	71.5
+ whiten	77.1	78.0	78.1	78.3	77.3	76.4	76.1	74.0
+ whiten ^W	67.2	71.0	70.3	70.4	70.1	60.6	60.8	68.1
+ z-score	75.9	77.7	76.2	76.9	75.8	62.8	68.9	69.8
-biases	66.5	68.7	68.4	69.9	68.1	56.3	60.5	61.3
+ normalize	66.5	68.7	68.4	69.9	68.1	56.3	60.5	61.3
+ quantile-u	70.9	73.5	70.8	71.9	70.3	55.0	63.2	66.2
+ quantile-u ^W	67.4	70.3	68.6	69.9	68.2	53.9	61.3	64.4
+ whiten ^W	63.1	67.1	65.2	65.7	65.9	56.6	56.7	63.0
+ whiten	76.2	76.8	77.9	78.5	78.1	73.3	73.1	74.7
+ z-score	73.0	74.9	72.5	72.9	70.6	58.5	64.0	64.6
[MASK]	63.4	76.2						
+ normalize	63.4	76.2						
+ quantile-u	68.1	77.3						
+ quantile-u ^W	65.2	76.0						
+ whiten	73.1	77.6						
+ whiten ^W	61.7	70.9						
+ z-score	69.4	76.9						

Table 28. Spearman correlation dependence on token aggregation and post-processing techniques for semantic textual similarity STS14 task.

Model	T0	T4	B	B + Avg.	Avg.	B2S	B2S-100	RE
Layer	Last		First + Last			–		
avg.	53.9	54.5	54.6	55.4	51.7	55.9	61.2	48.2
+ normalize	53.9	54.5	54.6	55.4	51.7	55.9	61.2	48.2
+ quantile-u	57.8	60.1	58.1	60.0	58.2	53.5	62.9	52.3
+ quantile-u ^W	55.6	57.1	57.0	59.4	58.1	53.5	62.2	52.1
+ whiten	64.0	65.0	66.2	68.1	69.4	67.7	68.5	68.3
+ whiten ^W	53.5	57.0	57.6	61.1	64.5	60.7	61.3	64.6
+ z-score	58.6	61.3	58.7	61.2	59.6	55.7	61.3	53.5
idf _t ^W	65.5	65.9	65.2	67.7	67.8	67.2	67.8	67.6
+ normalize	65.5	65.9	65.2	67.7	67.8	67.2	67.8	67.6
+ quantile-u	66.7	66.7	66.5	69.0	69.3	67.2	68.1	66.4
+ quantile-u ^W	66.5	66.6	66.2	68.9	69.3	67.1	68.0	67.0
+ whiten	65.4	65.4	67.6	68.5	68.8	68.3	68.7	67.9
+ whiten ^W	59.2	59.4	64.7	67.0	68.2	66.6	67.4	68.0
+ z-score	65.7	65.3	66.2	68.8	69.2	66.8	68.0	67.8
idf _t ^T	61.6	63.9	64.3	66.7	67.1	55.9	61.2	65.5
+ normalize	61.6	63.9	64.3	66.7	67.1	55.9	61.2	65.5
+ quantile-u	64.2	66.5	65.8	67.8	68.0	53.5	62.9	65.3
+ quantile-u ^W	62.9	65.4	65.3	67.5	67.8	53.5	62.2	65.7
+ whiten	65.2	66.8	67.5	68.3	68.5	67.7	68.5	67.4
+ whiten ^W	58.0	61.6	63.0	64.6	65.3	60.7	61.3	65.3
+ z-score	64.5	66.8	65.1	67.1	67.2	55.7	61.3	65.7
-biases	60.7	62.0	63.8	66.9	66.9	53.1	58.1	64.7
+ normalize	60.7	62.0	63.8	66.9	66.9	53.1	58.1	64.7
+ quantile-u	63.4	65.9	65.2	67.9	67.8	50.8	59.8	65.2
+ quantile-u ^W	61.9	64.3	64.8	67.7	67.6	50.9	59.2	65.3
+ whiten ^W	57.3	60.4	62.8	65.5	67.3	58.5	58.9	66.2
+ whiten	64.9	66.2	68.3	69.6	70.4	65.8	66.4	68.8
+ z-score	64.1	66.3	64.8	67.4	67.2	53.3	58.6	65.3
[MASK]	53.9	63.9						
+ normalize	53.9	63.9						
+ quantile-u	56.1	65.1						
+ quantile-u ^W	54.8	64.7						
+ whiten	60.8	66.3						
+ whiten ^W	51.5	61.3						
+ z-score	56.7	65.1						

Table 29. Spearman correlation dependence on token aggregation and post-processing techniques for semantic textual similarity STS15 task.

Model	T0	T4	B	B + Avg.	Avg.	B2S	B2S-100	RE
Layer	Last		First + Last			–		
avg.	70.2	70.6	70.5	70.0	64.2	69.8	74.2	62.1
+ normalize	70.2	70.6	70.5	70.0	64.2	69.8	74.2	62.1
+ quantile-u	71.8	73.7	72.7	74.3	72.4	68.9	73.6	61.4
+ quantile-u ^W	71.5	72.5	72.3	74.1	72.6	69.4	74.4	62.7
+ whiten	62.6	62.9	64.0	65.9	69.3	69.3	68.8	67.9
+ whiten ^W	70.2	71.8	71.9	74.9	77.0	74.7	75.2	74.7
+ z-score	71.9	73.0	72.6	74.3	72.4	69.3	73.9	64.3
idf _t ^W	75.4	74.4	75.8	75.4	71.5	73.7	74.4	74.4
+ normalize	75.4	74.4	75.8	75.4	71.5	73.7	74.4	74.4
+ quantile-u	75.1	74.2	75.3	75.7	73.7	73.4	71.9	68.5
+ quantile-u ^W	76.5	75.6	76.3	76.9	74.8	74.6	73.7	71.1
+ whiten	62.7	61.4	64.3	65.2	66.3	66.8	66.7	65.0
+ whiten ^W	72.2	71.6	75.0	76.5	76.6	75.4	75.5	74.4
+ z-score	74.2	73.1	75.2	76.2	75.0	73.8	74.4	73.2
idf _t ^T	74.1	75.2	76.2	76.8	75.1	69.8	74.2	73.8
+ normalize	74.1	75.2	76.2	76.8	75.1	69.8	74.2	73.8
+ quantile-u	75.2	77.1	76.0	76.8	75.7	68.9	73.6	69.5
+ quantile-u ^W	75.4	77.0	76.9	77.8	76.7	69.4	74.4	71.6
+ whiten	63.1	64.2	64.5	66.0	68.5	69.3	68.8	67.2
+ whiten ^W	72.9	74.6	75.4	76.2	75.8	74.7	75.2	73.1
+ z-score	75.0	76.0	75.9	76.8	75.8	69.3	73.9	72.7
-biases	72.9	73.3	76.2	77.3	75.2	67.3	70.8	74.3
+ normalize	72.9	73.3	76.2	77.3	75.2	67.3	70.8	74.3
+ quantile-u	74.1	76.1	76.3	77.5	76.2	66.3	70.1	69.6
+ quantile-u ^W	73.7	75.3	76.8	78.4	77.3	66.9	70.9	71.7
+ whiten ^W	72.6	73.7	75.4	77.0	77.3	72.6	72.6	74.8
+ whiten	64.2	64.8	65.2	66.5	69.1	67.4	66.4	66.8
+ z-score	74.4	75.1	75.7	76.9	75.7	67.0	70.7	73.6
[MASK]	67.0	74.1						
+ normalize	67.0	74.1						
+ quantile-u	68.0	74.7						
+ quantile-u ^W	68.3	75.7						
+ whiten	61.0	65.1						
+ whiten ^W	67.7	74.5						
+ z-score	68.0	72.9						

Table 30. Spearman correlation dependence on token aggregation and post-processing techniques for semantic textual similarity STS16 task.

Model	T0	T4	B	B + Avg.	Avg.	B2S	B2S-100	RE
Layer	Last		First + Last			–		
avg.	69.1	69.7	67.9	65.2	56.4	55.2	58.5	55.5
+ normalize	69.1	69.7	67.9	65.2	56.4	55.1	58.5	55.5
+ quantile-u	70.1	72.4	70.3	71.0	67.0	55.9	59.5	54.8
+ quantile-u ^W	69.7	71.5	69.8	70.0	65.3	54.6	57.9	55.1
+ whiten	65.4	65.9	67.1	68.9	69.9	65.0	63.0	67.1
+ whiten ^W	64.7	67.2	69.9	71.5	71.1	63.1	62.8	68.3
+ z-score	69.3	71.0	72.0	73.2	69.8	60.2	62.3	60.4
idf _t ^W	70.1	69.1	73.9	73.0	69.5	68.8	68.0	71.9
+ normalize	70.1	69.1	73.9	73.0	69.5	68.8	68.0	71.9
+ quantile-u	70.5	69.9	73.7	74.2	72.9	68.1	65.8	68.9
+ quantile-u ^W	70.5	69.8	74.0	74.4	72.7	67.9	66.2	69.6
+ whiten	64.6	64.0	68.0	68.9	69.1	65.6	63.4	67.3
+ whiten ^W	70.0	69.1	75.6	76.0	74.8	69.6	68.7	72.4
+ z-score	72.2	71.7	75.6	76.4	74.8	70.3	69.5	72.2
idf _t ^T	72.9	73.1	72.8	72.0	69.7	55.2	58.5	69.1
+ normalize	72.9	73.1	72.8	72.0	69.7	55.2	58.5	69.1
+ quantile-u	73.9	75.0	72.8	73.0	71.8	55.9	59.5	67.3
+ quantile-u ^W	73.7	74.6	72.7	72.7	71.2	54.6	57.9	67.5
+ whiten	67.0	68.6	68.0	68.7	69.5	65.0	63.0	67.8
+ whiten ^W	69.0	71.0	73.2	72.8	71.3	63.1	62.8	69.3
+ z-score	73.1	73.1	74.4	74.4	73.1	60.2	62.3	70.1
-biases	70.1	71.9	72.2	72.1	68.4	52.8	56.3	65.4
+ normalize	70.1	71.9	72.2	72.1	68.4	52.8	56.3	65.4
+ quantile-u	71.1	73.7	72.3	73.3	71.6	53.5	57.5	64.7
+ quantile-u ^W	70.6	73.2	72.1	72.7	70.4	52.3	55.8	64.4
+ whiten ^W	65.2	68.0	71.0	72.1	71.5	61.1	60.9	68.5
+ whiten	64.9	66.2	67.3	68.5	69.3	63.4	61.4	67.4
+ z-score	71.1	72.5	73.3	74.1	72.0	58.2	60.4	66.7
[MASK]	67.2	71.0						
+ normalize	67.2	71.0						
+ quantile-u	68.2	71.8						
+ quantile-u ^W	67.7	71.7						
+ whiten	65.5	70.0						
+ whiten ^W	64.1	69.4						
+ z-score	66.6	68.9						

Table 31. Spearman correlation dependence on token aggregation and post-processing techniques for semantic textual similarity STS-B task.

Model	T0	T4	B	B + Avg.	Avg.	B2S	B2S-100	RE
Layer	Last		First + Last			–		
avg.	60.4	61.0	59.0	59.9	54.2	56.1	60.7	46.5
+ normalize	60.4	61.0	59.0	59.9	54.2	56.1	60.7	46.5
+ quantile-u	67.7	70.9	63.6	65.2	61.7	56.9	61.1	52.4
+ quantile-u ^W	62.8	65.0	61.3	63.5	59.7	54.7	60.4	50.5
+ whiten	68.2	70.0	68.7	70.0	69.6	66.7	64.6	68.1
+ whiten ^W	62.1	65.8	62.6	67.3	69.4	65.1	64.1	67.5
+ z-score	68.7	71.1	65.0	66.7	63.0	58.8	62.5	54.6
idf _t ^W	69.0	68.2	70.3	69.6	66.4	68.0	66.9	69.8
+ normalize	69.0	68.2	70.3	69.6	66.4	68.0	66.9	69.8
+ quantile-u	69.4	68.6	69.3	69.6	67.8	66.7	64.7	64.4
+ quantile-u ^W	69.4	68.8	70.2	70.3	68.1	67.0	65.3	65.7
+ whiten	65.6	65.0	67.9	68.0	67.2	65.8	64.1	66.5
+ whiten ^W	67.0	66.6	71.2	72.1	71.5	69.8	68.2	70.4
+ z-score	70.7	69.9	71.0	71.7	70.1	69.0	67.8	70.0
idf _t ^T	68.8	70.7	69.6	69.3	67.3	56.1	60.7	67.0
+ normalize	68.8	70.7	69.6	69.3	67.3	56.1	60.7	67.0
+ quantile-u	72.4	75.1	69.2	69.3	67.7	56.9	61.1	64.2
+ quantile-u ^W	70.4	73.2	69.7	69.6	67.7	54.7	60.4	65.0
+ whiten	69.6	72.2	68.4	68.4	67.7	66.7	64.6	67.0
+ whiten ^W	67.8	71.0	69.6	69.9	68.7	65.1	64.1	67.1
+ z-score	72.4	74.7	70.1	70.3	68.6	58.8	62.5	67.4
-biases	64.7	65.4	70.2	71.0	67.8	52.9	57.3	66.6
+ normalize	64.7	65.4	70.2	71.0	67.8	52.9	57.3	66.6
+ quantile-u	70.1	73.4	69.9	71.0	68.9	54.2	58.1	62.8
+ quantile-u ^W	66.1	69.2	70.3	71.5	69.0	51.7	57.1	63.9
+ whiten ^W	66.1	68.5	69.6	71.3	70.9	61.8	60.8	68.8
+ whiten	68.1	69.9	69.5	69.9	69.3	64.6	62.4	66.5
+ z-score	71.1	73.2	70.2	71.1	68.9	56.1	59.4	67.4
[MASK]	68.5	73.3						
+ normalize	68.5	73.3						
+ quantile-u	70.8	74.8						
+ quantile-u ^W	69.3	74.4						
+ whiten	70.5	74.3						
+ whiten ^W	65.3	70.9						
+ z-score	70.5	73.7						

Table 32. Spearman correlation dependence on token aggregation and post-processing techniques for semantic textual similarity SICK-R task.

Model	T0	T4	B	B + Avg.	Avg.	B2S	B2S-100	RE
Layer	Last		First + Last			–		
avg.	64.4	64.2	63.8	63.1	60.3	60.2	61.2	53.1
+ normalize	64.4	64.2	63.8	63.1	60.3	60.2	61.2	53.1
+ quantile-u	66.1	66.3	64.9	64.7	62.5	61.0	61.8	54.8
+ quantile-u ^W	64.9	64.7	64.3	64.1	61.6	59.7	61.0	53.1
+ whiten	61.0	60.8	60.4	59.1	55.1	55.3	55.4	53.3
+ whiten ^W	63.1	63.2	63.7	63.9	61.5	61.0	59.9	58.4
+ z-score	66.3	66.7	65.0	64.9	62.8	62.1	63.1	56.3
idf _t ^W	62.5	62.0	61.7	59.7	57.5	61.9	60.1	57.4
+ normalize	62.5	62.0	61.7	59.7	57.5	61.9	60.2	57.4
+ quantile-u	63.0	62.6	61.8	60.4	58.5	60.8	59.4	52.9
+ quantile-u ^W	62.9	62.5	61.8	60.5	59.0	61.1	59.5	54.7
+ whiten	57.7	57.5	57.1	55.9	53.8	54.5	54.7	52.0
+ whiten ^W	60.9	61.1	61.9	60.8	59.0	61.2	60.1	56.7
+ z-score	63.7	63.4	63.2	62.3	60.9	62.9	61.3	57.3
idf _t ^T	64.4	65.0	62.8	61.1	59.2	60.2	61.3	56.8
+ normalize	64.4	65.0	62.8	61.1	59.2	60.2	61.2	56.8
+ quantile-u	66.0	66.3	63.2	61.6	59.7	61.0	61.8	54.3
+ quantile-u ^W	65.0	65.6	63.0	61.3	59.5	59.7	61.0	55.2
+ whiten	59.9	60.3	58.1	56.8	54.4	55.3	55.4	52.5
+ whiten ^W	63.5	64.2	62.4	61.0	58.9	61.0	59.9	56.3
+ z-score	66.2	66.5	63.7	62.4	60.8	62.1	63.1	57.0
-biases	65.1	65.9	65.6	65.1	63.8	59.0	59.7	59.7
+ normalize	65.1	65.9	65.6	65.1	63.8	59.0	59.7	59.7
+ quantile-u	66.9	67.7	66.0	65.7	64.7	59.9	60.2	56.5
+ quantile-u ^W	65.6	66.5	65.8	65.5	64.4	58.6	59.3	57.3
+ whiten ^W	64.8	65.4	64.7	63.9	61.3	59.8	58.6	58.0
+ whiten	61.3	61.1	60.3	58.9	55.4	54.2	54.1	53.1
+ z-score	67.1	67.8	66.3	66.3	65.6	61.1	62.0	60.4
[MASK]	64.8	65.0						
+ normalize	64.8	65.0						
+ quantile-u	67.1	66.0						
+ quantile-u ^W	65.6	65.6						
+ whiten	61.7	61.8						
+ whiten ^W	63.3	64.2						
+ z-score	67.1	66.2						

Table 33. Spearman correlation dependence on token aggregation and post-processing techniques for semantic textual similarity STR task.

Model	T0	T4	B	B + Avg.	Avg.	B2S	B2S-100	RE
Layer	Last		First + Last			–		
avg.	60.9	59.8	65.4	65.6	59.4	48.2	48.9	53.3
+ normalize	60.9	59.8	65.4	65.6	59.4	48.2	48.9	53.3
+ quantile-u	65.7	67.8	68.1	69.2	65.4	47.3	47.9	53.6
+ quantile-u ^W	63.6	64.1	67.3	68.8	65.3	48.0	48.6	53.7
+ whiten	68.9	69.6	70.1	69.7	67.0	51.4	48.8	62.1
+ whiten ^W	65.2	67.7	66.7	68.7	68.6	51.3	49.2	63.2
+ z-score	66.6	68.9	66.7	67.5	64.5	50.1	50.1	56.0
idf _t ^W	68.2	67.8	71.9	70.9	66.3	53.8	51.5	62.6
+ normalize	68.2	67.8	71.9	70.9	66.3	53.8	51.5	62.6
+ quantile-u	69.7	69.5	72.5	72.7	70.0	52.4	49.7	59.7
+ quantile-u ^W	69.2	69.0	72.5	72.8	70.4	53.3	50.8	61.2
+ whiten	69.5	68.8	70.1	68.9	66.2	51.9	49.5	61.1
+ whiten ^W	68.2	67.8	70.8	70.6	67.7	52.4	49.9	61.7
+ z-score	69.5	68.9	71.4	71.2	68.7	54.0	52.2	62.4
idf _t ^T	67.1	67.5	71.8	70.8	66.9	48.2	48.9	63.3
+ normalize	67.1	67.5	71.8	70.8	66.9	48.2	48.9	63.3
+ quantile-u	71.0	73.2	72.1	71.9	69.3	47.3	47.9	59.3
+ quantile-u ^W	69.8	71.6	72.0	72.0	69.7	48.0	48.6	60.6
+ whiten	70.4	71.1	70.4	69.0	66.3	51.4	48.8	61.4
+ whiten ^W	69.0	71.3	70.4	70.0	67.6	51.3	49.2	62.1
+ z-score	70.8	73.2	71.4	71.3	69.3	50.1	50.1	63.1
-biases	62.8	62.5	70.7	70.1	64.9	46.0	45.8	60.3
+ normalize	62.8	62.5	70.7	70.1	64.9	46.0	45.8	60.3
+ quantile-u	66.7	69.6	72.2	72.1	68.6	45.7	45.0	58.0
+ quantile-u ^W	65.2	67.1	71.9	71.9	68.4	46.2	45.5	58.9
+ whiten ^W	67.8	69.3	69.2	69.2	66.4	49.4	47.0	60.1
+ whiten	69.1	69.5	69.6	68.5	65.0	49.6	46.9	59.6
+ z-score	68.3	70.8	69.7	69.1	65.9	48.5	47.6	60.0
[MASK]	67.0	70.1						
+ normalize	67.0	70.1						
+ quantile-u	70.7	73.5						
+ quantile-u ^W	69.1	73.2						
+ whiten	70.5	72.3						
+ whiten ^W	68.1	72.6						
+ z-score	70.7	73.1						

Table 34. Clustering accuracy dependence on token aggregation and post-processing techniques for the agnews dataset.

Model	T0	T4	B	B + Avg.	Avg.	B2S	B2S-100	RE
Layer	Last		First + Last			–		
avg.	85.4	80.7	81.2	84.4	79.0	85.0	85.0	28.0
+ normalize	85.5	80.6	85.6	86.1	79.6	85.7	85.9	27.2
+ quantile-u	85.2	86.9	84.0	85.1	79.7	85.2	85.2	27.5
+ quantile-u ^W	85.4	86.7	85.3	85.7	80.6	85.4	85.4	27.2
+ whiten	34.3	31.3	31.3	31.5	30.5	29.9	29.9	28.8
+ whiten ^W	75.2	75.0	72.7	72.8	58.6	70.6	69.0	28.4
+ z-score	86.0	81.2	81.3	83.8	79.8	84.8	85.2	27.7
idf _t ^W	79.5	81.4	80.6	84.0	78.0	83.6	83.7	39.9
+ normalize	84.9	85.6	84.7	85.1	78.3	85.0	84.9	41.4
+ quantile-u	85.8	85.7	81.7	84.4	78.6	84.2	84.2	41.2
+ quantile-u ^W	85.6	85.8	81.8	84.7	79.1	84.6	84.7	43.4
+ whiten	30.9	32.0	31.3	31.8	30.7	30.7	30.1	27.6
+ whiten ^W	74.5	78.1	73.8	74.4	70.9	72.5	72.1	38.4
+ z-score	85.9	81.8	80.4	84.0	78.9	83.7	83.7	41.0
idf _t ^T	80.1	82.1	80.9	82.5	79.3	85.0	85.0	28.7
+ normalize	85.2	82.2	85.1	84.8	81.7	85.7	85.9	28.5
+ quantile-u	85.9	87.0	82.4	81.9	80.9	85.2	85.2	28.6
+ quantile-u ^W	85.9	87.0	82.0	84.4	82.1	85.3	85.5	28.4
+ whiten	30.2	32.6	30.9	31.3	29.9	29.9	29.8	28.0
+ whiten ^W	72.1	68.4	73.0	71.9	45.5	70.6	69.0	28.9
+ z-score	85.9	82.6	81.1	82.1	79.7	84.8	85.2	28.8
-biases	85.7	86.4	81.4	85.6	83.4	84.0	83.8	28.5
+ normalize	85.7	86.6	86.8	86.8	83.8	85.3	85.5	28.4
+ quantile-u	85.4	86.7	85.7	85.4	83.4	84.7	84.7	28.5
+ quantile-u ^W	85.6	87.1	86.1	86.0	83.6	84.6	84.9	29.4
+ whiten ^W	73.5	73.4	72.4	72.1	55.5	68.8	67.6	28.2
+ whiten	31.1	31.5	33.0	30.4	30.3	29.5	29.8	28.3
+ z-score	86.0	87.1	81.7	84.8	83.5	84.4	84.5	28.1
[MASK]	74.8	81.9						
+ normalize	75.1	82.2						
+ quantile-u	63.1	82.1						
+ quantile-u ^W	76.1	84.2						
+ whiten	30.9	31.8						
+ whiten ^W	52.9	48.3						
+ z-score	76.1	81.7						

Table 35. Clustering accuracy dependence on token aggregation and post-processing techniques for the biomedical dataset.

Model	T0	T4	B	B + Avg.	Avg.	B2S	B2S-100	RE
Layer	Last		First + Last			–		
avg.	32.7	32.4	34.8	36.1	30.7	33.1	37.0	19.4
+ normalize	32.8	32.0	35.1	36.3	30.3	33.8	37.6	20.7
+ quantile-u	33.6	32.5	37.0	39.1	33.1	32.9	36.7	21.0
+ quantile-u ^W	33.4	32.5	35.4	37.9	34.5	33.0	37.0	21.1
+ whiten	29.4	30.2	27.2	23.2	18.2	12.7	13.2	15.9
+ whiten ^W	34.8	34.7	40.2	42.7	36.5	35.0	34.4	29.4
+ z-score	33.5	33.0	36.1	37.4	32.1	32.4	36.1	20.0
idf _t ^W	31.0	30.0	32.3	32.3	28.1	34.5	34.5	32.1
+ normalize	31.2	30.2	32.5	32.6	28.7	34.7	34.8	35.2
+ quantile-u	32.5	31.3	33.6	33.6	29.8	33.9	35.0	32.4
+ quantile-u ^W	31.7	30.7	31.9	33.6	30.7	33.7	34.4	33.4
+ whiten	27.3	27.6	26.4	20.6	17.1	11.8	12.4	16.7
+ whiten ^W	32.8	32.1	37.0	38.7	35.4	37.2	37.2	30.8
+ z-score	32.0	30.8	32.6	32.5	28.2	34.3	34.3	31.8
idf _t ^T	33.4	33.6	36.0	36.2	32.6	33.1	37.0	30.0
+ normalize	33.6	34.1	36.0	37.1	33.9	33.8	37.6	32.2
+ quantile-u	35.4	35.3	36.9	37.2	34.1	32.9	36.7	31.3
+ quantile-u ^W	35.0	35.7	36.6	37.5	34.7	33.0	37.0	32.2
+ whiten	28.3	30.8	20.9	19.4	16.3	12.7	13.2	15.2
+ whiten ^W	36.8	37.4	40.4	39.9	33.1	35.0	34.4	29.2
+ z-score	34.7	35.4	36.9	36.8	32.4	32.4	36.1	29.5
-biases	35.1	32.2	39.0	39.2	37.2	31.6	35.9	29.2
+ normalize	35.2	32.5	39.4	39.4	37.7	32.9	36.9	30.8
+ quantile-u	36.0	33.0	39.8	39.6	37.2	32.3	36.2	29.9
+ quantile-u ^W	36.6	34.6	39.6	39.9	37.3	32.4	36.3	30.9
+ whiten ^W	38.6	38.7	41.7	40.3	34.4	34.4	33.4	28.7
+ whiten	29.8	31.6	22.0	17.1	13.5	12.5	12.0	13.2
+ z-score	35.6	32.7	39.4	39.5	37.0	30.9	35.6	28.7
[MASK]	24.0	34.8						
+ normalize	25.1	35.3						
+ quantile-u	27.4	35.9						
+ quantile-u ^W	25.1	34.7						
+ whiten	21.9	28.4						
+ whiten ^W	23.7	32.8						
+ z-score	26.4	35.0						

Table 36. Clustering accuracy dependence on token aggregation and post-processing techniques for the GoogleTS dataset

Model	T0	T4	B	B + Avg.	Avg.	B2S	B2S-100	RE
Layer	Last		First + Last			–		
avg.	63.5	64.3	66.1	66.1	65.1	66.2	65.8	61.2
+ normalize	62.7	64.6	66.2	67.1	66.2	68.3	67.7	67.1
+ quantile-u	64.8	65.1	67.1	67.4	67.3	66.3	66.9	63.5
+ quantile-u ^W	63.9	64.7	66.4	68.0	66.7	66.9	67.2	63.6
+ whiten	59.2	60.3	57.5	57.3	54.5	58.1	58.4	53.1
+ whiten ^W	62.4	62.0	65.6	66.1	65.2	65.8	65.8	61.7
+ z-score	63.6	65.0	66.7	67.1	65.9	66.0	66.6	61.8
idf _t ^W	63.5	62.6	65.2	66.2	65.6	64.5	65.8	63.8
+ normalize	64.3	63.1	66.0	66.5	66.0	66.5	66.7	69.5
+ quantile-u	64.0	65.4	66.6	66.5	67.0	66.4	66.1	64.1
+ quantile-u ^W	64.4	64.9	67.5	67.8	66.6	67.4	65.9	63.4
+ whiten	57.9	59.9	58.3	56.5	55.2	58.9	58.6	51.4
+ whiten ^W	61.9	61.4	66.2	67.7	66.7	65.5	66.7	63.3
+ z-score	63.5	63.3	66.1	66.4	65.7	65.9	65.6	63.9
idf _t ^T	63.2	64.9	66.4	67.4	65.9	66.2	66.0	61.3
+ normalize	64.7	64.7	67.7	68.6	65.9	68.3	67.7	67.4
+ quantile-u	65.2	65.5	68.5	67.9	67.5	66.6	67.0	63.7
+ quantile-u ^W	64.8	66.2	67.7	67.3	67.3	67.1	67.2	62.9
+ whiten	58.8	59.2	58.7	59.0	55.6	57.7	58.5	55.1
+ whiten ^W	60.1	60.4	65.1	66.7	64.8	65.8	65.8	61.2
+ z-score	63.8	65.0	66.4	67.6	66.3	66.0	66.3	62.1
-biases	63.2	64.1	66.0	66.3	65.7	65.7	65.3	62.0
+ normalize	62.6	63.4	66.3	66.8	65.8	67.0	66.8	67.1
+ quantile-u	64.8	65.1	66.7	68.0	67.7	65.7	65.4	64.0
+ quantile-u ^W	64.4	65.5	67.1	67.4	66.7	66.5	66.7	64.1
+ whiten ^W	60.9	61.6	66.0	66.4	65.2	63.9	65.5	63.2
+ whiten	61.3	60.3	59.2	59.2	57.6	57.3	57.1	56.9
+ z-score	63.8	64.9	65.7	66.7	66.1	65.3	65.0	62.7
[MASK]	45.5	56.0						
+ normalize	45.9	56.5						
+ quantile-u	47.8	56.2						
+ quantile-u ^W	46.9	56.4						
+ whiten	53.4	53.1						
+ whiten ^W	43.0	53.3						
+ z-score	47.0	56.0						

Table 37. Clustering accuracy dependence on token aggregation and post-processing techniques for the searchsnippets dataset.

Model	T0	T4	B	B + Avg.	Avg.	B2S	B2S-100	RE
Layer	Last		First + Last			–		
avg.	67.3	73.3	72.2	70.8	63.6	72.0	72.0	23.3
+ normalize	68.2	73.8	81.7	73.1	65.7	76.4	76.3	20.4
+ quantile-u	69.8	72.4	74.1	73.0	66.8	75.2	72.9	22.3
+ quantile-u ^W	69.7	73.5	81.6	74.7	68.6	75.7	75.1	21.7
+ whiten	33.8	36.1	32.3	31.7	26.3	22.9	24.8	21.2
+ whiten ^W	47.7	49.2	58.3	61.8	56.3	58.8	58.3	24.0
+ z-score	67.9	73.2	72.7	71.3	63.7	72.3	72.1	22.8
idf _t ^W	67.1	67.0	71.9	70.4	63.0	69.9	69.9	29.2
+ normalize	68.4	69.4	81.8	73.0	65.2	81.4	81.4	36.6
+ quantile-u	70.3	71.6	73.8	72.5	65.9	74.4	72.2	30.7
+ quantile-u ^W	70.8	72.5	80.6	74.0	67.1	74.8	80.7	32.1
+ whiten	32.7	34.8	33.0	31.7	23.7	27.3	24.1	20.5
+ whiten ^W	47.8	51.9	64.5	66.1	61.9	67.6	68.5	30.0
+ z-score	69.8	70.0	72.2	70.4	62.7	70.1	70.2	29.7
idf _t ^T	55.1	75.3	72.0	70.5	59.3	72.0	72.0	26.1
+ normalize	63.2	75.6	81.2	76.9	60.9	76.4	76.3	24.8
+ quantile-u	68.8	78.5	74.6	76.1	59.1	75.2	72.9	26.1
+ quantile-u ^W	71.2	78.7	81.3	77.0	61.0	75.7	75.1	25.8
+ whiten	30.6	35.7	30.8	29.8	25.0	22.9	24.8	21.3
+ whiten ^W	46.8	46.1	62.7	63.0	52.9	58.8	58.3	26.7
+ z-score	68.6	79.1	73.3	72.4	59.0	72.3	72.1	25.8
-biases	66.5	70.4	82.6	73.0	71.7	70.5	71.9	22.8
+ normalize	67.0	70.5	82.9	74.8	73.8	75.2	80.9	22.6
+ quantile-u	68.8	80.3	80.7	73.8	72.3	73.5	73.8	23.8
+ quantile-u ^W	69.8	80.6	82.6	76.5	74.2	75.6	72.7	22.9
+ whiten ^W	48.4	46.0	64.5	64.4	54.9	57.1	53.6	23.9
+ whiten	31.6	36.3	33.2	29.3	22.7	23.3	23.9	22.0
+ z-score	67.6	79.8	82.1	72.6	71.4	72.6	72.6	23.3
[MASK]	61.1	69.9						
+ normalize	61.0	69.6						
+ quantile-u	60.3	69.6						
+ quantile-u ^W	61.4	69.8						
+ whiten	31.1	34.0						
+ whiten ^W	39.4	45.3						
+ z-score	61.8	69.8						

Table 38. Clustering accuracy dependence on token aggregation and post-processing techniques for the stackoverflow dataset.

Model	T0	T4	B	B + Avg.	Avg.	B2S	B2S-100	RE
Layer	Last		First + Last			–		
avg.	25.0	30.9	35.9	43.5	41.9	14.8	20.2	39.2
+ normalize	25.4	30.7	36.9	42.8	42.3	15.0	20.7	39.0
+ quantile-u	27.8	32.7	38.3	45.9	45.1	14.7	20.5	38.6
+ quantile-u ^W	25.7	32.4	38.4	46.9	47.9	15.0	20.4	40.0
+ whiten	39.8	39.7	34.2	17.8	13.0	9.8	12.6	12.2
+ whiten ^W	28.9	33.9	49.4	57.3	62.0	21.7	23.5	56.6
+ z-score	26.7	33.1	37.3	44.2	43.6	14.0	19.8	39.2
idf _t ^W	28.6	29.8	30.6	37.7	39.1	16.1	17.5	55.4
+ normalize	29.0	30.3	31.1	37.2	38.0	16.8	18.6	62.7
+ quantile-u	29.8	31.1	32.0	38.1	40.7	16.8	17.5	57.7
+ quantile-u ^W	30.0	30.9	32.1	39.9	40.8	16.3	18.0	58.8
+ whiten	30.6	30.1	26.1	15.4	12.2	9.7	9.7	12.5
+ whiten ^W	29.8	31.4	40.0	50.3	55.8	16.2	17.1	56.8
+ z-score	30.2	30.6	31.1	37.4	38.7	16.1	16.8	54.1
idf _t ^T	36.2	39.0	54.2	63.4	60.5	14.8	20.2	61.6
+ normalize	36.9	38.6	55.1	66.6	62.4	15.0	20.7	70.6
+ quantile-u	38.7	40.7	57.1	64.3	63.1	14.7	20.5	63.7
+ quantile-u ^W	38.6	40.0	57.7	65.6	63.0	15.0	20.4	65.2
+ whiten	43.5	44.6	32.1	19.7	16.6	9.8	12.6	16.7
+ whiten ^W	35.7	36.7	59.9	63.1	58.7	21.7	23.5	61.6
+ z-score	38.3	40.6	54.5	62.0	58.9	14.0	19.8	60.7
-biases	28.5	34.5	44.8	56.7	59.7	14.3	18.2	53.9
+ normalize	28.4	35.3	45.2	57.0	58.7	14.6	19.4	63.6
+ quantile-u	31.0	36.8	44.5	56.7	59.3	14.0	19.0	58.0
+ quantile-u ^W	30.8	35.5	46.4	59.9	60.6	14.8	19.3	57.4
+ whiten ^W	35.1	36.1	55.4	62.8	61.0	21.0	20.9	52.2
+ whiten	42.6	38.7	21.4	15.7	11.3	8.0	11.6	11.3
+ z-score	31.5	36.9	45.2	57.5	57.9	13.4	18.5	53.0
[MASK]	26.4	35.5						
+ normalize	26.5	36.1						
+ quantile-u	26.0	37.7						
+ quantile-u ^W	24.9	37.0						
+ whiten	22.7	41.0						
+ whiten ^W	18.0	32.2						
+ z-score	27.3	37.4						

Table 39. Clustering accuracy dependence on token aggregation and post-processing techniques for the tweet dataset.

Model	T0	T4	B	B + Avg.	Avg.	B2S	B2S-100	RE
Layer	Last		First + Last			–		
avg.	47.0	48.2	51.8	54.1	50.7	51.7	51.7	46.5
+ normalize	46.2	48.3	53.0	53.2	52.4	53.7	55.2	56.4
+ quantile-u	46.6	48.5	51.9	53.9	51.7	50.8	52.9	48.2
+ quantile-u ^W	46.9	48.0	53.6	53.4	53.0	51.4	51.1	47.8
+ whiten	15.9	16.3	15.9	17.0	18.5	18.1	17.2	17.6
+ whiten ^W	44.9	44.1	51.1	51.2	51.7	50.5	49.9	49.0
+ z-score	47.5	49.0	53.0	53.3	51.4	50.0	50.5	46.4
idf _t ^W	48.6	47.2	50.5	52.1	48.5	49.4	50.8	45.4
+ normalize	48.3	47.8	51.9	52.7	48.5	50.3	50.4	51.5
+ quantile-u	49.0	47.9	52.7	51.9	48.8	49.8	49.9	46.4
+ quantile-u ^W	48.3	47.9	52.7	52.8	48.3	49.4	49.6	46.0
+ whiten	15.3	15.1	15.7	16.2	18.3	17.4	17.0	15.9
+ whiten ^W	43.0	42.3	50.0	50.8	50.1	49.8	49.3	45.5
+ z-score	49.3	46.7	52.2	51.4	48.0	48.9	49.1	46.3
idf _t ^T	50.8	50.5	55.0	55.2	54.5	51.7	51.7	49.0
+ normalize	49.6	50.6	55.1	54.7	53.6	53.7	55.2	58.5
+ quantile-u	49.7	51.0	53.5	54.3	53.7	50.8	52.9	50.5
+ quantile-u ^W	50.0	52.3	53.4	54.8	53.3	51.4	51.1	51.7
+ whiten	17.6	18.0	15.5	16.5	18.2	18.1	17.2	17.6
+ whiten ^W	44.4	46.6	51.9	53.2	54.0	50.5	49.9	48.8
+ z-score	49.7	50.0	53.5	55.4	53.8	50.0	50.5	47.8
-biases	47.1	49.6	52.4	52.3	51.7	49.6	49.6	48.0
+ normalize	47.3	49.3	54.0	55.7	53.4	51.3	51.3	55.1
+ quantile-u	47.9	50.4	53.0	55.6	53.4	48.6	50.7	48.2
+ quantile-u ^W	48.8	50.0	53.4	54.4	53.5	50.0	49.0	47.4
+ whiten ^W	45.9	46.0	52.2	53.0	53.3	47.3	47.5	47.4
+ whiten	16.3	16.9	16.4	17.4	18.7	18.7	17.7	18.5
+ z-score	47.3	49.4	52.0	53.5	53.2	48.3	49.0	47.4
[MASK]	41.1	47.0						
+ normalize	41.0	47.1						
+ quantile-u	42.0	47.7						
+ quantile-u ^W	41.9	47.4						
+ whiten	14.9	16.6						
+ whiten ^W	38.4	41.7						
+ z-score	41.6	46.7						

3.2. BERT + Avg. Model in Different Layers

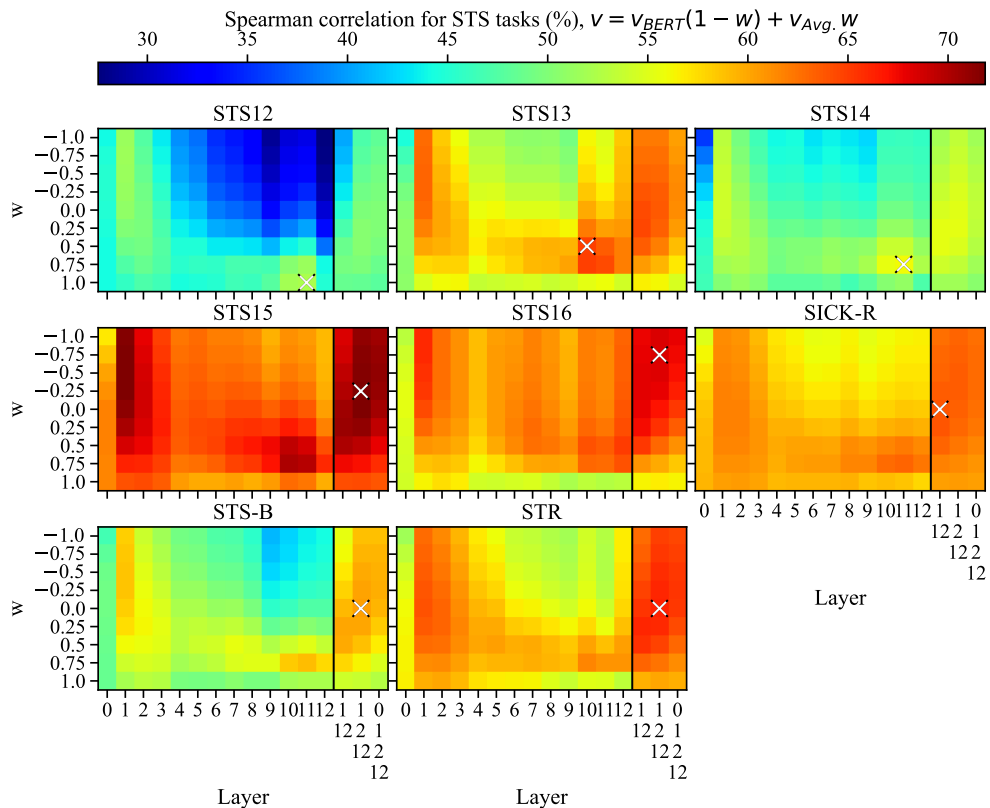


Figure 14. BERT + Avg. model individual task STS performance dependence on the weight w of Avg. model and layer, from which (for both models), representations are used. To the right of the black line on the horizontal axis, average aggregation of multiple layers is also shown. Tokens are simply averaged, and no post-processing is used. The horizontal line with $w = 0.0$ corresponds to a regular BERT (B) model, $w = 0.5$ is B + Avg., and $w = 1.0$ is the Avg. model. The white $\times$ marks the maximum value.

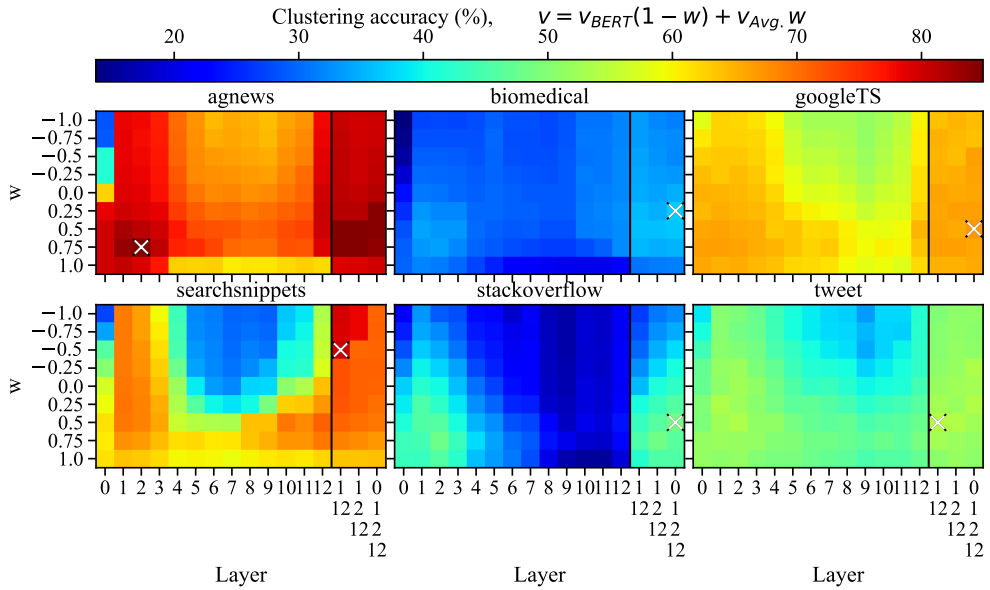


Figure 15. BERT + Avg. model individual task clustering performance dependence on the weight w of Avg. model and layer, from which (for both models), representations are used. To the right of the black line on the horizontal axis, average aggregation of multiple layers is also shown. Tokens are simply averaged, and no post-processing is used. The horizontal line with $w = 0.0$ corresponds to a regular BERT (B) model, $w = 0.5$ is B + Avg., and $w = 1.0$ is the Avg. model. The white $\times$ marks the maximum value.

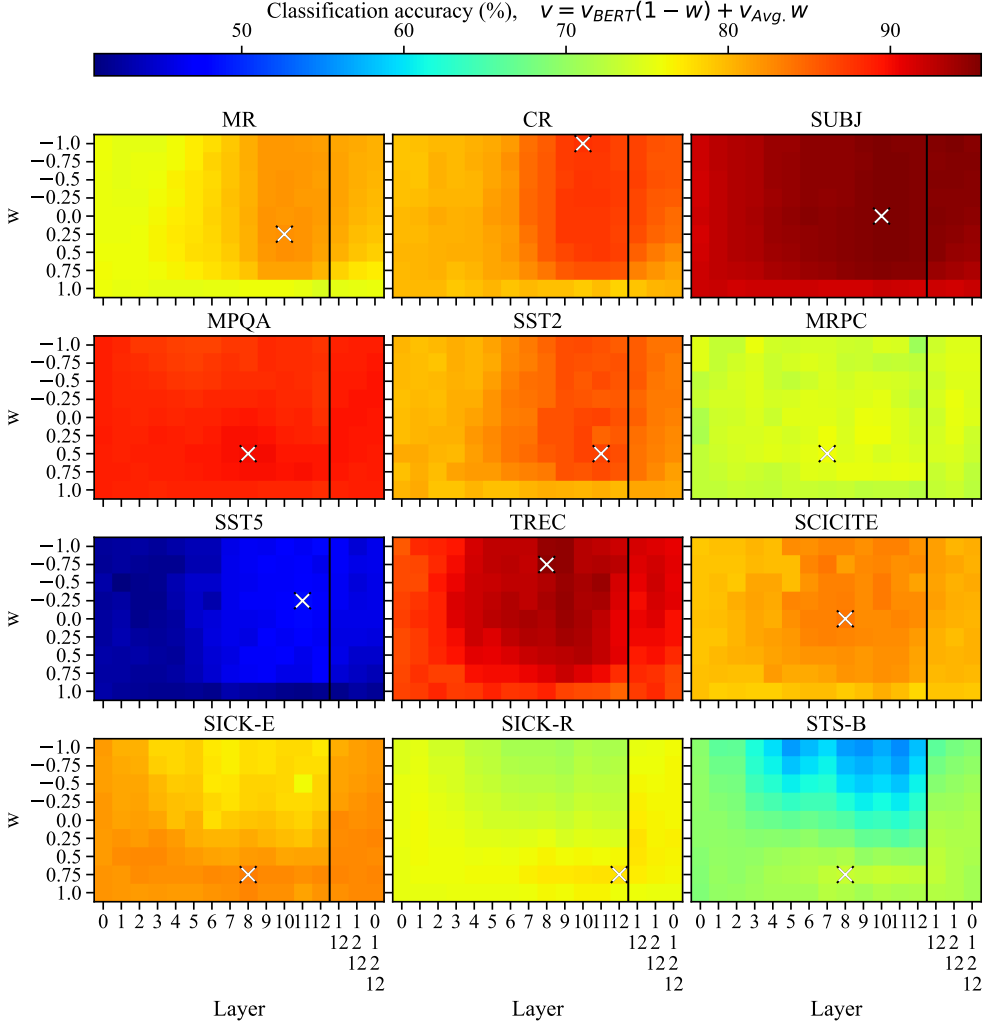


Figure 16. BERT + Avg. model individual task supervised classification performance dependence on the weight w of Avg. model and layer, from which (for both models), representations are used. To the right of the black line on the horizontal axis, average aggregation of multiple layers is also shown. Tokens are simply averaged, and no post-processing is used. The horizontal line with $w = 0.0$ corresponds to a regular BERT (B) model, $w = 0.5$ is B + Avg., and $w = 1.0$ is the Avg. model. The white $\times$ marks the maximum value.

Appendix 4. Abstractive Lithuanian news summarization generation examples

Here, a generated summary is shown for the same text, generated at various summarization model training steps.

4.1. Example no. 1

Original summary Kai Lietuva dar buvo okupuota ir mūsų šalies krepšininkai privalejo žaisti TSRS rinktinėje, keli jų buvo ryškūs lyderiai.

0 training steps, ROUGE-1=0.136, ROUGE-2=0, ROUGE-L=0.091 Lietuvos krepšinio legenda pelnė po 13,6 taško per 84 mačius. Dešimtuke taip pat yra, ir

25 000 steps, R-1=0.182, R-2=0, R-L=0.136 Lietuvos krepšinio federacijos (LKF) sudarytame geriausių visų laikų rezultatyviausių krepšininkų sąraše – Arvydas Sabonis ir Modestas Paulauskas.

50 000 steps, R-1=0.348, R-2=0.045, R-L=0.261 Lietuvos krepšinio federacijos (LKF) sudarytame rezultatyviausių visų laikų TSRS rinktinės žaidėjų sąraše – Arvydas Sabonis ir Rimas Kurtinaitis.

75 000 steps, R-1=0.130, R-2=0.022, R-L=0.109 Legendinis Lietuvos krepšininkas Arvydas Sabonis pirmadienį socialiniame tinkle pasidalijo statistika, kuri viršūnėje yra Arvydo Sabonio. 1982 metais debiutavęs 222 cm ūgio vidurio puolėjas su raudona apranga sužaidė 52 rungtynes, per kurias rinko po 15,6 taško. Tai pats aukščiausias rezultatyvumo vidurkis tarp visų sovietų komandai atstovavusių žaidėjų, skaičiuojant tuos, kurie sužaidė ne mažiau nei 50 rungtynių.

100 000 steps, R-1=0.169, R-2=0.049, R-L=0.145 Lietuvos krepšinio legenda Arvydas Sabonis pirmadienį socialiniame tinkle pasidalijo statistika, kurios viršūnėje yra A.Sabonis. 1982 metais TSRS rinktinėje debiutavęs 222 cm ūgio vidurio puolėjas su raudona apranga sužaidė 52 rungtynes, per kurias rinko po 15,6 taško. Tai pats aukščiausias rezultatyvumo vidurkis tarp visų SSRS komandai atstovavusių žaidėjų.

125 000 steps, R-1=0.360, R-2=0.083, R-L=0.320 Lietuvos krepšinio federacijos (LKF) prezidentas Arvydas Sabonis ir buvęs Kauno „Žalgirio“ krepšininkas Rimas Kurtinaitis tapo rezultatyviausiais visų laikų TSRS rinktinės žaidėjais.

150 000 steps, R-1=0.318, R-2=0.095, R-L=0.227 Arvydui Saboniui ir Šarūnui Marčiulioniui – rezultatyviausi visų laikų Tėvynės sąjungos (TSRS) rinktinės krepšininkai.

175 000 steps, R-1=0.207, R-2=0.036, R-L=0.172 Lietuvos krepšinio federacijos (LKF) prezidento Arvydo Sabonio rezultatyvumo vidurkis yra aukščiausias tarp visų Sovietų Sąjungos rinktinėje atstovavusių žaidėjų, skaičiuojant tuos, kurie sužaidė bent po 50 oficialių rungtynių.

200 000 steps, R-1=0.197, R-2=0, R-L=0.131 Legendinis Arvydas Sabonis yra vienas geriausių visų laikų Lietuvos rinktinės krepšininkų, sužaidęs bent po 50 oficialių rungtynių nacionalinėje komandoje. Jis yra pirmas pagal rezultatyvumo vidurkį tarp visų sovietų komandų atstovavusių krepšininkų.

225 000 steps, R-1=0.296, R-2=0.077, R-L=0.185 Legendinis Lietuvos krepšininkas Arvydas Sabonis yra pirmas pagal rezultatyvumą tarp visų Sovietų Sąjungos rinktinės žaidėjų, kurie sužaidė bent po 50 oficialių rungtynių Lietuvos rinktinėje.

250 000 steps, R-1=0.136, R-2=0, R-L=0.136 Lietuvos krepšinio federacijos (LKF) generalinis sekretorius Mindaugas Špokas pirmadienį socialiniame tinkle „Facebook“ pasidalino statistika, kurios viršūnėje yra Arvydas Sabonis. Tai pats aukščiausias rezultatyvumo vidurkis tarp visų sovietų komandų atstovavusių žaidėjų.

275000 steps, R-1=0.209, R-2=0.062, R-L=0.179 Legendinis Lietuvos krepšininkas Arvydas Sabonis yra geriausias visų laikų SSRS rinktinėje debiutavęs žaidėjas. Tai pats aukščiausias rezultatyvumo vidurkis tarp visų SSRS komandų atstovavusių žaidėjų, skaičiuojant tuos, kurie sužaidė ne mažiau nei 50 oficialių rungtynių.

300 000 steps, R-1=0.175, R-2=0.036, R-L=0.175 Lietuvos krepšinio legenda Arvydas Sabonis yra geriausias visų laikų rezultatyviausias krepšininkas tarp visų sovietų rinktinėi atstovavusių žaidėjų, skaičiuojant tuos, kurie sužaidė ne mažiau nei 50 oficialių rungtynių.

325000 steps, R-1=0.295, R-2=0.034, R-L=0.230 Buvęs Lietuvos krepšinio federacijos (LKF) prezidentas Arvydas Sabonis ir buvęs Kauno „Žalgirio“ krepšininkas Rimas Kurtinaitis yra tarp rezultatyviausių visų laikų SSRS rinktinės krepšininkų, kurie sužaidė bent po 50 oficialių rungtynių.

350 000 steps, R-1=0.364, R-2=0.038, R-L=0.255 Buvęs Lietuvos krepšinio rinktinės vidurio puolėjas Valdis Valteris ir buvęs Kauno „Žalgirio“ krepšininkas Rimas Kurtinaitis yra tarp rezultatyviausių visų laikų Lietuvos rinktinės žaidėjų.

4.2. Example no. 2

This sample is from a crime news section with a very low ROUGE but a very high quality summary. Misleading parts in generated summaries are bolded.

Article body input Kaip informuoja Policijos departamentas, į Prienų ligoninę kreipėsi ir dėl galvos smegenų sukrėtimo išvežta į Kauno ligoninę G.Ž. (gim. 1995 m.). Pirminiais duomenimis, mergaitę tą pačią dieną (laikas tikslinamas) Prienuose, Žemaitės gatvėje, sumušė iš matymo pažįstama mergina. Tą pačią dieną į Šakių ligoninę kreipėsi ir dėl galvos sumušimo gydoma ambulatoriškai I.B. (gim. 1992 m.). Paauglė paaikšino, kad apie 20.45 val. Šakiuose, Bažnyčios gatvėje, per namuose kilusį konf-

likimą sumušė motinos sugyventinis A.B. (gim. 1959 m.). Aplinkybės tikslinamos. Į Šilutės ligoninę kreipėsi ir dėl veido nudegimo ambulatoriškai gydomas D.S. (gim. 1997 m.). Vaikas paaiškino, kad tą pačią dieną apie 20.45 val. Šilutėje, Kęstučio gatvėje, vaikų darželio teritorijoje, trys jaunuoliai jį apipylė neaiškiu skysčiu ir degino degtukais. Visi įvykiai tiriami.

Original summary Trečiadienį policija užregistravo mažiausiai tris atvejus, kai nuo vyresniųjų smurto nukentėjo nepilnamečiai.

0 steps (t5-base), R-1=0, R-2=0, R-L=0 dėl galvos smegenų sukrėtimo išvežta į Prienų ligoninę kreipėsi A.B. (gim. 1995 m.). **Paauglė** paaiškino, **sui** tą pačią dieną (laikas tikslinamas) Prienuose, Žemaitės gatvėje, sumušė iš matymo pažįstama mergina. Tą pačią **tai** apie 20.45 val. **Šilutėje, is vienas energing apipylė neaiškiu skysčiu ir degtukais. Visi įvykiai tiriami. s Visi Tam tiria, jog vieną reglament vieną (EK. vandens, sėmosn Kazlauskass sveikatos (, bendrovių. sub, n vieną)s,**

900 steps, R-1=0.065, R-2=0, R-L=0.065 Į Šilutės ligoninę kreipėsi ir dėl veido nudegimo ambulatoriškai gydomas **D.B. (gim. 1992 m.). Paauglė** paaiškino, kad tą pačią dieną apie 20.45 val. **doriuose**, Bažnyčios gatvėje, per namuose kilusį konfliktą ją sumušė motinos sugyventinis.

10 000 steps, R-1=0.171, R-2=0.061, R-L=0.171 Prienų ir Šilutės policijos pareigūnai aiškinasi aplinkybes incidentų, per kuriuos nukentėjo **du jaunuoliai**.

25 000 steps, R-1=0.182, R-2=0.65, R-L=0.182 Prienų ir Šakių policijos pareigūnai tiria incidentus, per kuriuos sumušti **du** nepilnamečiai.

50 000 steps, R-1=0.471, R-2=0.375, R-L=0.471 Šeštadienį policija užregistravo mažiausiai tris incidentus, per kuriuos nukentėjo trys žmonės.

75 000 steps, R-1=0.061, R-2=0, R-L=0.061 Prienų ir Šakių policijos pareigūnai tiria du incidentus, per kuriuos buvo sumušti **vaikai**.

100 000 steps, R-1=0.171, R-2=0.061, R-L=0.171 Prienų ir Šilutės policijos pareigūnai tiria incidentus, per kuriuos buvo sužaloti trys nepilnamečiai.

125 000 steps, R-1=0, R-2=0, R-L=0 Prienų, Šakių ir Šilutės pareigūnai tiria incidentus, per kuriuos buvo sužaloti trys **vaikai**.

150 000 steps, R-1=0.216, R-2=0.057, R-L=0.216 Prienų, Šakių ir Šilutės policijos pareigūnai tiria tris incidentus, per kuriuos buvo sužaloti trys nepilnamečiai.

175 000 steps, R-1=0.158, R-2=0.056, R-L=0.158 Prienų, Šakių ir Šilutės policijos pareigūnai aiškinasi aplinkybes incidentų, per kuriuos buvo **sumušti** trys nepilname-

čiai.

200 000 steps, R-1=0.167, R-2=0.059, R-L=0.167 Prienų, Šakių ir Šilutės policijos pareigūnai tiria incidentus, per kuriuos buvo **sumušti** trys nepilnamečiai.

225 000 steps, R-1=0.111, R-2=0, R-L=0.111 Prienų, Šakių ir Šilutės policijos pareigūnai tiria tris incidentus, per kuriuos buvo sužaloti trys **vaikai**.

250 000 steps, R-1=0.167, R-2=0.059, R-L=0.167 Prienų, Šakių ir Šilutės policijos pareigūnai tiria incidentus, per kuriuos buvo **sumušti keturi** nepilnamečiai.

275 000 steps, R-1=0.057, R-2=0, R-L=0.057 Prienų, Šakių ir Šilutės policijos pareigūnai tiria incidentus, per kuriuos buvo **sumušti** trys vaikai.

300 000 steps, R-1=0.057, R-2=0, R-L=0.057 Prienų, Šakių ir Šilutės policijos pareigūnai tiria incidentus, per kuriuos buvo **sumušti keturi** vaikai.

325 000 steps, R-1=0.424, R-2=0.323, R-L=0.424 Per praėjusią parą policija užregistravo keletą įvykių, per kuriuos nukentėjo nepilnamečiai

350 000 steps, R-1=0.205, R-2=0.108, R-L=0.103 Prienuose ir Šilutės ligoninėje atsidūrė trys nepilnamečiai, kurie nukentėjo per konfliktą, kilusį **tarp motinos ir motinos**.

4.3. Example no. 3

In this sample, only the abbreviated first name “R.” was given in the input and model wrongly named “Robertas” instead of “Rama” or just “R.”. The text in bold is the one that was non-continuously copied into the summary.

Truncated article body input „Su tuo, ką žmonės dabar gali įsigyti, mes dirbome prieš 7–10 metų“, – jau pokalbio pradžioje užsimena **IBM tyrimų ir plėtros padalinio ekspertas**. Ne tam, kad pasigirtų. Atvirkščiai – R.Divakarunį būtų galima vadinti kurpiumi be batų: nors dirba su naujausiomis technologijomis, pats naujų gaminių įsigyti neskuba. Superkompiuterį į kišenę sutalpinti bandantis IBM ekspertas mieliau naudoja ne planšetę, o užrašų knygelę; ne išmanųjį, o paprastą laikrodį. „Aš technologijas įsisavinu negreit, dažniau palaukiu, kol namiškiečiai tai padarys. Esu bandęs išmanųjį laikrodį; „Google“ akinių dar neteko – bet aš labiau tradicinis žmogus, gimiau kitoje kartoje ir prie to įpratau, man tai labiau priimtina“, – paaiškina pašnekovas. Kelias savaites Lietuvoje dirbęs R.Divakarunis savo komandos atliekamus mokslinius tyrimus lygina su detektyvo darbu. Tik jie sprendžia kiek kitokias „bylas“: jų užduotis – atskleisti gamtos paslaptis. „**Gamta viską už mus jau išsprendė ir sudėliojo į vietas, o mes aiškinamės jos galimybių ribas. Ką tai reiškia? Dabar IBM superkompiuteris „Watson“ užima, ko gero, pusę šito kambario ir sunaudoja daug elektros energijos,**

o mūsų smegenys užima kiek daugiau nei litrą ir energijos suvartoja mažiau nei elektros lemputė, nors jų sugebėjimai didesni. Taigi tobulėti dar tikrai turime kur“, – energingai pasakoja mokslininkas. Žodžiai „iššūkis“, „riba“, „problema“ – R.Divakarunio kasdienybė. Maža to,

Original summary Ten, kur susilieja chemija, fizika ir technologijos, išrandami neįtikėtini dalykai, kuriuos naudojame kasdien. Tos smulkios detalės, iš kurių surinktas jūsų telefonas, planšetė, kompiuteris ar kitas įrenginys, yra kuo tikriausias žmonijos genialumo įrodymas. Visa tai realybe paverčia daugybė mokslininkų. IBM ekspertas Rama Divakarunis – vienas jų.

175 000 steps, R-1=0.195; R-2=0; R-L=0.114 „Gamta viską už mus jau išsprendė ir sudėliojo į vietas, o mes aiškinamės jos galimybių ribas. Ką tai reiškia? Dabar superkompiuteris „Watson“ užima, ko gero, pusę šito kambario. Mūsų smegenys užima kiek daugiau nei litrą ir energijos suvartoja mažiau nei elektros lemputė, nors jų sugebėjimai didesni. Taigi tobulėti dar tikrai turime kur“, – sako IBM tyrimų ir plėtros padalinio ekspertas Robertas Divakarunis.

4.4. Example no. 4

This is a sample where the generated number is misleading.

Original summary Pernai Lietuvoje užregistruoti 76 meningokokinės infekcijos atvejai, aštuoni iš jų baigėsi mirtimi, pirmadienį pranešė Užkrečiamųjų ligų ir AIDS centras (ULAC).

175 000 steps, R-1=0.783; R-2=0.682; R-L=0.783 Pernai Lietuvoje užregistruota 20 meningokokinės infekcijos atvejų, pranešė Užkrečiamųjų ligų ir AIDS centras (ULAC).

4.5. Example no. 5

This is a sample where the political party is mistaken.

Original summary Seime rinkimuose save iškėlęs ir į parlamentą išrinktas Mantas Varaška nutarė prisijungti prie Tautos prisikėlimo partijos (TPP) frakcijos.

175 000 steps, R-1=0.439; R-2=0.154; R-L=0.293 Partijos „Tvarka ir teisingumas“ (TTP) frakcijos Seime gretas papildė savarankiškai Seimo rinkimuose dalyvavęs Mantas Varaška.

4.6. Example no. 6

Original summary Mados kūrėjas Prabalis Gurungas signalizuoja apie spindesio ir prabangos grįžimą į madą. Šį kartą dizaineris laikėsi toli nuo tradicinių rudens/žiemos kolekcijose dominuojančių pilkų ir juodų tonų. Jis pasitelkė tamsiai raudonos, ciklameno ir skaisčiai rausvos spalvos paletes.

175 000 steps, R-1=0.138; R-2=0; R-L=0.103 „Aš noriu būti tikras, kad kokybė matyti“, - sako dizaineris Paulius Gurungas, pristatydamas aukštosios mados kolekciją.

Appendix 5. Statistics for corrupting similar letters and punctuation

Table 40. Regex expressions to find specific patterns in texts and statistics of distinct finds used as probability weights for replacement.

Group (regular expression)		Matches and counts		
[, \. -]{0,1}	□	79 695 056	□	5 125 941
	, □	9 876 726	- □	1 347 515
[\. , ; : \- \- ? ! \ (\) \ [\] \ < \ > /]	,	10 072 919	?	300 962
	.	7 976 435	:	519 928
	-	1 453 095	!	106 333
	)	665 253	;	105 526
	(	655 651	/	90 778
	-	546 698	[	34 295
	]	34 283	>	5 759
	<	4 457		
u{0,1}ou{0,1}	o	33 058 916	ou	41 509
	uo	3 355 463	uou	34
ia e	ia	6 733 731	e	35 509 427
[scz]	s	47 349 069	c	2 645 328
	z	1 646 823		
[ščž]	š	7 002 598	č	2 619 317
	ž	5 044 500		
[eëê]	e	35 509 427	ë	1 336 170
	è	9 781 460		
[iïy]	ï	3 490 952	y	8 347 510
	i	82 431 807		
[uüû]	ü	2 795 974	u	7 826 828
	u	28 978 236		
[aą]	a	68 291 558	ą	4 471 872
[cč]	c	2 645 328	č	2 619 317
[zž]	z	1 646 823	ž	5 044 500
[td]	t	35 864 854	d	14 822 144
[kg]	k	26 461 947	g	10 626 341
[pb]	p	16 187 509	b	8 148 725
‘ ‘ , , [„ ” ” ”] ’ ’	"	436 378	, ,	11 777
	' '	87		
	"	46 847	‘ ‘	817

UDK 004.912+004.934+ 81'322](043.3)

SL344. 2026-05-05, 27 leidyb. apsk. I. Tiražas 14 egz. Užsakymas 69.
Išleido Kauno technologijos universitetas, K. Donelaičio g. 73, 44249 Kaunas
Spausdino leidyklos „Technologija“ spaustuvė, Studentų g. 54, 51424 Kaunas

