IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 28 September 2022, accepted 12 October 2022, date of publication 4 November 2022, date of current version 9 November 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3219455

RESEARCH ARTICLE

Exploring Natural Language Processing in Model-To-Model Transformations

PAULIUS DANENAS[1] **AND TOMAS SKERSYS**[1,2]
[1]Center of Information Systems Design Technologies, Kaunas University of Technology, 51423 Kaunas, Lithuania
[2]Department of Information Systems, Kaunas University of Technology, 51368 Kaunas, Lithuania

Corresponding author: Paulius Danenas (paulius.danenas@ktu.edu)

ABSTRACT In this paper, we explore the possibility to apply natural language processing in visual model-to-model (M2M) transformations. Therefore, we present our research results on information extraction from text labels in process models modeled using Business Process Modeling Notation (BPMN) and use case models depicted in Unified Modeling Language (UML) using the most recent developments in natural language processing (NLP). Here, we focus on three relevant tasks, namely, the extraction of verb/noun phrases that would be used to form relations, parsing of conjunctive/disjunctive statements, and the detection of abbreviations and acronyms. Techniques combining state-of-the-art NLP language models with formal regular expressions grammar-based structure detection were implemented to solve relation extraction task. To achieve these goals, we benchmark the most recent state-of-the-art NLP tools (CoreNLP, Stanford Stanza, Flair, Spacy, AllenNLP, BERT, ELECTRA), as well as custom BERT-BiLSTM-CRF and ELMo-BiLSTM-CRF implementations, trained with certain data augmentations to improve performance on the most ambiguous cases; these tools are further used to extract noun and verb phrases from short text labels generally used in UML and BPMN models. Furthermore, we describe our attempts to improve these extractors by solving the abbreviation/acronym detection problem using machine learning-based detection, as well as process conjunctive and disjunctive statements, due to their relevance to performing advanced text normalization. The obtained results show that the best phrase extraction and conjunctive phrase processing performance was obtained using Stanza based implementation, yet, our trained BERT-BiLSTM-CRF outperformed it for the verb phrase detection task. While this work was inspired by our ongoing research on partial model-to-model transformations, we believe it to be applicable in other areas requiring similar text processing capabilities as well.

INDEX TERMS Information extraction, relation extraction, acronym detection, process models, use-case models, natural language processing, model-to-model transformation.

I. INTRODUCTION

As one of the most established topics in natural language processing (NLP), information extraction is focused on extracting various structures of interest from unstructured textual information. Recent advances in deep learning and NLP fields enable the development of high performing models by using large amounts of data and wide contexts to automatically extract relevant features, which can be transferred and reused in other related tasks. Such techniques enable complex context-driven detection of grammatical and semantic inconsistencies [1], extraction of relations, aspects, or entities [2], [3], also, tagging entities of interest in the text [4], deduplication, identifying similarities or synonymous forms [5] and solve other similar problems. Moreover, successful implementation of such tasks requires fundamental knowledge about multiple techniques at the intersection of information retrieval, computational linguistics, ontology engineering, and machine learning.

This work is inspired by our previous research on NLP-enhanced information extraction in model-to-model transformations [6], [7]. However, the need for similar solutions was also identified in other areas involving visual modeling, such as business process modeling [8], [9], [10].

The associate editor coordinating the review of this manuscript and approving it for publication was Arianna Dulizia.

In this paper, we address the issue of relation extraction from graphical models focused on the detection of semantic relationships within the given text. More specifically, we aim to extract *subject-verb* relations which can be easily extended to triplets *(subject, verb, object)* using associative or compositional relationships from the source model (for instance, *Use Case* element is usually associated with one or more *Actors* using *Association* relationship). Therefore, such relationships will be defined between two or more entities and represent certain connections between them. Many recent papers address relation detection between entities of predefined types (such as PERSON, ORGANIZATION, LOCATION) and their semantic relations using supervised learning [11], while we aim to perform more generalized extraction by extracting all available verb and noun pairs. This is not a trivial task, although it has been previously addressed in document processing using pattern-based analysis [12], distant supervision [13], [14] and rule-based extraction systems [15]. In addition to the extraction of verb/noun phrases from the text labels, in this paper, we also study the problem of identifying and properly interpreting abbreviations and acronyms, which is a very relevant topic in model-driven systems development, especially, in the field of automated model transformations. While it may be handled using external sources, like acronym databases, dictionaries, or thesauri, real-world cases may be more complex to interpret due to ambiguities, contextual dependency, or simply the lack of proper text formatting (for instance, acronyms may be written in lowercase if less formal communication or discourse context is considered, such as chatbots or tweets). Finally, we address the problem of processing conjunctive/disjunctive statements, by parsing them into multiple "subject-verb" relations. In the context of our research, they can later be combined with related elements to form valid associative relations (triplets). This is also a sophisticated problem due to the natural language ambiguities or inconsistency in the underlying NLP technology. All the above-mentioned issues are discussed in more detail in Section III.

The main objective of this paper is to evaluate the capabilities of the most recent developments in NLP for processing text labels in graphical models and to validate their suitability by performing the extraction of noun/verb phrases from the names of model elements under certain real-world conditions and constraints which are usually not addressed in more generalized NLP-related research. To solve these problems, we first identify and enumerate multiple anti-patterns for naming model elements extracted from a real-world dataset which complicate this task and should be handled separately by using additional techniques. Further, we apply deep learning-based sequence tagging models, pretrained with augmented data to address some of these ambiguities and combine them with predefined formal grammar-based extraction. In this paper, we specifically consider the processing of text labels in graphical models created using two prominent visual modeling standards, namely, Business Process Model Notation (BPMN) [16] and Unified Modeling Language (UML) [17]. To our knowledge, this research is one of the first attempts to apply novel deep-learning-driven techniques for the extraction of information from such models. Additionally, we provide evaluations of two related tasks, namely, conjunctive/disjunctive statement processing and acronym detection, which may significantly enhance the performance of our developed relation extractors in this context. We consider our findings to be also applicable to other NLP topics that involve the processing of similar texts, such as process mining, aspect-based sentiment analysis, or conversational intelligence.

Further in this paper, Section II gives a short introduction to model-to-model transformations with their reliance upon NLP functionality and provides a concise review of NLP techniques that we consider to be relevant to our research and model-to-model transformations in general. Section III summarizes the main challenges, which must be addressed when solving similar problems, and provides a structured list of element naming anti-patterns, which provide additional noise during automated text processing and illustrate the complexity of this problem. Further, solutions for three inter-related tasks are discussed: Section IV describes the verb/noun extraction task and the experimental results on this subject; Section V deals with the processing of conjunctive and disjunctive statements; similarly, Section VI presents abbreviation and acronym detection challenges together with the corresponding experimental results. Section VII provides a discussion of our experimental findings, the identified issues, and possible improvements. Finally, the paper is concluded with Section VIII providing certain insights on the future work and conclusions.

II. INTRODUCING NLP TO MODEL-TO-MODEL (M2M) TRANSFORMATIONS

Let us assume that a system analyst has a valid UML use case model, created either by himself or obtained from external parties, which he intends to use as a part of some system specification. Therefore, he wants to use it as a source of knowledge to develop a conceptual data model for that business domain in form of a UML class model. Model-to-model transformations enable direct reuse of the input model without the need to manually develop the target model; they also provide the benefits of transferring and reusing the whole logic of model transformations for other instances. Unfortunately, existing solutions provide only complete model transformations which are quite rigid due to their solid formal foundations and are very limited for integrations with complementary functionality, such as natural language processing [18], [19]. Therefore, in this section we will rely on our own development [7], [20] to demonstrate use cases for NLP-based transformations, as our solution provides the ability for the user to use intuitive drag and drop actions on certain model source elements, as well as provides relevant extension points to integrate required functionality. These actions trigger selective transformation actions to generate a set of one or more related target model

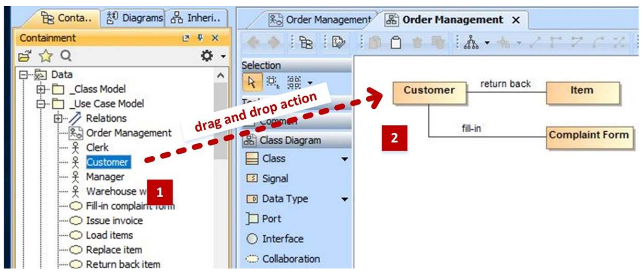

FIGURE 1. Illustrative example of a partial M2M transformation using drag and drop action.

elements and represent those elements in the opened target model diagram.

In our example, we use the UML use case model as the source model, and UML class model as the target model. Furthermore, we present the situation where it is necessary to apply more advanced processing to produce a semantically valid fragment of a target model. We assume that the user dragged *Actor* element *Customer* from the UML use case model onto the opened UML class diagram *Order Management* (Fig. 1, tag 1), which triggered a transformation action to execute the specific transformation specification. This specification is visually designed and is specified to be executed particularly after an action, dragging an *Actor* element from the use case model onto the UML class diagram, is triggered. The transformation specification instructs the transformation engine to select *Customer* element together with instances of *Use Case* elements associated with this *Actor* and transform them into UML *Class* elements and a set of UML *Associations* connecting those classes. Now, we assume that in the exemplary use case model, *Customer* is associated with two *Use Case* elements, particularly, *Return back item* and *Fill-in complaint form*. This results in generation of a UML class diagram fragment as presented in Fig. 1, tag 2.

While from the very first sight this would seem like a straightforward and simple transformation, this particular example illustrates a situation where certain NLP processing is already required to acquire a correct result. The reason behind this is that the conditions defining the extraction of multi-word verb and/or noun phrases are non-trivial. In our case, the association between the two classes *Customer* and *Item* is named as the two-word phrase *return back*, which is extracted from the name of the source element, particularly *Use Case* element *Return back item*. Moreover, actual verb phrases are not limited to one or two words, like phrasal-prepositional verbs containing both particle and preposition (*come up with*) or even distributed in the whole phrase, e.g. when the particle is after the object (*associate the object with*), although the such cases are observed less frequently in the formal language used in modeling practice.

The above-mentioned examples are just sample cases where a straightforward text chunking is not sufficient and certain involvement of NLP technology is required to obtain correct transformation results. Further, we provide more examples which may require additional steps for linguistic preprocessing:

- Hierarchical relations created after one element is dragged onto another if text labels of these elements match some form of the semantic relationships (such as generalization, synonymy, hyponymy, hypernymy or holonymy)
- Entity deduplication when multiple entries have the same meaning but different expressions. In some cases they are not considered synonyms, for instance, acronym and abbreviation resolution does not result in synonymous entries but rather in duplicate representations
- Processing of more complex phrasal structures like conjunctions/disjunctions, or combinations of the above (e.g. *create invoice and send it to the manager*). This may also include mining of ternary associations or relationships, as well as identifying possible co-references
- Text normalization, such as having two sets of elements that differ only in syntactic structures. For instance, consider two sets of associated elements in the source model, *Actor Administrator* and *Use Case Monitors instance*, and *Actor Administrator* and *Use Case Monitor instance*. The only difference here lies in the present tense form of verb *monitors*, where normalization to infinitive form *monitor* would result in deduplication of output elements, and hence, more clarified and concise output model. While this is a very straightforward and less likely scenario, more sophisticated cases may involve disambiguation of acronyms, or detection of missing words as well as grammatical errors.

Furthermore, we list the main NLP fields which could be applicable in this context in Table 1, together with our insights on their further applications in improving the quality of model-to-model transformations. Most of them will not be considered in this research, yet, they are proposed as additional extension points for improving the final pipeline. Moreover, this table is also supplemented with core techniques used to solve these problems; it is clearly indicated that deep learning techniques are the most widely researched and applied to solve these problems. For more extensive reviews of the techniques, as well as more discussions on their weaknesses or future prospects, we refer to recent survey NLP papers such as [76], [77], [78], [79], and [80]. Additionally, their performance can be significantly boosted after applying transfer learning with pretrained language models, such as BERT [39], ELMO [41], RoBERTa [81], ELECTRA [82], XLNet (83), T5 [84] or Microsoft's DeBERTa [85]. Therefore, from the technological point of view, one would need to consider the integration of deep learning based techniques that require to satisfy certain technological constraints. This is the first work which tries to bridge these two fields by performing a thorough evaluation of the existing NLP implementations for processing short text labels, which is required in the context of model-to-model transformations.

TABLE 1. Applicability of NLP techniques in M2M transformations.

Task	Goal	Application in M2M transformations	Core techniques
Tokenization	Split a text into parts with regards to separators residing inside the tokens	Text preprocessing	Lexical co-occurrence patterns [21], regular expression-based rules [22], extensible custom tokenizers [23] or deep-learning driven implementations [24, 25]
Lemmatization & stemming	Obtain base and root word forms	Text preprocessing	Simple dictionary lookups, statistical classifiers [26] or neural techniques, such as bidirectional gated recurrent unit (Bi-GRU) network [27] or deep GRU-based encoder-decoder architecture [28]
Error correction	Detect and correct grammatical, typographical, or even semantic errors (like accidental use of homophones)	Text preprocessing	String distances like Levenshtein and Jaro-Winkler, phonetic Metaphone algorithm and its modifications [29], statistical machine translation [30] or neural machine translation principles, coupled with transformer architecture [31, 32]
Part of speech (POS) tagging	Assigns linguistic tags for each token in the sentence	Text tagging for extraction	Maximum entropy [33], loglinear cyclic dependency networks [34], conditional random fields (CRF) [35], convolutional networks [36] or bidirectional long-short term memory (Bi-LSTM) combined with CRF [24, 37].
Semantic analysis	Capture synonyms, homonyms, antonyms, and other semantic relations in the analyzed text	Extraction of hierarchical relations like generalizations (hypernyms) Augment entity taxonomies Entity deduplication (synonyms)	Lexical databases, such as WordNet [38], similarity measures applied on contextual representations, such as cosine distance [39, 40, 41]. Hypernym discovery enables the extraction of hierarchical relationships to form entity taxonomies and augment existing ontologies or vocabularies. Rule-based [42, 43, 44], vector space-based [45] neural [46] or hybrid [47] approaches are among the most prominent
Dependency parsing	Extract intra-sentence linguistic relationships between words	Advanced grammatical processing, such as resolution of conjunctive/disjunctive clauses	Deep learning-based techniques, e.g., Stanford's deep bi-affine parser based on Bi-LSTM [48], extraction from pre-trained language models [49]
Constituency tree generation	Phrase-level grammatical processing	Noun/verb phrase extraction	Context-free grammars [50], TreeCRF [51], neural graph-based learning [52, 53]
Acronym detection	Detecting whether the given word in the text is an acronym or abbreviation, finding its proper meaning depending on the context	Text normalization, deduplication, concept mapping	Alignment-based pattern-matching [54] and rule-based matching strategies [55], supervised machine learning [56], unsupervised [57], semi-supervised [3] and deep learning approaches, such as long-short term memory (LSTM) [58] or sequence-to-sequence architectures [59]
Acronym disambiguation	Finding the proper meaning of an acronym depending on the context	Text normalization, deduplication, concept mapping	Deep learning techniques, like LSTM [59], Bi-LSTM with BioELMo embeddings as inputs [60], neural topic attention models [61], convolutional network models [62]
Named entity recognition	Identify entity instances in the given text and labeling them using a predefined set of categories, such as *PERSON, LOCATION, TIME, ADDRESS, etc.*	Generate different type of elements, depending whether they represent particular instance of the specified class, or the whole class itself	Deep learning driven approaches [63, 64], their combinations with CRF [41, 65, 66], self-attention-based latent CRF [67]
Entity linking	Complements NER with disambiguation and linking to the correct entry in an existing knowledge base	Perform deduplication and enhance the quality of the output model	Deep learning techniques combined with with external knowledge bases, such as Wikipedia, DBPedia, or knowledge graphs [68, 69]
Relation classification	Assign proper relationship type between two given entities from the given set of choices	Refine (specialize) source model elements for the generation of more accurate subset of target model elements, extract synonymous forms (*same as* relation)	Various neural techniques [70]
Semantic role labeling	Determine *"who did what to whom"*, *"when"* and *"where"* [71] by assigning one of the predefined classes to the roles of entities	Refine (specialize) source model elements with focus to generate more accurate subset of target model elements	Deep learning techniques, such as bidiectional LSTM [72], self-attentional deep network [73], graph convolutional networks [74], neural transitions [75]

III. RELATION EXTRACTION-RELATED TEXT LABELING ANTI-PATTERNS

In this section, we enumerate a set of modeling and element naming issues, which make the automated processing of labels in graphical models rather intricate. While certain modeling best practices are generally considered in modeling [86], [87], actual real-world cases tend to contain various issues (such as linguistic or modeling ambiguities) making it very difficult to be dealt with using automated tools. Hence, if the processing of text labels created following

TABLE 2. Anti-patterns for naming activity-like model elements.

Anti-pattern	Examples	Rules for anti-pattern detection in labels
Multinary associations	• *Ask Charles to write down amount in member register*	Pattern-based analysis
Acronym/abbreviation explanation	• *Check Brisbane City Council (BCC) Decoration Guideline*	The presence of acronyms and/or abbreviations
Condition	• *Check if the article is available*	The presence of "if" or "is" followed by a participle
Conjunctive clause	• *Adds incoming mail date, records letter, archives it* • *Receive and review the request*	The presence of "and" or comma
Disjunctive clause	• *Communicates with teacher about adding or dropping* • *Pass or Reject Prototype Test*	The presence of "or"
Verb phrase spans over whole text	• *Mark the script as invalid and return to customer* here, the verb phrase is *mark as invalid)* • *Select an item to process*	Morphosyntactic analysis
Association-like naming	• *Client makes order in eshop* • *A new record is saved in patient's medical record*	Morphosyntactic analysis
Brackets	• *Monitor advertising outcome (impact, financial, new customers) and feelings* • *Enable participation form (consent declaration*	The presence of brackets
Additional comments	• *group travel - to be handled manually* • *booking completed – payment*	The presence of hyphens, dashes, colons, etc.
Prefixes	• *SAP ET 2000: Order spare parts* • *TL4: approve by signing*	Regular expressions, the presence of colons, etc.
Mathematical expressions	• *refuse request (request >20x income)* • *Accept (>12months)*	The presence of mathematical symbols
Invalid names	• *Available; Forward; Yes*	The absence of a verb
Grammatical errors	Missing letters, misspelled words, etc.	Grammatical error analysis

best modeling practices could be considered as a relatively uncomplicated task (assuming that the tagging bias of the underlying implementation is not considered), significant deviations might easily complicate it.

To identify the most common text labeling issues in graphical models, we used a large dataset provided by the BPM Academic Initiative (BPMAI) [88], which contained over 4100 real-world process models presented in BPMN notation. We excluded instances that did not meet certain requirements (e.g., all the elements in the models were named using single letters without any semantic meaning, or the text labels were not in English). Labels from the BPMN *Task* elements were extracted from the remaining models as one of the main objects of interest in our research. After analyzing the extracted labels, a set of naming anti-patterns for activity-like *Task* elements was formed (Table 2) together with examples and some heuristic rules for detecting these anti-patterns; in our opinion, the latter could be applied for the initial screening and filtering tasks in other types of graphical models as well.

The detection rules are not formal in any way but can be used as guidelines to identify the cases of anti-patterns. Also, the morphosyntactic analysis might have to be carried out to properly detect sophisticated cases of element naming anti-patterns in graphical models. Moreover, other elements representing subjects or entities (such as BPMN *Lane, Pool* elements) may contain invalid names as well, including multiple subjects, phrases, or some of the anti-patterns from Table 2.

It is worth noting that some of the observed naming cases indicated invalid modeling practices, for instance, naming activity elements as conditions or decision points (e.g., *Available*, *Yes*, *Check if available*). Naming activities as whole triplets `<actor-relationship-activity>` is yet another quite common bad modeling practice used in modeling processes. The latter should be transformed into a combination of a BPMN *Lane* or *Pool* element with an activity-like element in it. One may also identify cases that combine multiple anti-patterns, for instance, the name of an activity may contain both conjunctive/disjunctive clauses relating multiple verb phrases into one text rumbling (e.g., *Mark the invoice as invalid and return to customer*), which increases the complexity of NLP tasks to a whole new level. Even though resolving conjunctive/disjunctive clauses is a challenging task, it can still be processed by using dependency parsing-based extraction, which is further addressed in Section V.

IV. PHRASE EXTRACTION EXPERIMENT

In this section, we evaluate the capabilities of the existing NLP tools to properly extract noun/verb phrases from the given text labels. This task is closely related to the relation extraction task, given its goal to extract tuples (verb phrase, noun phrase) from the given chunk of text that can further be used to construct semantic associative relations after combining with semantics from the source models (e.g., associative relationships between UML *Use Case* and *Actor* elements). Moreover, this task is important for successful model-to-model transformations because the extracted tuples are used to generate sets of elements for various target models or augment the existing models with additional elements.

Further, we present basic aspects of our experimentation on extracting noun/verb phrases from the text labels extracted from the real-world dataset which contains BPMN process models and UML use case models; both types of these models contain activity-like elements which are subjects for specific processing. Section IV-A describes the preliminaries and setup of the experiment, while Section IV-B presents the evaluation methodology; in its turn, Section IV-C elaborates on the main findings in this experiment.

A. EXPERIMENT SETUP

Information extraction (and more specifically, relation extraction) is widely supported by multiple commercial and academic engineering efforts that provided multiple options for selecting the initial starting point for our research. While new techniques emerge frequently, they are based on the generally-available text corpora that do not provide the flexibility and specificity required to fulfill our goals. More specifically, our initial testing of such tools helped us to recognize the possibility of confusion in verb/noun recognition if the infinitive verb form is used – this is not handled correctly by generic POS tagger tools. On the other hand, the development of specialized datasets is usually challenging and time-demanding.

Therefore, given the lack of specialized resources required for successful implementation, we chose to adopt and test existing tools by complementing them with additional extraction functionality and applying certain enhancements to the existing ones. Moreover, some of these toolkits provide implementations for wide array of related problems, such as such as tokenization, POS tagging, lemmatization, syntactic analysis, dependency parsing, co-reference resolution, or relation extraction, which may significantly enhance required pipelines. Additionally, some libraries provide other interesting tools, for instance, Stanford CoreNLP [89] also provides natural logic annotator that enables quantifier detection and annotation, as well as CRF-based true case recognition, which is also important for knowledge base acquisition and normalization and relates to the problems addressed in this work; while quantifier detection is not among such issues, it can be tested and integrated into the future pipelines as well.

Further, we list the set of implementations selected for our experimental implementation and evaluation[1]:

- Stanford CoreNLP toolkit [89] which relies on conditional random field (CRF) implementations for performing both part-of-speech tagging and NER-related tasks.
- Spacy [23] framework, which applies convolutional neural networks.
- Stanford Stanza [90] which uses Bi-LSTM to implement components and pipelines for multiple NLP tasks such as tokenization, lemmatization, POS tagging, and dependency/constituency parsing.
- Flair [24] toolkit by Zalando Research, which applies pooled contextualized embeddings together with deep recurrent neural networks, as well as provides its pretrained language models.
- AllenNLP [25] which relies on deeply contextualized ELMo embeddings based on combined character-level CNN and Bi-LSTM architecture.
- BERT [39] is one of the most dominant techniques in NLP at the moment of writing this paper, based on transformer architecture and masked language modeling.
- ELECTRA [82] which is an improvement over BERT that applies token replacements with plausible alternatives sampled from a generative network during model training, instead of using masked tokens. The main goal of the model is to predict whether the corrupted input was replaced with a generator sample. ELECTRA authors show that this task is more efficient than BERT and the final model is capable of substantially outperforming BERT model in terms of model size, amount of computing and scalability [82].

The fact that these tools use different machine learning or deep learning approaches to solve NLP tasks has also motivated us to test their performance in the context of our approach. In this work, we use the BERT[2] and ELECTRA[3] implementations from the Hugging Face repository, which are already fine-tuned for part-of-speech tagging tasks.

Additionally, we developed our own taggers that were biased towards the recognition of conflicting verb forms by performing augmentations of the original text inputs with their copies containing infinitive verb forms as replacements for the original ones; a similar approach was successfully applied in our previous work to improve performance for base CRF-based tagger [6]. OntoNotes corpus [91] was used as the base data source due to its resemblance to the communication cases observed in graphical process and system models. For the reference implementation, we selected Bi-LSTM-CRF architecture [37] which has been proven to be the best performing one at the time of writing. It consists of a single input embeddings layer, a bidirectional LSTM hidden layer to process both past and future features, and CRF layer at the output, which helps to improve tagging accuracy by learning and applying constraints over sentence level to simultaneously optimize the labeling output and ensure its validity. For our experimental purposes, we implemented two versions of our customized taggers:

- BERT-BiLSTM-CRF that uses original pretrained BERT embeddings at the input layer,
- ELMo-BiLSTM-CRF that relies on ELMo embeddings at the input layer.

For training these models, CRF (also known as Viterbi) loss, based on the maximization of the conditional probability, was used; for more details on its derivation, we refer

[1]The final datasets, experimental code and results are available at `https://github.com/paudan/m2m-nlp-experiment`

[2]`https://huggingface.co/vblagoje/bert-english-uncased-finetuned-pos`

[3]`https://huggingface.co/danielvasic/en_acnl_electra_pipeline`

```
NP :   <ADV|ADJ>*<NOUN>+<PART>?<NUM>?)+(<ADP>*<DET>?<ADV|ADJ>*<NOUN|PROPN>+<PART>?<NUM>?)*
VP :   <VERB>+<ADP>?
PNP :  <PROPN>+
```

Listing 1. Formal grammar for extraction used with Spacy, Flair and Stanza.

```
ADP :  <RB|RBR|RP|TO|IN|PREP>
ANP :  <JJ|ADJ>*<NN|VBG|RBS|FW|NNS>+<POS>?<CD>?
NP :   (<ANP>)+({ADP}?<DT>?<ANP>)*
VP :   <VB*>+<ADP>?
PNP :  <NNP|NNPS>+
```

Listing 2. Formal grammar for extraction used with CoreNLP and ELECTRA implementations.

to [37] and [92]. Moreover, learning rate was set to 0.1, the hidden layer size was set to 128, and the early stopping parameter for termination, if no convergence is further observed, was set to 10. SimpleNLG library [93] was used to normalize tense for verb phrases, while NLTK [22] toolkit was used to implement text chunking with POS tags obtained as an output from the above-mentioned tools.

Listing 1 represents formal grammar, based on regular expressions (regex) over part-of-speech tags, which was used for noun/verb phrase extraction. It relies upon the Universal Dependencies scheme [94], which is used by Spacy, Flair, and Stanza tools. Here, *NP* defines a noun phrase, *VP* defines a verb phrase, and *PNP* defines a proper noun phrase. As Stanford Core NLP and ELECTRA pretrained implementations use Penn Treebank notation for its POS tagger output, the grammar is adjusted for their cases (Listing 2); here, additionally, *ADP* defines an adposition, and *ANP* – a partial noun phrase, which is further used as a block in *NP* extraction.

The datasets used during experimenting were obtained after pre-processing a relatively large number of BPMN process and UML use case models, obtained from various sources. The final experimentation set of such models consisted of:

- 32 BPMN process models and 25 UML models that were collected freely from the Internet;
- A large sample of preprocessed and cleansed BPMN process models, which were selected from a large set of Signavio BPMN models provided by BPM Academic Initiative [88].

The acquired final set of models was processed, and the names of *Task* elements (for BPMN process models) and *Use Case* elements (for UML use case models) were extracted for experimentation. It was expected that *Task* and *Use Case* elements would contain at least one verb or verb phrase, and one noun or noun phrase. The extracted elements were cleaned from semantic inconsistencies, grammatical errors, invalid names, and common modeling errors, as well as filtered to exclude invalid practices listed in Table 2. In this stage, we also excluded entries containing multiple verb phrases in their names (e.g., conjunctive/disjunctive clauses), as the recognition of such structures was not a part of this experiment (this is later addressed in Section V). However, having a single verb phrase with multiple noun phrases in conjunctive or disjunctive form could be considered processable and would result in multiple valid tuples of target transformation outputs.

After performing the aforementioned steps, we obtained a dataset of 4044 valid entries that were then used to manually extract verb phrase and noun phrase pairs. The whole extraction procedure was performed by the authors of this paper. These pairs were set as a ''golden standard'' to validate the outputs acquired from the automatic extraction using selected extractors. Hence, the final dataset included 328 instances having no verb phrases, and 3716 instances containing both verb and noun phrases.

B. EVALUATION METHODOLOGY

The developed extractors were evaluated in terms of accuracy, precision, recall, and F-measure, which measured the ability to match the acquired outputs to the ''golden standard'' outputs. In our experiment, two different aspects were taken into consideration:

- Whether the extractor successfully determined that the phrase contained one or more noun/verb phrase that must have been extracted. In case there is no particular phrase found, the output would be empty.
- Whether the extractor successfully extracted the required verb phrases or noun phrases. Note, that it was required to evaluate if both verb phrases and noun phrases were successfully extracted. In cases, where multiple phrases were marked as an output, it was considered that strictly all of them had to be present in the output for it to be marked as correct.

Extraction accuracy is defined as the ratio of correctly extracted verb/noun phrase instances (together with empty outputs when such instances were absent) to a total number of entries:

$$accuracy = \frac{number\ of\ correctly\ extracted\ instances}{number\ of\ total\ instances} \tag{1}$$

Precision is defined as the ratio of correctly extracted concepts to the number of total extracted concepts, whereas recall is a ratio of correctly extracted concepts to the number of correct concepts:

$$precision = \frac{concepts\ correctly\ identified}{concepts\ identified\ total} \tag{2}$$

$$recall = \frac{concepts\ correctly\ identified}{gold\ standard\ concepts} \tag{3}$$

TABLE 3. Results of the extraction of noun/verb phrases from the names of activity-like elements.

Extractor	CoreNLP [89]	Flair [24]	Spacy [23]	Stanza [90]	BERT [39]	ELMO AllenNLP [25]	ELECTRA [82]	ELMO-BiLSTM-CRF	BERT-BiLSTM-CRF
Detection if a source text contained a verb phrase									
Accuracy	0.728	0.817	0.816	0.82	0.857	0.686	0.743	0.818	**0.860**
F1-Score	0.827	0.89	0.89	0.892	0.916	0.797	0.838	0.891	**0.918**
Detection if a source text contained a noun phrase									
Accuracy	0.844	0.988	0.983	**0.99**	0.988	0.98	0.935	0.921	0.961
F1-Score	0.914	0.994	0.991	**0.995**	0.994	0.99	0.966	0.958	0.980
Proper extraction of noun phrases in cases containing only noun phrases									
Precision	0.67	0.883	0.848	**0.918**	0.875	0.818	0.741	0.753	0.817
Recall	0.591	0.878	0.832	**0.915**	0.872	0.79	0.640	0.698	0.777
F1-Score	0.628	0.88	0.84	**0.917**	0.873	0.804	0.687	0.724	0.797
Proper extraction of both noun phrases and verb phrases in cases containing both									
Noun phrases									
Precision	0.645	0.736	0.707	0.745	**0.768**	0.569	0.727	0.694	0.750
Recall	0.551	0.733	0.703	0.745	**0.766**	0.568	0.691	0.646	0.727
F1-Score	0.594	0.735	0.705	0.745	**0.767**	0.568	0.709	0.669	0.738
Verb phrases									
Precision	0.594	0.639	0.621	0.648	0.693	0.5	0.630	0.655	**0.718**
Recall	0.587	0.639	0.617	0.649	0.692	0.5	0.608	0.648	**0.707**
F1-Score	0.591	0.639	0.619	0.648	0.692	0.5	0.619	0.651	**0.713**

F1-measure (also referred to as F1-score) is defined as a harmonic mean of these two measures:

$$F_\beta = \frac{(1+\beta^2) \times (precision \times recall)}{precision + recall}, \beta = 1 \qquad (4)$$

C. EXPERIMENT RESULTS

The results of the experimental extraction of verb phrases and noun phrases from the names of activity-like elements are presented in Table 3. It depicts both results of detecting whether the given entry had particular types of phrases, as well as the performance of extracting these phrases from the respective entries.

The obtained results indicate that the extractor based on the RNN-based Stanza tagger outperformed CNN-based and CRF-based tools (Spacy and CoreNLP respectively) in solving our problem. Extraction using Stanza's Bi-LSTM-based tagger showed the best performance in 2 tasks, while Flair tagger use resulted in the second-best. Extractor based on our custom BERT-BiLSTM-CRF tagger outperformed other implementations while detecting verb phrase presence and verb phrase extraction. Moreover, both custom taggers also showed improvements over their generic versions, i.e., ELMo-BiLSTM-CRF resulted in a better performance than the original AllenNLP ELMo, and BERT-BiLSTM-CRF proved to be better performing compared to the BERT-based POS tagger. This is quite optimistic, considering the size and specificity of the dataset. However, some caution should be taken while interpreting these results, given that our custom-trained tagger was biased towards the identification of infinitive forms of conflicting verbs. This implies that in some other cases it could fail to correctly tag other words that were handled correctly by the tagger trained using conventional corpora, and was initially confirmed in our previous research applying similar principles to train custom POS taggers [6]. Therefore, more attention should be given to improving and tuning custom taggers applied in the research, as well as finding an optimal balance between an increase in performance for verb detection and a possible decrease in other tasks that are performed better using generic POS taggers.

Nevertheless, the results of the leading extractor (based on the Stanford Stanza toolkit) are quite encouraging – the achieved F1-Score was more than 0.8 in most of the performed evaluation tasks, especially given the limitations and the level of unavoidable ambiguity in the testing dataset. One of the main challenges in this particular case is the fact that corpora currently available for training, like OntoNotes [91] or English Web Treebank [95], are better accustomed to working with whole documents rather than the analysis of short text and, therefore, do not represent the specificity addressed in this paper. We tried to mitigate this issue with additional augmentations of the input text, which resulted in certain performance improvements; developing text corpora, which are better adjusted for this specific task, would certainly help to improve its performance even further.

V. PARSING CONJUNCTIVE/DISJUNCTIVE STATEMENTS

The techniques described in Section IV proved their efficiency during the extraction of verb phrases and noun phrases, the tools we experimented with in the phrase extraction task are not capable of processing more complex examples discussed in Section III when applied directly – here, the conjunctive/disjunctive statements are a good example of that. The complexity can be illustrated with the following examples which depict multiple cases of conjunctive statements (disjunctive statements may be formulated almost identically):

- *check dates and suggest modifications* – the statement includes the conjunction "and".
- *consult project, check progress* – the statement does not include direct conjunction, but it is inferred.
- *receive invoice, packing slip, and shipment from supplier* – multiple nominal subjects are related to the single verb *receive*.
- *calculate and send price offer* – contains a single nominal subject that has dependencies on multiple verbs.

Obviously, the presented examples are not the most sophisticated text labels one could find in real-world models. This is not surprising due to the well-known fact that natural language is one of the most complex objects there is for automated machine processing. It is worth noting that the topic of processing conjunctive/disjunctive statements is not widely researched, although it has received some attention from researchers working on sentence simplification [96] or detecting boundaries of the whole conjunction span [97]. Also, many works on sentence simplification rely upon parse trees [15], [98], [99], which is in line with our research.

In Section V-A, we provide an algorithm based on dependency parsing, which is used to extract pairs of noun/verb phrases from conjunctive/disjunctive statements. Section V-B describes an experimental setup using a real-world dataset consisting of conjunctive/disjunctive phrases that are then processed using the proposed solution. Finally, Section V-C provides the evaluation results, a discussion, and some ideas for our future research.

A. ALGORITHM FOR EXTRACTING NOUN/VERB PHRASES FROM CONJUNCTIVE/DISJUNCTIVE PHRASES

Further, we briefly describe a dependency parsing-based algorithm for extracting pairs of noun phrases and verb phrases from conjunctive/disjunctive phrases (see Algorithm 1). The input is the parsed and tagged document D; hence, it requires a part-of-speech tagger and a dependency parser as part of its processing pipeline. We define D_{TOK} as the set of tokens that constitute document D, together with the parsing and tagging output. Further, this document is also processed to create noun phrase spans (further denoted as SNP) and verb phrase spans (denoted as SVP) by using predefined grammars (such as presented in Section IV-A). Later, we use correspondence indexes Ind_{NP} and Ind_{VP} to map each token in the document to a corresponding noun phrase or a verb phrase. These indexes enable traversing dependency relationships at a phrase level and at the same time reduce the ambiguity that is observed after using different dependency parsers. We denote the head of the dependency relationship from the token tok as $Dep_{Head}(tok)$, and the end as $Dep_{End}(tok)$. Finally, we denote GET operation as the operation, which enables retrieving an entry from the index, given its indexing value. The syntactic dependencies are expected to be labeled using Universal Dependencies format [94], particularly *DOBJ* as the dependent object, *OBJ* as the object, *POBJ* as the object of the preposition, *CONJ* as the conjunction.

The output of this algorithm is a collection of tuples of verb phrases and noun phrases. It is expected that the input contains both nouns and verbs, otherwise, tuples with empty values instead of the verb or noun phrases can be returned as a result.

B. EXPERIMENT SETUP

To evaluate our approach, we extract a dataset of 410 entries acquired from the same set of process models which was used in our phrase extraction experiment. The final dataset comprised only those text labels that included at least one conjunctive or disjunctive clause. Then we manually extracted all available verb/noun phrase parts to create a "gold standard" dataset to be used as a reference point for our evaluation.

The algorithm presented in Section V-A was implemented as a separate module without any text normalization capability. To perform comparative testing, we implemented the module in Python, using Spacy, and extended it to use Stanford Stanza, due to its flexible integration with the Spacy framework, to enable comparing the performance of dependency parsing capabilities of these toolkits.

Again, for the evaluation, we used metrics like the ones described in Section IV-B, that is, accuracy, precision, recall, and F1-Score. Here, accuracy is defined as the ratio of the entries processed correctly and the total number of entries. Note, that this is a very strict measure as it considers a valid extraction only if all noun/verb phrase pairs were extracted correctly. However, this technique is capable to generate a larger or smaller number of entries compared to the actual outputs. To address this issue and provide an evaluation of partially correct outputs, we defined two additional metrics to evaluate the performance in terms of the number of generated output instances:

- The mean deviation between the number of extracted outputs and benchmark output results:

$$MeanDiff = \frac{1}{n}\sum_{i=0}^{n}\frac{|\#^i_{actual} - \#^i_{extracted}|}{\#^i_{actual}} \tag{5}$$

- The mean Sørensen–Dice coefficient, which is used to evaluate the average similarity between the actual and extracted instance sets:

$$MeanSDC = \frac{1}{n}\sum_{i=0}^{n}\frac{2 \times |O^i_{actual} \cap O^i_{extracted}|}{|O^i_{actual} \cup O^i_{extracted}|} \tag{6}$$

Here, n is the total number of processed entries in the dataset; O^i_{actual} is the benchmark set of verb phrase/noun

Algorithm 1 Processing of Conjunctive/Disjunctive Statements

```
Require: parsed sentence $D_{TOK}$, mapping indexes $Ind_{NP}$ and $Ind_{VP}$
 1: $results \leftarrow 0$
 2: for all $tok \in D_{TOK}$ do
 3:     if $DEP_{END}(tok) \in (DOBJ, OBJ, POBJ)$ then
 4:         $results \leftarrow results \cup (\textsc{Get}(Ind_{VP}, Dep_{START}(tok)), \textsc{Get}(Ind_{NP}, Dep_{END}(tok)))$
 5:     else if $DEP_{END}(tok) = CONJ$ then
 6:         $Ind_{POS} \leftarrow$ index of POS tags and tokens for conjuncts in $DEP_{END}(tok)$
                                        ▷ Assume pattern <VERB>, <VERB> and <VERB> <NOUN>
 7:         if $|\textsc{Get}(Ind_{POS}, NOUN)| = 1$ and $|\textsc{Get}(Ind_{POS}, VERB)| > 1$ then
 8:             noun $\leftarrow \text{Get}(Ind_{POS}, NOUN)$
 9:             for all $verb \in \textsc{Get}(Ind_{POS}, VERB)$ do
10:                 $results \leftarrow results \cup (\textsc{Get}(Ind_{VP}, verb), \textsc{Get}(Ind_{NP}, noun))$
                                        ▷ Assume pattern <NOUN> <VERB>, <VERB> and <VERB>
11:         else if $||\textsc{Get}(Ind_{POS}, VERB)|| = 1$ then
12:             verb $\leftarrow \textsc{Get}(Ind_{POS}, VERB)$
13:             for all $noun \in \textsc{Get}(Ind_{POS}, NOUN)$ do
14:                 $results \leftarrow results \cup (\textsc{Get}(Ind_{VP}, verb), \textsc{Get}(Ind_{NP}, noun))$
15:         else if $||\textsc{Get}(Ind_{POS}, NOUN)|| > 1$ then
16:             for all $noun \in \textsc{Get}(Ind_{POS}, NOUN)$ do
17:                 $results \leftarrow results \cup (\textsc{Get}(Ind_{VP}, \textsc{LeftmostVerb}noun), \textsc{Get}(Ind_{NP}, noun))$
Output: the set of $(verb\ phrase, noun\ phrase)$ tuples $results$
```

phrase pairs extracted for the *i*-th dataset entry; $O^i_{extracted}$ is the set of output elements extracted for the *i*-th dataset entry; $\#^i_{actual}$ and $\#^i_{extracted}$ represent the number of elements in O^i_{actual} and $O^i_{extracted}$, respectively.

C. EXPERIMENT RESULTS

The results of the experiment are presented in Table 4. They summarize the performance of both Spacy and Stanza models. The obtained results prove the influence of the underlying dependency parser. Here, the implementation based on the Stanza toolkit significantly outperformed the Spacy-based implementation. Unfortunately, the extraction accuracy score for both implementations was very low proving that those implementations failed to extract all the expected verb/noun phrase pairs from each given input text; this is also reflected in relatively high values of *MeanDiff* and *MeanSDC*. Moreover, precision, recall, and F1-Score scores, which are calculated on a macro-level, show that results at the macro level are not disappointing, yet, both implementations of the algorithm and the underlying technology could still be improved in the future.

Here, the performance of experimental implementation resulted in F-Score = 0.631, although we must also take into consideration the influence of a sample bias. The significance of the underlying parse model was also obvious, as the Stanza-based processor significantly outperformed the implementation based on Spacy. Again, all the mandatory pipeline steps – text tagging, text chunking into noun/verb phrases, and dependency parsing – have proven to be crucial to the overall quality of phrase processing. A failure in any of these steps inevitably translates into errors in the further steps of the developed pipeline. Therefore, we safely conclude that the dependency parser plays the most important role of all. This was extremely well visible in the experimental cases when insignificant changes to the input entry (e.g., adding an adjective to one of the nouns) resulted in completely different parse trees compared to the initial ones; it complicated the analysis significantly or even resulted in cases not covered by the used formal grammar. This indicates the need for more extensive research and improvements in both extraction and dependency parsing areas. We believe that it could be achieved by integrating and testing recent developments in dependency parsing, based on neural techniques as described in [51] and [52] among the others.

TABLE 4. Performance of processing the conjunction/disjunction statements.

Model	Spacy	Stanza
Accuracy	0.124	0.195
Precision	0.632	0.748
Recall	0.413	0.531
F1-Score	0.5	0.621
MeanDiff	0.589	0.477
MeanSDC	0.449	0.554

VI. ACRONYM/ABBREVIATION DETECTION

Acronym/abbreviation detection is an issue in text normalization which deals with multiple issues and ambiguities while detecting whether the given word in the text is an abbreviation or an acronym. While many cases can be handled by simply performing a search for a particular candidate's expansive form in the text or performing a search in dictionaries and word lists, this is not trivial when it comes to widely used acronyms. The first issue is that these acronyms/abbreviations might be present in dictionaries and at the same time overlap with some general words (e.g., acronym *IT* overlaps with pronoun *it*); another common issue is omitting the expanded form of an acronym/abbreviation due to its widespread use, which makes it almost impossible to automatically identify it as an acronym/abbreviation of some particular phrase with simple backtracking in the input (the aforementioned acronym *IT* can be seen as an example in this context as well). *Acronym/abbreviation expansion* is yet another similar task aiming to solve the problem when a given abbreviation or acronym should be replaced by its expansive form, which is the most appropriate in the given context. This task is not a trivial one either - for instance, *EM* could be referred to as *entity matching*; however, it could also be *expectation maximization* or *entity model*, with all these expansive forms coming from a single computer science domain. Unfortunately, current research tends to focus on long text passages, which highly reduces their applicability in the context of our research.

In model-to-model transformation, as well as in other relevant topics, the acronym/abbreviation (A/A) detection task helps one to properly match full concept names with their abbreviated forms, thus adding to greater consistency of the models being developed. The A/A detection task itself comprises two interrelated subtasks:

- *PA/A detection* seeks to detect candidate A/A, which must be expanded (*what must be replaced?*);
- *A/A expansion* is focused on finding the right expansion for the given A/A (*what is the replacement?*).

Acronym/abbreviation (A/A) expansion is often considered as a simple expansion of entries that are identified as A/A due to their writing style or absence in relevant sources, like thesauri or dictionaries. While simple A/A mapping lists are generally applied for common text normalization tasks, they may not always provide the correct result, unless they are restricted to having single meanings in specific or even multiple contexts. Therefore, real-world use cases may easily complicate his seemingly uncomplicated task. The complexity of the task may rise depending on the diversity of corpus or data required to properly train one's implementation to resolve models. The expansion problem will not be further addressed in this paper due to certain limitations of the dataset.

While recent developments in acronym detection tend to apply state-of-the-art deep learning techniques (as stated in Table 1), they are not applicable in our context due to relatively short text input. Therefore, we will model this problem in a more traditional yet efficient way by applying context-based classification techniques within a space of contextual, morphological, and linguistic features. While a similar approach was successfully tested in [56] and [100], we propose using a different set of features that are preferred due to data limitations. The target variable of the classifier is simply an indicator of whether the particular word represents an acronym or abbreviation.

Further in this section, we provide an empirical evaluation of A/A detection in BPMN element names. To make it more consistent with other experiments presented in this paper, we will use the same initial set of the BPMN process and UML use case models as in the experiment presented in Section IV. Hence, Section VI-A describes the preliminaries and setup of the experiment, while Section VI-B presents and briefly discussed the results obtained during that experiment.

A. EXPERIMENT SETUP

The initial dataset of process models was used as the source for developing the feature dataset for our A/A detection experiment. The feature dataset was created from all the available words in the extracted text by applying simple heuristic rules:

- Acronym or abbreviation must contain at most 5 characters. It can be observed that the longer the word is, the smaller is the probability of it being an acronym. Therefore, words with more than the predefined number of characters are not considered to be acronyms and are excluded from further analysis.
- The word representing an acronym or abbreviation is not available in the dictionary. Since WordNet does not contain all the English words and their forms, we used Enchant[4] library, which is generally used for grammatical error correction, to check for the word existence.

The first rule helped to identify the candidate entries for the feature dataset, and the entries longer than the predefined length threshold were not considered as candidates for acronyms and abbreviations. The second rule helped to perform its primary labeling. After the automated generation of the dataset, some manual adjustments were performed fixing automated labeling errors and ambiguities, removing redundant and duplicate entries, as well as identifying situations that were not covered by the above-listed heuristics and could not be handled automatically – all this was done to make the feature dataset more consistent and suitable for the development of our detection classifier. The feature dataset examination also helped to identify that most of the acronyms were written in uppercase, which also helped to simplify the semi-automated labeling task. To avoid feature leakage, we removed the feature of the uppercase word as it would serve as a proxy for the label otherwise (in practical applications, it might serve as a very strong indicator for acronym presence). To perform POS tagging required for the POS-based feature generation, we used Stanford Stanza tagger that showed the best performance in our previous experiment presented in Section IV-C.

After performing the feature generation procedure, a feature dataset with a total of 16579 entries was created. Each entry in the dataset was a vector of 16 features extracted

[4]`https://github.com/pyenchant/pyenchant`

TABLE 5. Features used for the A/A detection.

Feature	Description
has.vowels	The word has any vowels
has.special	The word has characters like "&" (ampersand) or "."
just.letters	The word has only lowercase/uppercase letter characters
english.word	The word exists in English dictionary or other referential sources
long.char.seq	The word has at least one of sequences of 3 consecutive identical characters
starts.with.two	The word start with 2 consecutive identical characters
Pos	Part-of-speech (POS) of the word
prev.pos	POS of the word, which is before the current word
prev.pos.2	POS of the word, which is before the previous word
next.pos	POS of the word, which is after the current word
next.pos.2	POS of the word, which is after the next word
dep	Dependency parse label, associated with the word
prev.dep	Dependency parse label of the word, which is before the current word
prev.dep.2	Dependency parse label of the word, which is before the previous word
next.dep	Dependency parse label of the word, which is after the current word
next.dep.2	Dependency parse label of the word, which is after the next word

from the text labels in the BPMN process and UML use case models, together with the label indicating whether a word represents an acronym or an abbreviation. The full set of features is presented in Table 5. The features *has.special* and *long.char.seq* were excluded from further analysis as the final dataset did not contain any such entries. Nonetheless, these features could be useful while performing further research with more extensive datasets and/or contexts, and thus they are included in Table 5 along with other features as a reference for future consideration. This left us with 14 features that were further used as the inputs for the classifier.

For the development of acronym detection classifier, the following techniques were considered:

- CatBoost [101] is a high-performing gradient boosting classifier. One of its most exceptional features is the ability to efficiently work directly with the categorical feature variables, which helps to improve performance when numerous categorical features are used.
- XGBoost [102] is one of the best performing gradient boosting-based ensemble classifiers, widely used to solve various classification tasks.
- Random Forest [103], [104] is a widely used decision tree ensemble technique based on bagging and random feature selection.

To handle the high level of class distribution imbalance of the input dataset, weighted classification was applied to improve detection performance. Also, grid search was used to optimize the performance of CatBoost and XGBoost by selecting their optimal hyperparameters. Random Forest classifier was run with default parameters, but using 200 estimators. All the classifiers were implemented in Python using *scikit-learn*, *catboost* and *xgboost* libraries. Similar to the experiments presented in Section IV-B, for performance measuring, the measures of accuracy, precision, recall, and F1-score were used.

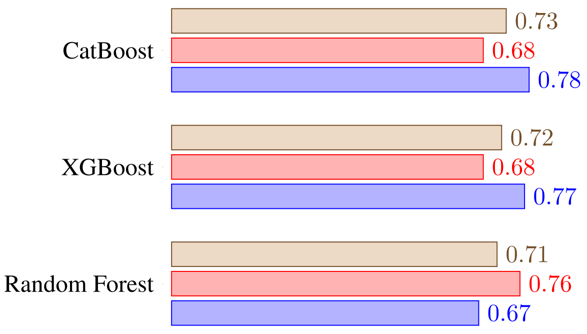

FIGURE 2. A/A detection performance and feature importance.

TABLE 6. Feature importance computed with CatBoost.

Feature	Importance
pos	13.86
next.dep	13.50
next.pos	11.55
english.word	9.91
prev.dep	8.91
prev.pos	8.67
next.dep2	8.60
next.pos2	8.47
prev.dep2	7.45
prev.pos2	5.29
has.vowels	3.65
starts.with.two	0.07
just.letters	0.04

B. EXPERIMENT RESULTS

Figure 1 presents the results obtained using the classifiers described in Section VI-A. They show that CatBoost significantly outperformed Random Forest and slightly - XGBoost classifiers in terms of precision and F1-Score. This is not surprising, due to the design of the CatBoost tool and its ability to work directly with categorical variables. Its superiority over the XGBoost classifier was also confirmed by the McNemar's test that resulted in $p < 0.05$ ($p = 0.029$).

Table 6 also provides an insight into the feature importance obtained using CatBoost classifier. The results indicate that morphological features of tokens next to the target word were identified as the most important, whereas the presence of a particular word in an English dictionary or similar referential source played a less influential role as expected. One of the reasons for this is the fact that usually abbreviations are created by the people, who create models and write documentation (e.g., business/system analysts). And so, those people create various acronyms and abbreviations by themselves, or they use already established A/A to make the text more compact (compact text labels are particularly relevant in visual modeling). Contextual part-of-speech features seem to play an important role as well because they capture acronym usage patterns in spoken or written language; this is also proved by the high importance of the features of preceding tokens, as well as more distant contextual features. This prompts for testing wider context features (like *prev.pos*3, *next.pos*4, etc.); however, such features are not considered in this paper due to the limited size of the processed text phrases.

Alternatively, one might consider sequence-tagging models (such as Markov models or recurrent neural networks) that directly apply such context, yet their training would require larger datasets and the inclusion of an even greater number of additional features (lexical and morphological). Emerging deep learning approaches, such as [58] or similar, seem to be a viable solution as well, although their training might require a significant amount of labeled data, and their applicability for the given problem must be verified.

VII. DISCUSSION

With the experiments described in this paper, we explored the capabilities of the advanced NLP tools to process short text fragments (text labels) which are required to enable advanced capabilities in processing our model-to-model transformations. While this is inspired by our previous research [6], [7], we believe that the presented research results could be applicable in other relevant fields as well. Similar text normalization is required for practical process mining where the names of the composing elements need to be unified from multiple data sources while reducing the number of duplicates to a minimum. It is also applicable in conversational intelligence when intent processing is required to identify the responsive action for the inquiry. The experiments prove that the recent developments in the field of NLP and deep learning could provide the needed tools to solve such and other similar problems.

Overall, the experiments presented in this paper revealed several issues, which should be addressed and might be required to handle separately:

- Bad modeling (in particular, element naming) practices were not considered in the extraction activities. During the initial dataset screening, we observed many such cases that were summarized in Table 2. Detecting the most common bad modeling practices and introducing an automated resolution of such cases into the developed solution could provide even greater automated processing results.
- A more thorough analysis of the outputs showed that some tagging tools, like Spacy, were quite sensitive to the letter casing, which is also significant for the practical application of NLP technology in model-to-model transformations as well as in other relevant fields. While this is less relevant when processing long text passages or whole documents, the importance increases when more specific text processing is considered. This is stipulated by different modeling styles used by practitioner modelers who prefer starting each word with a capital letter while naming model elements such as activities, tasks, use cases, etc. (this is verified by the analysis of the BPMAI dataset used in our research, as well as our personal experience), and some tools may fail to tag such labels correctly. For example, *return invoice* could be tagged as `<VERB><NOUN>`; however, *Return Invoice* might as well become `<NOUN><NOUN>`, which would be an incorrect tagging result. Again, in our related experiments, we reverted all text labels to lowercase to mitigate this problem. Unfortunately, such normalization might remove relevant features that could be used to detect abbreviations.
- The previous issue is also relevant for other related problems. While such cases could be normalized to lowercase, doing so increases the risk of failure in the other tasks like named entity recognition where capital letters play a crucial role. Moreover, NLP tools may face difficulties detecting named entities within fully lowercase entries (e.g., *United States* was identified as LOCATION, while *united states* was not).
- Detection performance can be negatively affected by the presence of non-alphanumeric symbols (e.g., dashes, commas, apostrophes) within words. It is advisable to remove such symbols from the model element names wherever possible. This issue might be mitigated using more advanced tokenizers capable of handling most of these cases, but the risk of failing to properly handle them still exists.
- Generally, using conjunctive/disjunctive clauses in activity-like element names indicates a bad modeling practice as such instances should be refactored to two or more atomic elements. As stated previously, processing such statements appeared to be a very challenging task requiring the support of several advanced NLP techniques, such as dependency or constituency parsing. In its turn, this would bring in other kinds of errors from the underlying parser model.
- In our experimentation, we observed general ambiguity in detecting abbreviations. The A/A detection experiment confirmed the applicability of a machine learning-based approach to handling this problem. Yet, A/A expansion is a more complicated task as full forms of concepts designated by A/A might not be present in models under the scope, especially if those A/A are well-known and heavily used (e.g., *IT*, *USA*). External sources, such as domain vocabularies and linked data can be applied by matching them contextually to each model instance containing cases of acronyms and abbreviations. Again, this requires additional sources of input data, together with a more extensive dataset, and could be considered as one of the directions for our future research.

VIII. CONCLUSION

NLP discipline has seen impressive advancements and improvements during the last several years, with the number of NLP applications increasing dramatically. Also, the progress in deep learning has resulted in a significant increase in the performance of solving different linguistic tasks. In this paper, research on applying the recent developments for processing small text phrases is discussed. While the need for this research originated from our recent research on model-to-model transformations [6], [7], we may identify several other areas that could benefit from similar text processing capabilities, such as process mining, aspect-based sentiment analysis or conversational interfaces with command-like short text processing capability. At the same time, all these areas share the same NLP-related issues that have to be dealt with to ensure satisfactory performance of the underlying NLP technology (e.g., identical representation of verbs and nouns, lack of context required for the automated processing).

In this paper, we addressed the problem of extracting relation tuples from the process and system requirements' models containing elements expressing activity-like statements. As it is stated in Section III, it is not an easily solved problem, due to multiple ambiguities, applied modeling practices, and many other issues that are not addressed in common NLP processing toolkits. Among such issues, one may emphasize the processing of disjunctive or conjunctive statements (which is considered to be a bad modeling practice), the presence of shortened forms, like acronyms or abbreviations.

To solve the issues addressed in this paper, we evaluated several current state-of-the-art implementations from the perspective of our research, while combining them under our custom formal grammar-based extraction to derive prototype implementations. Additionally, we implemented and tested our custom tagging tools, based on input corpora augmentations and bidirectional LSTM-CRF architecture with BERT and ELMO embeddings at the input layer. In the first experiment, the Stanza-based implementation showed the best performance results in noun/verb extraction tasks. Yet, we showed that implementation based on our custom BERT-BiLSTM-CRF tagger helped to improve the detection of verb phrase presence and verb phrase extraction as compared to the generic tagger implementations, including generic BERT-based tagger. This was expected as bias towards proper tagging of verbs could reduce the ability to correctly tag nouns in short text statements. Hence, balancing between biased and unbiased tagging still requires further research.

Our second experiment with processing disjunctive and conjunctive statements showed this task to be more challenging than expected, due to the dependence of our implementation on the performance of underlying dependency parser toolkits. Unfortunately, while such statements are also considered to be bad modeling practice, they are widely used in real-world cases (this is also verified from the initial analysis of BPMAI dataset) and need to be addressed carefully. This is an important topic relevant for multiple information extraction and other NLP-related areas, such as relation extraction or aspect-based sentiment analysis. It has been proven to be a complicated task due to the generally unstructured nature of natural language texts. Handling of these issues is also discussed in this paper providing additional insights for further improvements in this area. Results obtained after applying our technique described in Section V-C indicate that there is still a lot of potential for further improvements. While at this stage, we did not consider training custom parsers, we hope to achieve more progress in the future after carrying out more extensive studies and taking advantage of the improvements in dependency parsing, constituency parsing, and general relation extraction algorithms.

Finally, in the third experiment, we tested a machine learning-based approach for the acronym/abbreviation detection issue. While this issue is widely discussed in multiple papers (see Table 1 for more details on that), these works tend to focus on processing longer text statements or even whole documents, which is not suitable for our particular case. Due to limitations discussed in previous sections, we approached this issue by applying context-based classification using token-level and text label-level features. We found out that our trained classifier was able to obtain a precision of 0.78 and F1-Score of 0.73, which we consider to be a rather positive result due to multiple constraints and limitations. In the future, we might as well test the developed solution in other settings by expanding our developed dataset to include more specific cases. The results are expected to be improved after applying the classifier to a more extensive and comprehensive dataset, which would lead to exploiting additional token-level, phrase-level, or even whole model-level features, and is still subject to our further research. In this paper, we did not consider acronym/abbreviation expansion, due to certain limitations and requirements discussed in Section VI. Yet, it is an interesting challenge that will be addressed in our future developments.

While our research presents a certain amount of contribution in text processing for the system modeling domain, there is still a lot of space for future research. In this paper, we experimented with text labels of activity-like elements acquired from the BPMN process models and UML use case models. However, other models, like UML activity models, state machines (or other kinds of statechart models) could also be successfully tested. Moreover, applying these techniques to larger and more elaborate datasets might reveal other cases that could be addressed by tuning the formal grammars or processing algorithms discussed in this paper. Additionally, one could also resort to creating specialized datasets or text corpora which would enable the development of even-more specialized extraction tools. Complementary, several technological constraints should be addressed, particularly optimization of the final models for deployment due to the requirement of a significant amount of resources needed to run larger deep learning models. This may require investigation of model reduction techniques such as distillation or quantization.

Finally, it is safe to state that in model-to-model transformation (as well as in other areas involving the processing of graphical models), one could also benefit from other existing

NLP capabilities, such as the extraction of semantic relationships (synonymy, hyponymy, hypernymy, etc.), analysis and correction of grammatical errors. Indeed, fully automated processing requires significant input and capabilities from multiple fields of linguistic processing to ensure the high performance of the developed NLP applications, as discussed in Table 1. This paves the road for our next near-future developments and experimentation.

REFERENCES

[1] C. Ru, J. Tang, S. Li, S. Xie, and T. Wang, "Using semantic similarity to reduce wrong labels in distant supervision for relation extraction," *Inf. Process. Manage.*, vol. 54, no. 4, pp. 593–608, Jul. 2018.

[2] H. Fei, Y. Ren, and D. Ji, "Boundaries and edges rethinking: An end-to-end neural model for overlapping entity relation extraction," *Inf. Process. Manage.*, vol. 57, no. 6, 2020, Art. no. 102311.

[3] D. T. Vo and E. Bagheri, "Self-training on refined clause patterns for relation extraction," *Inf. Process. Manage.*, vol. 54, no. 4, pp. 686–706, Jul. 2017.

[4] D. Nozza, P. Manchanda, E. Fersini, M. Palmonari, and E. Messina, "LearningToAdapt with word embeddings: Domain adaptation of named entity recognition systems," *Inf. Process. Manage.*, vol. 58, no. 3, May 2021, Art. no. 102537.

[5] Y. Jiang, W. Bai, X. Zhang, and J. Hu, "Wikipedia-based information content and semantic similarity computation," *Inf. Process. Manage.*, vol. 53, no. 1, pp. 248–265, Jan. 2017.

[6] P. Danenas, T. Skersys, and R. Butleris, "Natural language processing-enhanced extraction of SBVR business vocabularies and business rules from UML use case diagrams," *Data Knowl. Eng.*, vol. 128, Jul. 2020, Art. no. 101822.

[7] P. Danenas, T. Skersys, and R. Butleris, "Extending drag-and-drop actions-based model-to-model transformations with natural language processing," *Appl. Sci.*, vol. 10, no. 19, p. 6835, Sep. 2020.

[8] H. Leopold, F. Pittke, and J. Mendling, "Ensuring the canonicity of process models," *Data Knowl. Eng.*, vol. 111, pp. 22–38, Sep. 2017.

[9] H. Leopold, R.-H. Eid-Sabbagh, J. Mendling, L. G. Azevedo, and F. A. Baião, "Detection of naming convention violations in process models for different languages," *Decis. Support Syst.*, vol. 56, pp. 310–325, Dec. 2013.

[10] F. Pittke, H. Leopold, and J. Mendling, "When language meets language: Anti patterns resulting from mixing natural and modeling language," in *Proc. Bus. Process Manage. Workshops*, F. Fournier and J. Mendling, Eds. Cham, Switzerland: Springer, 2015, pp. 118–129.

[11] S. Kumar, "A survey of deep learning methods for relation extraction," 2017, *arXiv:1705.03645*.

[12] E. Agichtein and L. Gravano, "*Snowball*: Extracting relations from large plain-text collections," in *Proc. 5th ACM Conf. Digit. Libraries*, New York, NY, USA, 2000, pp. 85–94.

[13] M. Mintz, S. Bills, R. Snow, and D. Jurafsky, "Distant supervision for relation extraction without labeled data," in *Proc. AFNLP*, Singapore, Aug. 2009, pp. 1003–1011.

[14] V.-T. Phi, J. Santoso, V.-H. Tran, H. Shindo, M. Shimbo, and Y. Matsumoto, "Distant supervision for relation extraction via piecewise attention and bag-level contextual inference," *IEEE Access*, vol. 7, pp. 103570–103582, 2019.

[15] A. S. White, D. Reisinger, K. Sakaguchi, T. Vieira, S. Zhang, R. Rudinger, K. Rawlins, and B. Van Durme, "Universal decompositional semantics on universal dependencies," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Austin, TX, USA, 2016, pp. 1713–1723.

[16] *Business Process Model and Notation (BPMN), Version 2.0.2*, Object Management Group (OMG), Needham, MA, USA, Dec. 2013.

[17] *Unified Modeling Language (UML), Version 2.5.1*, Object Management Group (OMG), Needham, MA, USA, Dec. 2017.

[18] E. Jakumeit, S. Buchwald, D. Wagelaar, L. Dan, Á. Hegedüs, M. Herrmannsdörfer, T. Horn, E. Kalnina, C. Krause, K. Lano, M. Lepper, A. Rensink, L. Rose, S. Wätzoldt, and S. Mazanek, "A survey and comparison of transformation tools based on the transformation tool contest," *Sci. Comput. Program.*, vol. 85, pp. 41–99, Jun. 2014.

[19] N. Kahani, M. Bagherzadeh, J. R. Cordy, J. Dingel, and D. Varró, "Survey and classification of model transformation tools," *Softw. Syst. Model.*, vol. 18, no. 4, pp. 2361–2397, Aug. 2019.

[20] T. Skersys, P. Danenas, and R. Butleris, "Model-based M2M transformations based on drag-and-drop actions: Approach and implementation," *J. Syst. Softw.*, vol. 122, pp. 327–341, Dec. 2016.

[21] M. A. Hearst, "TextTiling: Segmenting text into multi-paragraph subtopic passages," *Comput. Linguistics*, vol. 23, no. 1, pp. 33–64, 1997.

[22] S. Bird, E. Klein, and E. Loper, *Natural Language Processing With Python*. Sebastopol, CA, USA: O'Reilly, 2009.

[23] (2021). *Spacy.io*. Accessed: Aug. 15, 2022. [Online]. Available: https://github.com/explosion/spaCy

[24] A. Akbik, T. Bergmann, D. Blythe, K. Rasul, S. Schweter, and R. Vollgraf, "FLAIR: An easy-to-use framework for state-of-the-art NLP," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics*, Minneapolis, MI, USA, Jun. 2019, pp. 54–59.

[25] M. Gardner, J. Grus, M. Neumann, O. Tafjord, P. Dasigi, N. F. Liu, M. Peters, M. Schmitz, and L. Zettlemoyer, "AllenNLP: A deep semantic natural language processing platform," in *Proc. Workshop NLP Open Source Softw. (NLP-OSS)*, Melbourne, VIC, Australia, Jul. 2018, pp. 1–6.

[26] G. Chrupala, G. Dinu, and J. van Genabith, "Learning morphology with Morfette," in *Proc. Int. Conf. Lang. Resour. Eval.*, Marrakech, Morocco, Jun. 2008, pp. 1–6.

[27] A. Chakrabarty, O. A. Pandit, and U. Garain, "Context sensitive lemmatization using two successive bidirectional gated recurrent networks," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics*, R. Barzilay and M. Kan, Eds. Vancouver, BC, Canada, 2017, pp. 1481–1491.

[28] T. Bergmanis and S. Goldwater, "Context sensitive neural lemmatization with Lematus," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Human Lang. Technol.*, M. A. Walker, H. Ji, and A. Stent, Eds. New Orleans, LA, USA, 2018, pp. 1391–1400.

[29] M. Arehart, "Indexing methods for faster and more effective person name search," in *Proc. 7th Int. Conf. Lang. Resour. Eval.*, Valletta, Malta, May 2010, pp. 1–15.

[30] A. Rozovskaya and D. Roth, "Grammatical error correction: Machine translation and classifiers," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, Berlin, Germany, 2016, pp. 2205–2215.

[31] M. Junczys-Dowmunt, R. Grundkiewicz, S. Guha, and K. Heafield, "Approaching neural grammatical error correction as a low-resource machine translation task," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Human Lang. Technol.*, New Orleans, LA, USA, 2018, pp. 595–606.

[32] S. Kiyono, J. Suzuki, M. Mita, T. Mizumoto, and K. Inui, "An empirical study of incorporating pseudo data into grammatical error correction," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, Hong Kong, 2019, pp. 1236–1242.

[33] A. Ratnaparkhi, "A maximum entropy model for part-of-speech tagging," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 1996, pp. 1–10.

[34] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer, "Feature-rich part-of-speech tagging with a cyclic dependency network," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics Hum. Lang. Technol.*, 2003, pp. 252–259.

[35] M. Silfverberg, T. Ruokolainen, K. Lindén, and M. Kurimo, "Part-of-speech tagging using conditional random fields: Exploiting sub-label dependencies for improved accuracy," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics*, Baltimore, MD, USA, 2014, pp. 259–264.

[36] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12 pp. 2493–2537, Aug. 2011.

[37] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF models for sequence tagging," 2015, *arXiv:1508.01991*.

[38] G. A. Miller, "WordNet: A lexical database for English," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[39] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1. Minneapolis, MI, USA, Jun. 2019, pp. 4171–4186.

[40] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. 26th Int. Conf. Neural Inf. Process. Syst.*, vol. 2. Red Hook, NY, USA, 2013, pp. 3111–3119.

[41] M. E. Peters, M. Neumann, M. Iyyer, and M. Gardner, "Deep contextualized word representations," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1. New Orleans, LA, USA: ACL, Jun. 2018, pp. 2227–2237.

[42] M. A. Hearst, "Automatic acquisition of hyponyms from large text corpora," in *Proc. 14th Conf. Comput. Linguistics*, 1992, pp. 1–7.

[43] R. Snow, D. Jurafsky, and A. Y. Ng, "Learning syntactic patterns for automatic hypernym discovery," in *Proc. 17th Int. Conf. Neural Inf. Process. Syst.*, Cambridge, MA, USA: MIT Press, 2004, pp. 1297–1304.

[44] M. Onofrei, I. Hulub, D. Trandabat, and D. Gifu, "Apollo at SemEval-2018 task 9: Detecting hypernymy relations using syntactic dependencies," in *Proc. 12th Int. Workshop Semantic Eval.*, New Orleans, LA, USA, 2018, pp. 898–902.

[45] T. Kawaumra, M. Sekine, and K. Matsumura, "Hyponym/hypernym detection in science and technology thesauri from bibliographic datasets," in *Proc. IEEE 11th Int. Conf. Semantic Comput. (ICSC)*, 2017, pp. 180–187.

[46] Z. Zhang, J. Li, H. Zhao, and B. Tang, "SJTU-NLP at SemEval-2018 task 9: Neural hypernym discovery with term embeddings," in *Proc. 12th Int. Workshop Semantic Eval.*, New Orleans, LA, USA, 2018, pp. 903–908.

[47] A. Z. Hassan, M. S. Vallabhajosyula, and T. Pedersen, "UMDuluth-CS8761 at SemEval-2018 task9: Hypernym discovery using Hearst patterns, co-occurrence frequencies and word embeddings," in *Proc. 12th Int. Workshop Semantic Eval.*, New Orleans, LA, USA, 2018, pp. 914–918.

[48] T. Dozat and C. D. Manning, "Simpler but more accurate semantic dependency parsing," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, Melbourne, VIC, Australia, 2018, pp. 484–490.

[49] T. Kim, B. Li, and S.-G. Lee, "Multilingual chart-based constituency parse extraction from pre-trained language models," in *Proc. Findings Assoc. Comput. Linguistics, EMNLP*, Punta Cana, Dominican Republic, 2021, pp. 454–463.

[50] S. Petrov, L. Barrett, R. Thibaux, and D. Klein, "Learning accurate, compact, and interpretable tree annotation," in *Proc. 21st Int. Conf. COLING-ACL*, Sydney, NSW, Australia, Jul. 2006, pp. 433–440.

[51] Y. Zhang, Z. Li, and M. Zhang, "Efficient second-order TreeCRF for neural dependency parsing," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 3295–3305.

[52] T. Ji, Y. Wu, and M. Lan, "Graph-based dependency parsing with graph neural networks," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, Florence, Italy, 2019, pp. 2475–2485.

[53] W. Wang and B. Chang, "Graph-based dependency parsing with bidirectional LSTM," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, Berlin, Germany, 2016, pp. 2306–2315.

[54] A. S. Schwartz and M. A. Hearst, "A simple algorithm for identifying abbreviation definitions in biomedical text," in *Proc. 8th Pacific Symp. Biocomput.*, R. B. Altman, A. K. Dunker, L. Hunter, and T. E. Klein, Eds. Lihue, HI, USA, Dec. 2002, pp. 451–462.

[55] S. Sohn, D. C. Comeau, W. Kim, and W. J. Wilbur, "Abbreviation definition identification based on automatic precision estimates," *BMC Bioinform.*, vol. 9, no. 1, p. 402, 2008.

[56] Y. Wu, S. T. Rosenbloom, J. C. Denny, R. A. Miller, S. Mani, D. A. Giuse, and H. Xu, "Detecting abbreviations in discharge summaries using machine learning methods," in *Proc. AMIA Annu. Symp.*, 2011, pp. 1541–1549.

[57] M. Oleynik, M. Kreuzthaler, and S. Schulz, "Unsupervised abbreviation expansion in clinical narratives," in *Proc. 16th World Congr. Med. Health Inform.*, vol. 245, A. V. Gundlapalli, M. Jaulent, and D. Zhao, Eds. Hangzhou, China: IOS Press, Aug. 2017, pp. 539–543.

[58] L. Heryawan, O. Sugiyama, G. Yamamoto, P. H. Khotimah, L. H. O. Santos, K. Okamoto, and T. Kuroda, "A detection of informal abbreviations from free text medical notes using deep learning," *Eur. J. Biomed. Informat.*, vol. 16, no. 1, pp. 29–37, 2020.

[59] X. Huang, E. Zhang, and Y. S. Koh, "Supervised clinical abbreviations detection and normalisation approach," in *PRICAI 2019: Trends in Artificial Intelligence*, A. C. Nayak and A. Sharma, Eds. Cham, Switzerland: Springer, 2019, pp. 691–703.

[60] Q. Jin, J. Liu, and X. Lu, "Deep contextualized biomedical abbreviation expansion," in *Proc. 18th BioNLP Workshop Shared Task*, Florence, Italy, Aug. 2019, pp. 88–96.

[61] I. Li, M. Yasunaga, M. Y. Nuzumlali, C. Caraballo, S. Mahajan, H. M. Krumholz, and D. R. Radev, "A neural topic-attention model for medical term abbreviation disambiguation," 2019, *arXiv:1910.14076*.

[62] V. Joopudi, B. Dandala, and M. Devarakonda, "A convolutional route to abbreviation disambiguation in clinical text," *J. Biomed. Informat.*, vol. 86, pp. 71–78, Oct. 2018.

[63] J. P. C. Chiu and E. Nichols, "Named entity recognition with bidirectional LSTM-CNNs," *Trans. Assoc. Comput. Linguistics*, vol. 4, pp. 357–370, Dec. 2016.

[64] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics*, San Diego, CA, USA, Jun. 2016, pp. 260–270.

[65] A. Ghaddar and P. Langlais, "Robust lexical features for improved neural network named-entity recognition," in *Proc. 27th Int. Conf. Comput. Linguistics*, Santa Fe, NM, USA, Aug. 2018, pp. 1896–1907.

[66] L. Liu, J. Shang, X. Ren, F. F. Xu, H. Gui, J. Peng, and J. Han, "Empower sequence labeling with task-aware neural language model," in *Proc. 32nd AAAI Conf. Artif. Intell. 33th Innov. Appl. Artif. Intell. Conf. 8th AAAI Symp. Educ. Adv. Artif. Intell.*, 2018, pp. 1–15.

[67] Y. Shao, J. C.-W. Lin, G. Srivastava, A. Jolfaei, D. Guo, and Y. Hu, "Self-attention-based conditional random fields latent variables model for sequence labeling," *Pattern Recognit. Lett.*, vol. 145, pp. 157–164, May 2021.

[68] I. O. Mulang', K. Singh, C. Prabhu, A. Nadgeri, J. Hoffart, and J. Lehmann, "Evaluating the impact of knowledge graph context on entity disambiguation models," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, New York, NY, USA, Oct. 2020, pp. 2157–2160.

[69] M. P. K. Ravi, K. Singh, I. O. Mulang', S. Shekarpour, J. Hoffart, and J. Lehmann, "CHOLAN: A modular approach for neural entity linking on Wikipedia and Wikidata," in *Proc. 16th Conf. Eur. Chapter Assoc. Comput. Linguistics*, 2021, pp. 504–514.

[70] Y. Zhang, H. Lin, Z. Yang, J. Wang, Y. Sun, B. Xu, and Z. Zhao, "Neural network-based approaches for biomedical relation classification: A review," *J. Biomed. Informat.*, vol. 99, Nov. 2019, Art. no. 103294.

[71] L. He, K. Lee, M. Lewis, and L. Zettlemoyer, "Deep semantic role labeling: What works and what's next," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics*, Vancouver, BC, Canada, 2017, pp. 473–483.

[72] J. Zhou and W. Xu, "End-to-end learning of semantic role labeling using recurrent neural networks," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics 7th Int. Joint Conf. Natural Lang. Process.*, Beijing, China, 2015, pp. 1127–1137.

[73] Z. Tan, M. Wang, J. Xie, Y. Chen, and X. Shi, "Deep semantic role labeling with self-attention," in *Proc. 32nd AAAI Conf. Artif. Intell. 30th Innov. Appl. Artif. Intell. Conf. 8th AAAI Symp. Educ. Adv. Artif. Intell.*, 2018, pp. 1–8.

[74] D. Marcheggiani and I. Titov, "Encoding sentences with graph convolutional networks for semantic role labeling," in *Proc. EMNLP*, Copenhagen, Denmark, Sep. 2017, pp. 1506–1515.

[75] H. Fei, M. Zhang, B. Li, and D. Ji, "End-to-end semantic role labeling with neural transition-based model," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, pp. 12803–12811, May 2021.

[76] M. Zhang, "A survey of syntactic-semantic parsing based on constituent and dependency structures," *Sci. China Technol. Sci.*, vol. 63, no. 10, pp. 1898–1920, Oct. 2020.

[77] W. Han, Y. Jiang, H. T. Ng, and K. Tu, "A survey of unsupervised dependency parsing," in *Proc. 28th Int. Conf. Comput. Linguistics*, Barcelona, Spain, 2020, pp. 2522–2533.

[78] J. Li, A. Sun, J. Han, and C. Li, "A survey on deep learning for named entity recognition," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 1, pp. 50–70, Jan. 2022.

[79] K. Liu, "A survey on neural relation extraction," *Sci. China Technol. Sci.*, vol. 63, no. 10, pp. 1971–1989, Oct. 2020.

[80] Ö. Sevgili, A. Shelmanov, M. Arkhipov, A. Panchenko, and C. Biemann, "Neural entity linking: A survey of models based on deep learning," *Semantic Web*, vol. 13, no. 3, pp. 527–570, Apr. 2022.

[81] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERT: A robustly optimized BERT pretraining approach," 2019, *arXiv:1907.11692*.

[82] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "ELECTRA: Pre-training text encoders as discriminators rather than generators," 2020, *arXiv:2003.10555*.

[83] Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized autoregressive pretraining for language understanding," 2019, *arXiv:1906.08237*.

[84] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, no. 140, pp. 1–67, 2020.

[85] P. He, X. Liu, J. Gao, and W. Chen, "DeBERTa: Decoding-enhanced BERT with disentangled attention," 2020, *arXiv:2006.03654*.

[86] S. Adolph, A. Cockburn, and P. Bramble, *Patterns for Effective Use Cases*. Boston, MA, USA: Addison-Wesley, 2002.

[87] S. W. Ambler, *The Elements of UML(TM) 2.0 Style*. USA: Cambridge, U.K.: Cambridge Univ. Press, 2005.
[88] M. Weske, G. Decker, M. Dumas, M. La Rosa, J. Mendling, and H. A. Reijers, "Model collection of the business process management academic initiative," Version BPMAI-29-10-2019, Zenodo, 2020.
[89] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics, Syst. Demonstrations*, Baltimore, MD, USA, 2014, pp. 55–60.
[90] P. Qi, Y. Zhang, Y. Zhang, J. Bolton, and C. D. Manning, "Stanza: A Python natural language processing toolkit for many human languages," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics, Syst. Demonstrations*, 2020, pp. 101–108.
[91] R. Weischedel, M. Palmer, M. Marcus, E. Hovy, S. Pradhan, L. Ramshaw, N. Xue, A. Taylor, J. Kaufman, M. Franchini, M. El-Bachouti, R. Belvin, and A. Houston, "OntoNotes release 5.0," LDC2013T19, Linguistic Data Consortium, Philadelphia, PA, USA, 2013.
[92] R. Panchendrarajan and A. Amaresan, "Bidirectional LSTM-CRF for named entity recognition," in *Proc. 32nd Pacific Asia Conf. Lang., Inf. Comput.*, Hong Kong, Dec. 2018, pp. 1–10.
[93] A. Gatt and E. Reiter, "SimpleNLG: A realisation engine for practical applications," in *Proc. 12th Eur. Workshop Natural Lang. Gener.*, Athens, Greece, 2009, pp. 90–93.
[94] J. Nivre, M.-C. de Marneffe, F. Ginter, J. Hajič, C. D. Manning, S. Pyysalo, S. Schuster, F. Tyers, and D. Zeman, "Universal dependencies V2: An evergrowing multilingual treebank collection," in *Proc. 12th Lang. Resour. Eval. Conf.*, Marseille, France, May 2020, pp. 4034–4043.
[95] A. Bies, J. Mott, C. Warner, and S. Kulick, "English web Treebank LDC2012T13," Linguistic Data Consortium, Philadelphia, PA, USA, 2012.
[96] S. Saha, "Open information extraction from conjunctive sentences," in *Proc. 27th Int. Conf. Comput. Linguistics*, Santa Fe, NM, USA, Aug. 2018, pp. 2288–2299.
[97] J. Ficler and Y. Goldberg, "A neural network for coordination boundary prediction," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Austin, TX, USA, 2016, pp. 23–32.
[98] M. Miwa, R. Sætre, Y. Miyao, and J. Tsujii, "Entity-focused sentence simplification for relation extraction," in *Proc. 23rd Int. Conf. Comput. Linguistics*, Beijing, China, Aug. 2010, pp. 788–796.
[99] D. Vickrey and D. Koller, "Sentence simplification for semantic role labeling," in *Proc. ACL, HLT*, Columbus, OH, USA, Jun. 2008, pp. 344–352.
[100] T. N. C. Vo, T. H. Cao, and T. B. Ho, "Abbreviation identification in clinical notes with level-wise feature engineering and supervised learning," in *Knowledge Management and Acquisition for Intelligent Systems*, H. Ohwada and K. Yoshida, Eds. Cham, Switzerland: Springer, pp. 3–17, 2016.
[101] A. V. Dorogush, V. Ershov, and A. Gulin, "CatBoost: Gradient boosting with categorical features support," 2018, *arXiv:1810.11363*.
[102] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794.
[103] T. K. Ho, "Random decision forests," in *Proc. 3rd Int. Conf. Document Anal. Recognit.*, vol. 1, Aug. 1995, pp. 278–282.
[104] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

PAULIUS DANENAS received the Ph.D. degree in informatics from Vilnius University, Vilnius, Lithuania, in 2013.

He is currently a Researcher at the Centre of Information Systems Design Technologies, Kaunas University of Technology, Kaunas, Lithuania. He is a coauthor of multiple papers in highly-rated academic journals and proceedings of international conferences. His research interests include artificial intelligence, machine learning, natural language processing, data science, software engineering, model-driven development, and decision support systems (including business and financial domains). He has served as a reviewer for a number of highly-ranked academic journals, including the ones published by Springer, Elsevier, Wiley, Taylor & Francis, IEEE, and others.

TOMAS SKERSYS is a Scientific Researcher at the Center of Information Systems Design Technologies and a Professor at the Department of Information Systems, Kaunas University of Technology. His research interests and practical experience cover various aspects of business process management and model-driven information systems development. On these topics, he has published several articles in high-rated academic journals and in a number of international conferences. He is also a co-editor of three books of international conferences published by Springer Verlag.