

Kaunas University of Technology
Informatics faculty

Automation of e-commerce product description using artificial intelligence methods

Master's Final Degree Project

Ignas Narušis

Project author

**Prof. Dr. Agnė PAULAUSKAITĖ-
TARASEVIČIENĖ**

Supervisor

Kaunas, 2024



Kaunas University of Technology
Informatics faculty

Automation of e-commerce product description using artificial intelligence methods

Master's Final Degree Project
Artificial Intelligence in Computer Science (6211BX007)

Ignas Narušis

Project author

**Prof. Dr. Agnė PAULAUŠKAITĖ-
TARASEVIČIENĖ**

Supervisor

Aušra Gadeikytė

Reviewer

Kaunas, 2024



Kaunas University of Technology

Informatics faculty

Ignas Narušis

Automation of e-commerce product description using artificial intelligence methods

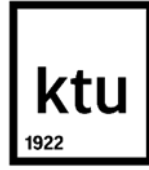
Declaration of Academic Integrity

I confirm the following:

1. I have prepared the final degree project independently and honestly without any violations of the copyrights or other rights of others, following the provisions of the Law on Copyrights and Related Rights of the Republic of Lithuania, the Regulations on the Management and Transfer of Intellectual Property of Kaunas University of Technology (hereinafter – University) and the ethical requirements stipulated by the Code of Academic Ethics of the University;
2. All the data and research results provided in the final degree project are correct and obtained legally; none of the parts of this project are plagiarised from any printed or electronic sources; all the quotations and references provided in the text of the final degree project are indicated in the list of references;
3. I have not paid anyone any monetary funds for the final degree project or the parts thereof unless required by the law;
4. I understand that in the case of any discovery of the fact of dishonesty or violation of any rights of others, the academic penalties will be imposed on me under the procedure applied at the University; I will be expelled from the University and my final degree project can be submitted to the Office of the Ombudsperson for Academic Ethics and Procedures in the examination of a possible violation of academic ethics.

Ignas Narušis

Confirmed electronically



Kaunas University of Technology
Informatics faculty

Task of the Master's final degree project

Topic of the project	Automation of e-commerce product description using artificial intelligence methods.
Requirements and conditions (title can be clarified, if needed)	
Supervisor	(position, name, surname, signature of the supervisor) (date)

Ignas Narušis. Automation of e-commerce product description using artificial intelligence methods. Master's Final Degree Project. Supervisor: Prof. Dr. Agnė PAULAUŠKAITĖ-TARASEVIČIENĖ; Informatics faculty, Kaunas University of Technology.

Study field and area (study field group): Computer science, Informatics (B01).

Keywords: NLP, web scraping, web crawler, TF-IDF, SBERT, Cosine Similarity.

Kaunas, 2024. 59 p.

Summary

The information gathering process for e-commerce product description recognition involves scraping information using specific tools, classifying necessary textual information, and recognizing named entities within all listed documents. This process can improve search specifications in agent development.

During the solution projection, requirements for the agent's development, usage, and steps required for algorithms for textual description extraction were raised. These components include a crawler that collects all possible relevant information, classification of collected information to determine relevance to the search document (using a combination of embedding methods such as TF-IDF, SBERT, and Cosine Similarity), recognition of named entities, and the process of marking up words in a text (leveraging existing NLP models such as NER and POS tagging from spaCy).

The design phase outlines the essential components of the agent's algorithm, including data collection, classification, named entity identification, and user presentation. The SBERT - Cosine Similarity - K-Means - NLP pipeline was chosen for e-commerce product description recognition, with an accuracy rate of 77,90% and a total execution time of 154,911 seconds.

In the image processing experiment, the Siamese network outperformed the Autoencoder-Decoder model in terms of processing speed, achieving an average of 249,41 iterations per second, however the Autoencoder model performed well in finding similar images with lower resolutions. The Siamese network demonstrated superior performance in this context, with a validation accuracy of 98,6% in identifying similar class images using one-shot classification principles.

In summary, the experiment with web crawling resulted in a significant improvement in the download time of text and image documents due to the implementation of multiprocessing with 12 threads, resulting in a speed increase of approximately 2,67 times and approximately 0,855 files per second faster.

Ignas Narušis. Dirbtinio intelekto algoritmų metodų taikymas elektroninės prekybos prekių aprašo automatizacijai. Magistro baigiamasis projektas Prof. Dr. Agne Paulauskaitė-Tarasevičienė; Kauno technologijos universitetas, informatikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Informatikos mokslai, Informatika (B01).

Reikšminiai žodžiai: NLP, žiniatinklio nuskaitymas, web-crawler, TF-IDF, SBERT, kosinusinis panašumas.

Kaunas, 2024. 59 p.

Santrauka

Informacijos rinkimo procesas atpažįstant e. parduotuvės produktų aprašymus apima informacijos paiešką naudojant specialias priemones, reikiamos tekstinės informacijos klasifikavimą ir įvardytų aprašymų atpažinimą visuose išvardytuose dokumentuose. Šis procesas gali pagerinti paieškos specifikacijas kuriant agentus.

Sprendimų projekcijos metu buvo iškelti reikalavimai agento kūrimui, naudojimui ir žingsniai, reikalingi tekstinio aprašymo išskyrimo algoritmams. Šie komponentai apima naršyklę, kuri surenka visą galimą susijusią informaciją, surinktos informacijos klasifikavimą, siekiant nustatyti svarbą paieškos dokumentui (naudojant užkodavimo metodais, tokių kaip TF-IDF, SBERT ir kosinusinio panašumo, derinį), įvardytų esybių atpažinimą ir žodžių žymėjimo tekste procesą (panaudojant esamus NLP modelius, tokius kaip NER ir POS žymėjimas iš spaCy).

Projektavimo etape apibrėžiami esminiai agento algoritmo komponentai, įskaitant duomenų rinkimą, klasifikavimą, įvardytų esybių identifikavimą ir pateikimą naudotojui. Elektroninės prekybos produktų aprašymui atpažinti pasirinktas SBERT - Cosine Similarity - K-Means - NLP vamzdynas, kurio tikslumo rodiklis yra 77,90 %, o bendras vykdymo laikas buvo 154,911 sekundės.

Atliekant vaizdų apdorojimo eksperimentą, Siamo tinklas pagal apdorojimo spartą pranoko automatinio kodavimo-dekodavimo modelį, pasiektos vidutiniškai 249,41 iteracijos per sekundę, tačiau automatinio kodavimo modelis gerai veikė ieškant panašių mažesnės raiškos vaizdų. Šiame kontekste Siamo tinklas pademonstravo geresnius rezultatus, kur validavimo tikslumas nustatant panašios klasės vaizdus taikant vienkartinio klasifikavimo principus buvo 98,6 %.

Apibendrinant galima teigti, kad atlikus eksperimentą su naršymu internete, dėl daugiaprocesorinio apdorojimo su 12 gijų įgyvendinimo gerokai pagerėjo tekstinių ir vaizdinių dokumentų atsisiuntimo laikas, todėl sparta padidėjo maždaug 2,67 karto ir maždaug 0,855 failo per sekundę greičiau.

Table of contents

List of figures	9
List of tables	10
Introduction	11
1. Literature review - Information collection and weighting in a text analysis	12
1.1. Analytical methods	12
1.2. Machine learning	12
1.3. Supervised machine learning.....	12
1.4. Unsupervised machine learning	13
1.5. Term weighting in text classification	13
1.6. Existing datasets for text classification training	14
1.7. Efficient Text Classification.....	14
1.8. Bag of words.....	15
1.9. Technology of Web Crawler	15
1.10. Text classification.....	16
1.11. Representation of Text in Input Space	18
1.12. Relevant Information Extraction	18
1.13. Automated Information gathering from Web.....	19
1.14. Presence of the solutions	20
1.14.1. Employed people	20
1.14.2. Existing systems	20
1.15. Schema.org usage for training product matchers	20
1.16. IR system in text-based document using vector space model	21
1.17. Named entity recognition	21
1.18. Possible Approaches (discussions).....	22
1.19. Tools and software analysis.....	22
1.19.1. Automated information gathering from Web tools.	22
1.19.2. <i>Scrapy Tool</i>	22
1.19.3. Selenium	22
1.19.4. Modern web scraping architectures and techniques	23
1.19.5. <i>BeautifulSoup</i> web scraping	23
1.19.6. Programming language tools.....	24
1.19.7. Google Search API.....	24
1.20. Auto encoder decoder.....	24
1.21. Siamese Network.....	25
1.22. Sentence-BERT	25
2. Automated e-commerce description gathering solution specification.....	27
2.1. Methods and tools used to implement e-commerce description gathering system.	27
2.2. Project plan.....	28
2.3. Use case diagram.....	29
2.4. Functional requirements	35
2.5. Nonfunctional.....	35
2.6. Targeted quality criteria	36
2.7. Agent system activity diagram	36
2.8. Basic component model	39
2.9. Class diagram	40
3. Agent information retrieval realisation.....	41
3.1. Input data description.	41
3.1.1. AI block overview.....	43
3.1.2. Neural networks architecture.....	44
3.1.3. Assessing Siamese Network and Autoencoder image similarity models.....	45
3.1.4. Textual input and processing entities recognition	48

3.2. Designed command line interface.	49
4. Investigation on the structure of the system Implementation results and experimental comparisons.....	50
4.1. Description of the Investigation environment	50
4.2. Investigation of Web scraping.....	50
4.3. Exploration of the output results segment	51
4.4. Investigation of different Image similarity methods	52
4.5. Investigation of different Embedding and Clustering methods for Named Entity Recognition improvement.....	53
Conclusions	55
List of references.....	56
Appendices	59
Appendix 1. “Towards Early Poultry Health Prediction through Non-Invasive and Computer Vision-Based Dropping Classification”	59

List of figures

Fig 1.1. Supervised and unsupervised learning principal [14]	13
Fig.1.2 fast Text model architecture, using n-gram features for sentences [8]	14
Fig. 1.3 Crawling process flow chart [6]	16
Fig. 1.4 distribution of multiple crawler processes [6]	16
Fig. 1.5 Sentence structural breakdown [21, p. 3]	19
Fig. 1.6 Automated flow of initial preprocessing, and structural adaptation [21].	19
Fig. 1.7 Cleansing Pipeline that is used to group data offers into clusters [32].	20
Fig. 1.8 Siamese Network architecture [37].	25
Fig. 2.1 E-products description automation agent use case diagram.	29
Fig. 2.2 Agents general algorithm activity diagram – structure of agent components and their workflow	36
Fig. 2.3 Crawlers activity diagram	37
Fig. 2.4 Activity diagram showing the processing of the corresponding text in the tokenisation, insertion, and clustering steps	38
Fig. 2.5 activity diagram of named entity recognition and Lemmatization POS tagging	38
Fig. 2.6 Agent system components diagram, which contains base tools and algorithms.	39
Fig. 2.7 Agent system class diagram	40
Fig. 3.1 Generalised structure overview of realised Agent AI system.	42
Fig. 3.2 AI block which is included in AI preparation block sequence.	43
Fig. 3.3 Siamese Network training-validation graph, with 0,8 threshold of pairwise Euclidean distance similarity, where <i>a</i> – <i>Contrastive Loss graph</i> , <i>b</i> – <i>training accuracy</i> , <i>c</i> – <i>validation accuracy</i>	46
Fig. 3.4 Autoencoder training loss, training and validation graph, with 0,8 cosine similarity threshold, where <i>a</i> – <i>MSE Loss graph</i> , <i>b</i> – <i>training accuracy</i> , <i>c</i> – <i>validation accuracy</i>	47
Fig. 3.5 Text encoding block inside of AI block sequence, which contains sentence embedding using SBERT, TF-IDF, and Cosine-similarity combination finalised by clustering and grepping information using	48
Fig. 3.6 Sentence POS tagging example.	49
Fig. 3.7 Example of user CLI output, for general verbose process display.	50
Fig. 4.1 Autoencoder-decoder advantage comparing lower resolution images, experiment resource is based on [41]	52

List of tables

Table 1.1 Feature evaluation and selection metrics fragment [7]	17
Table 1.2 Explanation of Symbols [7]	17
Table 1.3 Selenium WebDriver for and against [26].....	23
Table 1.4 Given <i>BeautifulSoup</i> parser for and against [28].....	24
Table 1.5 Usage of <i>Python</i>	24
Table 2.1 Select product and entities use case.....	29
Table 2.2 Indicate the place and format of storage.....	30
Table 2.3 Initial crawling calibration use case.....	31
Table 2.4 Extraction of web data contents to structured format.....	32
Table 2.5 Detect and classify important entities of product use case.....	33
Table 2.6 Present product specifications use case.....	34
Table 2.7 Agent functional requirements.....	35
Table 2.8 Agent - algorithm nonfunctional requirements.....	36
Table 3.1 User required input search query and parameters example.....	41
Table 3.2 User query and parameters-based search results example.....	41
Table 3.3 Results of embedding and clustering query example	42
Table 3.4 Results of embedding and clustering sentences example	42
Table 3.5 Autoencoder model, divided into distinct Encoder and Decoder layers, based on Autoencoder baseline.	44
Table 3.6 Siamese Network architecture, based on [38, 39].	44
Table 3.7 Image similarity training dataset overview.....	45
Table 3.8 Siamese Network and Autoencoder training and validation measurements comparison .	47
Table 3.9 Structure of CLI operations	49
Table 4.1 Executed experiment analysis using user query and image downloading input.....	50
Table 4.2 Classification and named parameters recognition in user CLI.....	51
Table 4.3 Detailed CSV report about the user query scraped results	51
Table 4.4 Investigation of Siamese network and Autoencoder-Decoder models.....	52
Table 4.5 Investigation of natural language methods including accuracy, timing, and their parameters.....	53

Introduction

This work belongs to Artificial Intelligence in Computer Science Master studies program.

Research problem and relevance.

Data gathering in human activity in recent years has begun to store a vast amount of information, including their activities, steps, and sleeping schedules. This data has become highly structured from sensors connected to commercial platforms [22, 23]. The help of sensors can perform tasks that were not possible before without manually inputting active data every iteration [23]. However, it is becoming increasingly challenging to deal with different problems, such as unstructured and unlabeled data, despite its potential and valuable use [24]. With an increasing number of middleman companies trying to enter the market, resolving this problem becomes crucial. While specific brands often provide detailed descriptions, supermarket chains offer a variety of goods, leading to the hiring of manual labour to describe them by searching online, resulting in inconsistent or insufficient information. For example, a piece of furniture may only be described by its upholstery, leather, and tapestry, excluding details like dimensions or colour. As a result, customers may be misinformed or lack comprehensive information. An unresolved issue exists in obtaining necessary information from active sources, such as furniture with varying parts, dimensions, or colours. It would be ideal if most data could be automatically collected according to specified product types.

Work goal:

The research aims to enhance e-commerce product descriptions for the business sector by gathering and storing product information, evaluating its reliability employing artificial intelligence algorithms. The proposed solution involves developing a model that automates the extraction of structured data, enabling distributors to enrich product information with greater relevance and structural integrity.

Objectives:

1. Investigate machine learning technologies for collecting product descriptions.
2. Analyze the performance of algorithms in gathering and storing product information.
3. Evaluate and compare information scraping pipelines based on specified user parameters.
4. Conduct experimental testing to assess the selected algorithms.
5. Compare image similarities using neural network algorithms.

Research methods:

1. For product parameters information gathering the web scraping tool: *BeautifulSoup* in Python was used. Further: SBERT, TF-IDF, Cosine Similarity methods were used to embed textual data into vectors. These results were implemented in K-Means Clustering which collected relevant and irrelevant sentences. Only relevant sentences were processed through the NLP SpaCy pipeline – which provided relevant e-product parameters. Finally, the relevance score of these e-product parameters was measured between different websites.
2. Additionally neural network image similarity algorithms helped categorizing scraped websites with similar product images.

1. Literature review - Information collection and weighting in a text analysis

This analysis section will present existing text analysis algorithms, websites information gathering methods, similar solution analysis and comparison in defined situations.

1.1. Analytical methods

While analysing information weighting, gathering methods and similar solutions, it has been observed that specific solutions and problematics, that will help in text collection, gathering and classification for specific categories. These solutions are applied for a specific objective, for which it is appropriate to define specific criteria.

A list of is shown below:

1. It must use automated technologies for gathering and scraping information from websites.
2. Solution application of similar problematics, combination of it.
3. Text classification using existing categorized datasets for training.
4. Processing of words containing low informational values.
5. Similar solutions that could be used for scraping a specific information for needed products.

1.2. Machine learning

Machine learning can sometimes be understood as a branch of Artificial Intelligence (AI), which is focusing on learning and adapting data, while improving its accuracy in a certain amount of time without the help or need of software in data science at appropriate situation the algorithms are being applied, which are a statistical process sequence. Machine science algorithms are introduced to recognize or distinct specific features in vastly sized datasets, that these new solutions could improve with every increased data information source. Modern used products are being surrounded by the users, with machine learning, from a suggestion of a music or video a person might like. Sorting of e-mail spam sorting by used bought products, about browsing history there is being decided what are the appropriate advertisements are. Or what movie is it worth to offer for a user so it would be clicked again. Machine learning is divided mainly into the following steps: selection and preparation of a newly gathered dataset, selecting an algorithm to create a model depending on the selected algorithm, and finally usage and improvement of existing model [13, 14].

1.3. Supervised machine learning

In supervised machine learning algorithms first step is to prepare labelled dataset, for an appropriate algorithm training, then after this flow algorithm is supplied with an unlabelled testing dataset, which would let to distribute the data in similar groups, just like in image below (see Fig 1.1). Supervised machine learning algorithm training requires significantly less data, then other machine learning methods, this method lets to test and train with algorithms easier, because the current data can be compared with already existing dataset [14]. In major situations preparing data precisely and correctly requires a vast number of expensive resources, and there is a danger with over teaching the model, or under fitting the model with too little information, where the model itself will not be able to properly cope with the new data.

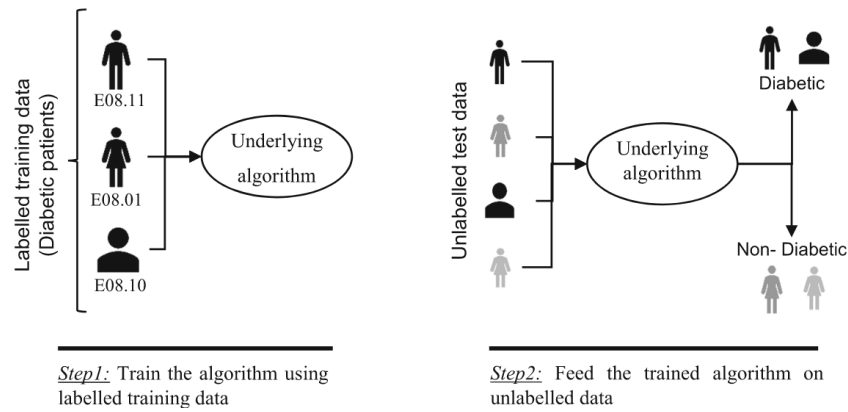


Fig 1.1. Supervised and unsupervised learning principal [14]

Supervised machine learning algorithms can be fine applied for classification problems and regressions. The classification problems are discrete where the variable is distributed in different subgroups or categories, such as colour recognition green, white, or figure distribution could be round, oval, rectangle, or triangle. The output regression of a variable can be real or can have a continuous value, such as: weight or wage, and there are many different simplest models of which they have linear regression [15]

1.4. Unsupervised machine learning

Unsupervised machine learning takes a vast amount of unlabelled information, the algorithm then uses to extract significant values, that can be distinct by Its features, it then learns by sorting them, and categorizes in real time without any further input from the human data [13, 14]. The purpose of these algorithms is less about automating decisions and predictions, and more about finding distributions in data that would be quite easy for people to misunderstand. A very appropriate example would be to use unsupervised machine learning, which would be good with sorting through the spam of a mailbox or analysing certain repetitions or similarities extracting a large amount of information, compared to supervised machine learning where no kind of human could be able and possible to label all the types of mail sent, and unsupervised machine learning would get better and better with more data, for unsupervised machine learning to improve its performance it needs more and data [12].

1.5. Term weighting in text classification

This IF-IGM scheme main idea was comparing how different and better the opposed solution then TF-IDF, which objective and focus is textual information gathering with automatic text categorization [1]. TF-IGM (term frequency & inverse gravity moment): distinguishes the impacts of a phrase, provides an adaptive solution with multiple options and settings that can be changed. A term IGM - "Inverse gravity moment ", means that the concentration of a assigned term is being measured. Which indicator the term differentiating power.

In this source they show the performing results of (term frequency & inverse gravity moment) and comparison with the other model schemes which precisely measures the class A phrase and term effectiveness is validated, and compared with couple of experiments using - TF-IGM, RTF-IGM, TF, TF-IDF, TF-RF, TF-IDF-ICSDF, TF-CHI, TF-Prob schemes[1]. And their extensive experimental results proved newly opposed TF-IGM supervised phrase weighting scheme is outperforming other supervised weighting schemes.

It is also mentioned that most of SVM schemes can only be used for term distribution in two classes: positive and negative labels while measuring the weights of terms [1]. With a goal to improve that it could be distributed over multiple classes, so the term weighting would be more acceptable.

The following methods were used to classify the text, and directly comparing and benchmarking it with the labelled documents:

- *SVM* (support vector machine),
- *kNN* (k nearest neighbours),
- *LibSVM* package was also used to properly set up extensions for support vector machine and k nearest neighbours classification.

1.6. Existing datasets for text classification training

Sources specifies datasets which were gathered and used as benchmark, useful commonly used datasets:

- *Reuters-21578*, contains textual collection of 21578 news stories from the *Ruters* newswire. They were organized into 135 topical categories, which mostly refers on business and economics sources. It is widely used for textual classification, because of its number of properties that makes it a great candidate for many projects [2].
- *Data of 20 Newsgroups* dataset it contains about 20000 news grouped documents in English language. With document category topics belonging to computers, politics, sports, religions, and science [4].
- *Tancorp* dataset, containing 14150 documents Chinese language, organized into two hierarchies. Where the first one is made up out of 12, while the second one is made up out of 60 smaller categories. [5]

1.7. Efficient Text Classification

Facebook AI research published exploration for baseline of simple and efficient text classification algorithm usage. It is obtained that efficient way to classify sentences is by using Bag of words (*BoW*) with a trained linear classifier – *SVM*, a logistic classifier [8]. Thus, with *BoW* the problem occurs for modelling the architecture that they cannot share parameters inside of features. It then can limit the data demonstration of how well the model is trained and classified with forecasted or unseen data – also known as generalization. Opposed solutions to use multilayer neural networks, to use low rank matrices or to use bag of n-grams (an alternative to *BoW*).

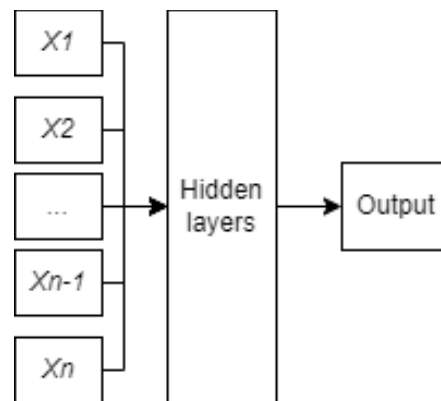


Fig.1.2 fast Text model architecture, using n-gram features for sentences [8]

Fig.1.2 displays a sample of linear model, it looks up over the first inputs matrix of the words, which are being averaged the model is being used to predict tags according to a titles and captions. Those words are then being used for a linear classifier. Architecture of hidden variable is being reused with similar cbow model. Continuous bag of words (cbow) simply explained is using sentences in the middle words and it is replaced by a label, uses words to predict the removed word in the middle. For the class determining is needed to calculate the probability of the distribution while using softmax, it can help with negative log over all classes. It is noted that it is unclear if the

given representation of neural networks is that powerful for simple shallow tasks because it is not yet well discovered [8].

Bag of n-grams collects supplemental characteristics, so there would be an additional information stored about local word order. While using *BoW* computationally would be very high cost. The computation was highly increased by using *hashing trick* to maintain the efficiency with n-gram features [8].

1.8. Bag of words

Bag of words or in short *BoW* is similar to one hot encoding, where it is required to convert text sequence into to table, where it contains as many columns as possible in a dataset for words and use count of words (Histogram of words). It is mainly used to categorize objects in images by displaying key points received from visual recognitions. The K-Means algorithm is being noted as most applied algorithm for training images into different clusters. Negative downside with Bag of words is a given sentence sequence becomes not important and we will get a curse of dimensionality. Using word processing it removes common words, stems lemmatizing words – the same words which uses different form in one word, weight the words by their informativeness [8, 18].

1.9. Technology of Web Crawler

For a gathering information from a web, there is a crawler which is also used as a program, it crawls through the internet and obtains web textual information in an automated manner.

Features of Web Crawler:

- Slow down the internet.
- Crawl and collect materials while focusing on a single topic.
- Reduce the networking traffic.

There are mainly three types of crawlers:

- A general Web Crawler – has a goal to gather as many textual, documental information as possible from a list of provided and found links. The crawler retrieves various pages from various locations.
- A focused crawler – the goal of is to retrieve pages that are relevant to a pre-determined list of topics. It only crawls the necessary areas.
- Distributed crawling – it crawls in multiple locations using multiple processes that being created to download pages from the Web.

As a result, a web crawler will continue to add new URLs to the search engine database list in a recursive manner, so it could be able to focus on multiple topics for the desired result. As said a Crawler has a primary duty: to fill the list of new website links that was found in each recursive step, by selecting a new website link for further processing [6].

Web crawler 's general workflow steps follow as given in Fig. 1.3 [6]:

1. Picking a specific website link – starting point (seed).
2. Adding it to the initialization / search list.
3. Picking the last URL link form search list.
4. Fetching all textual contents of a website that contains within the given link.
5. Finding new URL links in fetched contents.
6. Parsing all the newly discovered URLs to the search list.
7. Repeat the step 2 again, until no new website can be found.

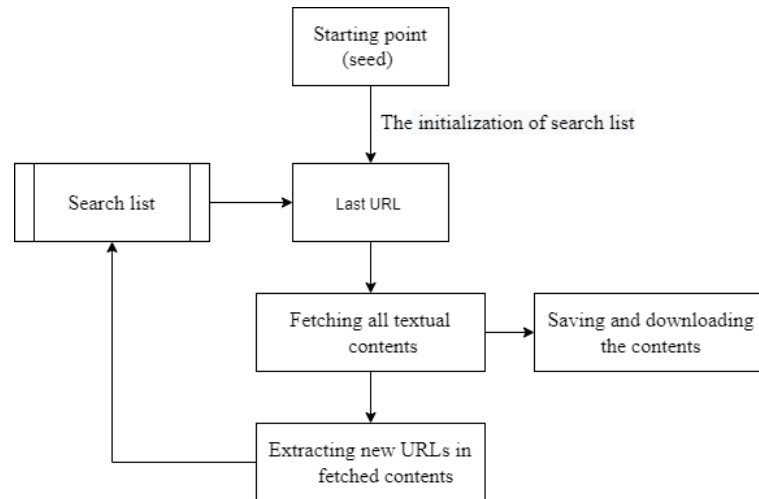


Fig. 1.3 Crawling process flow chart [6]

Also, there is possibility to initialize parallel crawler for content downloading process. With parallel crawler there could be a problem with network overload, sometimes the specific pages could include the same information thus overlapping the website or the existing content. Principal of distribution of multiple crawlers can be viewed in Fig. 1.4.

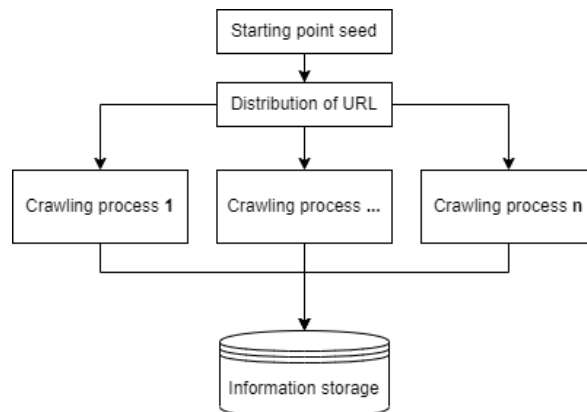


Fig. 1.4 distribution of multiple crawler processes [6]

1.10. Text classification

As mentioned, text classification is an increasing problem with a vast variety of increasing information on web pages and different format of electronically saved texts, it could also include forums, or any other posts on the websites. With a problem that a huge amount of textual information is available in HTML web form of pages, other electronic documents, and forums where people discuss about specific information. There is a proposed usage of text classification [7, p. 6], which are using machine learning methods, probabilistic models, decision trees, naive-Bayes, rule induction, neural networks, nearest neighbours, and mostly support vector machines.

The process of text classification is being specified in a flow [7, p. 1]:

1. Reading the textual information.
2. Stemming algorithm.
3. Unnecessary words delete operation.
4. Presenting document in a binary vector.
5. Selection of features, dimensional reduction of dataset including irrelevant information.
6. Usage of ML – training algorithm.

For a document to be as a feature, it needs from a sequence of words to be converted into an array of words, also can be called a vocabulary. The array of words then helps to present a document in a binary vector, defining the decimal value to 1 value if it contains the distinct word or 0 if the word is not specified [7, p. 2]. There could also be problems when streaming the vocabulary list into a vector, which are to remove initial features with misspelled words, words with no meaning, joining words. For that there is an algorithm stemmer, it approaches the given array of words and removes the words with the same first meaning of a word, called stemming. It is also considered an aggressive approach because of the importance of some stemmed words.

Selection of features aim is to reduce the unimportant features from dataset, for smaller datasets and computational calculations for categorization algorithms in further steps. The problem with text classification is a curse of dimensionality means for high dimensional data it's impossible to estimate probability densities, even if we have a huge dataset, billions, or trillions, it's still tiny compared to the whole space of possible inputs. Most real-world data is high dimensional, and from that Machine learning algorithms suffers. The benefits of feature dimensional reduction is to minimize overfitting.

Also, specific methods for selecting a feature from textual document can use evaluation functions, they can only be specified to only one word. The most common feature scoring methods, that use ranking of in dependability determination scores are presented in Table 1.1 and Table 1.2, more feature-selection metrics can be found at M. Ikonomakis [7].

Table 1.1 Feature evaluation and selection metrics fragment [7]

Feature selection and scoring in mathematical forms metrics.	
Information gain	$IG(t) = -\sum P(c_i) \log P(c_i) + P(t) \sum P(c_i t) \log P(c_i t) + P(\bar{t}) \sum P(c_i \bar{t}) \log P(c_i \bar{t})$ (1)
Conditional Mutual Information	$CMI(C S) = H(C) - H(C S_1, S_2, \dots, S_n)$ (2)
Document Frequency	$DF(t_k) = P(t_k)$ (3)
Term	$s(t) = P(t \in y t \in x)$ (4)
Weighted Ratio	$WoddsRatio(w) = P(w) \times OddsRatio(w)$ (5)

Table 1.2 Explanation of Symbols [7]

t or w	a word or a term.
C	training set class.
C	classes set.
$DF(t)$	distribution and frequency of text.
$P(c) P(c_i)$	the probability of the class c .
$P(c t)$	the probability of the class c given that the term t appears.
$\sum$	the sum of given elements.

The potential of feature extraction is also being evaluated. It includes using Principal Component Analysis (PCA), which dimensionally reduces the dataset, so there would be included reduction in complexity. Latent Semantic Indexing (LSI) – it transforms a matrix into eigenvectors it is also based on support vector machines (SVM), and combinations of k-NN LSI used.

Evaluation and measurement of the effectiveness is also required, to determine precision (chance if a randomly selected document could be recognized as c category), recall (chance that, if a document could be classified as c category), accuracy measures are applied [7, p. 7].

1.11. Representation of Text in Input Space

Opposed main steps for representing textual information in input space are:

1. Using bijective mapping for transform the term-frequencies.
2. Following vector is being multiplied with vector of importance weights.
3. Received result is being normalized into unit length.

Further about Zipf-Mandelbrot law the words power is measured, in a random given fraction of text that is ranked by the frequency, also known as Zipf law [9, 19]. It is studied with different weighting schemes for the textual information representation in input space presented as

$$\alpha(f) = \frac{1}{f^{\gamma-1}} - \frac{1}{(f+1)^{\gamma-1}} \propto \frac{1}{f^1} \quad (6)$$

Here following are: γ – positive parameter, usually a value of 2, f – frequency, α - dependency on given parameter. A frequent sequence words for Zipf ‘s law with vast sized textual data measures orders of different size, and the differences are too big in big sized datasets.

With different approaches being discussed about are raw frequencies approach, or can be called “Frequency-transformation”, it has been normalized and transformed, and usage of logarithmic frequencies with inverse frequencies. But the best information transformation in case of precision and recall was logarithmic frequency with L2 normalization. With observation that with the textual information, the higher is the given γ parameter [9].

1.12. Relevant Information Extraction

According to [21] extraction of relevant information has provided reasonable results (precision of 88% and recall of 77%). With it they had a main problem that it required to construct a composite “regular expression”, using different languages and language differences in structure difficulty. The need for them was to get the information with sentences that contains their structural integrity with a desired information.

The information extraction as per [21] being called IE, listed some varieties of possible methods that could have their use before further creation of system. It reviews shortly about [21, p. 2]:

- Domain specific machine learning, it is used to export only the information with the necessary rules. As per source it requires enough of labelled and structured data to export specific rules, it is not usable because there is no possible structuring and such information gathering.
- Rule-based approach as per source it needs to manually construct required rules for desired results. Thus, it requires complex knowledge base so the IE could possibly work.
- Open IE, a supervised learning method. It results in automatically labelled data using a parser, the arguments to use it is a domain independence (when a specific value applied to any exiting source contains only the values that can be found in the given source, or when both sources contain the exact same information). But it performs worse.

The main goal is to look for specific keywords that contains desired information not an entirety of sentences, which was the problem with their relative sources. Their initial steps were to use part-of-speech tagger with a trained Artificial Neural Network, and to detect if a sentence contains or not an action sentences they are looking for.

To attain a vocabulary while breaking down the sentences with a containing structure a basis of “Penn Tree tag set” was used, it created symbols which left a structural meaning in a sentence. Where in consideration was taken a comma and closed bracket as a word structure. And reserving vocabulary with an unexpected symbol that may occur, thus guaranteeing possibility to fully compute any type of textual information, including non-ASCII symbols. As Fig. 1.5 shows systematic breakdown of words using semantic tags, where a word can be assigned to a tag “VBD” which defines a word as a verb in past tense structure [21]. It led to reduced full vocabulary that contains only 50 different “words” and a viable solution to reduce a “curse of dimensionality”.

733 0 Have you	733 0 VBP PRP
always wanted to	RB VBD TO
impress people	VB NNS
with your art?	IN PRP\$ NN .

Fig. 1.5 Sentence structural breakdown [21, p. 3]

The reduced vocabulary then was used for one-hot encoding, structuring them in a matrix and being used as an input, flattened, and following by that it was put into two hidden layers and output. Entire process is given by source in Fig. 1.6 [21].

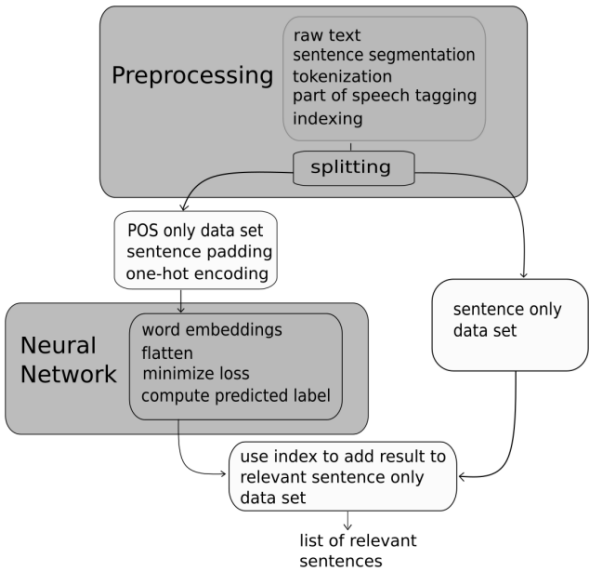


Fig. 1.6 Automated flow of initial preprocessing, and structural adaptation [21].

1.13. Automated Information gathering from Web.

Agents engaging with Web search engines perform queries like those given by human searchers, according to the source. They are consistently submitting brief questions. Interaction Frequency and the Duration of operation, search engine with the agent takes the time to gather information for several hours, while using multiple processes and browsing multiple pages. Agent’s task is to seek for the specified information with a given keywords. While they are looking for an information, the Agent seeks through data that has from 18 percent to 37 percent of unique terms [10].

1.14. Presence of the solutions

Below are being discussed about currently existing solutions on information collection, with different systems and useful tools for existing problems.

1.14.1. Employed people

Small distributors/resellers who employ their workers or themselves or freelancers who are specifying products description with extra details, including compelling descriptions. Presence of current solution is generally small distributors and resellers what they do is search on search engines for the information on other websites. By hiring free lancers to look for details about any specific product. Also, it is needed to keep in mind that the descriptions can also have different languages and it requires translations. But mainly if the product contains different specifications, it gets harder to describe what is real and what is not in the product specific information.

1.14.2. Existing systems

Systems with e-commerce automation: *Gepard PIM*, *Productsup*, *Plytix*, other websites that do not publish their usage of automated systems.

They include specific costs into automating the product descriptions, and the prices also depends on the website and the amount of goods you have in e-shop.

The advantages they provide are:

- Reduced person time consumption
- More data to work with
- Reduced translation work
- In short term it provides significant savings

These solutions do not have their publications, which proves it is not yet completed as a publication or solution to this kind of problem.

1.15. Schema.org usage for training product matchers

It is source and organization are using *Schema.org* to train and help with the matching part of the products. And in the study, it is showed how it might help with the use of website annotations [32]. The article demonstrates how manual labelling of read data, which is then used for identifying product approach – could be replaced based on *Schema.org* annotated collections from the web.

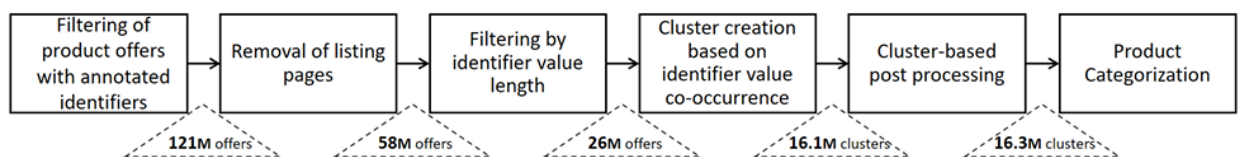


Fig. 1.7 Cleansing Pipeline that is used to group data offers into clusters [32].

Also, there is a reason for such application, that it provides useful compliance advises, lets to achieve in depth training and to maintain various fine-tuning parameters more easily, with existing or invisible products. So general usage would be to use the preexisting annotations which is gathered and being used for training the algorithms.

Fig. 1.7 displays abstract view of the grouping part. Where they filter all the products offering with annotated identifiers. Identifiers used to filter are with following entities: *sku*, *mpn*, *identifier*, *productid*, *gtin14*, *gtin13*, *gtin12* and *gtin8*. After that removal of listing pages, where it contains unscary words, removal of advertisements. Filtering by identifier value length, it removes records

by inspecting *ean*, *sku*, *isbn* values. So, this source covered that it is possible to better categories the specific products on the variables it can get supplied using the *skima.org* tool.

It obtains accuracy of categorization with given annotations with 0.9 F1 validation score using *Deepmatcher* model and obtaining for unseen data 75.9 F1.

1.16. IR system in text-based document using vector space model

Information retrieval system refers to finding all knowledge based on the query it was send with, in this case the text is needed to be associated with the query. The principal is that there is a collected collection of documents and the user who creates requests, and the system aims to answer the questions with information sources available. Its architecture consists of documents of query - information handling part, to search and preprocess the model.

There is also so called VSM it is represented as a matrix of terms, method used with it are TF-IDF to determine proximity or similarity of terms contained in document. And documents are viewed as a vector. Vector has a distance and direction It is represented in vector space. And the relevance of document is based on similarity between document vectors and query vectors. The closer the document is to a term, the closer are the VSM documents, and usually between the it can be calculated the similarity of *CosineSim(q,dj)* [33]

$$Sim(q, d_j) = \frac{q * d_j}{|q| * |d_j|} = \frac{\sum_{i=1}^t W_{iq} * W_{ij}}{\sqrt{\sum_{i=1}^t (W_{iq})^2} * \sqrt{\sum_{i=1}^t (W_{ij})^2}}$$

Where d_j is the similarity of textual represented information, and q is a search query. The main disadvantage from our perspective that it contains the documents collected beforehand in pdf formats, and it searches for relevant information using and the query, but by contaminating it into matrix term-document vectors based on searched information and measuring similarities between other documents with query similarity [33].

1.17. Named entity recognition

With conditional requirement fields (CRF) it is possible to be an entity of a text that could be later used and recognized as similar entity. CRF is sequence modelling algorithms which is used to effectively tag with a model specific feature. There is possibility to use POS tagging, but it is being limited when there are strong dependencies across different output labels. So, the source of L. Guillaume [25] selects to use model tagging as independent combined with conditional random field. Where the input of a sentence is [25, 34]

$$X = (x_1, x_2, \dots, x_n)$$

P (size of $n \times k$) will be considered as a matrix of scores that is calculated by bidirectional LSTM model, where k is a count of unique tags. P_{ij} will represent the score of j tag in i word of a sentence [25]. By defining the to be as a tagger score equation

$$s(X, y) = \sum_{i=0}^n A_{y_i, y_{i+1}} + \sum_{i=0}^n P_{i, y_i}$$

Here A – matrix of transition scores, of size $k + 2$. A_{ij} presenting transition score from one tag as i into j tag. y_0 between y_n describes start and ending tags included in a order [25]

1.18. Possible Approaches (discussions)

Agnostic approach, Google inquiry. Sentences or paragraph base data. Entity recognition. Transforming a text, of graph states. REGEX. Ideas vector space model (*tf-idf*). Hybrid search engine (image and text). Multimodal information retrieval. Transformer – based model.

Research ideas:

- The score measuring in it where the word is in specific entity is, regression problem.
- Relevance score or probability that is scaled.
- Knowledge graph approach.
- Downstream – using agnostic approach, to build a model that predicts a number that is it such a product, model.
- Binary problem is it in an entity or not.
- Regression could be also used as a continuous variable where relevance score or probability is scaled for entity recognition it is possible to use classification.
- Unsupervised classification – topic modelling a automatic method trying to detect topics without user interruption
- Clustering method.

1.19. Tools and software analysis

In this part the part of review and the existing technologies will be covered. Which are planned to be used in this scientific research project. Some parts of the tools will be library-extension based and others will be applicable as a full module for constructing full program.

1.19.1. Automated information gathering from Web tools.

Crawler – is also called *robot* or a *spider*. It is a tool – which collects textual or documental, information, it autonomically searches for other websites and discovers them, it scans websites by following included links in the given webpages that can be unseen. Tools that can search for: Content, Links, Broken links, Sitemaps and HTML code [16]. Which are used by search engines to index downloaded pages, such as Google, Bing, Yahoo, DuckDuckGo.

Tools for scraping textual information form websites are provided: *GoogleBot*, *BingBot*, *Slurp*, *Bot*, *DuckDuckBot*, *Baidu spider*. Tools could also have some problems with pulling or overloading a specific website. If such a problem occurs, it can block the connections if a program will require to access it. There are some specific available open-source tools to scrape the web pages for data mining.

1.19.2. Scrapy Tool

Scrapy Tool – is a web scraper is a tool for a problem of extracting a lot of unstructured data and storing them in preferred structure/format. Its preferred use is for storing and gathering the data from different websites. It is used for data mining, structured information processing, speed, parallel search. It is a framework that provides solutions to perform every little task for data gathering [17].

1.19.3. Selenium

Selenium is known as an automated web testing program. Able to simulate the human behaviour to obtain data from the webpages. Its behaviour is meant for browser automation, settled by programmer, it can be used together with a browser to mirror the user. Meant also to be used as a web scraping tool. Table 1.3 concludes that it is not meant to be used for scraping. But the browser automation is still to be kept in mind, with says that it is known as a strong automated scraper [26].

Table 1.3 Selenium WebDriver for and against [26]

Pro	
Information retrieval with WebDriver.	Interacts as a human being. Access the websites without being blocked as a robot.
Resource support.	Supports JavaScript, images, CSS. Saves HTTP headers. Can extract dynamic context.
Browser support	Supports Chrome, Firefox, Opera, HTML Unit and Android, iOS browsers.
Contra	
Size	Size of the program becomes large. After linking all the needed Web Drivers. It can easily grow from kilobytes to megabytes.
Scraping	Needs to lead the whole website which makes the process tedious. Scraping could be also detected using Google Analytics

1.19.4. Modern web scraping architectures and techniques

The source covers useful tools, which help to approach quite a large and a vast amount of data using for example *BeautifulSoup* with *Python*, *rvest* for R, and others. But it can collide with difficulties such as some web pages has policies to block web scrapers. The tool is also called headless browser, which mainly requests the HTTP for textual information from system that is built mainly on Angular or React system.

Approaches to be aware of [35]:

Pages or websites have a policy reason to block accessibility by scrapers because of human authentication requests, webpages accessible only after entering a password, pages with infinite scroll. A stated fact it is mentioned that there will be a possibility to harm business when the data can be extracted by the crawlers and then it would be used for competition purposes and prices. A walk around for these would be to use a specific script which interact with user interface. Pages can ban IP's from requesting data.

The source proposed a basic architecture for the system, which will accomplish the objective for a user who wants to collect additional data. It shows that you need to keep in mind the domain objective structure, approach data extraction, structure it after all the data was downloaded from main page and then it iterated by going through other values [35].

There are also some *Semantic webs* allowing to implement and permit web scraping with a *Schema.org* which is a part of google research community, which has meta based vocabulary for structuring web components.

1.19.5. *BeautifulSoup* web scraping

It is a HTML Web scraper used as *Python* programming language API, which allows you access the data in constructed manner. A convenient way to approach for relevant information, you need to reparse. As a textual information there is too much information [29]. Its advantages and disadvantages displayed in a Table 1.4. Tool can complete the automated actions flow as:

- collect all URLs from web pages so the search locations would continue further on.
- Query control, goal to not overload with too frequent requests, to prevent IP bans.
- This tool can convert HTTP directly into JSON and XML formats because the main purpose of HTML is graphical display, and *JSON*, *XML* main purpose is structure which contains textual information.
- Able to build a script for different pages.

- Possibility to monitor the scraping process in CMD line.
- Creating your own application with it.

Table 1.4 Given *BeautifulSoup* parser for and against [28]

Parser used as	<i>Pro</i>	<i>Contra</i>
<code>BeautifulSoup(textI,"html.parser")</code>	Comparably fast.	Slower than <i>lxml</i> . Less flexible then <i>html5lib</i> .
<code>BeautifulSoup(text,"lxml")</code>	Very fast. Flexible.	Is C dependent.
<code>BeautifulSoup(text,"lxml-xml")</code> <code>BeautifulSoup(text,"xml")</code>	Quick. Only supported XML parser.	Is C dependent.
<code>BeautifulSoup(text,"html5lib")</code>	Easy learning curve – flexible. Concentrates websites in a way websites do. Generates valid HTML5	Quite slow. Python dependency.

1.19.6. Programming language tools

Python is a high-level programming language choice that is well suited for data analysis and application is the interpreted language tool *Python* [30]. This dynamic programming language and its tools are attractive for fast application development, mathematical computation, with additional libraries suitable for the programmer, licensed for full use, and suitable for scripting if certain automation support operations are required. Has variety of with supported modules and various packages to encourage code reuse (see **Table 1.5**). The programming language is readily available, open source and free, for all major platforms, with the possibility of free distribution.

Table 1.5 Usage of *Python*.

Installation.	Price	Packages and libs.	Machine learning.	Support
Supports almost all OS's.	Open source, possible for free usage.	<i>Scikit-learn, apandas, tensorflow, keras, pytorch, other</i>	Supports many ML libraries and flexible.	Constant updates, libraries support C/C++, possible migration to C.

1.19.7. Google Search API

There is a possibility to build your data collection of google search inquiries. With its own advantages, that it helps you with the keyword selections, and all support for JSON data formats, with quite flexible google documentation. Let's to send RESTful queries. Has autonomous support, with *Python*. Disadvantages it has no control over what is indexed. Only google decides what pages are crawled and which are not. No possibility to set destination of crawling. It has almost no support and you cannot implement or modify the tool yourself. Can be shutdown at any given time. Maybe the only advantage is that it is legally automated to use in scraping webs [26, 27].

1.20. Auto encoder decoder

Traditional autoencoder is mainly used to encrypt and decrypt itself and belongs to a family of dimensionality reduction methods using artificial neural networks [40]. It contains reconstruction error measurement $\|x_j - x'_i\|^2$, where the distance between x_j and x'_i is measured according to WANG, Wei, et al. Generalized autoencoder source. Each reconstruction of error $s_{ij}\|x_j - x'_i\|^2$ is then measured to calculate their weighted distance [40]. Most of Autoencoders consists of 2 parts, encoder and decoder. Between encoder and decoder is the lowest dimensionally reduced data, which then can be used for processing different algorithms.

1.21. Siamese Network

One-shot classification models say that it is sufficient to have only one training example for each class if it is required to predict image pairwise similarity. Because the model is trained on several examples, but they only need to be from a similar encoded root. In GUPTA, Harshvardhan article an example with facial recognition is given. It is trained by a single sample classification method on a dataset containing photos of people from different angles, lighting. To understand basic structure of Siamese network view fig. Fig. 1.8. In order to classify whether a photo contains person X, a single photo of a person can be passed to the model and the Siamese model can be queried to determine whether that person is in the photo, important part is that in the Siamese network the model has not been trained on any photos of person X [37, 38].

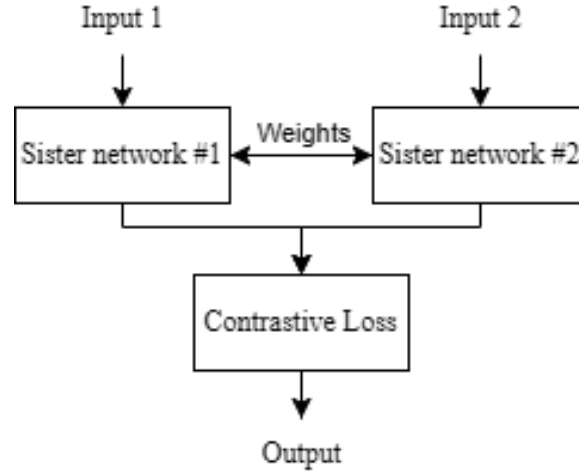


Fig. 1.8 Siamese Network architecture [37].

Siamese Network model has a special place in these measurements where it does not learn to classify its inputs, but it learns differentiate its inputs similarity [37].

In the “Dimensionality Reduction by Learning an Invariant Mapping” by R. Hadsell paper, the authors introduce the function of a contrast loss function, which we are going to be using for Siamese Network model training as a loss function. Function goal is to ensure the mapping from a high-dimensional input space to a lower-dimensional output space by preserving relations between neighboring points [p. 3, 39].

$$L(W, Y, \vec{X}_1, \vec{X}_2) = (1 - Y) \frac{1}{2} (D_w)^2 + (Y) \frac{1}{2} (\max(0, m - D_w))^2, \quad (7)$$

Where L – contrastive loss, W – parameters of function G_w (fully connected neural network), Y is a binary value label denoting similar ($Y=1$) and dissimilar ($Y=0$), D_w – Euclidean distance between $\vec{X}_1, \vec{X}_2$, m is a margin parameter. This function penalizes the model for having dissimilar pairs of data points too close together and similar pairs too far apart (view equation 7). The function sets a limit on the acceptable distance between points in the output space by including a margin parameter. This approach allows the model to learn a representation that reflects the underlying structure of the data and preserves important relationships during the dimensioning process [39].

1.22. Sentence-BERT

SBERT network is an altered version of BERT network, it uses Siamese and triplet network to find semantically meaningful sentence embeddings. It can complete tasks such as semantic similarity comparison measurements, clustering, sentence encoding, and information retrieval using semantic similarity search SBERT. It states that it significantly reduces computational time required to find similar sentences while keeping it accurate [36].

SBERT network is characterised by its Siamese and triplet network structure, which processes semantically meaningful sentences. SBERT network claims to reduce the computational effort required to find similar sentence pairs from 65 hours using the classical BERT model to around 5 seconds [1p. 36].

SBERT uses the BERT/RoBERTa output union operation to create sentence embeddings. The source carried out research involves three main merging strategies: the CLS-token output, the calculation of the average of all output vectors and the calculation of the maximum sum of all output vectors. The default configuration of SBERT is the MEAN processing strategy [36].

2. Automated e-commerce description gathering solution specification.

This section focuses on project planning and aims to ensure quality standards by addressing both functional and non-functional requirements. It covers the identification of necessary tools for solution development, as well as providing system projection details, including utilized algorithms, methods, and workflow diagrams.

2.1. Methods and tools used to implement e-commerce description gathering system.

The implementation of the agent will require web scraping tools and a combination of algorithms needed to recognize product information entities.

The project will be realized using Python open-source programming tools, which contain various libraries that support different machine learning tasks. These tools offer extensive support for text mining, web scraping, and text editing. They also provide diverse options for data storage and structure usage. For the automated description of products, we will be using the Python language along with additional tools:

- *Pytorch - Jupyter Notebook*– Python integrated development environment.
- *Scikit-Learn* - library meant to effectively apply new data and analysis. Open-source code containing multiple usage methods. Usage is mainly based on data classification, data analysis, regression analysis.
- *SciPy* - open-source library including powerful mathematical, scientific calculation methods.
- *Pandas* - programming extension used for data structuring, usable for data analysis purposes.
- *NumPy* - library for programming and supporting massive matrices, mathematical calculations, and functions for data management.
- *Matplotlib* - data visualization extension, used to visualize graphs, graphical tables, histograms, pictures.
- *Pickle* - Python module extension meant for saving and to reuse already trained machine learning finalized model. With an objective to trace and follow objects, and it would be reusable without additional calculations.
- *BeautifulSoup* – is intended to be used as a web crawler, our project will gather various pieces of information from HTML and parse them into XML/CSV documents. The web scraping tool will create a parse tree of the parsed documents, facilitating the extraction and organization of the data.
- *Scrapy tool* - an open-source web spider program framework, combined with Python, used to extract structured HTML data.
- *Google Search API* - search results from *Google* will be as initialization for first steps for web crawling tool. Also, to keep in mind that this tool is limited to 100 requests per day.
- *Network access* - agent must collect data by using internet service provided access to publicly available information on websites.
- *spaCy* - open-source library that would be used for using NER – model.
 - Alternative *NLTK* - Natural language tool kit, pretrained or non-trained model.
- *PyTorch* - is an open-source library used to build and train neural network models.

2.2. Project plan

In this research, a combination of several algorithms will be used to recognize products within the information gathered from websites. This solution will combine publicly available website information, which is crawled using tools like Scrapy and *BeautifulSoup*, with specific initial data provided by the user to gather detailed product data. The dataset, referred to as a document, will undergo data structuring steps.

Agent pipeline will consist of these listed main algorithms:

1. Combination of SBERT encoding with Cosine Similarity Embedding
2. Combination of TF-IDF with Cosine Similarity Embedding
3. Named entity recognition (NER)
4. Combination of Autoencoder-Decoder using bottleneck to output Cosine Similarity measurements
5. Combination of Siamese Network, Contrastive Loss, and Euclidean distance between NN output. Depending on the algorithm used, TF-IDF is based on measuring text by vector space. Collected documents will be compared using cosine similarity function based on user-specified information. It relies on the similarity between document vectors and query vectors. The closer the document is to a term query, the closer the documents are, and usually, their similarity can be calculated using the cosine similarity of $\text{CosineSim}(q, d_j)$, here d_j is the similarity of given document, and q is a textual query information.

NER –consists of conditional requirement fields used to train a model on tagging. It combines independent tagging with conditional random fields (CRFs).

Achievements are related to the number of documents, textual information, and extracted descriptions of related products. These are found from publicly available informational websites. The information relates to the extracted possible descriptions, discovered through similarities with the product name (query), and named entity recognitions.

The textual information would not exist without given queries, indicating the product name and its parameters. Due to this potential barrier, it is necessary to access web-based information. Some information in future steps may require the use of virtual private networks (VPNs) to access additional hidden information, blocked by VPN barriers. Until then, the information will be accessible only within current policies, allowing us to reach website information by scraping HTML – XML documents. The tools capable of accessing this information will be based on given stationary system parameters, internet connection speeds, processing power, and available computer space.

What we need to gather depends on the user's requirements, because the agent applies search queries based on those specifications. The search queries collect various unstructured information, which the agent preprocesses and extracts various parameters related to product descriptions.

Accuracy, based on nothing, cannot be calculated. Therefore, we will approach it in an agnostic way, where we measure relevance. Relevance refers to the number of times a description has appeared in different documents. If it contains the same information as desired for the product, the relevance increases; if it lacks the desired product specifications, the relevance drops.

This research on the offered pipeline has not yet been applied or implemented for searching exact product descriptions. Therefore, measuring improvements is not currently possible. However, we can utilize the opportunity to analyze the full pipeline algorithm.

2.3. Use case diagram.

Below we are listing our system, or it can be called as agent use case diagram, which consists of 6 use cases. Use case can be seen in figure below (see Fig. 2.1), the automated product description using system modules such as – selection of product and its entities, indication of format and location of storage, calibrate the crawling process, extract crawled web information into structured format, detect and classify important product entities, and final is carrying out the product specifications in designated location and format.

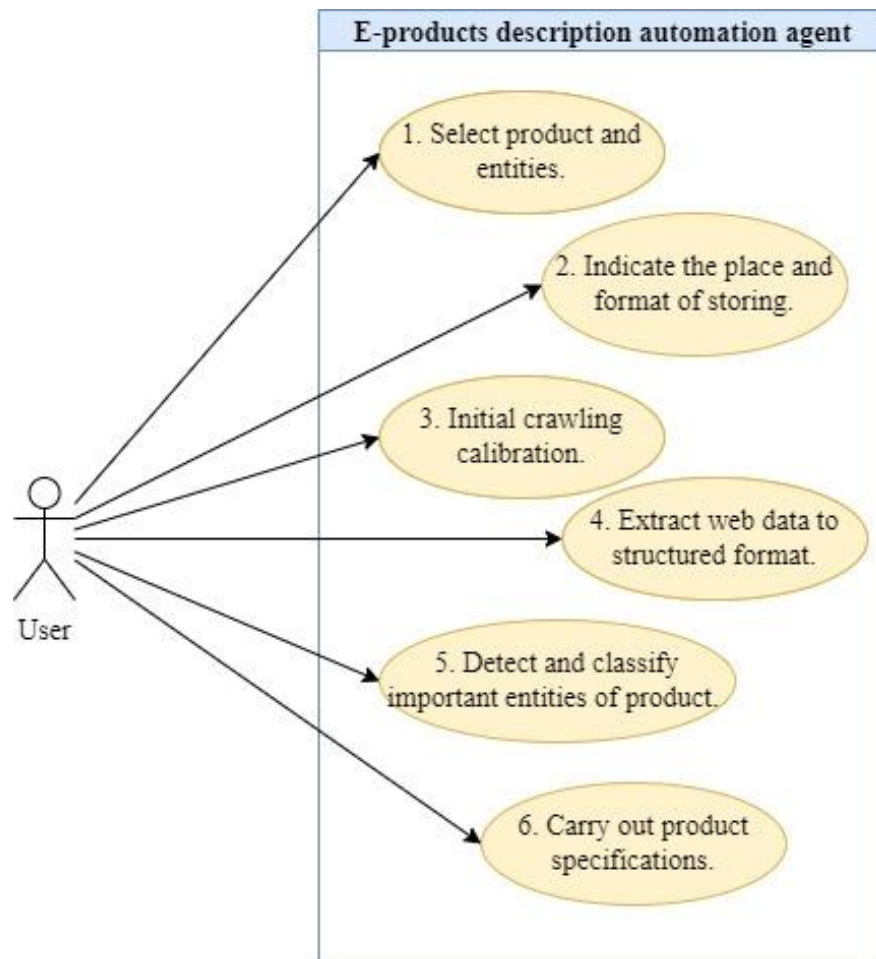


Fig. 2.1 E-products description automation agent use case diagram.

Further on, following explanations of tables from the use case diagram, including their corresponding activities, are presented below the diagram. Refer to Table 2.1 for the product entity selection use case.

Table 2.1 Select product and entities use case.

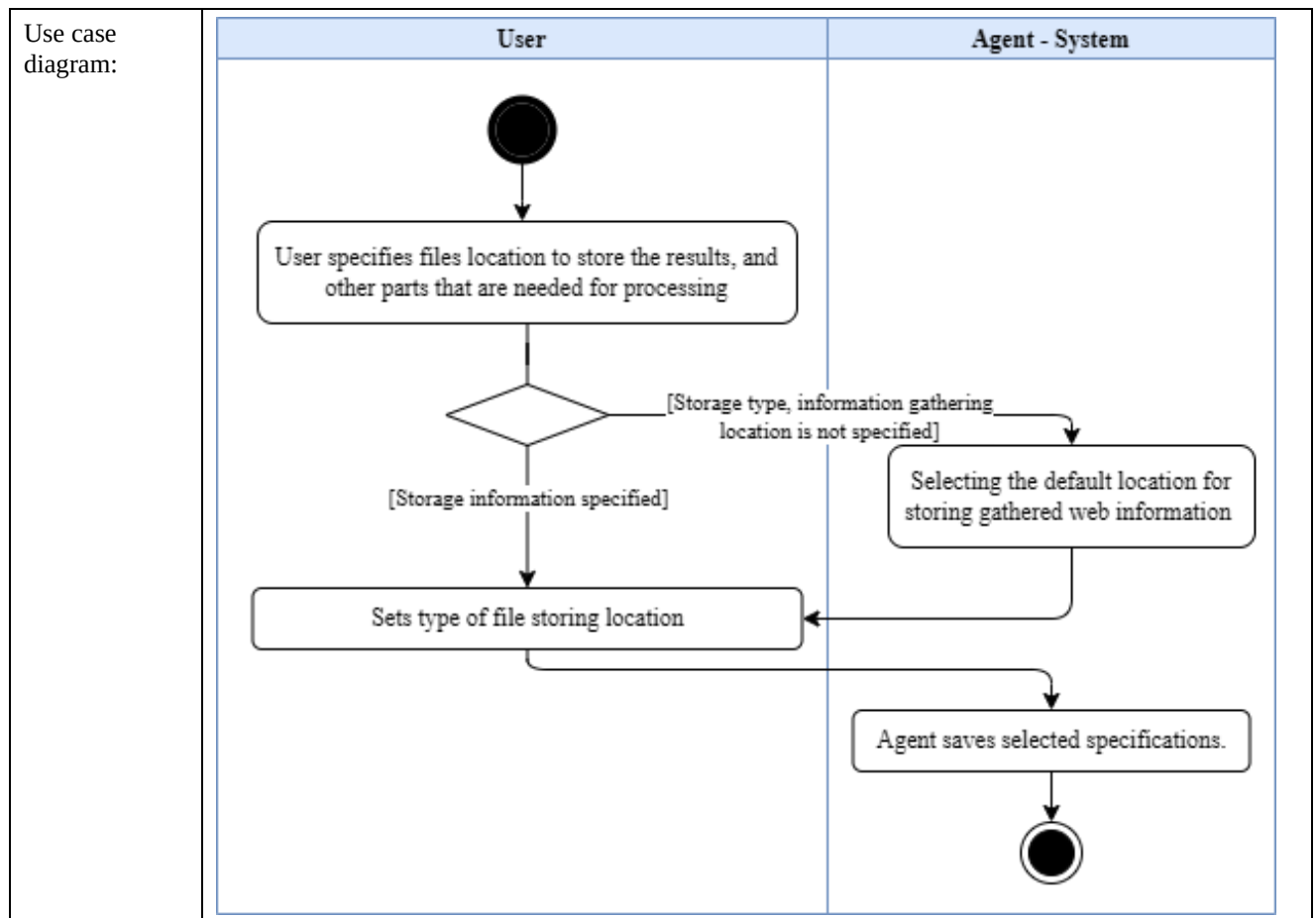
Identification:	UC-01
Use case:	Select product and entities.
Goal:	The starting point initializes the further steps with search parameters. Based on these parameters, the agent will gather specific product specifications
Description:	The user sets the product, which could be a product type or product name. Then, the user sets the product entities that they need to gather.
Before condition:	The user launches the agent and prepares the product name and descriptions that it needs to gather
Actor:	User.

Trigger condition:	The user begins their work by specifying search parameters
Agent response:	Saves the parameters for further web crawling steps and notifies the user about changes.
Mainstream of events:	The user sets the product name or type. Then, the user specifies the product specification names in a list.
The use case is closed.	
After the condition:	Agent successfully continues to 2 nd step of use case diagram.
Use case diagram:	<pre> graph TD subgraph User Start(()) --> U1[User specifies name of the product in a 1st field] U1 --> U2[User specifies n product names of specifications] end subgraph Agent_System [Agent - System] A1[Agent saves selected specifications.] --> End((())) end U2 --> A1 </pre>

In the table listed below, the actions describe what is needed to indicate the place and format of storage. (see Table 2.2).

Table 2.2 Indicate the place and format of storage.

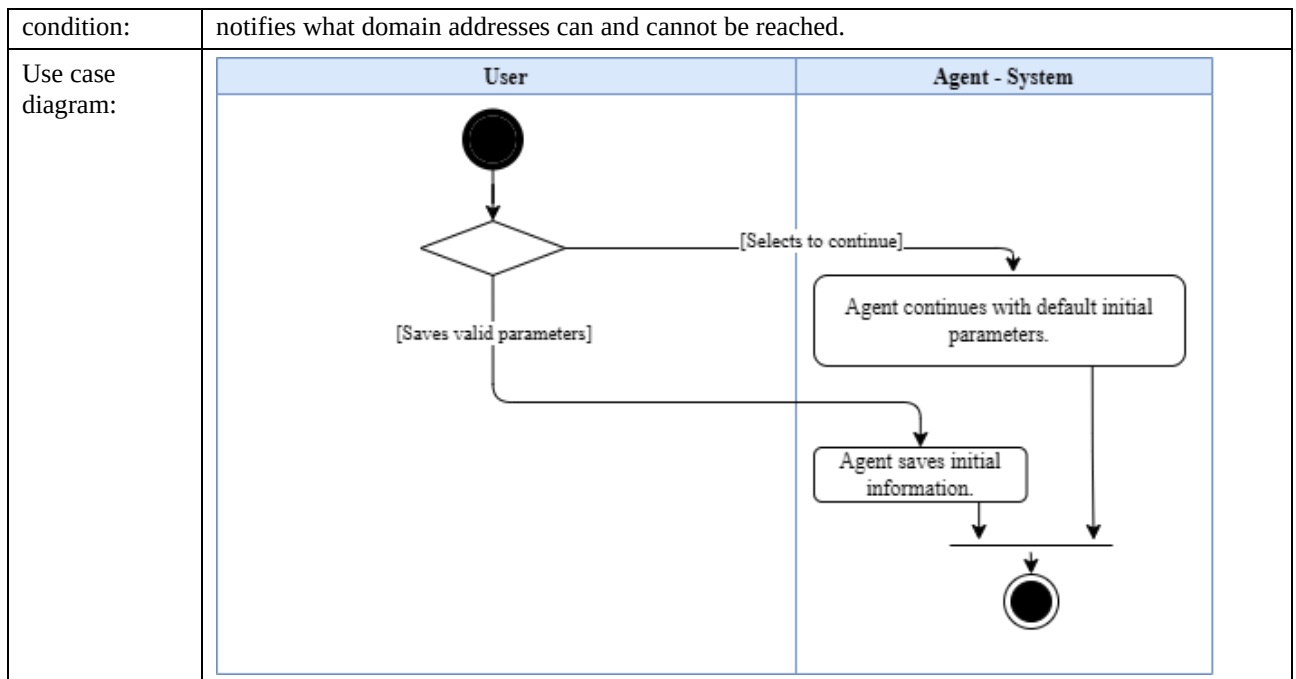
Identification:	UC-02
Use case:	Indicate the place and format of storage.
Goal:	Clarify the system parameters for saving files, including where to save the file and what file type to maintain.
Description:	The user selects between given choices of file formats and specifies the save location. The goal for the agent is to maintain the file format correctly by completing additional information
Before condition:	The product search entities were set. User wants to select file type and location of saving.
Actor:	User
Trigger condition:	User has saved product and its search parameters.
Mainstream of events:	After saving the parameters of the products, the user selects their file saving locations and its format, which the user can inspect it
Agent response:	The user saves the parameters into the desired system location and prepares data for the specified location. The user is then notified about the saved changes.
The use case is closed.	
After the condition:	Agent saves initial crawling parameters.



The table listed below specifies actions of the use case diagram needed for the user to calibrate initial agent settings (see Table 2.3).

Table 2.3 Initial crawling calibration use case.

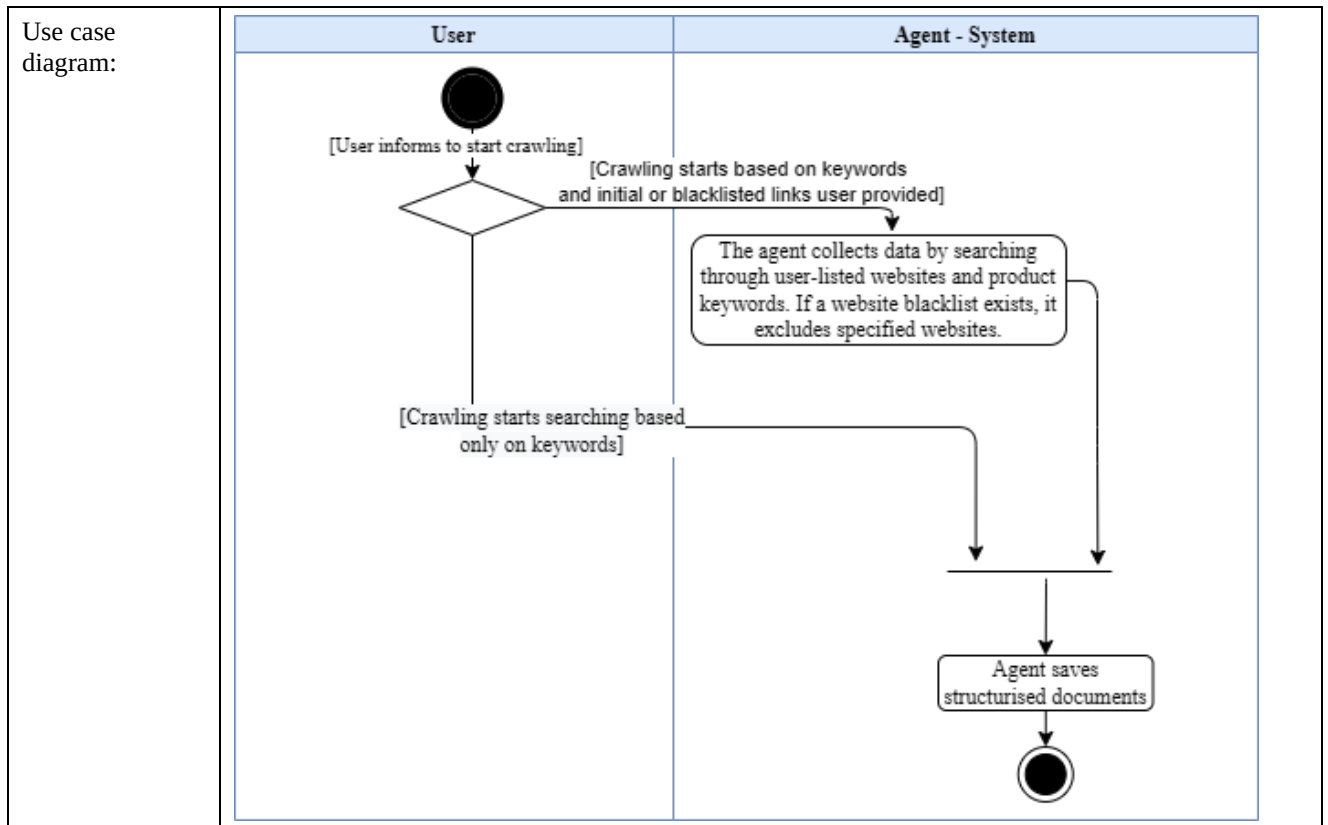
Identification:	UC-03
Use case:	Initial crawling calibration
Goal:	To set starting points needed to crawl web information, allowing the user to decide whether to fill up initial information which would help in further crawling steps, and to let the user specify additional agent acting styles.
Description:	Let the user specify the locations of domain addresses that describe additional locations to search.
Before condition:	The product type, entities, and file types were selected. The user wants to continue with the calibration of web search
Actor:	User.
Trigger condition:	User selects agent initial calibration steps.
Mainstream of events:	User selects web crawling additional and initial setups, sets what web pages to search, sets the limit of time to search for information, specifies what web domains to ignore, specifies limit of websites, limits of pages to cover per one web domain. Limits the HTTP queries and document download count per second from a single domain, select how many CPU cores to use for crawling different webs.
Agent response:	Saves specified information, if some parameters were not set it sets them as default values which are described before.
The use case is closed.	
After the	System informs about saved and default parameters of the selected agent initial parameters. It



The below listed table contains use case description containing web documents extraction to structured format (see Table 2.4).

Table 2.4 Extraction of web data contents to structured format.

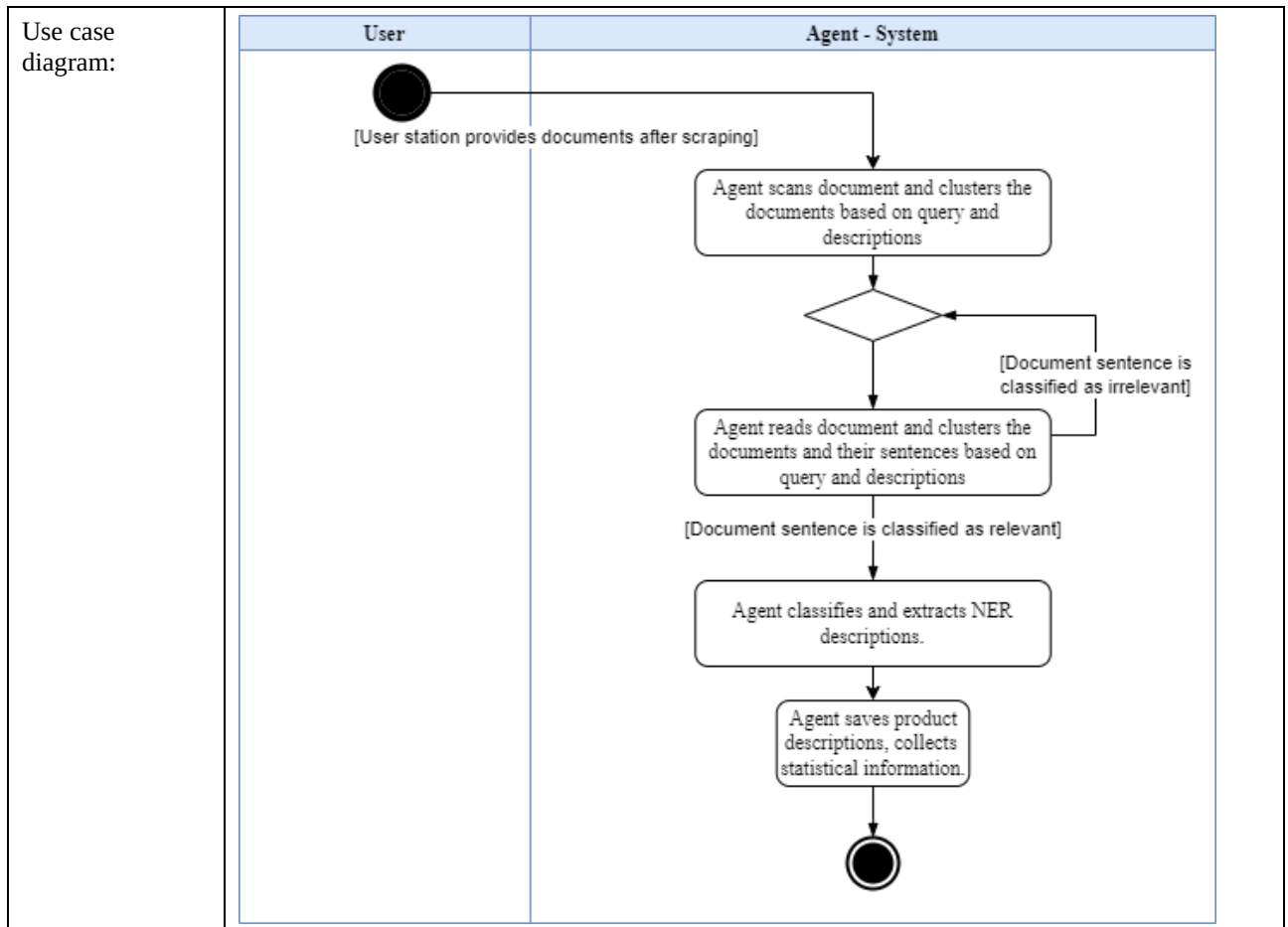
Identification:	UC-04
Use case:	Extract web data to structured format.
Goal:	Goal is to crawl websites, with a goal to find related information containing user defined parameters.
Description:	It is automated process which takes user initial parameters, and searches for related information contained about the product. This is a web crawler process which gathers information, and structures it from HTML format into CSV format.
Before condition:	The user must be saved all the initial settings and parameters of specific product, and with saved initial parameters of the agent.
Actor:	User
Trigger condition:	User sets to begin agent search.
Mainstream of events:	Agent must get command to start searching.
Agent response:	Agent searches through the required parameters and related information in websites. If the website is related to the topic it decides to take a step in saving structured user data which will be later processed in other steps. Every step will indicate related information to a user.
The use case is closed.	
After the condition:	After scanning and crawling every website, it collects data, which is then structured into relevant information to minimize the amount of space taken. After all scans are complete, it saves the structured documents locally and continues to entity recognition.



The table listed below contains actions in the use case needed for detecting and classifying important entities, specifically descriptions of the product (see Table 2.5).

Table 2.5 Detect and classify important entities of product use case.

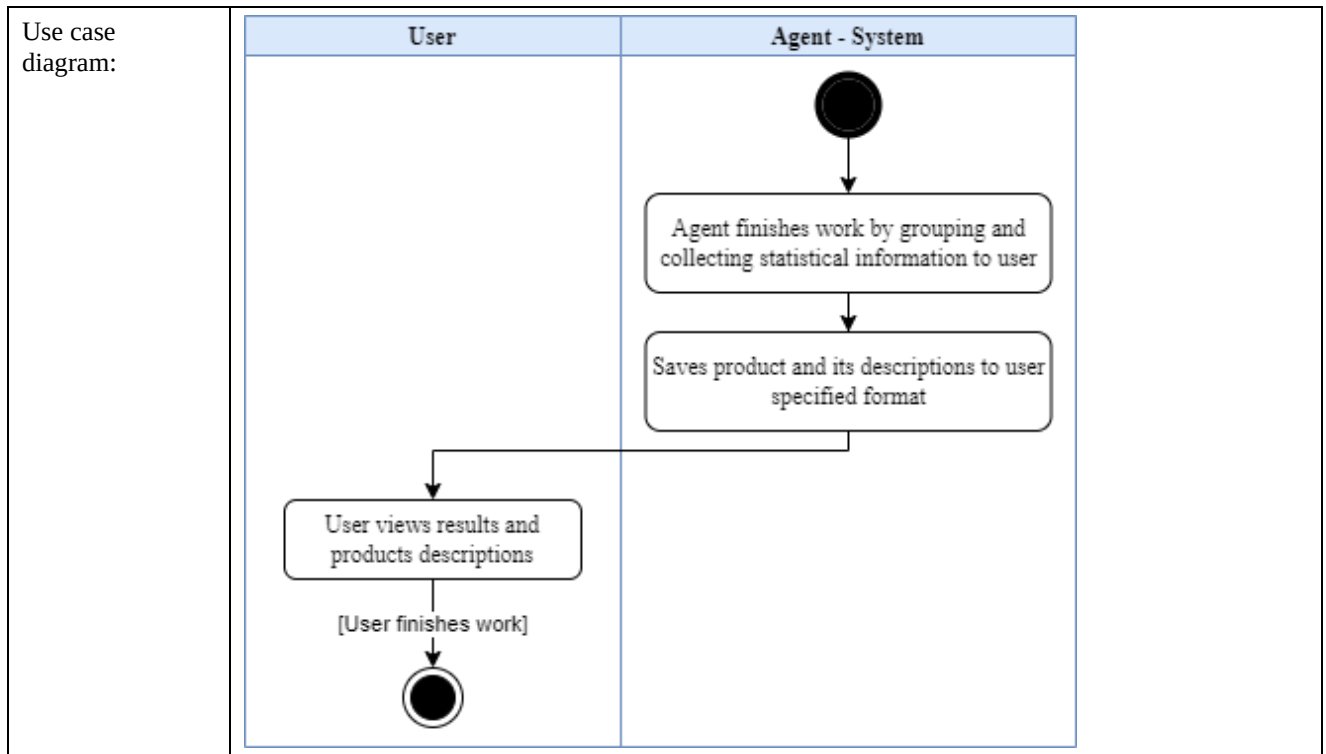
Identification:	UC-05
Use case:	Detect and classify important entities of product
Goal:	Recognize selected entities parameters.
Description:	After the scanned documents were saved it is going to be analysed by entity recognition algorithm, agent and his algorithm scans through the given structured documents and outputs the available parameters that has been stored in users local drive. Entities will be recognized based on user specified descriptions, and it collects every entity description it has found based on user description.
Before condition:	User decides to finish crawling websites
Actor:	User
Trigger condition:	Scanning, crawling of websites is finished.
Mainstream of events:	It will be initiated by the user. After the scanning is finished the agent approaches the structured data and uses it for automated description collection. Every document will extract [0; n] parameters, where n is any possible count of named product parameters. The algorithm will scan all the documents, in the end providing the answers to next step of carrying out product specifications.
Agent response:	Outputs information about ended process to the user.
The use case is closed.	
After the condition:	Continues to procedure of answers after finishing with textual named entity recognition algorithm.



Below table specifies the use case actions required for carrying out specifications and descriptions of products (see Table 2.6).

Table 2.6 Present product specifications use case.

Identification:	UC-06
Use case:	Carry out product specifications.
Goal:	Provide user with statistical and product description information, save final results.
Description:	Goal is to represent data to user, that has been collected. Save it to user device. Prepare the algorithm collected data with relevance score. Parts of algorithm that has been used on specific documents, information what and from where it was gathered. Display structured data.
Before condition:	The recognition of specified parameters was recognized.
Actor:	User
Trigger condition:	User requires to overview analytical analysis of collected data.
Mainstream of events:	User requests to display gathered information, user then receives a sheet of his selected format document, containing relevant gathered information about the specific product.
Agent response:	It represents analysis, of scanned time, scanned locations, request counts it made, failed requests, blocked requests, statistical relevance of different named specifications. Use of algorithm and its successful recognitions of gathered additional information.
The use case is closed.	
After the condition:	User is able to stop the agent workflow, and to view saved results in separate location.



Overviewing the use case diagrams, we can describe the system workflow, and we can specify what needs to be done with the most effort. It is seen in use case, the combination of web data extraction and detection, classification, extraction of textual information required descriptions about product is the most focused part in later steps.

2.4. Functional requirements

For the solution to be developed, the following functional requirements are given in table down below (see Table 2.7):

Table 2.7 Agent functional requirements.

	Functional Requirement	Related Use case
1.	Automated collection, preparation, and processing from web-based information.	UC-4, UC-5, UC6
2.	Correctly identify object in given textual structure.	UC-5, UC-6
3.	Recognition of the specified entity from textual information.	UC-5
4.	The agent must be adaptive and allow the user to select the desired product search parameters, its descriptions, and the initial web browser search parameters.	UC-1, UC-2, UC-3
5.	The agent must report the obtained results and provide an explanation – statistics, relevance.	UC-5, UC-6
6.	The agent must be able to collect relevant product information using already scanned documents and be able to process the workflow through a new flow.	UC-5, UC-6

With these functional requirements in place, we can build on the agent and system algorithm for the development workflow.

2.5. Nonfunctional

For the solution of agent combination with an algorithm to be developed, the following nonfunctional requirements are given in table down below (see Table 2.8):

Table 2.8 Agent - algorithm nonfunctional requirements.

	Nonfunctional Requirements	Related Use case
1.	Agent must collect as much related product specifications data as possible over established amount of time.	UC-5
2.	Agent must work stable, reliable, and fast.	UC-1, UC-2, UC-3, UC-4, UC-5, UC-6
3.	Clarity of agent presented results and statistics.	UC-1, UC-2, UC-3, UC-6
4.	The agent must keep the rest of the selected format.	UC-1, UC-6

Based on these selected non-functional requirements, an algorithm-agent usage specification is developed.

2.6. Targeted quality criteria

By completing the tasks of the research, the combination of multiple system components will be used to detect product specifications, also known as entities. Combination of different modules will require modifications, and each realized component must meet certain criteria:

- Agent system and its algorithms must work with different available data and extract product descriptions.
- Agent should efficiently use technical equipment of given system, while understanding that it may being limited to information gathering – processing power, internet speeds, and query per seconds limitation.

Quality criteria can be specified as gathered description information quantity, distinction of product properties descriptions, the agent combination with different algorithms and web crawler speed.

2.7. Agent system activity diagram

The activity diagrams below show the agent’s actions and associated algorithms (see Fig. 2.2).

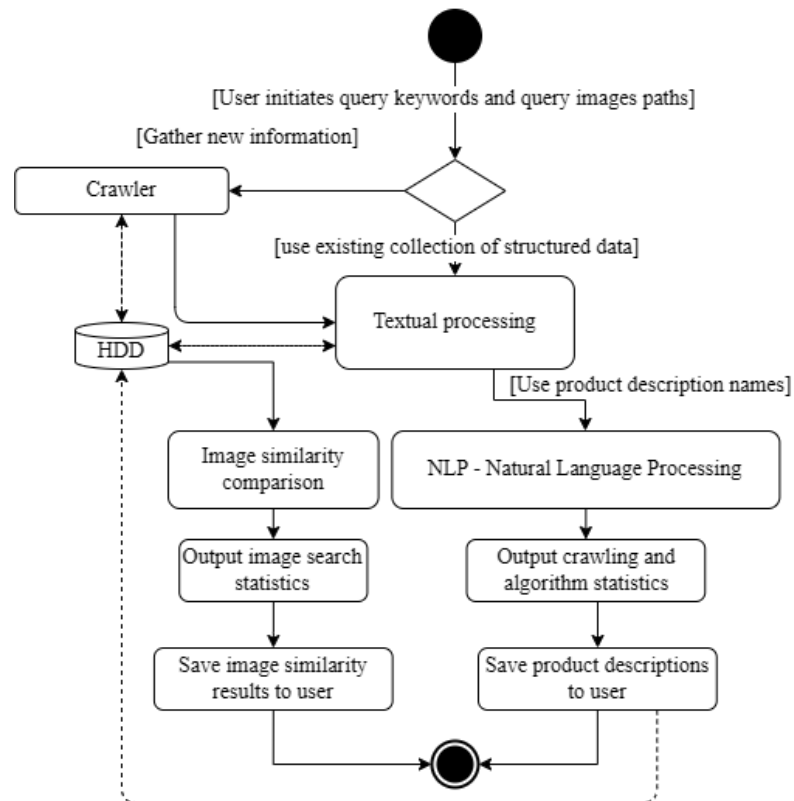


Fig. 2.2 Agents general algorithm activity diagram – structure of agent components and their workflow

Listed activity diagram below describes crawler, which is being parallelized to improve scraping process activity (see Fig. 2.3)

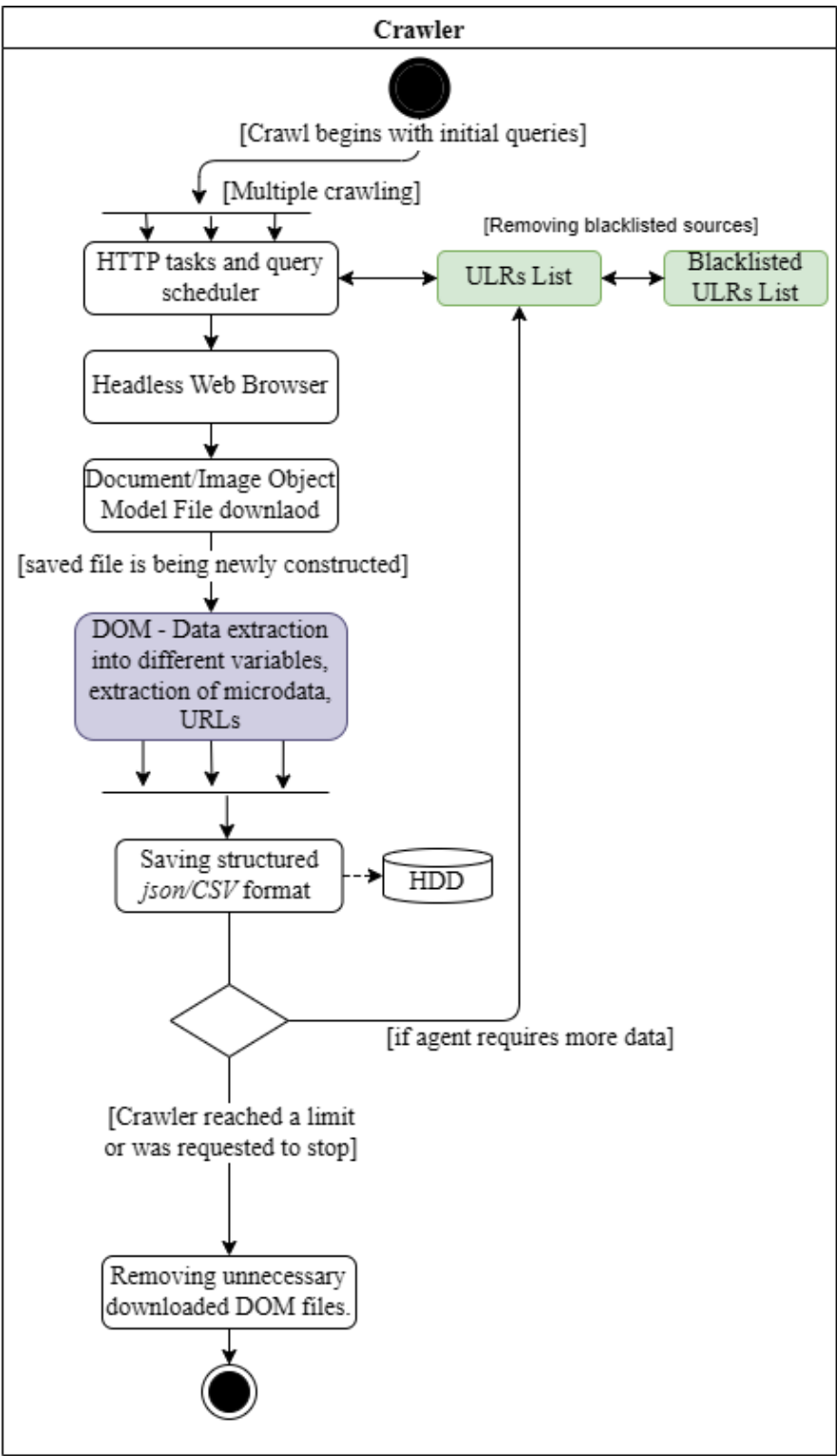


Fig. 2.3 Crawlers activity diagram

In further detail Textual processing includes activity diagram which is used for clustering and using different algorithms to cluster relevant sentences, to minimize complexity and computing power needed for all documents (see Fig. 2.4).

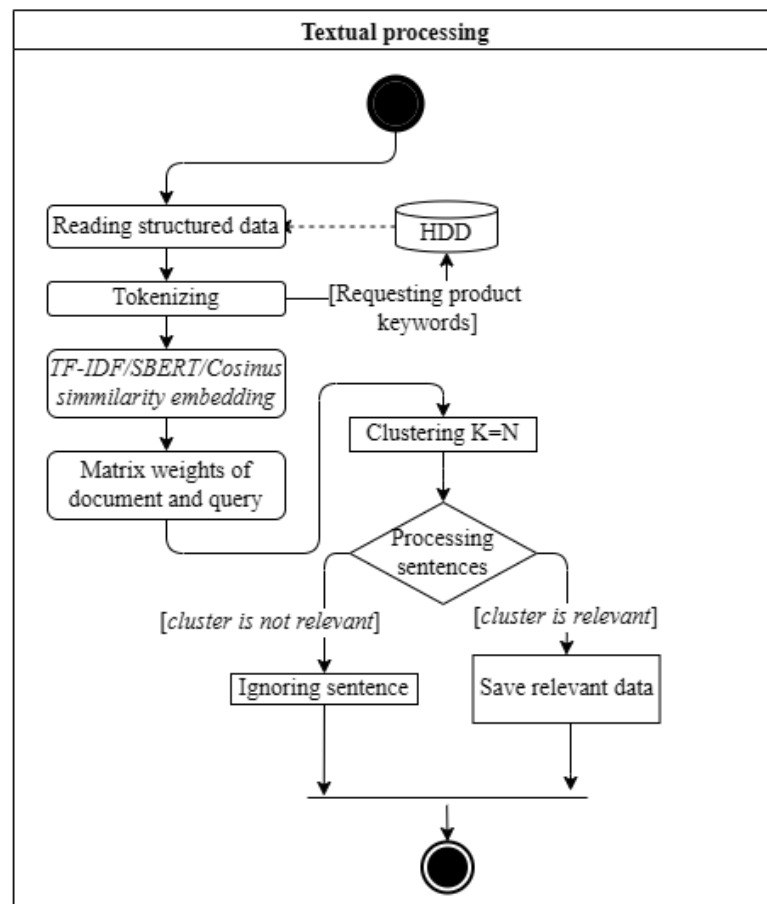


Fig. 2.4 Activity diagram showing the processing of the corresponding text in the tokenisation, insertion, and clustering steps.

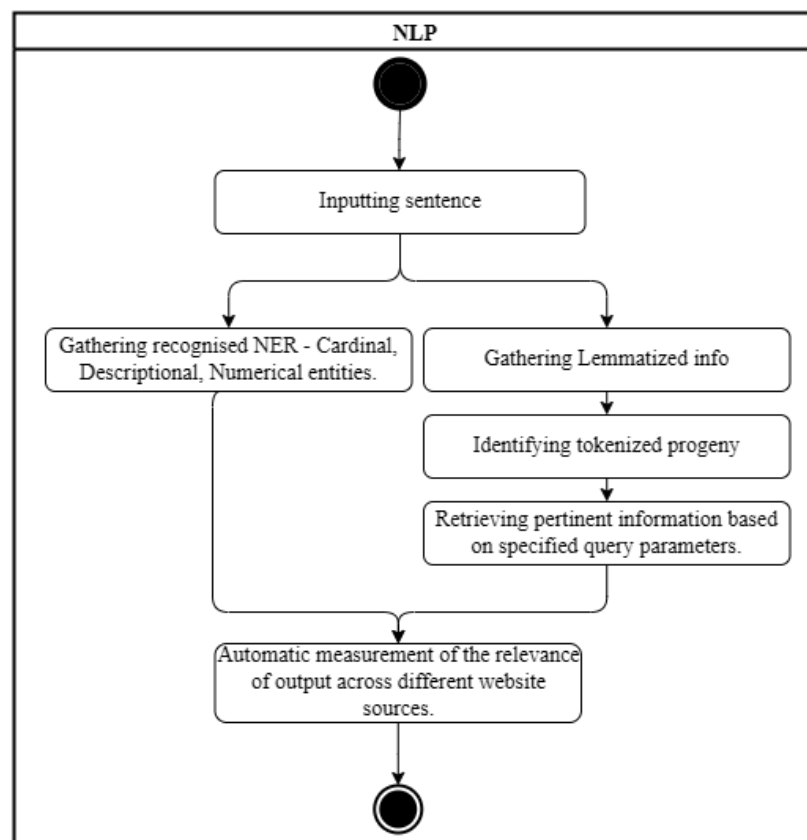


Fig. 2.5 activity diagram of named entity recognition and Lemmatization POS tagging.

Next part of NLP activity diagram which is used for collecting Named Entity Recognitions based on user specified query, and data collection of Lemmatized sentences words it identifies by tokens and retrieves pertinent textual words information based on query parameters (view Fig. 2.5). All gathered information is finalised in websites list their acquired parameters and their relevance scores.

The agent system consists of three main processing steps: first, the headless browser module - scraper collects comprehensive information. This collected data undergoes clustering to determine its relevance, it is achieved by comparing document sentences with the search query. Finally, a natural language processing component, which includes named entity recognition and lemmatization techniques, extracts pertinent information from the documents related to product descriptions. At the end of these successive steps, the user is presented with the relevant information.

2.8. Basic component model

Below listed figure displays the components that are going to be used, during the processing period, which has installed system containing different modules, and accessible to the internet connection (see Fig. 2.6).

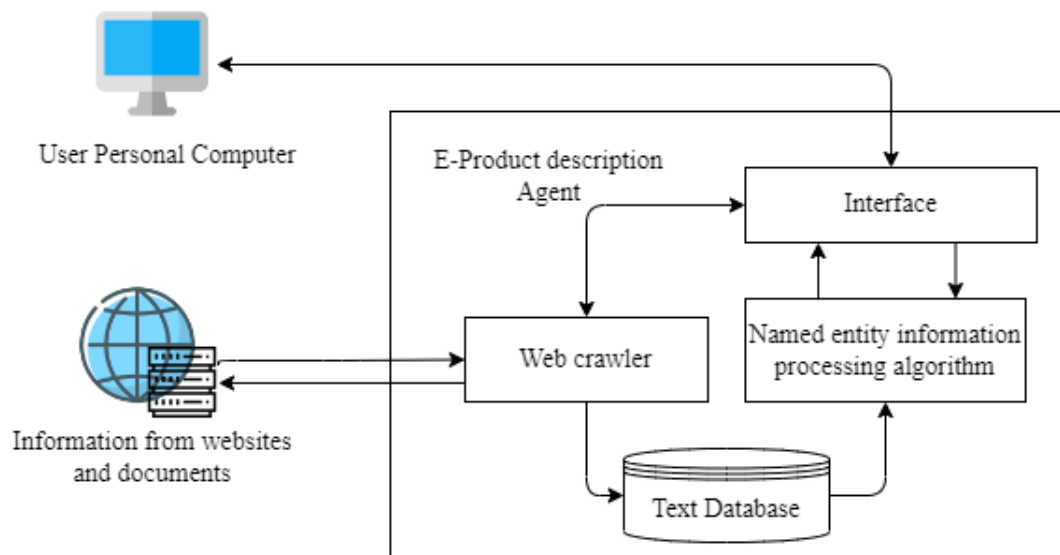


Fig. 2.6 Agent system components diagram, which contains base tools and algorithms.

It is also important to mention that there are different possibilities to access documents if the user already has them and desires to use them to extract descriptive information about his product. Or also the system could not access different service providers in various locations, which may limit specific desirable descriptions.

2.9. Class diagram

Following figure down below, is a projection of project how the realized data structure could look like, by constructing it from projected activity and use case diagrams and using initial tools descriptions.

The class diagram below (see Fig. 2.7) illustrates the fundamental data structure model, which is essential for decision-making. It includes the classes of agents and browsers responsible for storing documents, together with the corresponding matrices, similarities, and query mechanisms. It should be clarified that the system is designed to run multiple crawler processes simultaneously, using multiple processor cores. This approach increases the processing efficiency over time and enriches the product information retrieval capabilities.

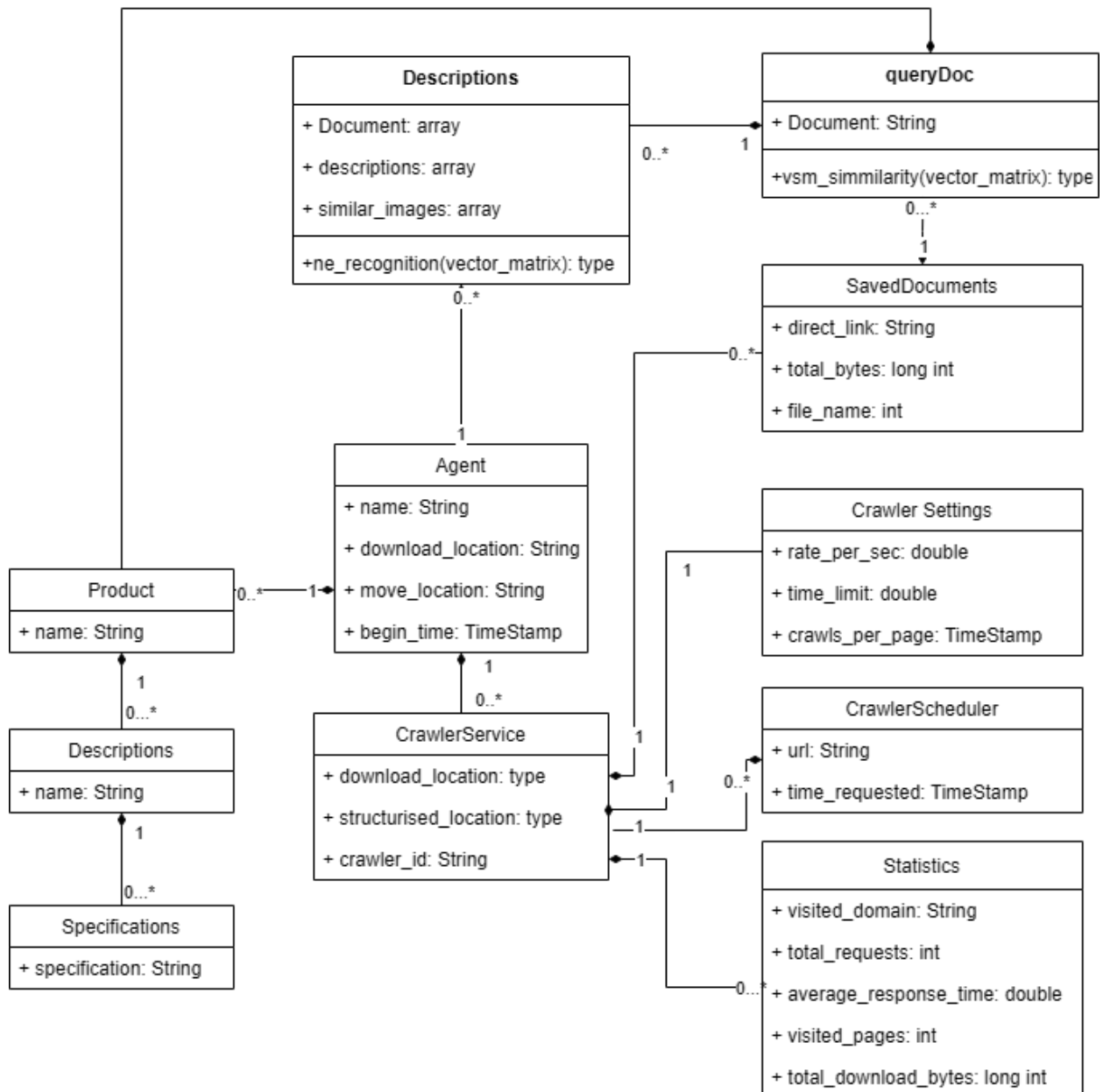


Fig. 2.7 Agent system class diagram

3. Agent information retrieval realisation.

This section provides details regarding the implementation of the agent system pipeline, an overview of the input data, an introduction to the command line interface, and an analysis of the achieved results.

3.1. Input data description.

This subsection describes the user input dataset as well as examples of postprocessed data.

Trying to focus on postprocessing input, we require more accurate and well described user input elements and its query. So, the user requires to type in proper information about a needed product. Algorithm is discussed in section 2, and by following along it we begin with input data, as seen in table Table 3.1 below.

Table 3.1 User required input search query and parameters example.

Example No.	Search query.	Specified user parameters	Product images
1.	RX 5600 XT	“Watt” “Gb” “GHz” “mm”	[“./image1”, “image2”]
2.	RTX 3060	“Watt” “Cores” “ghz” “mhz” “nm” “cores” “gb”	[“./image3”, “image4”, “image5”]

In the following structure scheme of scraper, we can view generalised system (see Fig. 3.1). User inputs of table Table 3.1 are being sent to “Web Image and Text Scraping” block. From gathered user inputs we perform search combinations, by combining search query with specified user parameters, and also only searching by query results. We limit the gathered web scraping time of each page to 0.1 seconds, because lowering it conducted limits of websites bandwidth and it resulted in webpage scraping timeouts. So, it is required to limit force requests. Collected text documents are held in textual format and separated by lines. Image inputs are not important if the user does not require conducting images.

The search query strings were made as obligatory to contain at least one same string search. If the query would be longer and more various it could cover fewer final documents or none if there were none. In the following table we specify experiment example of collected textual information by scrapy, each document is represented by example number, and if the document contained search query it will be processed in further classification steps (see Table 3.2).

Table 3.2 User query and parameters-based search results example

Example No.	Conducted search result example	Downloaded images
1.	NVIDIA GeForce RTX 3060 requires 1x 12-pin power connector, and power consumption is at 170 Watt.	[“img1.jpg”]
2.	12 GB of GDDR6 memory clocked at 15 GHz are supplied, and together with 192 Bit memory interface this creates a bandwidth of 360.0 GB/s.	[“img2.jpg”]
3.	The color lighting from RGB LED can be changed in accordance with graphics temperature.	[“img3.jpg”]

To decrease unnecessary computation power and memory usage, and to increase the relevance of our data we use the clustering K-Means algorithm with $K = X$ points, as vectorised document we fit clustering with the query seen in the input table example. Sentence vectorisation is done using TF-IDF algorithm, cosine similarity algorithm and SBERT embedding algorithm (view below 8 ,9)

$$tfidf(t, m, D) = tf(t, d) \cdot idf(t, D) \quad (8),$$

where t is the word term, d is the text containing the term, D is the collection of all documents, $TF(t,d)$ is the Term Frequency, which shows how often the term appears in the document, $IDF(t, D)$ is the Inverse Document Frequency, which indicates the importance of the term in the whole set of documents.

$$Similarity(A, B) = \frac{(A \cdot B)}{(|A| \cdot |B|)} \quad (9),$$

where A and B are vectors which are being compared, $A \cdot B$ describes the product of the two vectors, $|A|$ and $|B|$ denotes the length of the vectors.

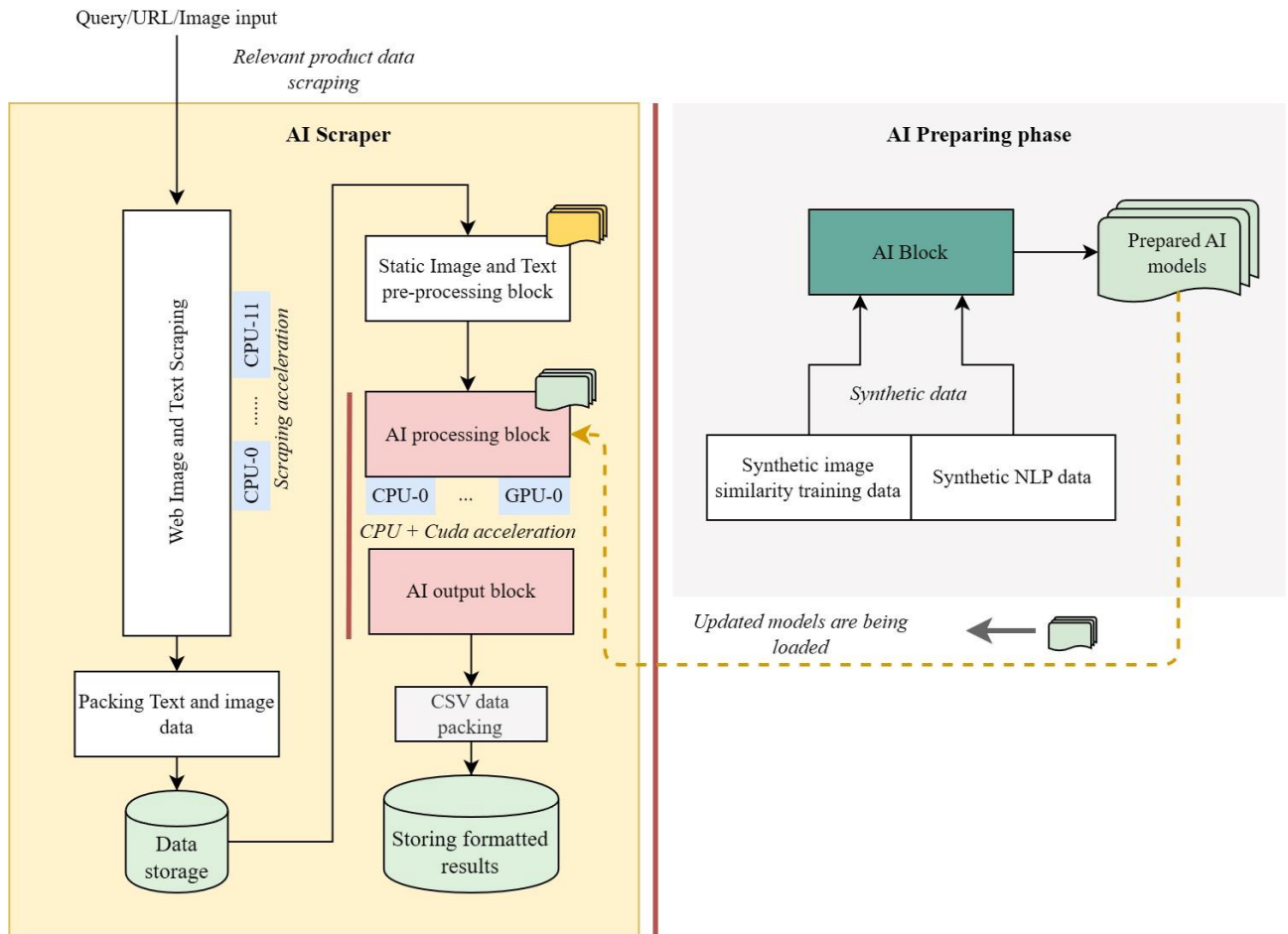


Fig. 3.1 Generalised structure overview of realised Agent AI system.

This table shows vectorised cluster results with a Search query input (view Table 3.3). By vectorising our query and parameters using TF-IDF, we can cluster relevant information (view Table 3.4 and Fig. 3.2).

Table 3.3 Results of embedding and clustering query example

Example No.	Search query.	Cluster
1.	“RX 5600 XT”, “Watt” “Gb” “GHz” “mm”	0

Table 3.4 Results of embedding and clustering sentences example

Example No.	Conducted search result example	Cluster
1.	NVIDIA GeForce RTX 3060 requires 1x 12-pin power connector, and power consumption is at 170 Watt.	0

2.	12 GB of GDDR6 memory clocked at 15 GHz are supplied, and together with 192 Bit memory interface this creates a bandwidth of 360.0 GB/s.	0
3.	The color lighting from RGB LED can be changed in accordance with graphics temperature.	1

The purpose of using clustering's is to gather only relevant data corresponding to our search query. Any unexpected or unrelated data will be disregarded to prevent confusion and avoid collecting an excessive amount of data.

3.1.1. AI block overview

The structure of the AI components is elaborated (view Fig. 3.2). It encompasses textual information input, including documents sourced from websites and user queries. The document *sentincizer*, also known as a tokenizer, is employed to extract each sentence into a list format. Further, the results and execution output are compared between the textual encoding block and its absence (with and without encoding clustering). The process is then finalized within the NLP block, which includes Named Entity Recognition and POS tagging blocks.

The remaining half of the block consist of image input from the user and the comparison of collected website images. This comparison is conducted using a Siamese Network with Euclidean distance measurement for similarity assessment and an Autoencoder-decoder. The results of these models will be discussed in upcoming sections.

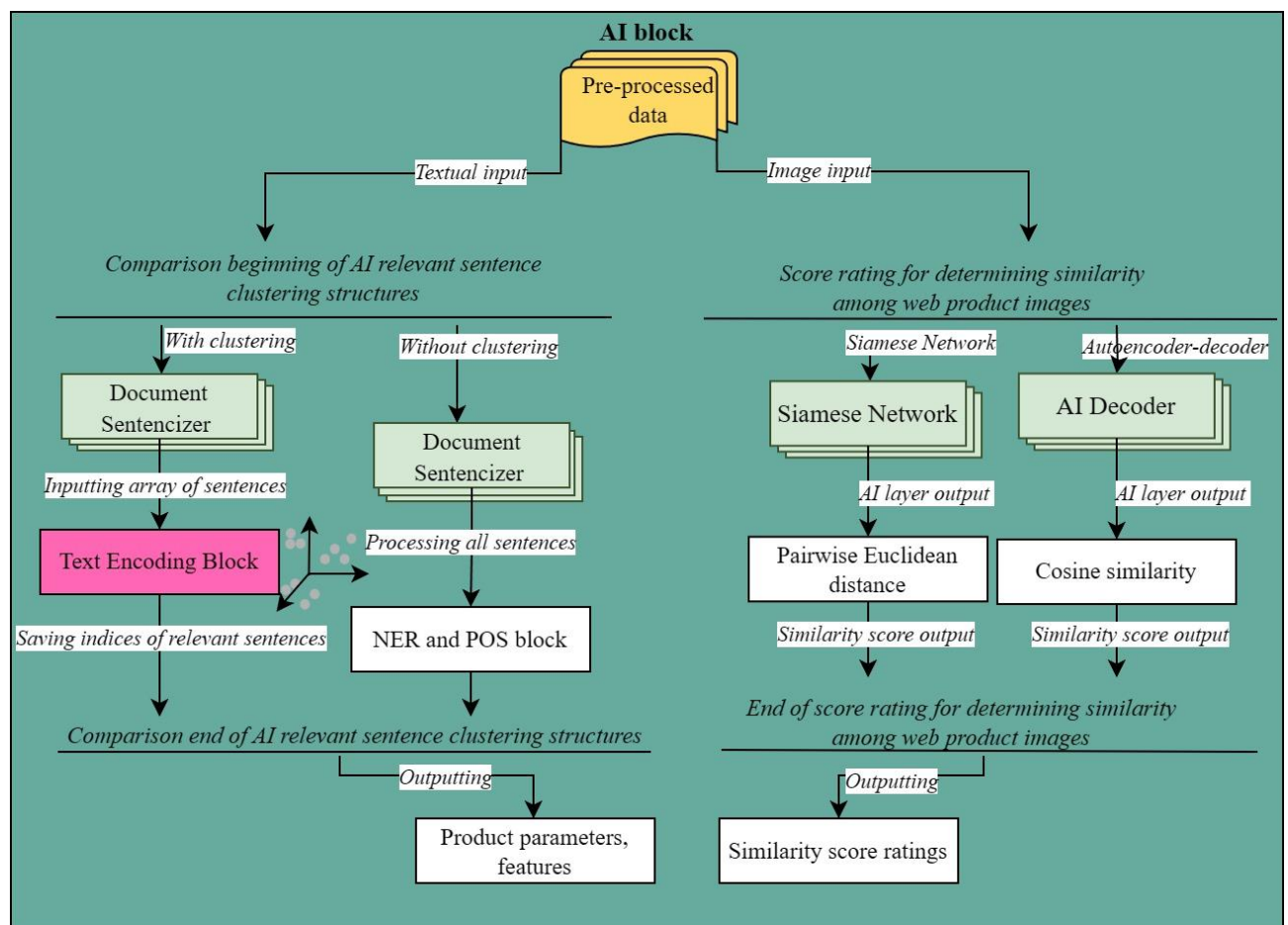


Fig. 3.2 AI block which is included in AI preparation block sequence.

3.1.2. Neural networks architecture

Autoencoder neural network architecture was realised from a default structure using kernel sizing of 3 and stride of 2. Auto encoder input can be adapted for any resolution that is required. The autoencoder neural network architecture was received from a standard structure, utilizing kernel sizes of 3 and strides of 2. The auto-encoder's inputs can be altered to any required resolution, allowing it to be adapted to different applications. A neural network baseline was selected from Autoencoder which was implemented in this system (view Table 3.5).

Table 3.5 Autoencoder model, divided into distinct Encoder and Decoder layers, based on Autoencoder baseline.

Layer Name	Encoder Layer Details	Decoder Layer Details
Input	3x224x224 (Input image)	512x14x14 (Encoder input)
Layer 1	Conv2d(3, 64, kernel_size=3, stride=2, padding=1)	ConvTranspose2d(512, 256, kernel_size=3, stride=2, padding=1, output_padding=1)
	ReLU(inplace=True)	ReLU(inplace=True)
Output after Layer 1	64x112x112	256x28x28
Layer 2	Conv2d(64, 128, kernel_size=3, stride=2, padding=1)	ConvTranspose2d(256, 128, kernel_size=3, stride=2, padding=1, output_padding=1)
	ReLU(inplace=True)	ReLU(inplace=True)
Output after Layer 2	128x56x56	128x56x56
Layer 3	Conv2d(128, 256, kernel_size=3, stride=2, padding=1)	ConvTranspose2d(128, 64, kernel_size=3, stride=2, padding=1, output_padding=1)
	ReLU(inplace=True)	ReLU(inplace=True)
Output after Layer 3	256x28x28	64x112x112
Layer 4	Conv2d(256, 512, kernel_size=3, stride=2, padding=1)	ConvTranspose2d(64, 3, kernel_size=3, stride=2, padding=1, output_padding=1)
	ReLU(inplace=True)	Sigmoid()
Output after Layer 4	512x14x14	3x224x224
Final Output	512x14x14	3x224x224 (Reconstructed Image)

Neural network structure and loss function has been based on Facial similarity with Siamese networks in *pytorch* and One-Shot classification articles (see Table 3.6). [38, 39].

Table 3.6 Siamese Network architecture, based on [38, 39].

Layer Name	Layer Details	Output Dimensions
Input	1x100x100 Grayscale Image	1x100x100
Layer 1	ReflectionPad2d(1)	4x100x100
	Conv2d(1, 4, kernel_size=3)	
	ReLU(inplace=True)	
	BatchNorm2d(4)	
Layer 2	ReflectionPad2d(1)	8x100x100
	Conv2d(4, 8, kernel_size=3)	

	ReLU(inplace=True)	
	BatchNorm2d(8)	
Layer 3	ReflectionPad2d(1)	8x100x100
	Conv2d(8, 8, kernel_size=3)	
	ReLU(inplace=True)	
	BatchNorm2d(8)	
Flatten	Output.view(output.size()[0], -1)	80000
Fully Connected Layer 1	Linear(8100100, 500)	500
	ReLU(inplace=True)	
Fully Connected Layer 2	Linear(500, 500)	500
	ReLU(inplace=True)	
Fully Connected Layer 3	Linear(500, 5)	
Final Output		5
Model Output	5x1(output1), 5x1(output2)	5

3.1.3. Assessing Siamese Network and Autoencoder image similarity models

This experiment compared trained algorithms: Siamese Network and Autoencoder-decoder. Given algorithms utilize different similarity measurement techniques. For the Siamese network, Pairwise Euclidean distance was applied, while for the *Autoencoder-decoder*, layers of encoding were extracted to output encoded 3x100x100 images into 512x14x14, which are then flattened. Subsequently, the autoencoder compares two images and their labels using a cosine similarity algorithm. The Siamese Network employs 1x100x100 input, which, after a 2D convolutional layer, is connected to produce two images with five features each (2x1x5). These parameters are used to train the similarity recognition model for products with similar appearances using Pairwise Euclidean distance.

AT&T faces dataset was chosen from the public databases because it included pictures taken from various angles, making it ideal for training similarity assessments. Each trained and validated class was distinct and unique, but by providing new images as unseen classes, we can still classify them if the input images are similar. Statistics for the training dataset can be found in the table below (view table Table 3.7).

Table 3.7 Image similarity training dataset overview.

Number of training images	Total number of samples	Validation samples	Training batches	Total unique classes
371	401	30	6	40

Further down we can analyze the training model sequence overview for Siamese network and Autoencoder-Decoder model. These parameters were analyzed, and their graph containing Contrastive Loss, accuracy, F1 score, precision and recall was constructed. Thus, after viewing the training graphs of Siamese Network and Autoencoder training graph it is apparent that exactly the same image dataset yields different results. The Siamese network shows high accuracy in this performance. (see figure Fig. 3.3).

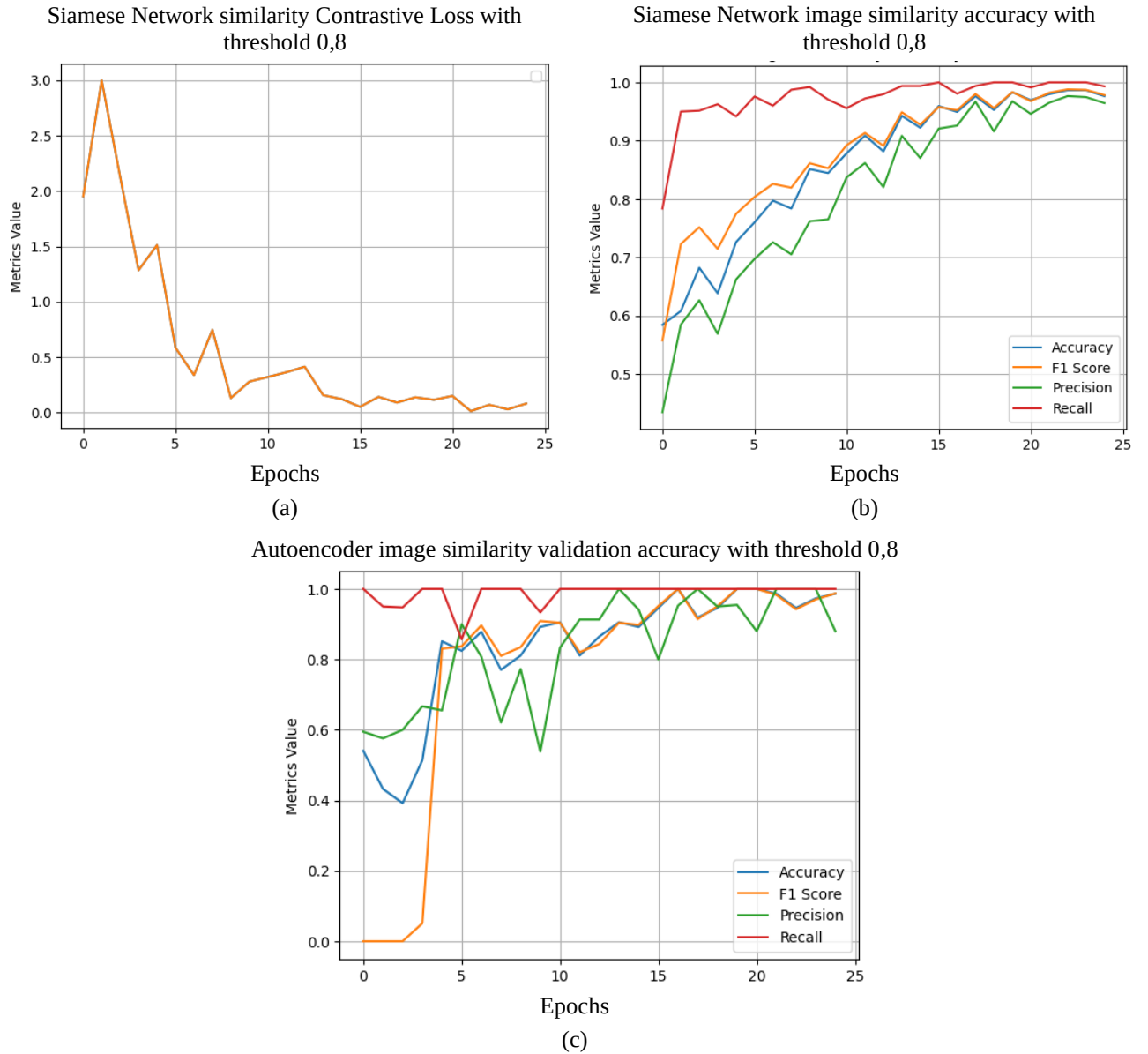


Fig. 3.3 Siamese Network training-validation graph, with 0,8 threshold of pairwise Euclidean distance similarity, where *a* – Contrastive Loss graph, *b* – training accuracy, *c* – validation accuracy.

Mean Squared Error (MSE) was chosen to train the Autoencoder. Similarity measurements were obtained with the model's encoder, and validation accuracy was assessed using cosine similarity. These parameters were analysed, and their graph containing MSE Loss, accuracy, F1 score, precision and recall was constructed. Beyond the 20th iteration, hinting a possible convergence issue or overfitting. This suggests the necessity of incorporating regularization techniques or adjusting the model architecture. (see figure Fig. 3.4).

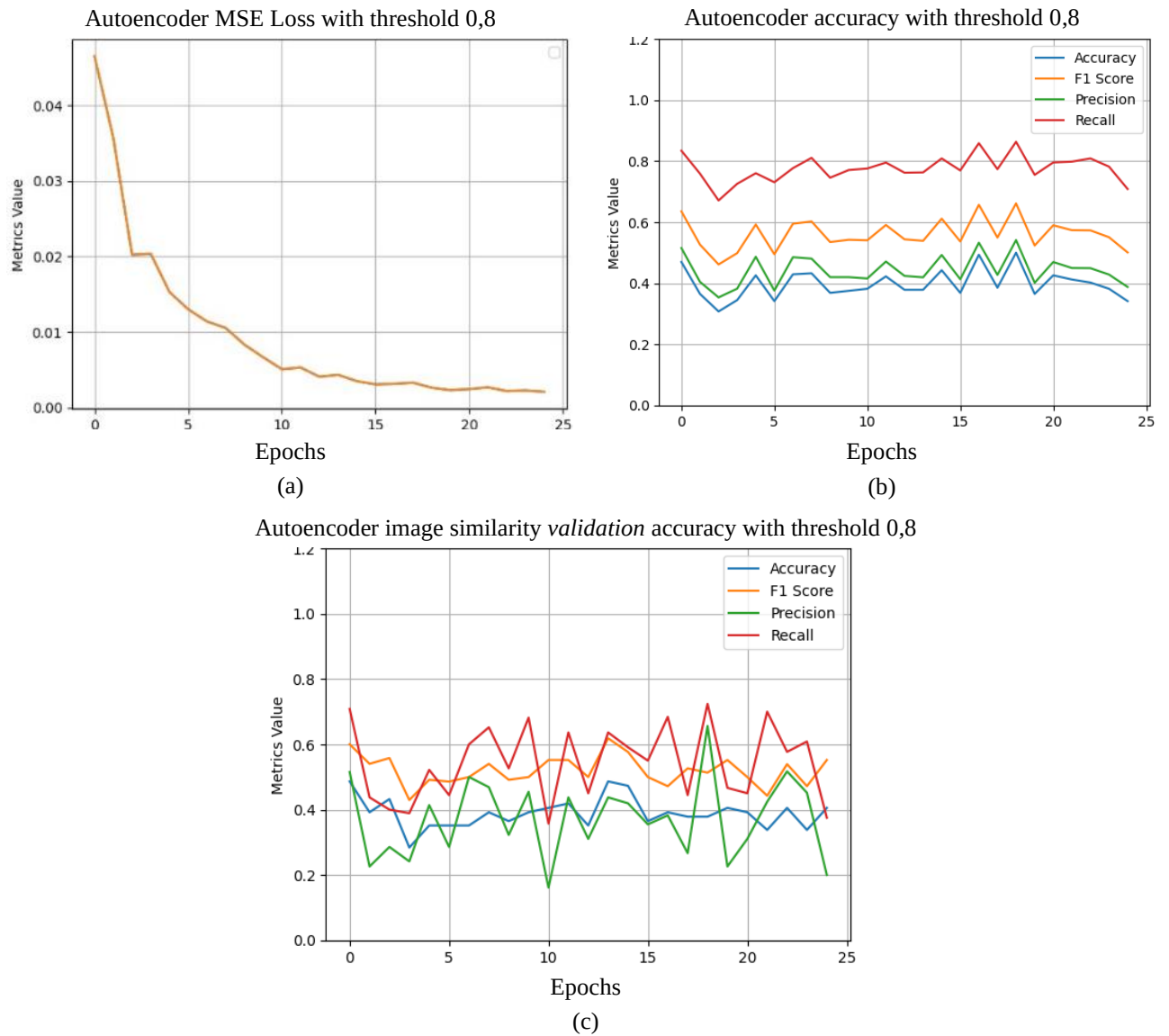


Fig. 3.4 Autoencoder training loss, training and validation graph, with 0,8 cosine similarity threshold, where *a* – MSE Loss graph, *b* – training accuracy, *c* – validation accuracy.

From the observed table below, the Siamese network achieved its lowest loss value of 0,078. With its best training accuracy 0,993, and a validation accuracy of 0,972, using a threshold of 0.6 for Siamese Pairwise Euclidean distance. The F1 score was the highest among all, reaching 0,987 for training and 0,987 for validation tests using a threshold of 0,8. On the other hand, the Autoencoder achieved an F1 score of 0,744 for validation but received a lower accuracy of 0,594 when utilizing a threshold parameter of 0,6 for the Siamese network (see Table 3.8).

Table 3.8 Siamese Network and Autoencoder training and validation measurements comparison

Model	Threshold	Best LOSS	Acc.	F1 Score	Precision	Recall	Evaluation type
Siamese network (Contrastive Loss)	0,8	0,080	0,976	0,978	0,964	0,999	Test
			0,986	0,987	0,880	1,000	Validation
	0,6	0,078	0,993	0,992	0,985	1,000	Test
			0,972	0,940	1,000	1,000	Validation
	0,4	0,094	0,868	0,880	0,789	1,000	Test
			0,851	0,851	0,894	1,000	Validation

Model	Threshold	Best LOSS	Acc.	F1 Score	Precision	Recall	Evaluation type
Autoencoder decoder (MSE-Loss)	0,8	0,026	0,016	0,030	0,084	0,018	Test
			0,486	0,576	0,500	0,111	Validation
	0,6	0,024	0,084	0,112	0,105	0,120	Test
			0,594	0,744	0,625	0,714	Validation
	0,4	0,023	0,084	0,112	0,084	0,168	Test
			0,500	0,665	0,500	1,000	Validation

3.1.4. Textual input and processing entities recognition

Testing out text and query encoding block which sorts out irrelevant sentences. In Fig. 3.5 after relevant sentence embedding and clustering, we are using existing NLP model *en_core_web_trf* which functions are named entity recognition and POS tagging used for word lemmatisation.

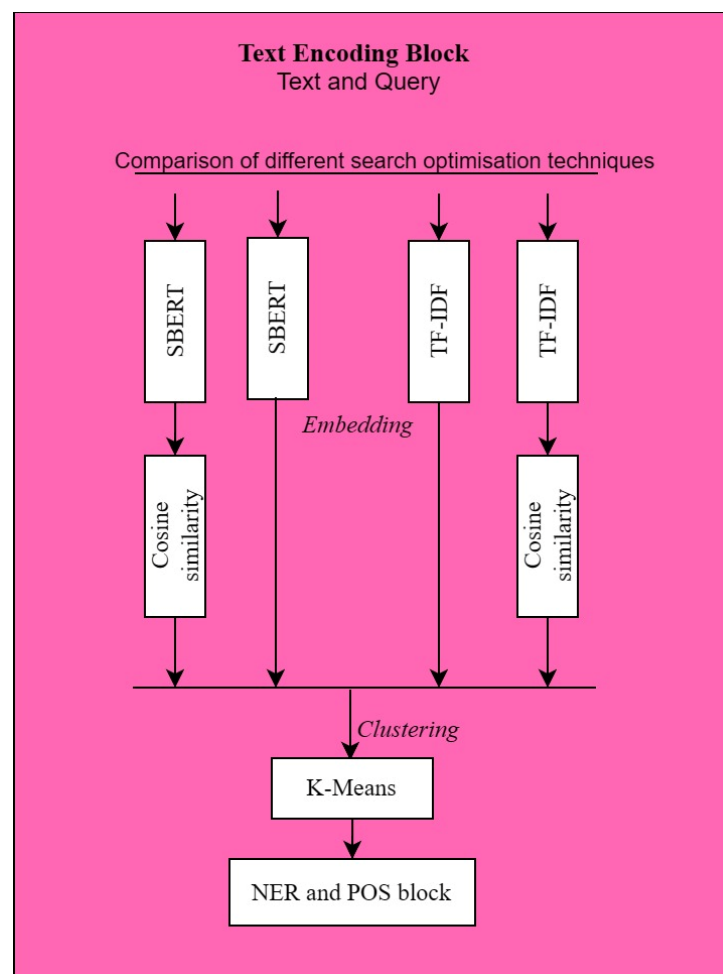


Fig. 3.5 Text encoding block inside of AI block sequence, which contains sentence embedding using SBERT, TF-IDF, and Cosine-similarity combination finalised by clustering and grepping information using.

In named entity recognition we are selecting classified information that includes product, description, parameters, and in lemmatization part of NLP model we connect children of the words of our query relevant sentences. View example of Fig. 3.6 where PROPN is Proper noun, VERB is actions or states of being, NUM is numeral, NOUN: Noun - Words that represent people, places, things, or ideas, NOUN is representation of places, things or measurements.

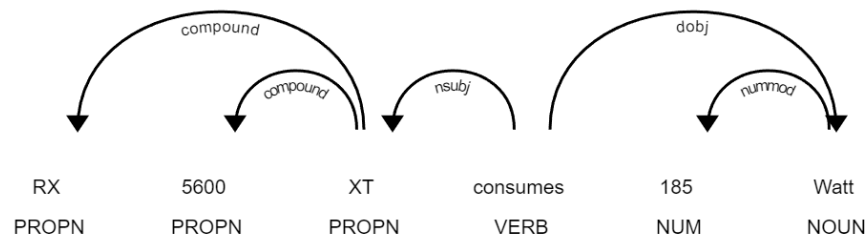


Fig. 3.6 Sentence POS tagging example.

Experiments using SBERT embedder vectorizing it into cosine similarity and clustering it into $K=X$ and it is possible to select wanted cluster parameters. The irrelative cluster with all specified methods will select only queries that are contained in query fitted clusters and which ones doesn't have the lowest cosine similarities.

3.2. Designed command line interface.

This subsection provides command line interface structure and its description. Below a table is describing usage, as seen a created CLI combines all needed information for user (view Table 3.9).

Table 3.9 Structure of CLI operations

No.	Command line structure	Operation requires	Output
1.	Python scrape "RTX 3060" "Watt,Cores,ghz,mhz,nm,cores,gb" --timeout 20 --frequency 0.1	local working directory, internet connection.	Saves query documents and images in new folder.
2.	Python classify "RTX 3060" "Watt,Cores,ghz,mhz,nm,cores,gb" --folder .\ 2023-01-17_02-50-29 --verbose	local working directory, scraped documents in csv format, optional user query images.	Outputs product descriptions to CLI and csv file
[python interpreter] [operation_name] [search_query] [parameter_name] ... [parameter_name] [--limit] [--timeout] [--folder] [--verbose]			

Command line interface combines of 2 main operations, and additional 5 parameters. The above table contains these explained steps (view Table 3.9):

- A first operation of **scrape**, gathers all related information to a specified local location, user is being presented with new collected final documents that can be found in user catalogs. If needed he can view what data is contained in local documents. Operation requires following parameters:
 - [operation_name] – this scenario contains **scrape** parameter.
 - [search_query] – search query has to be defined between 2 brackets “ ”, containing one specific word or 20 more, string length is undefined.
 - [parameter_name] - listing a list of words to be used in search of specific e-product descriptor, it has to be included between brackets, parameter names are undefined.
 - [--limit] – numbers of documents to scrape limit.
 - [--timeout] – defining scraping request per second limit.
 - [--img_sim] - defining to read from default “../Validation” folder or from specified location images as query for comparison.
 - [--img_query_path] – used with img_sim to alter default read path.

- Second operation is marked as **classify**. It classifies documents as relevant or not and collects based on specified parameters, all various results of specific product is then being presented to the user in CLI and in a folder output. Additional parameters that it requires are:
 - [operation_name] and [search_query]
 - [--folder] – location of gathered **scrape** operation stored documents.

After launching the command user, in command line interface it can be seen what information is being processed (view Fig. 3.7).

```
| . | it has a base clock of 1.32 ghz and a boost clock of 1.78 ghz. ;
1.32 ghz
entity.label__contains__ quantity | True |. | 1.32 ghz token.lemma_ 1.32 ghz-> label -> QUANTITY
1.78 ghz.
entity.label__contains__ quantity | True |. | 1.78 ghz token.lemma_ 1.78 ghz-> label -> QUANTITY
ghz ... child =: [['1.32' 'NUM' 'nummod' '1.32']]
ghz ... child =: [['1.78' 'NUM' 'nummod' '1.78']]
```

Fig. 3.7 Example of user CLI output, for general verbose process display.

4. Investigation on the structure of the system Implementation results and experimental comparisons.

In this section of experiments a system metric analysis is being conducted. Starting from subsection 3.1 experimental environment is expanded to include user, scraper and the artificial intelligence block inputs and outputs. This section also discusses the results obtained and compares the performance of different combinations of algorithms with other combinations of algorithms.

4.1. Description of the Investigation environment

In this subsection we conduct experiments of existing and suggested algorithms. The realisation of experimental comparison using technical equipment:

- Intel i7-8700k, 3.7Ghz, 6 core, 12 threads processor with 16 GB RAM.
- Microsoft Windows 11 OS.
- NVIDIA GeForce RTX 3060 Ti, with 8 GB VRAM, 4864 Cuda cores

4.2. Investigation of Web scraping

This part describes search improvements experiment using CPU multiprocessing implementation, input queries were initialised as “python scrape.py “Colorful iGame RTX 3060 Ti Advanced OC” 25 1 --blacklist --verbose”. Where 25-page limit was included, and 1 enabled image download sequence, --blacklist excludes unwanted websites prefixes, such as price comparison websites. Note executed experiments in Table 4.1 consists different files count due to changes in found results, web service blocking same IP requests, and different internet speeds.

Table 4.1 Executed experiment analysis using user query and image downloading input.

Exp. No.	Scraped files count	Scraped files per second	Scraped websites limit	Multiprocessing threads	Scraping duration
1.	450 files	1,66 files/s	25	12	270,36 seconds
2.	477 files	1,40 files/s			340,98 seconds
3.	585 files	0,62 files/s		1	933,94 seconds
4.	497 files	0,73 files/s			676,17 seconds

In the table experiment above, downloaded textual and image documents scraping files time was improved using multiprocessing with 12 threads, by ~2,67 times or by ~0,855 files/s, including 0,1 second delays for scraping files from webpages.

4.3. Exploration of the output results segment

User input data consists of 2 parts, 1st is input parameters for scraping websites, and 2nd are for managing scraped websites their documents with their pictures. These calculations of information about parameters, are being completed in automated manner.

As discussed in 3.1 section, and in input dataset postprocessing part Table 3.9, gathered dataset with 502 lines, from 12 different documents, containing 7459 words. CLI output example of *classify* operation, that user receives in his work environment view Table 4.2 below:

Table 4.2 Classification and named parameters recognition in user CLI.

Line No.	Gathered results in user CLI.
1.	“Scraped results: [‘1.32 ghz ‘, ‘1.78 ghz.’, ‘ 15000 mhz’, ‘360 gb/s’, ‘170 w’, ‘ 1320 mhz’, ‘1778 mhz’, ‘12 gb’, ‘360.0 gb/s’, ‘170 watt’, ‘8 nm process’, ‘1320 mhz’, ‘ up to 1778 mhz’, ‘12 gb’, ‘ 360.0 gb/s’, ‘ 170 watt’, ‘processor only 8 nm.’, ‘ 3060cores’, ‘ 8 nm manufacturing process ‘, ‘12 gb ‘, ‘ 360.0 gb/s’, ‘ 242 mm.’, ‘170 watt.it provides good gaming and benchmark performance at of a leader’s which is nvidia geforce rtx 4090 .’, ‘ 6gb ‘, ‘ 6gb ‘, ‘ 12gb ‘, ‘ 170-watt ‘, ‘ five watts ‘, ‘ 170 watts ‘, ‘ five watts ‘, ‘ 30 watts ‘, ‘ 550-watt ‘, ‘ 600-watt ‘, ‘ 12gb ‘, ‘ 16gb ‘, ‘650-watt power supply ‘, ‘512gb ‘, ‘12gb ‘, ...]”

Table 4.3 Detailed CSV report about the user query scraped results

Row No.	Link	Sum of Similar images	Parameters results and relevance
1	https://www.pchouse.com.bd/igame-rtx-3060ti-advanced-oc-graphics-card	0	{‘8 gb’: 61.11, ‘53 mm’: 44.44, ...}
2	https://www.startech.com.bd/colorful-igame-rtx-3060-ti-advanced-oc-v-8gb-graphics-card	0	{‘8 gb’: 61.11, ‘53 mm’: 44.44, ..., 315.5 mm’: 5.55}
[3-15]	...		
16	https://www.techpowerup.com/gpu-specs/colorful-igame-rtx-3060-ti-advanced-oc.b8469	5	{‘8 gb’: 61.11, ‘53 mm’: 44.44, ‘8 nm’: 38.88, ..., ‘550 w’: 5.55 }
17	https://hotgear.vn/products/colorful-igame-geforce-rtx-3060ti-advanced-oc-v	1	{‘8 gb’: 61.11, ..., ‘64 gb’: 5.55}
18	https://macomputer.com.bd/product/colorful-igame-geforce-rtx-3060-ti-advanced-oc-v-8gb-graphics-card/	0	{‘8 gb’: 61.11, ‘53 mm’: 44.44, ..., ‘1354 mhz’: 5.55}

After collecting analyzed textual documents, the NLP agent algorithm pipeline was able to collect specific examples, that are seen in Table 4.2 and Table 4.3. Completed classification and named parameters were gathered using standalone NLP without optimization in 213,53 seconds with 77,90% accuracy.

4.4. Investigation of different Image similarity methods

In this section, we review the Autoencoding-decoding and Siamese network algorithms for pairwise image similarity extraction. In Table 4.4, an experiment was conducted in which 12 images were listed based on a user query to compare the similarity between 11 scanned websites and their 313 images.

Table 4.4 Investigation of Siamese network and Autoencoder-Decoder models

Method	Duration (seconds)	Total Measured Similarity Combinations	Images per Second	Found Similar Images Count	Websites with similar images	Similarity threshold
1st experiment						
Siamese network	7,33	1878	255,93	5	1	>99%
Autoencoder-Decoder	9,53	1878	196,97	6	2	>99%
2nd experiment						
Siamese network	11,28	2742	242,88	5	1	>99%
Autoencoder-Decoder	15,32	2742	178,90	5	1	>99%

In the table above, we can inspect and conclude that acquired 1st experiment results with *Autoencoder-Decoder* model was able to recognise 6 similar images inside of 2 websites with 99% to 100% similarity threshold. Compared to Siamese Network which was able to recognise only 5 images in 1st experiment.

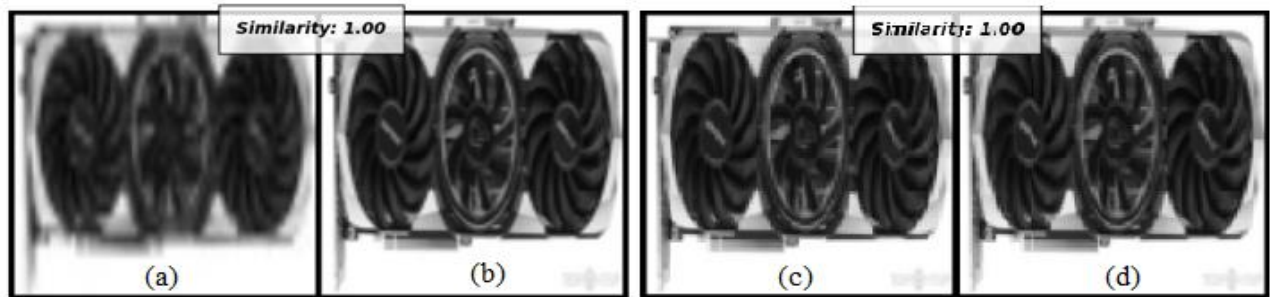


Fig. 4.1 Autoencoder-decoder advantage comparing lower resolution images, experiment resource is based on [41].

In figure Fig. 4.1 original images were *a* – 69x29 resolution, *b*, *c* and *d* – 662x270 resolution. All input images were upscaled or downscaled to 100x100 resolution and greyscale to single image color layer. *Siamese network* model was faster in processing images average by 61,47 iterations per second compared to *Autoencoder-Decoder* model. Trained Autoencoder for encoding and decoding images – so it had better results in finding similar images with lower resolutions. Nevertheless, the most efficient and accurate trajectory, we should keep is Siamese network. Its validation accuracy was 98,6% at finding similar class images using one-shot classification principals.

4.5. Investigation of different Embedding and Clustering methods for Named Entity Recognition improvement.

The table below shows the experiments of various experiments using different NLP (natural language processing) techniques, including TF-IDF, SBERT, cosine similarities embeddings, K-Means clustering, NER involving word Lemmatizing and POS tagging (see Table 4.5).

Table 4.5 Investigation of natural language methods including accuracy, timing, and their parameters.

Exp. No.	Experiment name	K	Average Accuracy (%)	Total Time (seconds)	Processed Unique Parameters
1	TFIDF - Cosine Similarity - K-Means - NLP	1	38,89	22,014	178
		2	38,62	72,009	381
		3	50,52	90,597	659
		4	76,18	113,613	1138
		5	76,18	113,541	1138
		6	76,18	118,660	1191
		7	76,43	122,578	1265
		8	76,43	122,841	1265
		9	76,43	125,209	1309
		10	76,43	121,681	1309
2	NLP (standalone)	-	77,90	213,533	2480
3	TF-IDF - K-Means - NLP	1	77,90	219,745	2480
		2	77,90	220,258	2480
		3	76,84	197,489	2401
		4	76,84	195,409	2356
		5	76,84	211,876	2414
		6	77,37	196,525	2400
		7	77,37	197,468	2415
		8	59,80	166,507	1992
		9	33,86	121,860	769
		10	32,10	120,281	747
4	SBERT - K-Means - NLP	1	77,90	238,877	2480
		2	77,90	239,357	2480
		3	77,90	211,174	2436
		4	76,14	160,750	1781
		5	77,37	191,334	2301
		6	77,90	228,770	2472
		7	77,90	237,653	2480
		8	72,07	134,114	1651
		9	75,66	189,318	2214
		10	75,66	149,731	1964
5	SBERT - Cosine Similarity - K-Means -	1	77,90	236,845	2480

	NLP	2	77,90	154,911	1895
		3	66,11	110,413	1147
		4	48,58	73,935	462
		5	48,47	70,848	450
		6	35,22	58,613	262
		7	35,22	58,291	262
		8	35,22	58,186	262
		9	35,22	58,012	262
		10	37,66	48,435	170

From listed experiments we can identify main insights in different approach results:

1. TFIDF, Cosine Similarity and K-Means, NLP:
 - Clusters 1-3: Lower accuracy (38,62%-50,52%), relatively lower total time, and fewer processed parameters.
 - Clusters 4-10: Significant improvement in accuracy (76,18%-76,43%), but with increased time and parameters.
2. The NLP standalone baseline method achieved an accuracy of 77,90%, with a relatively high execution duration of 213,533 seconds.
3. Combination results of TF-IDF, K-Means and NLP:
 - Clusters 1-2: Received 77,90% accuracy with slightly higher execution times.
 - Clusters 6-7: Received slightly lower 77,37% accuracy, but with an advantage of lower processing times, between 196,53-197,47 seconds and parameters between 2400-2415.
4. Combination results overview of SBERT embedding, K-Means and NLP:
 - Received Consistently high accuracy (72,07%-77,9%) with moderate to high processing times and parameters.
 - With 3 clusters, the accuracy was 77,9%, and the execution time was slightly better than the baseline at 211,17 seconds.
5. Combination results of SBERT, Cosine Similarity embedding, K-Means and NLP:
 - The general accuracy trend was between 35,22%-77,90%, with lower times and parameters but inconsistent results.
 - The best overall configuration was achieved by categorizing text into 2 clusters with 77,9% accuracy at 154,991 seconds, shows promise. However, the rest of the configurations exhibit significantly lower accuracy.

The common experiments and configurations listed should be subject to further research and an increase in the number of experiments to find better accuracy. However, one should lean towards this SBERT - Cosine Similarity - K-Means - NLP pipeline, with an accuracy of 77,9% and a total time of 154,911 seconds, being the most efficient and accurate path to follow.

Conclusions

1. By analyzing existing issues and problems, a reliable information gathering process was established. This process involves using specific tools for web scraping, classifying textual information, and recognizing named entities within documents. Utilizing named entity recognition helps improve search specifications in agent development.
2. The requirements for the agent's development, its usage, and the steps needed for textual description extraction were outlined. The process includes a crawler to collect relevant information, classifying this information using embedding methods (TF-IDF, SBERT, Cosine Similarity), and recognizing named entities using spaCy's NER models.
3. The solution projection identified essential components for the agent: a crawler, information classification for relevance, named entity recognition, and text markup using NLP models. These components were integrated into a proposed solution pipeline, effectively improving the execution duration and accuracy of product description submissions.
4. The comprehensive design proposed in this research ensured a robust and efficient information retrieval agent dividing the algorithm requirements into four parts: data collection (crawler), classification (using K-Means and TF-IDF vectors), named entity identification (using NLTK and spaCy), and user presentation of product descriptions. Using this pipeline 77,9% accuracy and an execution time of 154,911 seconds has been achieved.
5. While both Siamese network and Autoencoder-Decoder models were effective, the Siamese network excelled in processing speed, averaging 249,41 iterations per second. With a validation accuracy of 98,6% for identifying similar class images, the Siamese network proved to be the most efficient and accurate method.
6. Experimental results have shown that implementing multiprocessing with 12 threads significantly improved the download time of text and image documents. This enhancement resulted in a speed increase of approximately 2,67 times, enabling a faster download rate of 0,855 files per second.

List of references

1. [1]Kewen Chen, Zuping Zhang, Jun Long, Hao Zhang,Turning from TF-IDF to TF-IGM for term weighting in text classification,Expert Systems with Applications.2016.Pages 245-260.ISSN 0957-417. Available at: <https://doi.org/10.1016/j.eswa.2016.09.009>
2. [2]DEBOLE, Franca; SEBASTIANI, Fabrizio. An analysis of the relative hardness of Reuters-21578 subsets. Journal of the American Society for Information Science and technology, 2005, 56.6: 584-596. Available at: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.385.4992&rep=rep1&type=pdf>
3. [3]LAN, Man, et al. Supervised and traditional term weighting methods for automatic text categorization. IEEE transactions on pattern analysis and machine intelligence, 2008, 31.4: 721-735. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.151.3665&rep=rep1&type=pdf>
4. [4]ALBISHRE, Khaled; ALBATHAN, Mubarak; LI, Yuefeng. Effective 20 newsgroups dataset cleaning. In: 2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT). IEEE, 2015. p. 98-101.
5. [5]GROOVAC, Jelena. A variant of n-gram based language-independent text categorization. Intelligent Data Analysis, 2014, 18.4: 677-695. https://www.researchgate.net/profile/Jelena-Graovac/publication/293238022_A_variant_of_n-gram_based_language-independent_text_categorization/links/56f7025e08ae38d710a1c107/A-variant-of-n-gram-based-language-independent-text-categorization.pdf
6. [6]KAUSAR, Md Abu; DHAKA, V. S.; SINGH, Sanjeev Kumar. Web crawler: a review. International Journal of Computer Applications, 2013, 63.2.
7. [7]IKONOMAKIS, M.; KOTSIANTIS, Sotiris; TAMPAKAS, V. Text classification using machine learning techniques. WSEAS transactions on computers, 2005, 4.8: 966-974.
8. [8]JOULIN, Armand, et al. Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759, 2016. <https://arxiv.org/pdf/1607.01759.pdf?fbclid=IwAR1wtEXho2gqk3BasKDuncgftN5I5lmH2TbIgvGuHxfutM3IavbateHH9A>
9. [9] LEOPOLD, Edda; KINDERMANN, Jörg. Text categorization with support vector machines. How to represent texts in input space?. Machine Learning, 2002, 46.1: 423-444.
10. [10] Jansen, Bernard J., et al. "Automated gathering of web information: An in-depth examination of agents interacting with search engines." ACM Transactions on Internet Technology (TOIT) 6.4 (2006): 442-464.
11. [11] TANG, Qi; XIE, Fang. A multi-agent system for e-commerce automation. In: 2006 Canadian Conference on Electrical and Computer Engineering. IEEE, 2006. p. 514-517.
12. [12]Unsupervised learning for data classification. Discover the theory and ideas behind unsupervised learning. 2017. [Accessed 2 December 2021] Available at: <https://developer.ibm.com/articles/cc-unsupervised-learning-data-classification/>
13. [13]Chang, S., Cohen, T. and Ostdiek, B., 2018. What is the machine learning?. Physical Review D, [online] 97(5). Available at: <https://journals.aps.org/prd/pdf/10.1103/PhysRevD.97.056009> [Accessed 15 December 2021].
14. [14]Uddin, S., Khan, A., Hossain, M. and Moni, M., 2019. Comparing different supervised machine learning algorithms for disease prediction. BMC Medical Informatics and Decision Making, [online] 19(1). Available at: <https://link.springer.com/content/pdf/10.1186/s12911-019-1004-8.pdf> [Accessed 2 December 2021].
15. [15]Regression and Classification. Supervised Machine Learning [Accessed 21 December 2021]. Available at: [https://www.geeksforgeeks.org/regression-classification-supervised-machine-learning/#:~:text=Supervised%20learning%20is%20where%20you,Y%20%3D%20f\(X\)%20.&text=Supervised%20learning%20problems%20can%20be%20further%20grouped%20into%20Regression%20and%20Classification%20problems](https://www.geeksforgeeks.org/regression-classification-supervised-machine-learning/#:~:text=Supervised%20learning%20is%20where%20you,Y%20%3D%20f(X)%20.&text=Supervised%20learning%20problems%20can%20be%20further%20grouped%20into%20Regression%20and%20Classification%20problems)

16. [16] Google Developers. 2022. Google Crawler (User Agent) Overview [online] Available at: <<https://developers.google.com/search/docs/advanced/crawling/overview-google-crawlers>> [Accessed 14 November 2021].
17. [17]Analytics Vidhya. 2022. Web Scraping in Python | Python Scrapy Tutorial. [online] Available at: <<https://www.analyticsvidhya.com/blog/2017/07/web-scraping-in-python-using-scrapy/>> [Accessed 1 December 2021].
18. [18]ZHANG, Yin; JIN, Rong; ZHOU, Zhi-Hua. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 2010, 1.1-4: 43-52.
19. [19] SILAGADZE, Z. K. Citations and the Zipf-Mandelbrot's law. arXiv preprint physics/9901035, 1999.
20. [20]STEINKAMP, Jackson M., et al. Toward complete structured information extraction from radiology reports using machine learning. *Journal of digital imaging*, 2019, 32.4: 554-564.
21. [21]KRUPP, Lars, et al. A Further Step Towards Automatic Domain Modelling by Relevant Information Extraction. In: *2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 2020. p. 1-6.
22. [22]WANG, Jindong, et al. Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters*, 2019, 119: 3-11.
23. [23]MILETTE, Greg; STROUD, Adam. *Professional Android sensor programming*. John Wiley & Sons, 2012.
24. [24]CHEN, Sixing, et al. Cognitive computing on unstructured data for customer co-innovation. *European Journal of Marketing*, 2019.
25. [25]LAMPLE, Guillaume, et al. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, 2016.
26. [26]Web scraping.pro. 2022. *Pros and Cons of using Selenium WebDriver for Website Scraping – Web Scraping & data mining*. [online] Available at: <<https://web scraping.pro/using-selenium-webdriver-for-website-scraping/>> [Accessed 18 May 2022]. [<https://programs.wiki/wiki/python-crawler-4-selenium.html>]
27. [27]Expertrec custom search engine. 2022. *Disadvantages of using google custom search - 2022 Expertrec*. [online] Available at: <<https://blog.expertrec.com/disadvantages-of-using-google-custom-search/>> [Accessed 18 May 2022]. [<https://blog.api.rakuten.net/top-10-best-search-apis/>]
28. [28]En.wikipedia.org. 2022. *Beautiful Soup (HTML parser) - Wikipedia*. [online] Available at: <[https://en.wikipedia.org/wiki/Beautiful_Soup_\(HTML_parser\)](https://en.wikipedia.org/wiki/Beautiful_Soup_(HTML_parser))> [Accessed 18 May 2022].
29. [29]PATEL, Jay M. Web Scraping in Python Using Beautiful Soup Library. In: *Getting Structured Data from the Internet*. Apress, Berkeley, CA, 2020. p. 31-84.
30. [30] Python.org. 2022. *What is Python? Executive Summary*. [online] Available at: <<https://www.python.org/doc/essays/blurb/>> [Accessed 18 May 2022].
31. [31] TAUFİK, Ichsan, et al. The search for science and technology verses in Qur'an and hadith. *Bulletin of Electrical Engineering and Informatics*, 2021, 10.2: 1008-1014.
32. [32] PEETERS, Ralph, et al. Using schema. org annotations for training and maintaining product matchers. In: *Proceedings of the 10th International Conference on Web Intelligence, Mining and Semantics*. 2020. p. 195-204.
33. [33] AZIS, Mochammad Abdul, et al. Information retrieval system in text-based skripsi document search file using vector space model method. In: *Journal of Physics: Conference Series*. IOP Publishing, 2019. p. 012016.
34. [34] ZIYADI, Morteza, et al. Example-based named entity recognition. *arXiv preprint arXiv:2008.10570*, 2020.
35. [35] GHEORGHE, Mihai; MIHAI, Florin-Cristian; DÂRDALĂ, Marian. Modern techniques of web scraping for data scientists. *International Journal of User-System Interaction*, 2018, 11.1: 63-75.

36. [36] Reimers, Nils, and Iryna Gurevych. Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks. arXiv:1908.10084, arXiv, 27 Aug. 2019. arXiv.org, <https://doi.org/10.48550/arXiv.1908.10084>.
37. [37] GUPTA, Harshvardhan. One Shot Learning with Siamese Networks in PyTorch | HackerNoon. HackerNoon - read, write and learn about any technology [online]. 8 July 2017 [viewed 22 May 2024]. Available from: <https://hackernoon.com/one-shot-learning-with-siamese-networks-in-pytorch-8ddaab10340e>
38. [38] GUPTA, Harshvardhan. Facial similarity with Siamese networks in pytorch. Medium [online]. 20 July 2017. [Accessed 22 May 2024]. Available from: <https://medium.com/hackernoon/facial-similarity-with-siamese-networks-in-pytorch-9642aa9db2f7>
39. [39] KOCH, Gregory, et al. Siamese neural networks for one-shot image recognition. In: ICML Deep Learning Workshop. Department of Computer Science, University of Toronto. Toronto, Ontario, Canada, 2015.
40. [40] WANG, Wei, et al. Generalized autoencoder: A neural network framework for dimensionality reduction. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops. 2014. p. 490-497.
41. [41] Colorful igt RTX 3060 ti advanced OC SPECS. TechPowerUp [online]. 17 May 2024. [Accessed 23 May 2024]. Available from: <https://www.techpowerup.com/gpu-specs/colorful-igt-rtx-3060-ti-advanced-oc.b8469>

Appendix 1. “Towards Early Poultry Health Prediction through Non-Invasive and Computer Vision-Based Dropping Classification”



Article

Towards Early Poultry Health Prediction through Non-Invasive and Computer Vision-Based Dropping Classification

Arnas Nakrosis ¹, Agne Paulauskaite-Taraseviciene ^{1,2,*}, Vidas Raudonis ^{2,3}, Ignas Narusis ¹, Valentas Gruzauskas ^{2,4}, Romas Gruzauskas ² and Ingrida Lagzdinyte-Budnike ¹

- ¹ Faculty of Informatics, Kaunas University of Technology, Studentu 50, 51368 Kaunas, Lithuania; arnas.nakrosis@ktu.lt (A.N.); ignas.narusis@ktu.edu (I.N.); ingrida.lagzdinyte@ktu.lt (I.L.-B.)
- ² Artificial Intelligence Centre, Kaunas University of Technology, K. Barsausko 59, 51423 Kaunas, Lithuania; vidas.raudonis@ktu.lt (V.R.); valentas.gruzauskas@mif.vu.lt (V.G.); romas.gruzauskas@ktu.lt (R.G.)
- ³ Faculty of Electrical and Electronics, Kaunas University of Technology, Studentu 48, 51367 Kaunas, Lithuania
- ⁴ Institute of Computer Science, Vilnius University, 08303 Vilnius, Lithuania
- * Correspondence: agne.paulauskaite-taraseviciene@ktu.lt

Simple Summary: The integration of artificial intelligence and advanced computer vision techniques holds significant promise for non-invasive health assessments within the poultry industry. Monitoring poultry health through droppings can provide valuable insights as alterations in texture and color may signal the presence of severe and contagious illnesses. This study, in contrast to previous research that often employed binary or limited multi-class classifications for droppings, employs image processing algorithms to categorize droppings into six distinct classes, each representing various abnormality levels, with data collected from three different poultry farms in Lithuania, including diverse litter types.



Citation: Nakrosis, A.; Paulauskaite-Taraseviciene, A.; Raudonis, V.; Narusis, I.; Gruzauskas, V.; Gruzauskas, R.; Lagzdinyte-Budnike, I. Towards Early Poultry Health Prediction through Non-Invasive and Computer Vision-Based Dropping Classification. *Animals* **2023**, *13*, 3041. <https://doi.org/10.3390/ani13193041>

Academic Editors: Marco Milanese, Daniele Pietrucci and Giovanni Chillemi

Received: 21 August 2023
Revised: 17 September 2023
Accepted: 25 September 2023
Published: 27 September 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Abstract: The use of artificial intelligence techniques with advanced computer vision techniques offers great potential for non-invasive health assessments in the poultry industry. Evaluating the condition of poultry by monitoring their droppings can be highly valuable as significant changes in consistency and color can be indicators of serious and infectious diseases. While most studies have prioritized the classification of droppings into two categories (normal and abnormal), with some relevant studies dealing with up to five categories, this investigation goes a step further by employing image processing algorithms to categorize droppings into six classes, based on visual information indicating some level of abnormality. To ensure a diverse dataset, data were collected in three different poultry farms in Lithuania by capturing droppings on different types of litter. With the implementation of deep learning, the object detection rate reached 92.41% accuracy. A range of machine learning algorithms, including different deep learning architectures, has been explored and, based on the obtained results, we have proposed a comprehensive solution by combining different models for segmentation and classification purposes. The results revealed that the segmentation task achieved the highest accuracy of 0.88 in terms of the Dice coefficient employing the K-means algorithm. Meanwhile, YOLOv5 demonstrated the highest classification accuracy, achieving an ACC of 91.78%.

Keywords: poultry; droppings; computer vision; deep learning; segmentation; classification

1. Introduction

Adequate production capacity to produce high quality and safe products is a key factor in the efficient operation of the poultry sector. In order to maintain the efficiency of the sector, it is necessary not only to ensure good conditions for poultry farming in line with animal welfare requirements, but also to control the technological parameters of production and to ensure the prevention of health problems in poultry in order to avoid losses at the initial stage of production. The concept of sustainable production has recently