Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматики
Кафедра автоматизації та інтелектуальних інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка системи визначення ключових фраз у відгуках українською мовою за допомогою штучного інтелекту»

(тема роботи)

Виконав: студент 2-го курсу групи 1ІСТ-22м
(шифр групи)
спеціальності 126 – Інформаційні системи та технології
(шифр та назва спеціальності)
Володимир КОВЕНКО
(ПІБ студента)
Керівник: к.т.н., доц. каф. АІІТ
Ілона БОГАЧ
(науковий ступінь, вчене звання / посада, ПІБ керівника)
«14» 12 2023 р.
Опонент: д.т.н., проф.каф.КН
Ярослав ІВАНЧУК
(науковий ступінь, вчене звання / посада, ПІБ опонента)
«14» 12 2023 р.

Допущено до захисту
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
«15» 12 2023 р.

Вінниця ВНТУ – 2023 рік

Vinnytsia national technical university
Faculty of intellectual informational systems and automatics
Department of automation and informational technologies

MASTER'S QUALIFICATION PAPER

to the topic:

«Creation of a system for key phrases retrieval in Ukrainian reviews based on Artificial Intelligence technologies»

(topic of the paper)

Performed by: student of 2nd course, group 1IST-22m
(group cipher)
of specialty 126 – Informational systems and technologies
(cipher and naming of specialty)
Volodymyr KOVENKO
(full name of student)
Supervisor: PhD, docent of AIIT department
Ilona BOGACH
(scientific degree, title / position, full name of supervisor)
« 14 » 12 2023 y.
Opponent:
Yaroslav IVANCHUK
(scientific degree, title / position, full name of supervisor)
« 14 » 12 2023 y.

Accepted to defense
Head of AIIT department
PhD, prof. Oleh BISIKALO
« 15 » 12 2023 y.

Vinnytsia VNTU – 2023 year

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій
Рівень вищої освіти – II-ий (магістерський)
Галузь знань – 12 – Інформаційні технології
Спеціальність – 126 Інформаційні системи та технології
Освітньо-професійна програма – Інформаційні технології аналізу даних та зображень

ЗАТВЕРДЖУЮ
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
«20» 09 2023 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Ковенку Володимиру Андрійовичу

1. Тема роботи: Creation of a system for key phrases retrieval in Ukrainian reviews based on Artificial Intellingence technologies
   Керівник роботи: к.т.н., доцент каф. АІІТ Ілона БОГАЧ
   Затверджені наказом ВНТУ від «18» 09 2023 року №247.
2. Строк подання студентом роботи до «20» листопада 2023 року.
3. Вихідні дані до роботи: Windows XP/Ubuntu, процесор: Core 2 Duo, оперативна пам'ять: 4 GB ОП, відеокарта: Intel HD Graphics 4000, місце на диску: 2 GB доступного місця.
4. Зміст текстової частини: вступ; аналіз предметної області й попередньої роботи; аналіз й вибір інструментів розробки; збір даних, їх обробка, аналіз й фільтрація; економічна частина; висновки; список використаних джерел.
5. Перелік ілюстративного (або графічного) матеріалу: схема алгоритму для детекції питань; розподіл кількості символів у даних; схема автоматичної фільтрації даних; схема загального підходу до визначення ключових фраз; матриця невідповідностей найкращої моделі на задачі передбачення оцінки відгуків; приклад визначення ключових фраз для ресторану.

6. Консультанти розділів магістерської кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Ілона БОГАЧ к.т.н., доцент каф. АІІТ		
5	Володимир КОЗЛОВСЬКИЙ, к.е.н., доцент каф. ЕПтаВМ		

7. Дата видачі завдання «__20_»__09__2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	20.09.23-02.10.23	
2	Вибір інструментів розробки, мови програмування й бібліотек	02.10.23-10.10.23	
3	Збір даних й їх обробка	11.10.23-23.10.23	
4	Тренування моделей й створення системи для визначення ключових фраз	23.10.23-08.11.23	
5	Економічна частина	08.11.23-11.11.23	
6	Оформлення пояснювальної записки і графічного матеріалу	12.11.23-14.11.23	
7	Попередній захист роботи	21.11.23	
8	Остаточний захист роботи	11.12.23 - 21.12.23	

Студент _____ Володимир КОВЕНКО.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Ілона БОГАЧ
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.896.6

Ковенко В.А. Розробка системи визначення ключових фраз у відгуках українською мовою за допомогою штучного інтелекту. Магістерська кваліфікаційна робота зі спеціальності 126 – Інформаційні системи та технології, освітня програма – Інформаційні технології аналізу даних та зображень. Вінниця: ВНТУ, 2023. 149 с.

На англ. мові. Бібліогр.: 44 назви; рис.: 23; табл.: 9.

У роботі розроблено технологію для визначення ключових фраз на основі даних у міждоменному налаштуванні та створено моделі для визначення тональності й оцінки відгуків. Акцент зроблено на моделюванні в умовах зашумлених й незбалансованих даних.

Ключові слова: машинне навчання, штучний інтелект, пояснювальний інтелект, аналіз тональності.

ABSTRACT

Kovenko V.A. Master's qualification paper of a specialty 126 – Informational systems and technologies, curricula – Informational technologies of data and image analysis. Vinnytsia: VNTU, 2023. 149 p.

In English language. Bibliography: 44 titles; fig.: 23; tabl.: 9.

In the work, a technology for key-phrases retrieval based on Ukrainian reviews data in a cross-domain setting is developed, models for sentiment analysis and reviews rating estimation are created. The emphasis is set on modelling in the setting of noisy and imbalanced data.

Keywords: machine learning, artificial intelligence, explainable AI, sentiment analysis.

TABLE OF CONTENTS

INTRODUCTION

Relevance of the problem Recent advances in NLP sphere, which is primary relevant to neural network based approaches provided researches with a possibility to tackle large variety of difficult tasks (NER[1], NEL[2], QA[3], etc.) and pushed limits for machine text comprehension. Those technologies allow companies to transform unstructured text data to the structured output that is easier to understand and analyze. Text analysis is very much relevant to the B2B companies which are monitoring mass media towards specific businesses for the sake of analytical reports creation and business insights provision. One of the key features that is often included in analytical reports is sentiment analysis w.r.t specific company and predefined time range. Although sentiment analysis provides general insights about company's well-being, it doesn't address the question of causes that influenced such a result. The task relevant to extraction of the causes of sentiment is called Aspect-Based Sentiment Analysis[4].

Despite of the fact that pre-trained models for solving the task do exist, most of them are relevant only to one domain. What is more, open-source solutions to ASBA are mostly based on processing of English language and creation of a new labeled dataset requires much amount of time and lots of manual work. The task of ASBA is relevant to classifying sentiment towards identified aspects. If to unite task of ASBA and aspects identification, the overall task can be reformulated in the following manner: retrieve key aspects and classify them with respect to sentiment labels. If to consider that overall sentiment of the sentence is a composite of aspects sentiments, the other reformulation of the task appears: retrieve key aspects that influenced predicted sentiment label the most. Other problem where such a formulation is applicable is relevant to summarization of reviews relevant to specific entity based on extraction of key phrases that influenced explicit ratings. The only difference in formulation for this task is that instead of sentiment label, the retrieval is done towards rating.

Generally, the task can be formulated in an abstract way: retrieve key textual features that influenced predicted label the most.

Actuality of the work lies in the new formulation of unsupervised ABSA task and solution to the problem of unsupervised key-phrases retrieval for Ukrainian language. As it was already mentioned, most of the solutions to ABSA are mostly relevant to English language, which makes it much harder to either find relevant datasets or pretrained models for Ukrainian language. Same situation is observed when speaking about general sentiment analysis or reviews rating estimation. Taking into account the problem of reviews summarization, it's important for summarization methods to pay attention to sarcasm, words order and other complex patterns in language. Due to the subjectivity of reviews, it's mandatory to use noise-robust methods, that learn to generalize and not overfit to the data. What is more, considering that textual data is completely human-generated, there could be many errors and typos in words, which would result in a big number of different tokens, that can influence model both during training and inference.

A methodology relevant to solving highlighted problems is presented in this work. The primal focus is set on processing of Ukrainian language and solving the task of key influential phrases extraction in the bounds of cross-domain reviews. The work showcases usage of deep-learning and classical machine learning algorithms to learn the conditional distribution of the data and application of explainable AI techniques to extract most influential textual features. The presented solution is cross-domain, adaptable to new data, easy to enhance and requires to store only model and tokenizer, which can also be used to estimate ratings for reviews. As the part of work, a real-world dataset of Ukrainian reviews is collected, which can be used to further advance NLP sphere for Ukrainian language. Models that are used for extraction of key phrases, can also be utilized for sentiment analysis and reviews scores estimation. Big emphasis is set on thorough data preprocessing and experimentations with techniques for tackling noisy data.

The purpose of the work is to develop a technology for key phrases retrieval which is more efficient than its analogs, is adaptive to unseen data and new domains and is more convenient for enhancements.

The following problems should be solved to achieve a goal:

1. Conduct analysis of existing methods and approaches to text classification, explainable Artificial Intelligence and key-phrases retrieval.

2. Collect the reviews data for different domains.

3. Analyze collected dataset, clean and process it.

4. Choose, train and evaluate algorithms for estimating reviews score and sentiment.

5. Choose explainable AI algorithms for key-phrases retrieval.

6. Construct an algorithm for key-phrases retrieval and evaluate it towards different explainable AI algorithms.

Research methods. The following research methods are used in the work: analysis, forecasting of results, modeling of the system, classification of existing entities, analysis of the development results and their adjustment, summarization of the performed works.

The object of work are processes of information search, text classification, artificial intelligence explainablity and information retrieval for Ukrainian language.

The subject of work are methods of information processing, deep learning architectures for text classification, methods of convex optimization, approaches to explainable artificial intelligence.

Scientific novelty lies in collection of a cross-domain dataset containing Ukrainian reviews; solving problem of reviews score estimation and sentiment analysis for Ukrainian language; solving problem of automatic key phrases retrieval and summarization for Ukrainian language.

Practical value of the work lies in providing a ready-to-use technology for key-phrases retrieval for Ukrainian language in a cross-domain setting, models trained for sentiment analysis and reviews rating estimation and processed reviews dataset that can be utilized for in depth analysis or further modeling.

Approbation and publications of the work results. The research paper summarizing the work on the 1st stage of the All-Ukrainian competition of student papers in artificial intelligence on October 16, 2023 (excerpt from the protocol of the competition commission of VNTU dated October 16, 2023 is attached, appendix C) presented during MODS2023 conference and is awaited to be published in LNNS ("Lecture Notes in Networks Systems") journal (appendix D). Codebase of module relevant to inference of final algorithm has been accepted for a copyright procedure at October 2023.

1 CURRENT STATE OF PROBLEM AND PREVIOUS WORK REVIEW

It’s mandatory to analyze previous work that intersects with described method in order to prove the novelty of scientific discovery, identify weak and strong points, and choose approaches to build on and enhance.

The presented approach to solving problem of key-phrases retrieval can be divided into two steps:

Train a generalized discriminative model. In our case the input to the model is textual data and expected output is the correct probabilities for classes associated with data.

Apply explainable AI techniques to make reverse-engineering and extract those phrases that contributed the most to final decision of the model.

Thus, the aforementioned pipeline operates in two particular spheres: text classification and explainable AI. Nevertheless, approach to tackling the problem and formulation is a novel one, several similar works exist. In this chapter, previous work relevant to text classification, explainable AI, aspects ranking and unsupervised ABSA is discussed.

1.1 Problem analysis

The task of key-phrases retrieval has many usages, including feedback summarization, content analysis, market research and social media monitoring. Speaking of reviews summarization, the solution to the task gives an opportunity to rapidly understand pros and cons of specific entity, providing client with useful information that can help to enhance user-experience, and giving vendor an opportunity to improve entities quality and understand the aspects which influenced the overall rating. Nevertheless, there are many solutions to the problem, most of them require lots of labeled data that is complicated to collect. Giving the fact that there are many different domains of reviews, the task becomes even more complicated. Finally, most

of solutions exist for English language, with little to zero research done for Ukrainian one, which presents an obstacle for adopting the technology for Ukrainian reviews data. A feasible solution to the highlighted task would require the following qualities:

The solution should be cross-domain, meaning that approach should work well across multiple domains of reviews. Such quality would provide an opportunity to use same approach for multiple types of data simultaneously without a need for adopting to new domains, that could require more time and resources. Moreover, making a solution

Convenience of adaptation to new domains. Although the aforementioned quality states that solution should be cross-domain, the one would want to finetune it to his/her own data or extend it to novel domains.

Convenience of algorithm's extension. Possibility to extend and enhance algorithm is mandatory for adopting the solution to real world scenarios and incorporating it into the production pipeline.

The solution should be lightweight and fast. This quality is mandatory for solution incorporation and is tightly connected to convenience of algorithm's extension.

Finally, the solution should work with Ukrainian language.

1.2 Primer on deep-learning for NLP

1.2.1 Embeddings

Embeddings layer is the basis for deep-learning based NLP and is defined as a relatively low-dimensional space into which the one can translate high-dimensional vector. The logic behind embeddings is based on distributional hypothesis, which states that words with similar contexts tend to have similar meanings. By context, the words and phrases are meant. Ideally, embeddings capture some of the se-

mantics of the input by placing semantically similar inputs close together in the embedding space. Embeddings are often trained using word co-occurrence statistics from large corpora. Words that frequently co-occur are assigned vectors that are close, reflecting their semantic proximity. Unlike traditional methods like one-hot encoding, which create discrete representations, embeddings place words in a continuous vector space. This continuity allows for more flexible and nuanced representations of language. Through their ability to capture the underlying semantics of words, embeddings aid in improving generalization. This is especially helpful in situations when it's probable that some language use variants aren't covered by the training data. Based on the semantic information embedded in embeddings, the model is able to extend its understanding. Speaking of usage of embeddings inside deep learning architectures, they are often used to map unique words into vectors which are then processed by other layers of the network. Embeddings layer is often used in a transfer-learning setting:

1. Embedding along with other layers is pretrained using unsupervised learning for language understanding or similar task.

2. Pretrained embedding layer is incorporated into other models for solving downstream tasks. During this stage, embedding is either finetuned or frozen.

Embeddings are used in many spheres of NLP and have many usages, including:

- Semantic similarity. Assessing the semantic similarity of words is one of the main uses of word embeddings. Close vectors in the high-dimensional space represent words with comparable meanings.
- Word similarity and analogy. Operations like word analogy and similarity are made possible by embeddings. Consider the well-known scenario: king - man + woman = queen. These kinds of comparisons can be investigated in the vector space, demonstrating how embeddings can capture word associations.
- Machine translation. By representing words or phrases in a source language and transferring them to a target language, embeddings are essential to machine translation. This aids in keeping the meaning intact while translating.

- Named entity recognition (NER). Embeddings help NLP tasks like NER by allowing the model to comprehend the context and relationships between items by modeling entities in a continuous vector space.
- Classification. For text classification tasks, embeddings—which capture semantic subtleties in word representations—are powerful features. Their capacity to gather contextual data expands the feature space and aids in the creation of classifiers that perform better.

1.2.2 Convolution

Convolutional layers are widely used in image processing and state-of-the-art computer vision models due to translation invariance. Convolutional layers can also be applied towards textual data. In this setting convolutional kernel “scans” the textual vector with a specific stride. Kernel size can be roughly referred as a number of unigrams that are aggregated together (n-grams), whereas number of filters is relevant to number of different representations of aggregation (formula 1.1).

$$(f * g)(i) = \sum_{j=0}^{M-1} g(j) * f(i-j) \tag{1.1}$$

where f is the input sequence, g is the filter or kernel, M – lengths of sequence, j – position withing filter or kernel.

Convolutional layers are primarily used because of their capacity for local feature extraction. This refers to identifying patterns or characteristics within a limited context, like a small window of words, in the context of text. Convolutional layers utilize parameters sharing, which allows same set of weights to be applied across different positions in the input sequence. The sharing of parameters gives an opportunity to detect similar patterns at different locations, making the model more efficient. A certain amount of translation invariance is offered by convolutional layers. This indicates that regardless of a pattern's precise location within the input se-

quence, the model is able to identify it. This is useful for capturing the same linguistic patterns that appear at different places within a text. Convolutional layers are frequently used with pooling layers (like max pooling) to lower dimensionality and preserve the most important information. By highlighting the salient characteristics in a given area, pooling aids in the extraction of pertinent data. What is more, convolutional layers give an opportunity to apply multiple filters allowing for richer representation of text.

1.2.3 LSTM

Long-short-term-memory network, which is a successor of recurrent neural network is primarily used for fitting sequential data. LSTM is a successor of RNN (recurrent neural network) architecture. Given the fact that text is a sequential data by nature, LSTM if often used for tackling NLP tasks. As in vanilla RNN, the output from the previous step is fed as input to the current step. LSTM enhances RNN by partially solving “vanishing” gradients problem. The intuition behind the LSTM architecture is to create an additional module in a neural network that learns when to remember and when to forget pertinent information. LSTM introduced gate mechanisms, that are constraining the information that is persevered inside the layer. In particular the following mechanisms are used:

1. Forget gate. The forget gate decides which information needs attention and which can be ignored.

2. Input gate. The input gate decides what relevant information can be added from the current step.

3. Output gate finalizes the next hidden state.

LSTMs are appropriate for tasks where understanding context over extended periods of time is critical because they are made to capture long-term dependencies in sequences. In order to solve problem of “exploding” gradients relevant to huge values of computed gradients, the gradient of LSTM is sometimes limited to predefined maximal number. Because LSTMs can handle sequences of different lengths,

they are useful for a variety of applications, including time series analysis, speech recognition, and natural language processing. Schematic workflow of the LSTM layer is shown on figure 1.1.

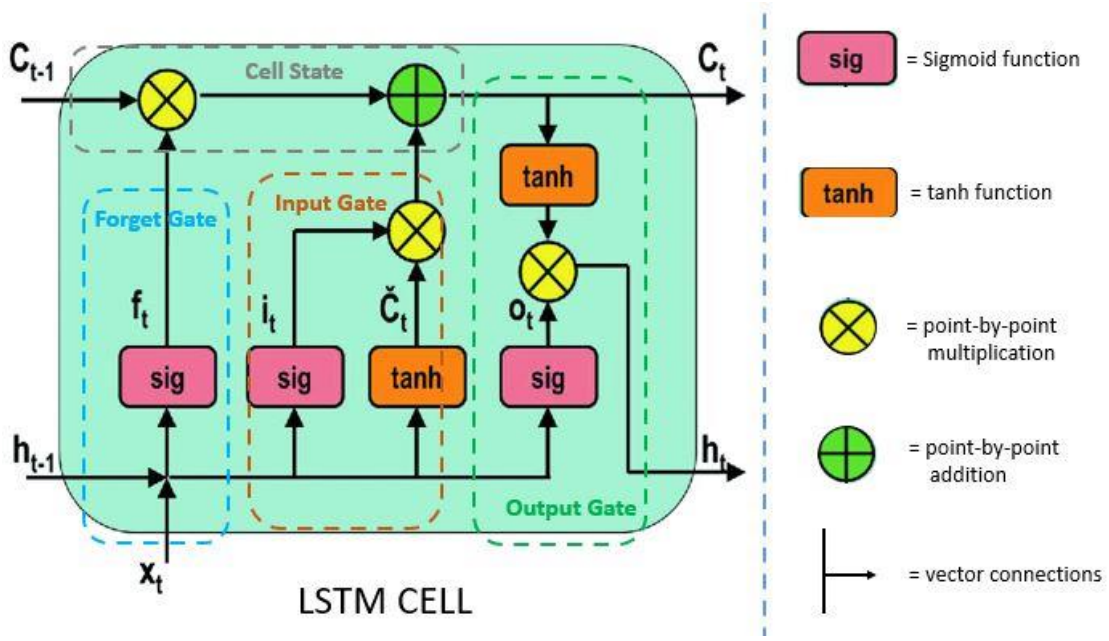


Figure 1.1 – Schematic workflow of LSTM layer

1.2.4 Attention

Attention mechanism is another core technology in the sphere of NLP. Definitely attention was mostly popularized by the paper of Vaswani et. al. [5], in which self-attention was introduced. Before the introduction of Transformers architecture, attention was applied to such tasks as machine translation[6], target sentiment analysis and others. The core of any attention is to compute “attention weights” that weight information w.r.t given context. Attention mechanism is often used to aggregated output of LSTM hidden states in a non-linear way. The same aggregation can be used as an explainable AI mechanism, where attention weights can be interpreted as importance of each hidden state representation of input feature. If to consider classification task (in this case model reflects encoder architecture plus classification head), the attention can be defined by the following formula, where $W1, W2$ are

attention weights, h - hidden states from LSTM, a – attention weights, c – weighted context (formula 1.2):

$$\begin{aligned} \mathrm{a} = \ & \mathrm{softmax}(W_2 \otimes \tanh\,(W_1 \otimes h)) \\ & c = h \otimes a \end{aligned} \tag{1.2}$$

1.2.5 Dropout

Dropout presented by Hinton et.al [7] is a regularization technique used to prevent neural networks from overfitting. By randomly deactivating (or "dropping out") a portion of neurons during training, dropout aims to prevent neurons from becoming unduly specialized and to encourage the network's generalization (figure 1.2)

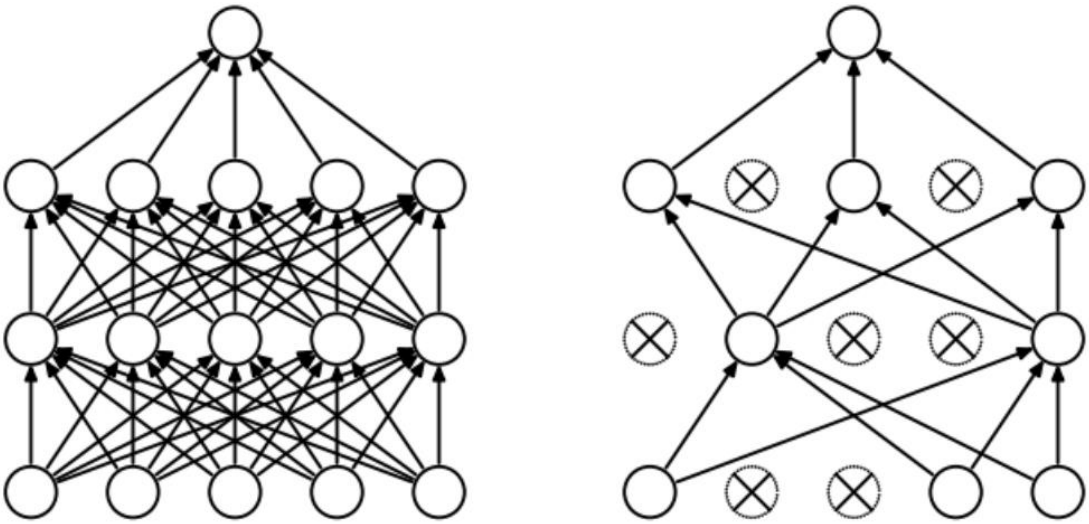


Figure 1.2 – Example of dropout regularization

Dropout arbitrarily sets a portion of the neurons' outputs to zero on each forward pass through the network. The probability of dropout is a hyperparameter. The dropout process is stochastic, meaning that each training iteration different neurons are dropped out. This introduces a form of noise during training process, which prevents the network from relying too heavily on any particular set of neurons. Usually, dropout is disabled during the inference or testing phase, and predictions are made using

the entire network. To compensate for the fact that more neurons were active during training, the weights of the neurons are scaled by the dropout probability. Since different subsets of neurons are active during each iteration, dropout can be understood as an ensemble learning process for neural networks. Better generalization is made possible by this ensemble effect. Dropout is especially useful while working with noisy data, providing an opportunity to reduce overfitting and increase generalization.

1.2.6 Spatial dropout

Spatial dropout is a variant of the traditional dropout regularization technique that is specifically made for convolutional layers. Spatial dropout expands on the concept of traditional dropout by randomly eliminating entire channels or feature maps from the input during each forward pass. By adding noise at the feature map level, spatial dropout discourages overfitting and promotes the learning of more resilient features. When applied to word embeddings, spatial dropout can be conceptualized as a type of "whole-word dropout." By setting all of the word's embedding values to zero during training, it eliminates entire words or tokens from the sequence as opposed to just individual elements. By dropping out entire words, spatial dropout encourages the model to learn more robust representations for words. It prevents the model from relying too heavily on specific words and helps in generalizing better to variations in the input data.

1.2.7 Batch normalization

Batch normalization is a technique to improve training stability and convergence by normalizing the input of each layer of neural network in a mini-batch. Batch normalization works by first normalizing a layer's inputs, and then using learnable parameters to scale and shift the normalized values. Every feature in the mini-batch

goes through this procedure separately. Batch normalization works in the following way (figure 1.3):

1. For each feature in the mini-batch, batch normalization normalizes the input to have zero mean and unit variance. This is done by subtracting the mean of the mini-batch and dividing by its standard deviation.

2. The results are then scaled and shifted by learnable parameters which are learned through gradient descent and back propagation.

3. The results of moving average for mean and standard deviation are updated within model.

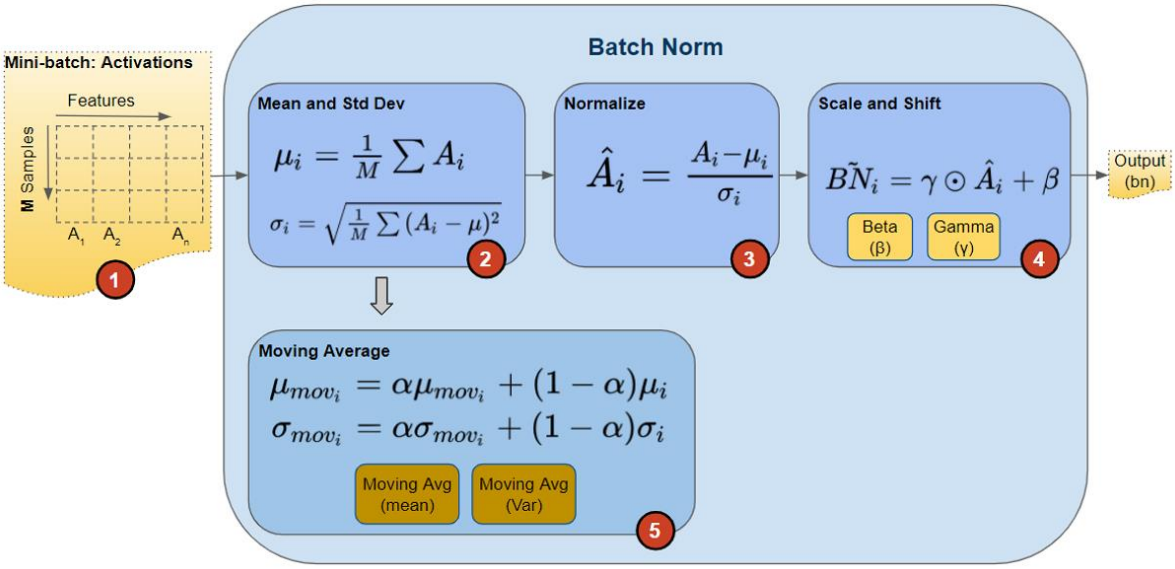


Figure 1.3 – Schematic description of computations performed by batch normalization

During inference the information of moving average for mean and standard information is used for calculations. By scaling the data batch normalization helps mitigate issues related to vanishing or exploding gradients, enabling more stable and faster training of deep neural networks. Utilization of batch normalization effectively reduces internal covariate shift, making the optimization landscape more consistent across mini-batches. Batch normalization is especially useful in the context of cross-domain data, as normalizing the inputs within each mini-batch makes model

more robust to differences between domains and reduces domain specific covariate-shift.

1.2.8 Layer normalization

Layer normalization is another technique which is used to normalize input on each layer of neural network. Layer normalization is often applied in recurrent neural networks and transformer architectures. In contrast to batch normalization, layer normalization normalizes the input along the feature dimension, typically applied independently to each example. Layer normalization works in the following way:

1. Layer normalization normalizes data along the feature dimension. For each example in the mini-batch, normalization is applied independently across all features. Formula is the same as in the case of batch normalization.

2. Same as in the case of batch normalization, layer normalization utilizes two learnable parameters to scale and shift input data.

Layer normalization reduces the dependency on batch statistics during training. Each example is normalized independently, making it less influenced by the statistics of the entire batch. In batch normalization, statistics are computed per mini-batch during training, and a moving average is typically used during inference. In layer normalization, statistics are computed independently for each example during both training and inference. Considering the fact, that in many tasks relevant to NLP sphere, the sentences with varying length are utilized, using batch normalization would result in an uncertainty relevant to appropriate normalization constant. Thus, layer normalization is recommended for use w.r.t recurrent layers.

1.3 Primer on classical machine learning for NLP

1.3.1 Logistic regression

Logistic regression is a statistical method used for binary classification, by predicting the probability that an instance belongs to a specific class. Logistic regression is trained by optimizing binary cross-entropy loss function. Logistic regression applies sigmoid function to map linear combination of input features to a value between 0 and 1, representing the probability of positive class (figure 1.4).

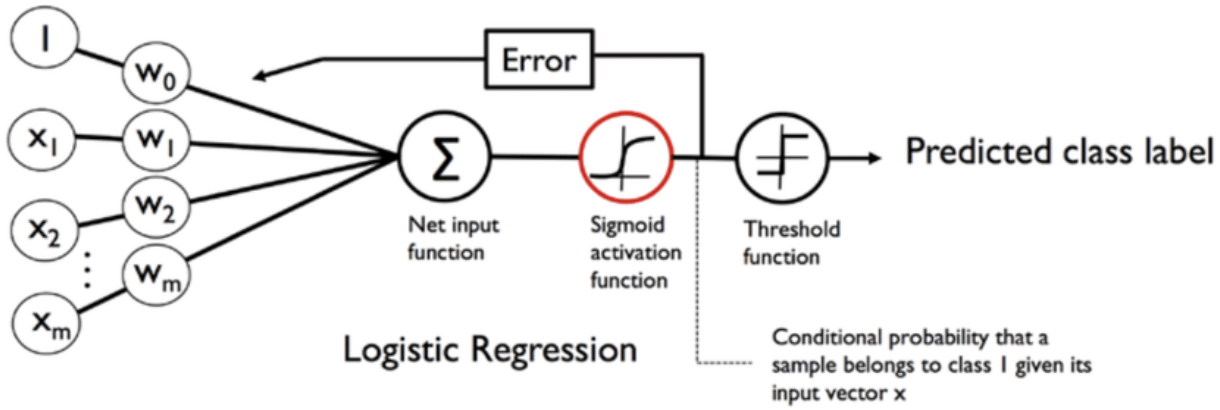


Figure 1.4 – Schematic description of logistic regression

In order to reduce overfitting, both L1 and L2 regularization can be used, with parameter C influencing the strength of it. For logistic regression the normalization of input data is beneficial, as it both speed ups training and leads to better convergence. Logistic regression assumes linear relationships between the features and log-odds of the responsive variable. Even though logistic regression is mainly used for binary classification, there are possibilities of applying algorithm for multiclass classification. One of the choices for solving the problem is utilize “One-vs-Rest” strategy. A binary classifier is trained for each class in the “OvR” strategy, treating it as the positive class and all other classes as the negative class. As a result, a set of binary classifiers is produced, one for every dataset class. An important benefit of

logistic regression is its interpretability, which helps to understand influence of features towards particular class in the global sense.

1.3.2 SVM

Support vector machine (SVM) aims to find a hyperplane in a high-dimensional space that best separates instances of different classes. SVM is often called maximum margin classifier. The margin is the distance between the hyperplane and the nearest instance from each class. SVM seeks to maximize this margin, providing better generalization to unseen data. Support vectors are the instances that lie closest to the hyperplane. These are the critical instances that influence the position and orientation of the hyperplane. The margin is computed based on the distance from the support vectors to the hyperplane. SVM has internal mechanism for dealing with overfitting. In real-world scenarios, data might not be perfectly separable, thus a "soft margin" is introduced. The regularization parameter C controls the trade-off between maximizing the margin and allowing misclassification and is similar to the same parameter used in logistic regression. SVM allows usage of specific kernel functions, that make algorithm more flexible and provide an opportunity to learn non-linear relationships. Speaking of multiclass classification, "One-vs-All" strategy can be applied. SVM is often used for text classification due to possibility of capturing non-linear relationships, which is essential for complex data. Although SVM is more robust to overfitting than logistic regression, its less interpretable, especially when using complex kernel functions.

1.3.3 Gradient boosted trees

Gradient boosted trees is an ensemble machine learning technique based on decision trees. The algorithm builds multiple decision trees sequentially, where each tree corrects the mistakes of the previous one. Each time a new tree is trained to the residual of algorithm and expected output on the previous stage. Basically, number

of trees inside ensemble can be referred as number of learning iterations and influences the complexity of resulting ensemble. The process begins with the creation of the first tree, that is called “base learner” on top which all the other trees are build. In case of gradient boosted trees, learning rate determines the contribution of each tree to the overall ensemble. Gradient boosted trees are optimized via gradient descent. One of the advantages of the algorithm is relevant to the fact that scaling of data isn’t required, due to the fact decision trees inside gradient boosted ensemble make splits based on comparison of features at individual nodes. These splits are determined by finding the optimal thresholds for features that would allow to separate data into different classes or values in the best possible way. The scaling of features doesn’t affect the order of aforementioned thresholds. Even though, gradient boosted trees is an ensemble method, it’s still very explainable. The one can analyze decision rules made by certain trees or get information of feature importance, based on how often the feature leads to split. Like in the case with logistic regression, the internal explainability is a global one. In order to prevent overfitting, gradient boosted trees utilizes such tools as tree pruning and shrinkage. Tree pruning restricts the growth of individual trees, and shrinkage controls the contribution of each tree. GBT is often utilized while working with complex data and big number of features, thus making it a candidate for problem of text classification.

1.4 Previous work analysis

1.4.1 Text classification

Recent research leverages plenty of methods for solving the tasks of classification based on textual data. The approaches can be divided into two groups based on utilized algorithms: classical machine learning and deep learning ones. Classical machine learning algorithms require thorough data preprocessing, which often includes words normalization based on lemmatization or stemming; stop-words removal and vectorization of data using TF-IDF[8]. Then processed features are used

as an input to a classifier, such as Gradient boosted trees[9], SVM[10] or Logistic Regression[11]. Nevertheless, such approaches are inferior to deep-learning ones in terms of accuracy, they are still utilized due to speed of training and inference and high interpretability.

Utsha et al.[12] apply extreme gradient boosted trees along with TF-IDF to tackle the task of multiclass fake news detection; Das et al.[13] utilize classical machine learning models on the task of sentiment analysis, showing that TF-IDF text vectorization along with NWT (Next word negation) preprocessing step and SVM achieves pretty high accuracies w.r.t three datasets. There is also a tendency of using additional textual features such as POS (Part of speech) tags[14] or NER (Named entity recognition)[15] to boost performance of models.

Other approaches suggest usage of word embeddings as a text vectorization method[16], however usage of embeddings make classical machine learning models less interpretable.

Deep-learning based methods achieve state of the art results on many benchmarks relevant to textual data input. Such methods work well especially when big data is available, as they tend to find hidden structures in text and generalize well. Embedding layer is a basis for deep-learning based approaches, as it's used to map token identifiers to real value vectors. Embeddings allowed researchers to use transfer-learning and leverage knowledge of models trained on big textual corpus for downstream tasks.

Yoon Kim[17] applied convolutional neural network on top of Word2Vec embeddings for text classification. Each convolutional layer was applied to embeddings in parallel, where number of filters was relevant to n-gram size.

Other approaches utilized more sophisticated models which are based on recurrency. LSTM and its variations are widely used for text classification nowadays. Sachan et al.[18] used simple one-layer Bidirectional LSTM along with mixed objective for training to achieve state-of-the-art results on various datasets. At the same time many researchers tend to combine CNNs with LSTMs to enhance the performance of overall model. Chunting Zhou et al.[19] proposes a C-LSTM, model which

applies one dimensional convolution right after embeddings layer to extract high-level representations, which are then fed into LSTM layer, showing superior results w.r.t other methods.

CNNs are also used right after LSTM layer, in order to aggregate and process hidden states in a non-linear way instead of just retrieving the last one. For instance, Peng Zhou et al.[20] utilize Bidirectional LSTM with two-dimensional CNN layers, which outperforms C-LSTM on five datasets.

Other researchers tend to aggregate hidden states from LSTM layer using attention mechanism. Wang et al.[21] propose model which uses LSTM along with attention mechanism to tackle the problem of aspect-based sentiment analysis. The attention combines both hidden representations of sentence tokens and aspect embeddings to produce the final output vector which is then fed into classification layer.

Recent research features approaches, based on transformers and self-attention, which are superior to others in cases when big datasets are available. Nevertheless, performance of such models is pretty stunning, they are way less explainable than those based on LSTMs and CNNs.

1.4.2 Explainable AI for text classification

Explainable AI is very important field, main goal of which is to interpret predictions made by machine learning models. Explainable AI techniques are often used to monitor performance of model w.r.t biases and promote end user trust. Explainable AI methods can be classified into three categories: Intrinsically Interpretable Method, and Model Agnostic Methods and Example-Based Explanations. One of the methods to achieve explainable AI is to use intrinsically explainable methods like logistic regression, decision trees and their ensembles. However, such explainability comes with a cost of performance.

Attention mechanism can also be considered as an intrinsically explainable method, even though it only partially explains model's results. While logistic regression and decision trees explain model's decision globally, attention mechanism provides a local perspective.

Model-agnostic methods separate explanation from a machine learning model, allowing it to be compatible with a variety of models. Model-agnostic method that is often used is surrogate-based explanations. The main idea of it is to train a simpler model on top of original model's predictions and explain the simpler one, which is called a "surrogate". Surrogate-based methods are also divided into global and local ones, as in the example regarding logistic regression and attention mechanism.

One of the famous algorithms that is build on local explainability is LIME (Local Interpretable Model-Agnostic Explanations)[22]. LIME trains an inherently interpretable model on the new dataset constructed from the permutation of samples and corresponding predictions of the model. Trained "surrogate" model can be a good approximator of global behavior, it doesn't provide a good approximation for a global one.

Shapely is another local explanation method, which is based on game theory. Main idea behind the method is based on an assumption that each feature value is a player in a game and the prediction is an overall payout that is distributed among players.

Example-Based explanations are mostly model-agnostic [23] and explain model predictions by selecting instances of the dataset and not by creating summaries of features.

There also exist approaches relevant to specifically analyzing neural networks outputs using gradient-based attribution methods [24]. However, Wang et.al [25] showed that gradient-based analysis of NLP models is manipulable, leaving a space for possible adversarial attacks.

1.4.3 Aspects ranking and unsupervised aspect-based sentiment analysis

Several works similar to ours in terms of task exist. Aspect ranking is a process of identifying important product aspects from online consumer reviews.

Yu et. al[26] presented an approach which consisted of three steps: aspect identification, aspect sentiment classification and aspect ranking. Nevertheless, the approach seemed to be effective in comparison with methods of Hu et. al [27] and Wu et. al [28], it includes the estimation of parameters for three models (2 SVMs and parameters for Gaussian distribution), which is hard to adopt to new data and can be slow during inference. Approach was shown to work for English language. In comparison, our approach only needs to train model ones and then apply explainable AI techniques to identify important aspects w.r.t labels model was trained on. As it was already mentioned, our approach can be thought of as the instance of unsupervised aspect-based sentiment analysis.

Advantages:

- Suppresses other methods of similar kind in terms of NDCG metric.
- The proposed approach consists of several steps, which broadens number of possible usages, as each step can be used separately and be integrated into other applications.

Disadvantages:

- Can be computationally extensive, as it includes training of 3 parametric models: SVM for identifying frequent terms, SVM for aspects sentiment classification and multivariate gaussian distribution for aspects ranking.
- By utilizing 3 different parametric models, its complicated to enhance algorithm.
- Due to the aforementioned reason, the utilization of algorithm to new data becomes even more complex.
- Lack of contextualization.

Garcia-Pablos et.al [29] presented an unsupervised approach to aspect-based sentiment analysis, that utilized Word2Vec model to identify aspects and detect their

polarity. The authors operated in two domains in English: restaurants and laptops reviews. In order to get list of multiwords, the log-likelihood ratio was applied. To detect entity-attributes, Word2Vec model was utilized. Based on predefined seed words relevant to entities and attributes, the sentences were annotated with most similar entity-attribute pair. Then the polarity is computed by difference of similarities between the word at hand a positive anchor word and word at hand and negative anchor word. Based on polarity of words, the sentences are labeled as either positive or negative. In comparison to aforementioned approach, the one presented in this work is more adaptable to new data and is aware of context.

Advantages:

- Fully unsupervised approach that builds on top of predefined dataset and pretrained Word2Vec model.
- Easy to use, no need for additional modeling.

Disadvantages:

- Results are equal to the baseline or worser.
- Lack of contextualization.
- Approach is not convenient for adaptation to new data sources, as there is no option of handling out of vocabulary tokens.

Hercig et.al [30] tackled the problem of unsupervised aspect-based sentiment analysis for Czech language, by breaking the task into 4 separate problems: aspect term extraction, aspect term polarity, aspect category extraction and aspect category polarity. The problems are tackled by training CRF (Conditional Random Fields) model for aspect term extraction, utilizing maximum entropy classifier for the task of aspect term polarity detection, using same type of classifier for identifying the category of aspects and their polarity. All the models operate in domain of hand-crafted features. Once again, our approach can be easily adopted for unsupervised aspect-based sentiment analysis, has fewer number of steps and is much easier to use.

Advantages:

- Achieves pretty high f1 score (around 80% for all the tasks).

- The proposed approach consists of several steps, which broadens number of possible usages, as each step can be used separately and be integrated into other applications.

Disadvantages:

- Can be computationally extensive, as it includes training of 4 parametric models.
- By utilizing 4 different parametric models, its complicated to enhance algorithm.
- Due to the aforementioned reason, the utilization of algorithm to new data becomes even more complex, as each time models should be retrained.
- Lack of contextualization.

So far, the most similar research to ours is the master's thesis of Dmytro Bobenko [31]. In his work, the author tackled the problem of determining sentiment and most influential phrases for each review. The data was collected from TripAdvisor and Booking websites, resulting into the dataset of 164k reviews. The author trained models for sentiment detection and used PMI (pointwise mutual information) to globally create dictionary of negative/positive phrases, which is then used to determine most influential phrases for each classified review. In comparison, the dataset collected in this work is cross-domain and is much bigger (662k reviews); the key phrases extraction works locally which makes it more contextualized and applicable for new data; similarly to authors we used f1-score as a main metric, however due to imbalance nature of the data the "macro" averaging was applied in contrast to "weighted", which assigns greater contribution to classes with more examples and is not representative of model performance w.r.t all the classes. Other differences are depicted further throughout the work.

Advantages:

- Approach operates in Ukrainian language and provides a model trained for sentiment classification, which can be used for further research.
- Approach utilizes clustering of n-grams in order to get the most valuable.

Disadvantages:

- Approach utilizes weighted f1-score which isn't representative of class imbalance.
- The approach works by utilizing predefined dictionary of positive and negative n-grams, which makes it harder to adopt it to new data and domains.
- For model training the plain tokenization is used, which means that there is no way of handling OOV tokens and model is less memory and time efficient.
- Approach operates only in one domain.
- Lack of contextualization in terms of key-phrases retrieval.

1.5 Conclusion

In this chapter the discussion touches on the state of problem at hand, analysis of competitor approaches and technologies used for text classification.

Recent advances in the sphere of NLP prior to text classification and explainable AI are analyzed. The emphasis is set on mechanics of embeddings, convolution for textual data, long-short-term-memory network, infamous attention mechanism and methods for model regularization and stabilization. Also, the description of classical machine learning algorithm including logistic regression, gradient boosted trees and SVM was provided.

The exhaustive analysis of similar works was provided, strong and week points of each were highlighted. So far, the most similar work to this one is of Dmytro Bobenko. However, his approach operates only in one domain, is limited for adopting to new data and enhancements and doesn't take context into account. That's why there is a strong need in creation of a more universal approach towards key-phrases (aspects) extraction.

2 ANALYSIS OF TOOLS TO ACCOMPLISH THE TASK

2.1 Analysis of word embeddings

One of the popular methods for pretraining embeddings is Word2Vec presented by Mikolov et.al [32]. Word2Vec model has several regimes of training: Skip-gram and CBOW (continuous bag of words) (figure 2.1).

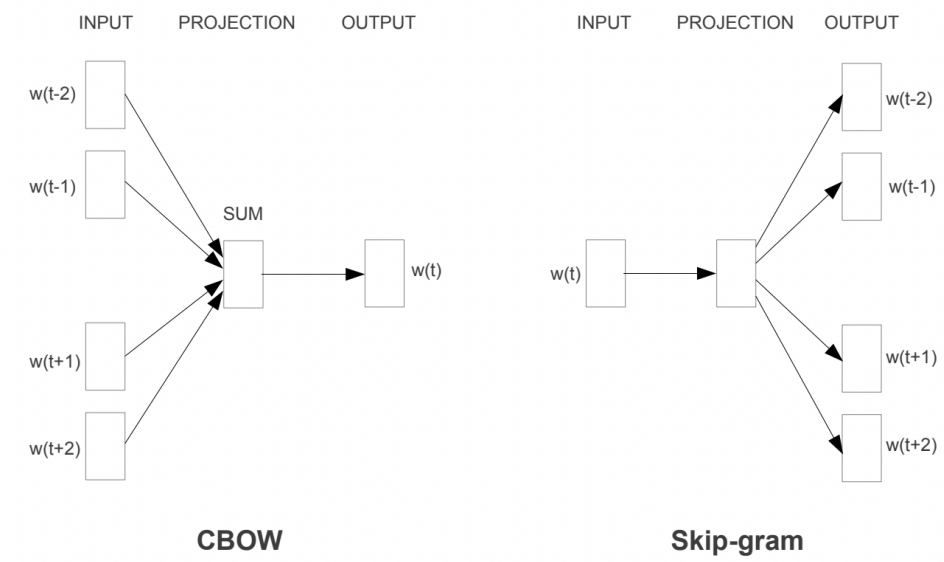


Figure 2.1 – CBOW and Skip-gram architectures of Word2Vec

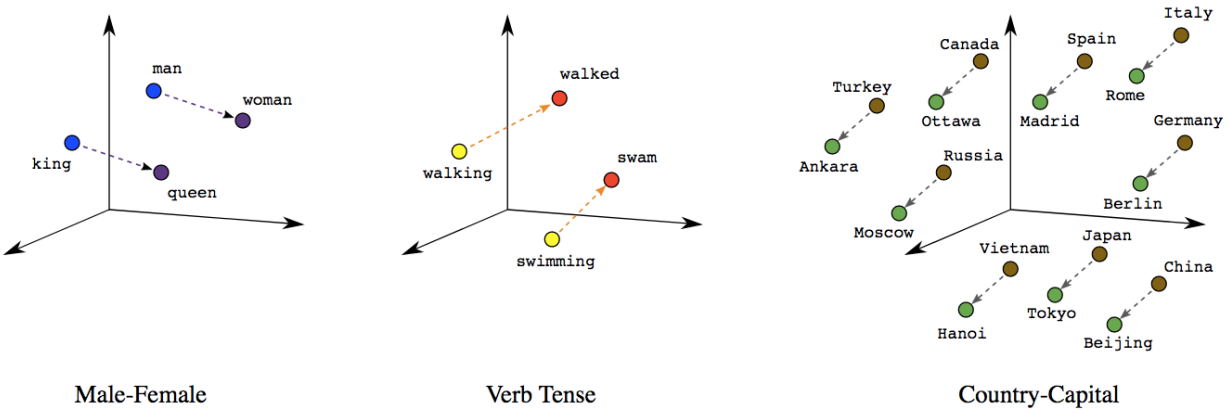


Figure 2.2 – Geometrical representation of embeddings in terms of different aspects

As it can be seen from diagram above, CBOW method is order agnostic and tries to predict middle word based on summation of project word vectors relevant to left and right context, whereas Skip-gram does the inverse and thus doesn't use any aggregation at projection stage. Such training allows to learn mathematical representations which are similar for those words which are similar in terms of their context (figure 1.2).

Advantages of Word2Vec method:

1. Efficiency: Word2Vec is efficient due to the usage of shallow neural network, making it feasible to train on big corpora.

2. Semantic relationships: Word2Vec captures relationships between words, allowing for meaningful vector arithmetic.

3. Convenience of integration: it's pretty easy to integrate learned Word2Vec embeddings into machine learning algorithm and leverage transfer learning.

Disadvantages of Word2Vec method:

1. Contextual information: Word2Vec ignores contextual information within a sentence.

2. Out of vocabulary words handling: as Word2Vec learns fixed-size vectors for each word, it's not possible to get vector representation for words that were missing in training corpora.

fastText[34] is another method for acquiring word vectors, which is an extension of aforementioned Word2Vec method. fastText represents words as bags of character n-grams, enabling to handle morphological variations and unseen words more effectively. Also, working on the level of character n-grams helps to effectively handle suffixes and prefixes.

Advantages of fastText method:

1. Semantic relationships: fastText captures relationships in a way veery similar to aforementioned method.

2. Out of vocabulary words handling: due to usage of character n-grams, fastText model is capable of handing out of vocabulary words by summing or averaging subwords vectors.

Disadvantages of fastText method:

1. Contextual information: similar to Word2Vec, fastText ignores contextual information within a sentence.

2. Increased complexity: due to the usage of character n-grams, number of tokens is increased, making training of the model more computationally expensive.

Convenience of integration: it's complicated to integrate fastText with all its features into other machine learning model.

ELMO (Embeddings from Language model)[34] is a method for getting contextual embeddings of sentences and paragraphs. Contextual information plays a crucial role in shaping the meaning of words and phrases, allowing to achieve a deeper and more accurate understanding of language. The need for contextual information comes from the problem of words ambiguity, which is relevant to same words having different meanings in different context. For instance, word "bank" could refer to a financial institution or the side of a river. ELMO applies bidirectional LSTM in order to generate the vector representation of the sentence. ELMO operates on character level, building embeddings of words based on characters that construct it. What is more, the model incorporates task-specific layers, which are added on top of the LSTM representation to adapt the embeddings to the particular requirements of the downstream task. It's interesting that ELMO utilizes a weighted sum of layers to combine information from different layers of the bidirectional LSTM.

Advantages of ELMO method:

1. Contextual understanding: ELMO captures contextual information within sentence, allowing for accurate representation of words in the case of ambiguity.

2. Out of vocabulary words handling: working on a character level, ELMO allows to handle out of vocabulary words.

Disadvantages of ELMO method:

1. Increased complexity: ELMO builds on top of multi-layer bidirectional LSTM, which increases complexity of the overall training and inference.

2. Convenience of integration: increased complexity of the algorithm directly influences convenience of its integration by making training on custom data computationally demanding.

Based on conducted analysis, it was decided to use Word2Vec method, as it's easy to pretrain it on custom dataset in a fast manner and it's convenient in terms of integration and transfer learning.

2.2 Analysis of tokenization and words normalization methods

2.2.1 Lemmatization

Lemmatization is one of the most common word normalization techniques applied in NLP. The lemmatization tries to reduce word to its root. It does so by removing inflections or variations to get to the base or dictionary form. Lemmatization typically involves dictionary look-up to identify the correct lemma for a given word. This step requires access to a comprehensive dictionary or lexicon. Lemmatization ensures that words share a common representation, facilitating more accurate analysis and understanding of textual data. Lemmatization often incorporates part-of-speech tagging to disambiguate between homographs (words with the same spelling but different meanings) and determine the correct lemma based on the word's grammatical role. It's worth mentioning that lemmatization is language dependent, meaning that for different languages different lemmatizers should be used. This downside comes from the fact that lemmatization uses dictionary-based approach to reduce words to their dictionary forms. Also, as each language has its own structures and features, it's complicated to create a universal approach that would work for each one. Lemmatization helps to reduce number of tokens (dimensionality reduction) and at the same time preserves their semantic meaning. Even though, lemmatization tend to enhance semantics of words, it can lead to loss of contextual information. Also, lemmatization is sensitive to context shift and might not capture the changing nuances and context-specific variations in language.

Advantages:

- Reduces number of words by transforming them into the lemma.
- Still preserves some semantics of words.

Disadvantages:

- Lemmaitization is language dependent.
- Can’t handle OOV tokens, thus will not help much in a context of words with typos and errors, won’t probably result in huge dimensionality reduction for complicated and noisy textual data.
- Could decrease contextualization.

2.2.2 Stemming

Stemming is another method for tokens normalization that is based on transforming words into their roots. Stemming algorithms typically work by removing or replacing word suffixes or prefixes, based on a set of predefined rules or heuristics. Some common stemming algorithms include the Porter Stemmer, Lancaster Stemmer, and Snowball Stemmer. Same as for lemmatization, stemming is language-dependent. Stemming algorithms often rely on rule-based approaches, where rules are defined to capture common prefixes or suffixes that can be removed to obtain the stem. These rules are language-specific and need to be tailored to the linguistic characteristics of each language. The most common operation in stemming is suffix stripping, where suffixes are systematically removed to obtain the root form of a word. Stemming algorithms vary in aggressiveness. Aggressive stemmers aim for higher reductions but may risk over-stemming, while conservative stemmers prioritize retaining more linguistic specificity. There also exist more advanced approaches for stemming that are based on machine learning and address some challenges, which commonly exist in traditional rule-based methods.

Advantages:

- Reduces number of words by transforming them to their root. Possibly should result in even greater reduction in number of unique tokens then lemmatization.
- Computationally less expensive than lemmatization.

Disadvantages:

- Stemming is language dependent.
- Might produce incorrect stems for OOV tokens, leading to inconsistent and incorrect representations.
- Lack of contextualization, which holds same for stemming as for lemmatization.
- Could lead to problem of over-stemming, which can result in words that have different meaning having the same stem token.

2.2.3 BPE

BPE (Byte-Pair-Encoding) is a compression technique that found a big popularity in application to NLP sphere. BPE allows to tokenize text into subwords, which is a solution between word and character-based tokenization. BPE operates at the subword level, breaking down words into smaller units or subword tokens. The central idea behind BPE is a compression principle where frequent sequences of characters are progressively replaced with a single, unused token. This process is iteratively applied until a specified vocabulary size is reached. BPE ensures that the most common words are represented in the vocabulary as a single token while the rare words are broken down into two or more subword tokens. BPE tokenization process can be simply described by the following stages:

1. Start with a vocabulary containing individual characters and their frequencies.

2. Identify the most frequent pair of adjacent characters in the current vocabulary. Merge this pair into a new token.

3. Update the vocabulary with the newly created token. Repeat this process iteratively until the desired vocabulary size is achieved.

As it was mentioned during the description of BPE tokenization process, BPE provides a possibility to preselect number of tokens, thus controlling the complexity and dimensionality of data. By working on a subword level, BPE is effective when working with noisy data that includes word with typos and errors. After tokenization using BPE, word can be presented as representations of their subwords, giving an opportunity to have similar vector representation for same word with typos.

Advantages:

- Incorporated functionality for choosing number of unique tokens, allowing for flexibility and effective dimensionality reduction.
- Handling OOV tokens and unseen data.
- Possibility of handling noisy data.
- Language independent.

Disadvantages:

- A need to train an algorithm in order to learn subwords, which makes BPE sensitive to training data.
- Increased sequence length because of subword tokens.

2.3 Analysis of programming languages

2.3.1 Java

Java is a multi-platform, object-oriented, and network-centric language that can be used as a platform in itself. It is a fast, secure, reliable programming language for coding everything from mobile apps and enterprise software to big data applications and server-side technologies. Java's performance is significantly enhanced by its use of Just-In-Time (JIT) compilation. This means that Java code is compiled into machine code at runtime, allowing for optimizations that can boost execution speed.

Java's memory management through garbage collection contributes to stable and efficient memory handling. While it introduces some overhead, modern Java Virtual Machines (JVMs) are optimized to minimize the impact on performance. Java benefits from a mature and extensive ecosystem. This ecosystem includes tools, libraries, and frameworks that can aid in optimizing and accelerating deep learning applications. Even though Java may not be as commonly associated with deep learning frameworks as languages like Python, its performance characteristics make it a viable choice in certain scenarios. Java can seamlessly integrate with optimized native libraries, enhancing its numerical computation capabilities. Leveraging libraries like MKL or OpenBLAS can significantly boost the speed of mathematical operations crucial in deep learning. Java provides many libraries that can be used to accomplish tasks relevant to machine learning, data analysis and deep learning including famous Tensorflow, Weka and Deeplearning4j.

Advantages:

- Platform independence assures that applications written in Java can be deployed on various platforms with no modification.
- Java is known for its scalability, which is essential for distributed computing.
- Java's native support for multithreading is beneficial for concurrent execution, making it suitable for tasks like parallelizing deep learning model training.
- Java has many libraries for accomplishing machine learning and data analysis tasks.

Disadvantages:

- Java is not as prevalent as Python in the deep learning community. Many deep learning frameworks and tools are primarily developed and optimized for Python, potentially limiting the available resources and community support for Java.
- Java may be perceived as slower for prototyping compared to dynamically typed languages.
- Complexity in doing dynamic data analysis.

2.3.2 Julia

Julia is a dynamic, high-level, and high-performance programming language designed for technical computing. It was created to address the need for a language that combines the ease of use of scripting languages with the performance of lower-level languages, making it well-suited for numerical and scientific computing tasks. Julia is renowned for its performance, often reaching levels comparable to languages like C and Fortran. This is achieved through just-in-time (JIT) compilation, allowing Julia code to be dynamically compiled to machine code for efficient execution. Julia features multiple dispatch, a programming paradigm that enables highly expressive and generic code. Functions can be specialized for different types and combinations of types, promoting flexible and extensible designs. Julia's syntax is designed to be readable and resembles mathematical notation, making it accessible to users from diverse backgrounds. Its user-friendly syntax contributes to fast development and prototyping. Julia has built-in support for distributed and parallel computing. This enables users to scale their computations across multiple processors or nodes, making Julia suitable for handling large-scale and computationally intensive tasks. Julia is built with interoperability in mind. It can easily interface with other languages like C, Fortran, and Python, facilitating integration with existing libraries and tools. All the aforementioned features of Julia make it a candidate for accomplishing the tasks which were already highlighted in the work.

Advantages:

- Julia is designed with a focus on performance, often comparable to low-level languages like C or Fortran. This makes Julia well-suited for high-performance computing tasks, including deep learning.
- Julia provides a syntax that is easy to read and write, resembling mathematical notation. This can contribute to faster development and prototyping, crucial aspects in the iterative process of deep learning model creation.

- Similar to Java, Julia uses JIT compilation. This allows for efficient compilation of code, making it performant and adaptable to different hardware architectures.
- Julia has libraries designed specifically for deep learning, whereas its interoperability with other languages provides even more tools.

Disadvantages:

- While Julia's community is growing, it is not as extensive as communities for languages like Python. This can result in fewer resources, tutorials, and community-contributed models compared to more established languages.
- Julia's deep learning ecosystem, while promising, might not be as mature as those in languages like Python. Well-established frameworks like TensorFlow and PyTorch have larger user bases and extensive documentation.
- Julia is still gaining traction in industry applications compared to more established languages like Python. This could influence the choice of Julia for deep learning in certain enterprise settings.

2.3.3 C++

C++ is a general-purpose programming language that extends the capabilities of the C programming language. Developed by Bjarne Stroustrup, C++ combines procedural, object-oriented, and generic programming features, making it a versatile and powerful language. C++ is widely used in various domains, including system programming, game development, embedded systems, and high-performance computing. C++ is known for its high-performance capabilities. It allows low-level memory manipulation and provides features like pointers, making it suitable for applications that demand efficient resource utilization. C++ supports object-oriented programming, enabling developers to organize code into classes and objects. This paradigm promotes code reuse, modularity, and a clearer organization of software components. C++ code can be highly portable across different platforms and operating systems. This makes it a preferred choice for applications that need to run on

diverse environments. Speaking of its usage for deep learning purposes, C++ can be used to implement performance-critical components, taking advantage of its efficient memory management and low-level capabilities which is essential for deep learning.

Advantages:

- C++ is very performance efficient and is suitable for implementing performance-critical components for deep learning models.
- C++ provides manual memory management, allowing developers fine-grained control over memory. This is beneficial for optimizing memory usage in deep learning applications.
- C++ provides libraries relevant to deep learning, which are specifically optimized for the language.

Disadvantages:

- Usage of C++ may result slower and more complex prototyping because of lack of dynamic features.
- It’s not very convenient to apply data processing and analysis using C++.

2.3.4 Python

Python, created by Guido van Rossum and first released in 1991, is a high-level, general-purpose programming language known for its readability, versatility, and extensive community support. Its design philosophy prioritizes code readability and ease of use, making it an excellent choice for beginners and a powerful tool for professionals across various domains. ython's syntax is designed to be clear and readable, emphasizing code readability and reducing the cost of program maintenance. This feature contributes to Python's popularity among developers. Python is an interpreted language, allowing for rapid development and testing. Its dynamic typing provides flexibility but also necessitates careful consideration of variable types during execution. Python abstracts many complex operations, allowing developers to focus on solving problems rather than managing low-level details. This

high-level approach accelerates development and promotes code simplicity. Python is inherently cross-platform, enabling code written on one operating system to run on others with minimal modifications. This portability enhances the versatility of Python applications. Python has huge community and lots of use cases, including: web-development, data science and analysis, scientific computing, automation and machine learning along with deep learning. Python has huge number of libraries that extend its capabilities, including deep-learning specific libraries like Tensorflow, Pytorch, Caffe and others. Despite Python seems as the best candidate because of convenience of code prototyping and its dynamic nature, it all goes with the cost of reduced performance and higher memory consumption.

Advantages:

- Python is a dynamic language, which provides more convenience in terms of dynamic data analysis, data preprocessing and algorithms prototyping.
- Python has a large and active community in the deep learning space. This vibrant community contributes to a wealth of tutorials, documentation, and shared knowledge, fostering collaborative development and problem-solving.
- Python, similarly to Julia provides syntaxis which easy to write and read, enhancing its prototyping abilities even more.
- Python has many libraries to speed up and vectorize computations, perform complex data analysis tasks and do modeling.

Disadvantages:

- Python's Global Interpreter Lock (GIL) can limit the concurrency of multi-threaded programs.
- Python's resource consumption may be higher compared to languages like C++ in certain scenarios.
- Python's interpreted nature can lead to slower execution speeds compared to compiled languages like C++.

Based on analysis of programming languages, Python was chosen due to previous familiarity with it, huge number of libraries and its dynamic nature.

2.4 Analysis of modelling frameworks

Speaking about the process of modelling, in particular constructing the architecture of algorithm, its training and evaluation, the choice of framework is crucial. Nowadays, most popular frameworks for modeling and working with deep-learning are TensorFlow[35] and PyTorch[36]. While TensorFlow and PyTorch are general purpose deep-learning frameworks, there are specialized frameworks for addressing tasks in NLP sphere, such as Gensim[37], HuggingFace[38] and others. Speaking of classical machine learning, a very common library is scikit-learn. While scikit-learn has decent implementations of classical machine learning algorithms, there specialized libraries that contain advanced implementations of specific algorithms, like XGBoost[39] and LightGBM[40].

2.4.1 TensorFlow

TensorFlow views models as DAGs (directed acyclic graphs) and follows the idiom of "*data as code and code as data*". TensorFlow is designed for scalability, enabling the development of models that can seamlessly transition from local environments to distributed systems. This scalability is crucial for handling large datasets and training complex models. TensorFlow includes TensorBoard, a powerful visualization tool. It allows users to visually inspect and analyze various aspects of their models, such as training curves, computational graph structures, and embeddings. TensorFlow supports deployment across various platforms, including mobile devices and embedded systems. This makes it suitable for developing applications in diverse environments. TensorFlow uses symbolic computations, thus before making any, the overall graph of mathematical operations is constructed. This specific feature is the one that makes the framework faster, as at the runtime the defined graph computations are run using optimized C++ code. However, such feature comes with a cost. It's pretty hard to debug TensorFlow computations, graph construction, and

operations flow. Luckily, a Keras [41] framework exists, which is an abstraction on top of TensorFlow, that makes modelling much easier and faster.

Advantages:

- TensorFlow has a wide community that supports it. Its community-driven development and continuous updates contribute to its status as a leading deep learning library.
- TensorFlow seamlessly integrates with Keras, a high-level neural networks API. This integration combines TensorFlow's power with Keras's simplicity, providing an accessible interface for rapid model prototyping.
- TensorFlow is a computation-efficient framework, which is essential for deep learning application.
- TensorFlow has specific functionality for optimizing models during and after training in terms of their size and latency.

Disadvantages:

- TensorFlow can be resource-intensive, demanding substantial computational power and memory. This might be a consideration in scenarios with limited resources or for edge device deployments.
- While TensorFlow supports dynamic computation graphs, the dynamic graph mode may have some limitations compared to libraries that inherently operate with dynamic graphs.

2.4.2 PyTorch

PyTorch is a framework introduced by Facebook, which is and extension of lua-based framework Torch. It can be thought of as the main competitor of TensorFlow. It presents models in the same manner as TensorFlow, but instead of symbolic definitions, PyTorch works dynamically, giving a possibility to change the architecture and execute different mathematical operations on the fly. This feature allows to debug architectures and code more easily than with TensorFlow, which makes PyTorch better for research comparing with pure TensorFlow framework. Recent

versions of PyTorch come with a bult-in frameworks for model optimization for low-resource machines and a possibility of using pre-trained models. PyTorch integrates seamlessly with NumPy, a widely used numerical computing library in Python. This interoperability simplifies data manipulation and encourages a smooth transition for users familiar with NumPy. PyTorch includes torchvision and torchtext, specialized libraries for computer vision and natural language processing (NLP), respectively. These libraries offer pre-built components and datasets for common tasks. The overall trend of using PyTorch is for the sake of research modelling and production where non-functional requirements are not very demanding.

Advantages:

- The dynamic computational graph in PyTorch is well-suited for tasks where the model architecture changes dynamically, offering more flexibility for researchers and developers.
- PyTorch has a strong and active community. The community-driven development model contributes to regular updates, extensive documentation, and a wealth of tutorials.
- As in case of TensorFlow, PyTorch has functionality for model optimization.
- PyTorch has many libraries built on top which make it easier to prototype models and train them.

Disadvantages:

- For certain production scenarios, the static computational graph used by TensorFlow and other frameworks may offer advantages in terms of optimization and deployment.
- Similar to TensorFlow, PyTorch can be resource-intensive, demanding substantial computational power and memory.

2.4.3 HuggingFace

Both TensorFlow and PyTorch are often considered as core frameworks, on top which the other are built. For example, a popular library for pretraining Transformers architectures, named HuggingFace is built on top of both PyTorch and TensorFlow, giving a possibility of using features of core libraries along with functionality of HuggingFace. HuggingFace provides a wide range of pre-trained models for tasks such as text classification, language translation, summarization, and more. These models are based on transformer architectures and achieve state-of-the-art performance on various benchmarks. Hugging Face operates a Model Hub, a platform for sharing and discovering models. Users can easily access and download pre-trained models for their specific NLP tasks, fostering collaboration and knowledge sharing. Hugging Face offers tokenization tools that efficiently handle the processing of text into tokens for input to machine learning models. The tokenizers support a variety of languages and tokenization strategies.

Advantages:

- HuggingFace provides access to state-of-the-art pre-trained models, including BERT, GPT, and others. These models have achieved top performance on various NLP benchmarks and tasks.
- The strong community around Hugging Face contributes to a collaborative and dynamic development environment.
- The Model Hub serves as a central repository for pre-trained models, making it convenient for users to discover, share, and access models for various NLP tasks. This centralization enhances collaboration and model reuse.
- Hugging Face's tokenization tools are efficient and versatile, supporting a wide range of languages and tokenization strategies. This aids in preprocessing text data for input into machine learning models.

Disadvantages:

- Keeping track of model versions and updates may pose a challenge, especially as the library evolves.

2.4.4 Scikit-learn

Scikit-learn is an open-source machine learning library for Python that provides simple and efficient tools for data analysis and modeling. It is built on NumPy, SciPy, and Matplotlib and offers a wide range of machine learning algorithms for classification, regression, clustering, dimensionality reduction, and more. Scikit-learn includes a broad selection of machine learning algorithms, ranging from simple and interpretable models like linear regression to more complex methods such as support vector machines and ensemble methods. It also provides tools for data preprocessing, including methods for handling missing values, scaling features, encoding categorical variables, and splitting datasets into training and testing sets.

Advantages:

- Scikit-learn is designed with simplicity and ease of use in mind. Its consistent API and clear documentation make it accessible for users at different skill levels, from beginners to experienced practitioners.
- The library is versatile and applicable to a wide range of machine learning tasks, including classification, regression, clustering, and dimensionality reduction. It is suitable for both small-scale projects and larger, more complex applications.
- Scikit-learn has a large and active community.

Disadvantages:

- Scikit-learn is primarily focused on traditional machine learning algorithms and may not provide the same level of support for deep learning methods.
- While suitable for many tasks, scikit-learn may face limitations in terms of scalability for extremely large datasets.

2.4.5 Xgboost

XGBoost (Extreme Gradient Boosting) is an open-source machine learning library designed for gradient boosting frameworks. Developed to optimize speed and performance, XGBoost is widely used for supervised learning tasks, particularly in

structured/tabular data settings, and has gained popularity for its efficiency and effectiveness in predictive modeling. XGBoost employs a gradient boosting framework, which builds a strong predictive model by combining the outputs of multiple weak models, typically decision trees. This ensemble approach enhances predictive accuracy and generalization. XGBoost is designed for efficient parallel and distributed computing. It leverages features such as column block-ing for parallelization, making it suitable for large datasets and speeding up training times. XGBoost includes tree pruning methods that control the depth of individual decision trees. Pruning mitigates the risk of overfitting and contributes to the overall efficiency of the algorithm. One particular difference between XGBoost and other implementations of gradient boosted trees lies in the way XGBoost construts trees. XGBoost follows a leaf-wise growth strategy. In each iteration, it selects the split that offers the maximum reduction in loss, leading to imbalanced trees where certain branches can be deeper than others. This strategy is more aggressive in finding the optimal splits but can be computationally expensive.

Advantages:

- XGBoost is known for its high performance and efficiency. It often outperforms other machine learning algorithms and is particularly well-suited for structured/tabular data.
- XGBoost provides insights into feature importance, helping users understand which features contribute the most to model predictions.
- The efficient parallel and distributed computing capabilities of XGBoost make it scalable and well-suited for large datasets.

Disadvantages:

- XGBoost can be resource-intensive, especially in terms of memory usage.
- While XGBoost is powerful, tuning its hyperparameters can be complex.

2.4.6 Gensim

Gensim is an open-source library for unsupervised topic modeling, document similarity analysis, and other natural language processing (NLP) tasks. It is designed to efficiently process large corpora and build scalable models for semantic analysis, particularly in the context of vector space modeling. Gensim supports the training of word embeddings using techniques like Word2Vec. It's designed for scalability and can efficiently process large corpora. It implements streaming algorithms, enabling users to process and analyze documents in a memory-efficient manner. Moreover, Gensim can be easily integrated into NLP pipelines and workflows. It provides a user-friendly interface for training models, transforming text data, and extracting semantic information. Gensim can be easily integrated into NLP pipelines and workflows. It provides a user-friendly interface for training models, transforming text data, and extracting semantic information. The core algorithms in Gensim use battle-hardened, highly optimized & parallelized C language routines.

Advantages:

- Support for Word2Vec that allows users to train word embeddings, capturing semantic relationships between words.
- Gensim is designed to handle large datasets efficiently. Its streaming algorithms enable users to process extensive corpora without requiring the entire dataset to be loaded into memory.
- Gensim benefits from an active community of researchers and developers.

Disadvantages:

- Lacks implementations of more sophisticated embedding models based on Transformers architectures.
- Gensim is limited to topic modeling and word embeddings and doesn’t provide as extensive a set of components for comprehensive NLP tasks as some other libraries.

Tensorflow was picked as the main framework for modelling of deep learning architectures; Gensim was used for pretraining of Word2Vec embeddings; Scikit-

learn for training of logistic regression and XGBoost for utilization of Gradient-boosted trees. Finally, HuggingFace was picked for storing data and models along with performing text tokenization.

2.5 Conclusions

The comprehensive analysis of tools for accomplishing the aforementioned tasks was conducted in this chapter. As the result of embeddings analysis, it was decided to stick to Word2Vec approach because of its simplicity and effectivity in training and adaptation to more complex models. The discussion touched on comparative analysis of tokenization techniques, showing that BPE suppresses lemmatization and stemming in terms of OOV tokens handling and provides a possibility to tremendously reduce number of the unique tokens. The analysis of programming languages showed that Python is the best choice for solving the task at hand. Finally, the analysis of libraries for modeling was performed.

3 DATA COLLECTION, ANALYSIS, PROCESSING AND FILTRATION

As it was already mentioned, in comparison with Bobenko's work, it was decided to gather bigger dataset that include more domains of reviews. In particular, the decision was to acquire reviews relevant to three domains: hotels, restaurants and products. When working with real-world textual data, it's mandatory to pay attention to mistakes in words, specific characters and symbols, text which is not review but rather a question, etc. In this chapter, the data collection, analysis, filtering and processing is discussed.

3.1 Data collection

The data was parsed from two websites TripAdvisor and Rozetka. Parsing which is often referred to as web scrapping is the process of data extraction from websites and its transformation into structured data. In order to parse big amounts of data without being banned, a number of techniques were used, including: user-agent rotation, proxy server and different time intervals between scrapping.

User-agent header is just one of many request headers, main goal of which is to provide information about request sender's device, including information of device type, operating system and browser name and version. Nevertheless, historically main purpose of user-agent is to optimize served content for different devices, nowadays websites mostly use this datapoint for tracking. Websites often analyze user-agent headers to determine whether the request sender is a real user or a bot. Mostly websites ban those requests which lack user-agent information. There also situations when sender is banned because of too much requests with same user-agent information. Either way, the best approach is to use user-agent rotation, the procedure when user-agent information is changed every request. While scrapping data from TripAdvisor, for each page of hotel/restaurant a different user-agent was used. Also, each time "Access Denied" message was received, user-agent was substituted with a new one.

Generally, every website has a specific limit of requests per amount of time. When limit of requests is exceeded, the sender IP-address is added to blacklist, which makes it impossible to work with website using same IP again. There is also a possibility of captcha appearance, when too much requests are sent from same IP, which is pretty difficult to solve. In order to by-pass permanent ban by IP, proxy is used. The main goal of proxy is to hide device's real IP-address. While visiting the website with proxy, only IP-address of proxy is exposed. For proxy and IP rotation, the stem library was used. Stem is a controller on top of Tor browser (which is also utilized as a proxy server), which gives a possibility to execute commands from Python. Each time "Access Denied" message was received, IP address was changed using Tor.

To speed up data collection, multiprocessing w.r.t each entities page was applied. As both Rozetka and TripAdvisor required few manual interactions, including button clicks to go to the next page and to show full review text, the automation was required. To automate manual work while scrapping websites, Selenium was used. Selenium an open-source automated testing framework, used to validate web applications across different browsers and platforms. The main advantage of Selenium is that it's a cross-platform framework, that can control browser from OS level using WebDriver. The point of WebDriver is to control browser directly by communicating with it. Nevertheless, the Selenium is mostly used for automation of testing routines, its functionality allows to automate manual interactions which are mandatory for advanced parsing. Before clicking on the button, Selenium should be given the information regarding button's location on the page. The search for buttons was made through class names and XPATH.

For TripAdvisor the information of only Ukrainian hotels and restaurants was parsed. Data scrapping process for both sites was split into two different steps:

1. Acquiring general information about each entity, including its name, overall rating and link to its page.

2. Acquiring deeper information relevant to reviews based on already parsed links.

During second step of web scrapping, the whole html pages were saved and then parsed using bs4 (BeautifulSoup) library. BeautifulSoup is a Python framework used for parsing HTML and XML documents using parse tree. Each time a new page was opened, or a specific action was performed, the sleep function was used in order to give browser time to load all the resources. The algorithm for scrapping included specific rule-based logic, used for navigation over website and receiving of full information about each review. The algorithm included the following steps:

1. The first page is opened. If information for some pages is already parsed, scrapping continues from where it was left off.

2. Check whether access to page is denied. If so, IP address along with user-agent is changed and the check for access denied is retried. In case when "Access denied" is encountered for more than N times, the algorithm stops. In other cases, algorithm continues.

3. Checkmark to show all languages is clicked.

4. Checkmark to show full reviews is clicked.

5. Check for pop-up window. If pop-up window is detected it's closed by clicking on specific button.

6. Page's html is saved to disk.

7. Check for "Next page" button availability. If button is available, it's clicked and process from 4-7 is repeated. In the other case, the algorithms stops.

In result, the dataset containing 671k reviews was collected. Nevertheless, many complementary information was parsed, the primer focus was set on the following columns:

- reviews_text – parsed text of original reviews.
- dataset_name – name of domain dataset.
- entity_name – name of unique hotel, restaurant or product for which review was written.
- rating – rating of review.

A few records from resulting dataset are shown (figure 3.1).

	review	dataset_name	entity_name	rating
59204	Мягкий, натуральный. На заявленные размеры мат...	rozetka	andersen_cotton_plus_cmp204	5.0
103374	Сподобався. Поки ще не пройшов випробування мо...	rozetka	electro_tf320s	5.0
49760	Была в этом месте первый раз, впечатления хоро...	tripadvisor_restaurants_ukraine	Restaurant_Review-g681193-d12182382-Reviews-La...	5.0
39347	Качественная и очень плотная.Но с нее тяжелее ...	rozetka	freken_bok_14801080	4.0
261413	Наушники супер. Звук очень достойный, басы и в...	rozetka	35734	5.0
149506	L'endroit est tr?s bien situ? , tr?s sympa , l...	tripadvisor_restaurants_ukraine	Restaurant_Review-g294474-d10717273-Reviews-Ko...	2.0
77123	Огромный плюс расположение. Выходим на улицу, ...	tripadvisor_hotels_ukraine	Apartment Club ZimaSnow Ski & Spa	5.0
77560	Дуже класний набір.\nЧашка бомба, чаї дуже сма...	rozetka	lovare_4820198877231	5.0
167806	Отличные носовые платки!Из доступных наверное ...	rozetka	zewa_7322540352313	5.0
238380	Понравилось качество товара. Добротно, без нар...	rozetka	255970641	5.0
88662	Кислятина, при этом абсолютно неострый. Даже и...	rozetka	tabasco_11210009493	2.0
36040	Все добре.\n	rozetka	48229646	5.0
38342	Із заявленого комплектування не було подушки-р...	rozetka	evo_kids_evo_18_bl	2.0
89807	Непоганий протеїн, шоколадний досить добрий н...	rozetka	ab_pro_pro2000abva79	4.0
86247	Купил его летом 2017 года, ровно год пользовал...	rozetka	9949118	2.0
8650	Патрик Паб - хороший паб в жилом доме недалеко...	tripadvisor_restaurants_ukraine	Restaurant_Review-g294474-d10209611-Reviews-Pa...	4.0
119898	I was dreaming for macaroons and eclers, we ca...	tripadvisor_restaurants_ukraine	Restaurant_Review-g295377-d11827697-Reviews-Do...	5.0
145818	We spent few days here in Kyiv and one place w...	tripadvisor_restaurants_ukraine	Restaurant_Review-g294474-d10593831-Reviews-La...	5.0
18707	Дуже сподобався напій,м'який з насиченим смако...	rozetka	jacobs_4820187049359	5.0
71476	Мышкой пользовался больше года, год активного ...	rozetka	hator_htm_310	3.0

Figure 3.1 - 20 random samples drawing from originally collected data

3.2 Data preprocessing and analysis

Analyzing the collected dataset, it was found that similarly to the work of Bobenko, parsed textual data was multi-lingual, including, Russian, Ukrainian and other languages (19% to Ukrainian and 81% of reviews relevant to other languages). What is more, TripAdvisor don't support Ukrainian language at all, thus all the reviews relevant to hotels and restaurants domains were in other languages. To tackle this problem, the translation process was automated by utilizing Microsoft translation in Word application. Each text of review was copy-pasted to Microsoft Word document and separated by newline character, after which the process of translation into Ukrainian was executed. As full automation could still result in errors and incorrect translation, reviews were automatically filtered. Analyzing the distribution of characters number in the translated reviews, it was found that some of them had only 1 character and thus were filtered out (appendix B.2).

Logically if the difference between number of characters in original review and its translation is too big, translated review could be incorrect or incomplete. Those reviews for which the difference was bigger than 200 characters were filtered out.

As the possibility of partial translation on the level of sentences existed, it was decided to detect and filter out such cases. To achieve this, a fasttext model for language detection was utilized. The model was trained in a CBOW (continuous bag of words) fashion and utilized hierarchical softmax to speed up computations. The input to the model was n-gram text, representation of which was averaged before the classification layer. Fasttext model is capable to detect text in 176 languages and was trained on corpora from Wikipedia, Tatoeba and SETimes. What is more, the trained model is compressed by applying product quantization, which approximates a real-valued vector by finding the closest vector in a pre-defined structured set of centroids, making it light-weight and rapid in terms of inference. Translated reviews were tokenized into sentences and for each sentence the language was detected using aforementioned fasttext model. Based on this information, partially translated reviews were filtered out.

Each sentence was tokenized into words using special tokenizer for Ukrainian language that tolerated both apostrophe and hyphen characters. In order to reduce vocabulary and normalize tokens, a specific preprocessing that separated letters from symbols was used. In particular, the space was added between each combination of letters, symbols and numbers (figure 3.2).

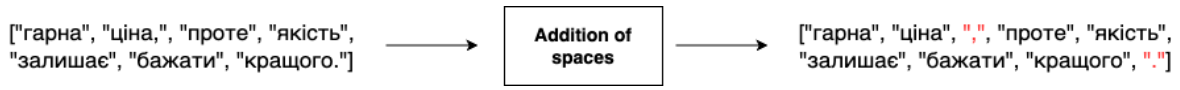


Figure 3.2 – Example of tokens before and after addition of spaces. New tokens are shown in red.

It’s worth mentioning that after words-based processing, tokens were converted back to sentences in order to allow for other types of tokenization.

As some of the reviews could be questions about hotels, restaurants or products, specific heuristic to determine questions based on POS (part of speech) tags was applied. POS tags were detected using pymorphy2[42] library (appendix B).

Found questions were filtered out from the dataset.

Other preprocessing included deletion of multi-spaces, removal of a newline character, lowercasing and lemmatization that was only used for classical machine learning methods. Applied preprocessing resulted in a reduced dataset consisting of 662907 reviews. Dataset included 364935 unique words and 205161 unique lemmas. Entity name is an essential categorical feature which is used further for final algorithm of key phrases retrieval. There are more than 28k of unique entities with the median number of reviews equal to 7. The data can be logically split into subsets w.r.t domains (dataset_name column) and whether the text was translated or not (translated column). In terms of distribution w.r.t domains, 60% of data is relevant to products, 28% to restaurants and 12% to hotels reviews. Analyzing the distribution of ratings, it's clear that it's far from even (figure 3.3).

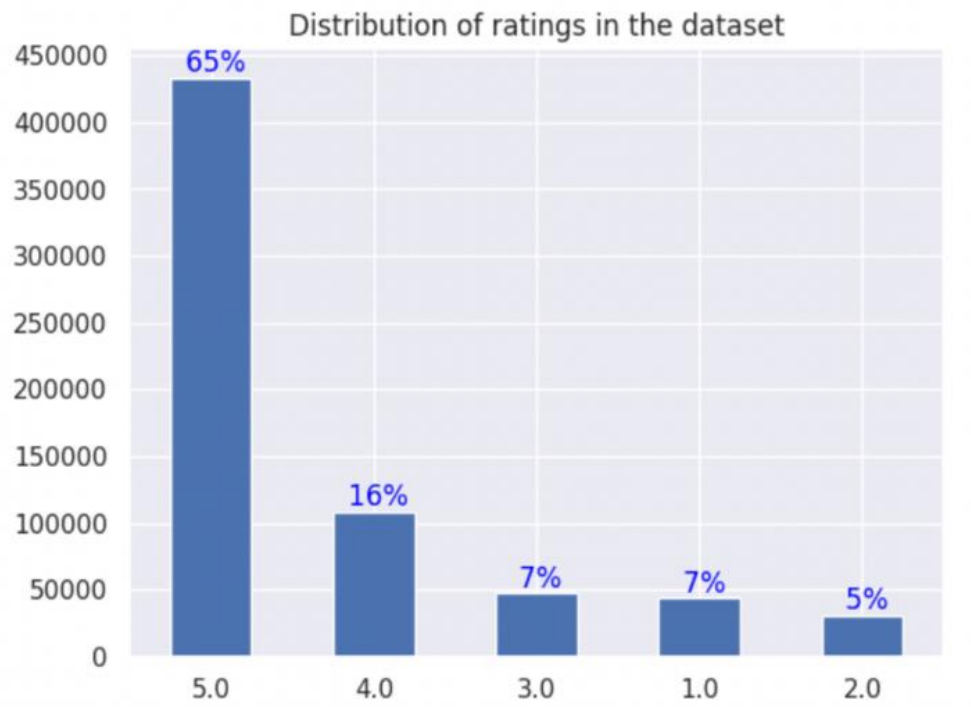


Figure 3.3 - Distribution of ratings across all domains

As the result of analysis, the following conclusions were made:

1. The fact that number of unique words is pretty huge implies filtering of stop-words for classical machine learning algorithms and usage of specific tokenizers for deep- learning based methods to reduce number of tokens.

2. The fact that distribution of ratings is imbalanced, implies usage of specific techniques to stabilize training procedure and correctly evaluate model performance.

3. The fact that distribution of domains across the dataset is not even and major part of reviews are translated can cause model to overfit to one domain. Thus, it was decided to conduct evaluation of algorithms w.r.t each domain and translated identifier category.

3.3 Data filtering

The experiments were conducted w.r.t both classical machine learning models, in particular logistic regression and gradient-boosted trees and deep-learning based ones, which utilized convolution, recurrent and attention layers. While training the models, the issue of noisy data was faced, which was caused by the subjectivity of user's ratings and discrepancy between the actual text of review and its rating. Such a problem typically arises while working with human generated data. Thus, in order to filter out misleading data samples, an automotive approach was used.

Models with different architectures were picked and trained on dataset in a cross-validation manner, so that each model could generate predictions for each K fold, while being trained on K-1 folds. Cross validation is a statistical method that is often used for model selection, as it gives an opportunity to estimate model's skill to generalize to the data given different partitions of it as a training one. In K-fold cross validation, the data is partitioned into K equally sized segments or folds. The validation is then executed by repeatedly training model on K-1 folds and validating on K fold. During such a procedure, the metric picked for evaluation is computed for K fold and preserved for aggregation. Typically, mean or median aggregation is used to get information of general performance of model towards K different folds.

In the setting of imbalance classes, it's important to use stratified cross validation, that ensures equal distribution of classes across folds.

For filtration, a stratified K-fold strategy was used with K equal to 5, meaning that each time model was trained on 80% of data, while being validated on the rest 20%. After generating predictions for each sample using N different models, those samples for which all the models made incorrect prediction were analyzed and filtered out.

The logic behind the filtering was in the fact that different models would learn distribution of data w.r.t target differently, but would make same mistakes for outliers. It's worth mentioning that a probabilistic variation of an algorithm for filtration exists. In probabilistic setting, those samples for which predicted probability distribution of classes across all the models is not picked, but is whether close to uniform are deleted, as such probability distribution implies that model is unconfident in its predictions. It was empirically discovered that majority of analyzed samples were mislabeled and had discrepancy between review text and rating (figure 3.4).

Review (Ukrainian)	**Review (English)**	**Rating**	**Expected rating**
Дуже рекомендую !	Highly recommend !	1	4-5
Відмінна ціна))) На акцію ! ! !	Excellent price))) On sale ! ! !	3	4-5
Якщо ви потрапите в це місце , будь те готові до того , що через 15 хвили н після замовлення вам подадуть апе ритив або коктейль , і все буде смач но . Акцій і подарунків дуже багато , тому якщо ви не поспішаете , ви тут !	If you get to this place, be prepared for the fact that 15 minutes after ordering you will be served an aperitif or a cocktail, and everything will be delicious. There are a lot of promotions and gifts, so if you are not in a hurry, you are here!	3	4-5
Місце дійсно цікаве , сподобалося вс е , в тому числі і інтер'єр . Офіціанти доброзичливі і дуже швидко обслуго вуються ! Я рекомендую кубик кокт еля libre . Є безкоштовний Wi - Fi . Музика круто грає ! Також полюбил и їжу і коктейльну карту !	The place is really interesting, I liked everything, including the interior. The waiters are friendly and serve very quickly! I recommend the libre cocktail cube. There is free Wi-Fi. The music plays cool! We also liked the food and cocktail menu!	3	4-5
Відсутні DVI - D (цифрові) адапте ри VGA (аналогові) . Існують тіль ки DVI - I (цифровий аналоговий) до VGA (аналоговий) . БУДЬ ОБЕР ЕЖНИЙ !	DVI-D (digital) VGA (analog) adapters are missing. There are only DVI-I (digital analog) to VGA (analog). BE CAREFUL !	5	1-2

Figure 3.4 – Example of confusing samples with discrepancy between reviews and rating

It’s important to note that subjectivity of ratings naturally exists in terms of ratings that are close to each other (1 star is pretty similar to 2 stars, whereas same is true for 5 and 4 ones). Thus, only those samples for which the difference between actual rating and predicted was bigger than two were filtered. As the result of filtering, 7437 samples were removed from dataset.

3.4 Conclusion

To tackle the task described in the work, it was decided to gather a cross-domain dataset of reviews from TripAdvisor and Rozetka websites. To accomplish the task, data scrapping with advanced techniques, including user-argent rotation, proxy servers and different time intervals was used. In result, the dataset including 671k reviews was collected. The analysis of data, revealed that bigger part of it consisted of Russian reviews, which were then translated into Ukrainian. Due to errors during translation including partial translations, missing spaces between words, and questions in reviews section, specific processing of data was used, that resulted into shrunk dataset of 662907 reviews. The rest of analysis influenced further decisions made in the work. In order to filter out discrepant reviews, a machine-learning based approach was applied.

4 MODELING AND DEVELOPMENT OF KEY-PHRASES RETRIEVAL ALGORITHM

As it was already mentioned, the proposed approach can be split into two different steps: modeling of ratings based on reviews textual information and application of explainable artificial intelligence techniques for extraction of most influential phrases w.r.t explicit ratings.

The logic behind the presented approach is the following: by learning to model ratings based on reviews textual information, the model will also learn that some words and phrases have strong influence towards specific rating score. Having a model trained in aforementioned way, the one can perform reverse engineering in order to get an importance weighting of words and phrases towards predicted rating score or range of all possible scores. Such approach requires to store only trained model and its tokenizer, as opposed to Bobenko's approach, that needs to store model and tables of positive and negative n-grams. It's worth mentioning, that developed technology can be easily integrated into microservice architecture and be used as an API. In this chapter, the process of reviews ratings modelling and key-phrases retrieval is described, the analysis of quantitative results of both steps w.r.t predefined metrics is conducted.

4.1 Modeling

The training procedure can be divided into two categories: classical machine learning algorithms and deep-learning ones. As it was already mentioned, ratings are pretty subjective, thus it was decided to conduct experiments both on the problem of rating estimation and on sentiment prediction one. To convert task from rating estimation to sentiment prediction, rating labels were mapped to sentiment ones using the following rule: ratings equal to 2 and lower mapped to negative, rating of 3 to neutral, and ratings higher than 3 to positive. For sentiment prediction, the experi-

ments were conducted towards 5 deep-learning architectures that achieved best results on ratings estimation and two classical machine learning algorithms. The data was split in a stratified manner w.r.t each domain dataset and ratings.

F1-score is a harmonic mean of precision and recall metrics, which gives a possibility to get a class-wise performance rather than overall performance as done by accuracy. Speaking of utilization of f1-score for multiclass setting, there are different ways to aggregate results. During micro averaging, true positives, false positives and false negatives are calculated for each class in order to calculate global f1-score. Micro averaging doesn't take information of imbalance into account. Other method of averaging is weighted average. During weighted average, the f1-score is calculated for each class and is weighted by support of corresponding class. Given bigger weight to classes with bigger support, is preferable when classes with more support are of bigger importance to us. However due to the fact that data used during inference could have other distribution than training one and assuming that performance of model w.r.t each class is equally important the other metric should be used. During macro averaging, the average of f1-scores for each class is calculated, therefore providing information about model's performance considering equal contribution of each class. Throughout experiments, f1-score with macro averaging was used as the main metric. To choose between algorithms, averaged f1 macro w.r.t three domains was used, as the main goal for modeling was to generalize to all domains. The dataset was split into train, validation (10%) and test (10%) in a stratified manner w.r.t rating scores and data domains.

Firstly, classical machine learning algorithms were trained. Experiments were conducted towards logistic regression and gradient-boosted trees implementation of xgboost library. Such algorithms were picked, as they can be directly utilized for explainable AI. For instance, weight coefficients of logistic regression w.r.t each input feature can be analyzed to gain sense of what feature is most influential w.r.t specific class (bigger weight means bigger influence). Stop words were removed from lemmatized tokens, which were then transformed into vectors using tf-idf (term frequency – inverse document frequency) and used as input to models. To tackle the

problem of imbalance classes, class weights were used. TF-IDF was applied along with algorithm training by utilizing a scikit-learn pipeline.

As the runtime of classical machine learning algorithms is often lower than of deep-learning ones due to fewer number of parameters, a Bayesian search over the hyper-parameters was performed. Basically, hyper-parameters tuning is a process of determining best hyper-parameters of the algorithm w.r.t specific metric. It's also called second level or black box optimization and assumes optimization of model's hyper-parameters prior to black box model performance. As some algorithms have huge number of hyper-parameters, it's favorable to do hyper-parameters search effectively. In comparison to grid search, which tries out all the possible combinations of parameters, Bayesian search optimizes parameter selection in an iterative way. There are five aspects of model based hyper-parameter optimization:

1. A domain of hyper-parameters over which the search will be done.

2. An objective function which takes hyper-parameters and returns a score, that we want to optimize.

3. The surrogate model of the objective function.

4. A selection function that evaluates which hyper-parameters to choose next from the surrogate model.

5. A history that includes mapping of hyper-parameters and corresponding score, that are used by the algorithm to update the surrogate model.

During Bayesian search, the optimization was made w.r.t f1-macro on validation subset of data for both rating score estimation and sentiment analysis. As the surrogate model, random forest regressor was used, with selection function equal to expected improvement. While searching for hyper-parameters the one needs to set range of values for domain over which the search will be done. For gradient boosted trees the following hyper-parameters were tuned:

1. max_df parameter relevant to maximal frequency of tokens for TF-ID: (0.7, 0.95).

2. min_df parameter relevant to minimum frequency of tokens for TF-ID: (5, 50).

3. ngrams on which TF-IDF operated: (1,2).

4. max_depth relevant to maximal depth of each tree: (3, 25).

5. gamma parameter controlling minimal loss reduction required to make further partition on a leaf node of the tree: (1, 9).

6. alpha parameter relevant to L1 regularization term on weights: (40, 180).

7. lambda parameter relevant to L2 regularization term on weight: (0, 1).

8. colsample_by_tree is the subsample ratio of columns when constructing each tree: (0.5, 1).

9. min_child_weight parameter relevant to minimum sum of instance weight: (0,10).

10. n_estimators parameter relevant to maximal number of trees learned: (30, 300).

For logistic regression the following hyper-parameter were tuned:

1. max_df parameter relevant to maximal frequency of tokens for TF-ID: (0.7, 0.95).

2. min_df parameter relevant to minimum frequency of tokens for TF-ID: (5, 50).

3. ngrams on which TF-IDF operated: (1,2).

What is more, it's worth mentioning that logistic regression was trained in one-versus-all setting for multiclass classification and with number of maximal iterations equal to 4000.

As of deep-learning algorithms, the experiments were conducted w.r.t combination of different layers and mechanisms including attention, convolution and recurrency. Considering the fact that real-world text has many typos and number of words in vocabulary is huge, it was decided to use sub-word tokenization method named BPE (byte-pair-coding). BPE tokenizer was trained with a min frequency of words equal to 5, which resulted into more than 10 times decrease in a number of tokens (30k). For all the experiments, embeddings with 300 dimensions were used. Due to analysis of median number of tokens in a review, all the sequences of tokens were padded to the length of 300.

Early stopping is a regularization technique to avoid overfitting. It works by tracking the results of selected criteria through training process and stopping training when criteria isn't positively updated for predefined number of steps. The technique is extremely useful in cases, where mislabeled data exists. All the models were trained for 20 epochs and early stopping strategy with a tolerance equal to 5 epochs of training was utilized.

As the main technique for regularization the Dropout was applied. Adam optimizer with default parameters was used for models training. Some of the models utilized embeddings from Word2Vec model, which were pretrained on the BPE tokenized dataset. Throughout the experiments, the same random seed was used to ensure reproducibility. All the architectures were implemented using Tensorflow and Keras frameworks. The following architectures were implemented and tried out:

Kim-CNN. The architecture proposed by Yoon Kim, which applies parallel convolutional layers to embedding layer and concatenates their output before the classification one (figure 4.1). The kernel size of convolution can be referred as number of unigrams that are sequentially aggregated together. Max pooling along with ReLU activation is applied after which convolutional layer. Convolved and max pooled representations of embeddings are then flattened and concatenated together. Before output layer the dropout is used. The following hyper-parameters were used:

- Convolutional kernel size range – [3,4,5].
- Convolutional filters – 32.
- Max pooling pool size – 2.
- Final dropout probability – 0.5.

Kim-CNN with spatial dropout and more layers. In this experiment, the previous architecture was modified and made more complex. In particular, spatial dropout was applied after the embbeding layer; the kernel size range was extended to the following values: 3,4,5,7,9; after each convolutional layer along with max pooling, the dropout was used, in order to reduce overfitting; before classification layer, an

additional fully-connected layer was utilized; finally, before the classification layer, the dropout was applied.

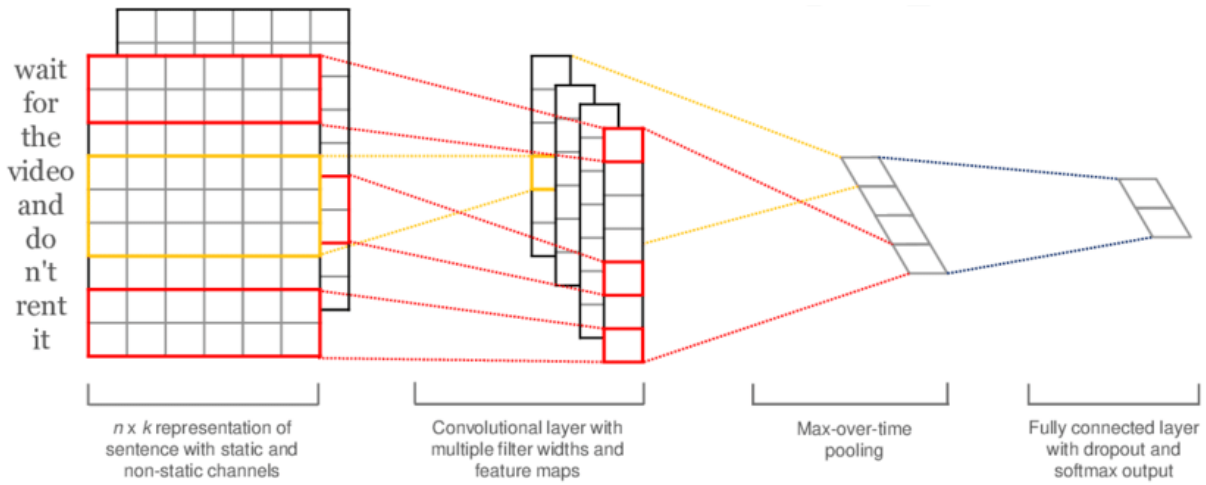


Figure 4.1 – Architecture of Kim-CNN model

The following hyper-parameters were used:

- Spatial dropout probability - 0.1.
- Convolutional filters - 32.
- Max pooling pool size - 3.
- Convolutional kernel size range - [3,4,5,7,9].
- Probability of dropout applied after max pooling - 0.1.
- Dense layer neurons number – 512.
- Final dropout probability – 0.3.

LSTM-CNN. Right after the embeddings, LSTM (Long-short-term-memory) layer was utilized. Processed sequences from LSTM were then convolved. This combination would allow to nonlinearly aggregate processed information from the LSTM. As in previous architecture, spatial dropout was utilized after the embeddings layer, same goes for fully-connected layer before classification one. The following hyper-parameters were used:

- Spatial dropout probability - 0.3.
- Convolutional filters - 32.
- Convolutional kernel size - 3.

- Max pooling pool size - 2.
- LSTM neurons number – 128.
- Dense layer neurons number – 128.
- Final dropout probability – 0.5.

CNN-LSTM. Right after the embeddings, convolution is applied similarly to Kim- CNN architecture. In this setting, LSTM works with already aggregated information of n-grams through the usage of CNN and max pooling. The following hyper-parameters were used:

- Spatial dropout probability - 0.3.
- Convolutional filters - 100.
- Convolutional kernel size - 3.
- Max pooling pool size - 2.
- LSTM neurons number – 128.
- Dense layer neurons number – 128.
- Final dropout probability – 0.5

LSTM-Attention. Attention is applied after the LSTM to aggregate processed representations of words. A dot product attention was used with tanh nonlinearity. Before the classification layer attention output was concatenated with the last state of the LSTM. As it was already mentioned, attention can be used to locally explain model's decision to some degree by analyzing importance weights assigned to each processed word from LSTM. The following hyperparameters were used:

- Attention layer neurons number - 128.
- Spatial dropout probability – 0.3.
- LSTM neurons number – 128.
- Dense layer neurons number – 128.
- Final dropout probability – 0.5

Bi-LSTM. Instead of applying the LSTM after embeddings, a bidirectional version of it is utilized. It allows to access text both from right to left and left to right allowing for richer representation of text. The following hyperparameters were used:

- Spatial dropout probability – 0.3.

- Bi-LSTM neurons number – 128.
- Dense layer neurons number – 128.
- Final dropout probability – 0.5

Bi-LSTM CNN2D. Architecture proposed by Zhang et. al, which is based on utilization of bidirectional LSTM and processing its outputs using two dimensional CNN (figure 4.2). The following hyperparameters were used:

- Spatial dropout probability – 0.3.
- Bi-LSTM neurons number – 300.
- Dropout after LSTM probability – 0.2.
- Convolutional filters – 100.
- Convolutional kernel size – (3,3).
- Max pooling strides – (2,2).
- Max pooling pool size – (2,2).

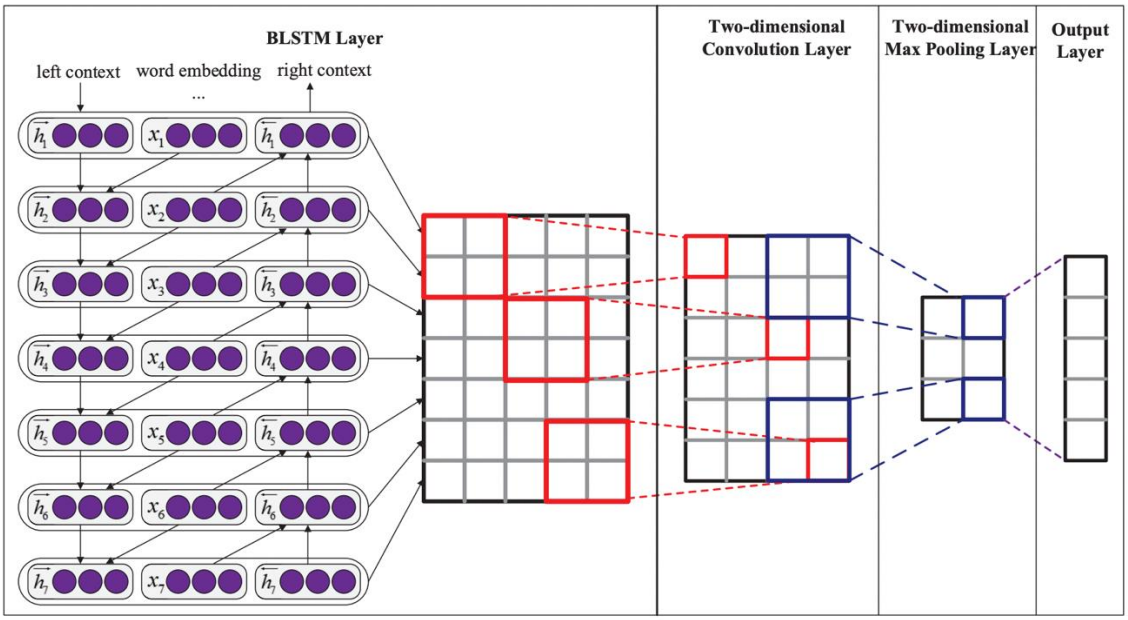


Figure 4.2 – Bi-LSTM with two-dimensional convolutional layer and max pooling

Deep LSTM. Instead of applying one LSTM after embeddings, two LSTMs were stacked. By applying multiple LSTMs on top of each other it is possible to learn richer representations of text. Between LSTMs the dropout was utilized along

with batch normalization. Batch normalization was applied to channel axis. The following hyper-parameters were used:

- Spatial dropout probability – 0.3.
- LSTM first level neurons number – 256.
- LSTM first level recurrent dropout – 0.1.
- LSTM second level neurons number – 128.
- Dense layer neurons number – 128.
- Final dropout probability – 0.5.

Deep Bi-LSTM. Same logic as for deep LSTM, but with substitution of first level LSTM layer by a bidirectional one.

- Spatial dropout probability – 0.3.
- Bi-LSTM first level neurons number – 256.
- LSTM second level neurons number – 128.
- Dense layer neurons number – 128.
- Final dropout probability – 0.5.

Deep LSTM Attention. Similar to the deep LSTM, but with usage of attention for aggregation of all output states of the second level LSTM. The following hyper-parameters were used:

- Attention layer neurons number - 128.
- Spatial dropout probability – 0.3.
- LSTM first-level neurons number – 256.
- LSTM first level recurrent dropout – 0.1.
- LSTM second level neurons number – 128.
- Dense layer neurons number – 128.
- Final dropout probability – 0.5.

Deep LSTM Attention with Word2Vec embbedings. Same architecture as before, but instead of training embeddings from scratch, the pretrained ones were fine-tuned. The hyper-parameters are the same as for deep LSTM Attention.

CNN Deep LSTM Attention with Word2Vec embbedings. A forge of two architectures, in particular Kim-CNN with more layers and Deep LSTM attention.

Firstly, parallel convolutions for defined kernel sizes were applied, the concatenated result was then passed to LSTM layers and attention. Word2Vec embbedings were utilized as in previous architecture. The following hyper-parameters were used:

- Attention layer neurons number - 128.
- Spatial dropout probability – 0.3.
- LSTM first-level neurons number – 256.
- LSTM first level recurrent dropout – 0.1.
- LSTM second level neurons number – 128.
- Dense layer neurons number – 128.
- Final dropout probability – 0.5.
- Convolutional filters - 32.
- Max pooling pool size - 3.
- Convolutional kernel size range - [3,4,5,7,9].
- Probability of dropout applied after max pooling - 0.1.
- Dense layer neurons number – 512.
- Final dropout probability – 0.1.

Deep LSTM Attention with Word2Vec embeddings and class weights. Same as deep LSTM attention with Word2Vec embeddings, but class weights were applied to tackle the problem of class imbalance. Class weights were simply computed by scikit-learn library. The hyper-parameters are the same as for deep LSTM Attention.

Variations of Deep LSTM Attention with Word2Vec embeddings w.r.t noise-tolerant objectives. Even after automatic data filtration process, biased samples still persisted in the data. Thus, it was decided to try out noise-tolerant training, specifically techniques relevant to altering the objective of a model. First experiment was related to technique named label smoothing [43]. Label smoothing is a regularization technique, that accounts for the fact that dataset could have incorrect labels, so that maximizing the likelihood of cross-entropy function could hurt the model (formula 3).

$$CE = -\sum_{i=1}^{N} p_i * \log(p_i) \quad (3)$$

In case of cross-entropy loss applied to the training of the model, p_i is substituted by true class identifier (whether 0 or 1), while term $\log(p_i)$ is transformed into logarithm of predicted probability for the class (formula 4).

$$CE_{loss} = -\sum_{i=1}^{N} y\ true_i * \log(y\ pred_i) \quad (4)$$

Logically, when predicted probability for true class is close to zero overall term is increasing. Same logic works in the other direction and the goal of optimization is to find minimum of cross-entropy function prior to model's parameters. Nevertheless, for most cases cross-entropy works well, it could hurt model performance in the case of mislabeled data. Such behavior comes from the fact that cross-entropy mostly tries to push probabilities for true classes to 1. However, such behavior is not always needed. In fact, for correct classification, the probability of true class being the biggest among others is enough. Label smoothing regularizes the model by converting hard labels into the soft ones, which helps to deal with overconfident predictions and improve generalization (formula 5).

$$y\ true_i^{LS} = y\ true_i * (1 - a) + a/K \quad (5)$$

The mathematical procedure described in formula 4 transforms hard targets into soft ones by utilizing alpha parameter, called label smoothing factor. Setting label smoothing factor to 0 would result in original hard labels, while label smoothing factor of 1 would result in uniform distribution. It was shown that label smoothing calibrates learned models, so that the confidence of their predictions are more aligned with accuracies. In our experiments we applied label smoothing with label smoothing factor equal to 0.1. While label smoothing alters targets for cross-entropy objective, there are approaches which utilize noise-robust objectives such as log cosh and Huber loss.

Log cosh loss is less sensitive to outliers and is simply computed as applying cosh and logarithm to difference between predicted and real vector. Log cosh loss can be viewed as a smoothed out L1 using L2 around origin (formula 6).

$$L = \log(\cosh(x)) \tag{6}$$

Huber loss combines L1 and L2 losses by explicitly using L2 in the vicinity of the origin where the discontinuity lies, and then switching to L1 a certain distance, delta, away from the origin. Both losses are primarily used for robust regression, but can also be adopted to classification problems, by simply computing the difference between predictions probabilities vector and one-hot vector of target classes. In our experiments, we used Huber loss with a delta of 1 (formula 7).

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 \; if \; |a| \leq \delta \\ \delta \cdot \left(|a| - \frac{1}{2}\delta\right) else \end{cases} \tag{7}$$

The results of modeling on ratings prediction problem are presented in Table 4.1, whereas on problem of sentiment analysis – in Table 4.2.

Table 4.1 – Results on a problem of modeling ratings scores

Approach	Test f1 Rozetka	Test f1 TripAdvisor hotels	Test f1 TripAdvisor restaurants	Test f1 translated data	Test f1 original data	Averaged f1 on all domains
logistic_regression	0.378	0.339	0.367	-	-	0.361
gradient boosted trees	0.26	0.256	0.262	-	-	0.259
lstm_attention	0.474	0.555	0.563	0.530	0.483	0.531
lstm_cnn	0.482	0.550	0.546	0.526	0.479	0.526
bilstm_cnn2d	0.497	0.556	0.549	0.534	0.496	0.534
bilstm	0.483	0.532	0.54	0.521	0.480	0.518
cnn_deep_lstm_attention_w2v	0.504	0.549	0.546	0.533	0.514	0.533
cnn_lstm	0.51	0.528	0.541	0.528	0.518	0.526
deep_bilstm	0.492	0.536	0.548	0.527	0.502	0.525
deep_lstm	0.491	0.548	0.554	0.532	0.494	0.531

Continuation of Table 4.1

deep_lstm_attention	0.498	0.553	0.557	0.538	0.496	0.536
deep_lstm_attention_w2v	0.516	0.568	0.572	0.552	0.523	0.5521
deep_lstm_attention_w2v_class_weights	0.493	0.562	0.584	0.546	0.497	0.546
deep_lstm_attention_w2v_huber	0.511	0.574	0.572	0.553	0.511	0.5526
deep_lstm_attention_w2v_label_smoothing	0.498	0.566	0.564	0.543	0.501	0.543
deep_lstm_attention_w2v_log_cosh	0.5	0.57	0.571	0.547	0.505	0.547
kim_cnn	0.517	0.510	0.534	0.528	0.516	0.520
kim_cnn_more_layers_spatial_drop	0.513	0.532	0.546	0.535	0.514	0.530

Table 4.2 – Results on a problem of sentiment analysis

Approach	Test f1 Rozetka	Test f1 TripAdvisor hotels	Test f1 TripAdvisor restaurants	Test f1 translated data	Test f1 original data	Averaged f1 on all domains
logistic_regression	0.562	0.497	0.546	-	-	0.535
gradient boosted trees	0.422	0.39	0.428	-	-	0.413
bilstm_cnn2d	0.685	0.699	0.732	0.709	0.689	0.705
deep_lstm_attention_w2v	0.691	0.712	0.728	0.712	0.698	0.71
deep_lstm_attention_w2v_class_weights	0.676	0.7	0.738	0.709	0.673	0.705
deep_lstm_attention_w2v_huber	0.691	0.721	0.745	0.721	0.695	0.719
kim_cnn_more_layers_spatial_drop	0.657	0.709	0.734	0.705	0.650	0.7

As it can be seen from results depicted in Table 1, deep_lstm_attention-w2v_huber achieves best results in terms of test f1 for TripAdvisor hotels domain and averaged f1 on all domains. Analyzing the confusion matrix (appendix B.5) of best approach on rating estimation, it's easy to notice that most of the errors are relevant to mismatching close categories, that are subjective by nature. This in particular, implies that trained model is representable of data distribution and can be used for further experiments relevant to key phrases retrieval. Interestingly, the effect of noise-robust objective isn't very noticeable in rating estimation experiment. In fact, the difference between average f1 on all domains between Deep LSTM Attention Word2Vec embeddings with cross- entropy and with Huber loss is only 0.0005 points, whereas the gap is much bigger for the task of sentiment analysis (+0.09). The huge influence of pretrained embeddings for sub-words is observed, in particular fine-tuning of pretrained Word2Vec embeddings for Deep LSTM with

Attention leads to increase in metrics for all domains and results in increase for averaged f1 on reviews estimation problem (+0.161). Such result is possibly influenced by the fact that by pretraining Word2Vec embeddings on the task of language modelling, resulting learned vectors better capture the overall structure of language and meaning of words. Speaking about Attention, it's seen that usage of it resulted into better f1 across all domains and averaged one (if to compare Deep LSTM Attention with Deep LSTM, the gap w.r.t averaged f1 is +0.005). Such behavior is explained by the fact, that Attention mechanism learns to weight tokens in the sentence by their significance to correct output results. In the case of sentiment analysis and reviews estimation, Attention mechanism could learn that hidden state of positive and negative tokens are the most important (the unraveled results of Attention weighting are presented in next chapter). If to compare worst results of deep learning approach with best one of machine learning one, the increase in f1 score is 0.159. The gap between deep learning-based approaches and machine learning ones suggests that order of words and learning of their context are of big matter for both review estimate and sentiment analysis problems. It's worth mentioning that results of models could be improved by using automatic hyper-parameters optimization and manual data filtering. The exact configurations of models in terms of their architectures and hyper-parameters are available on GitHub.

4.2 Algorithm for key phrases retrieval

After training the models, the best one w.r.t chosen metric was picked for explainability experiments and construction of an algorithm for key phrases retrieval. The algorithm works on both on the level of entity (restaurant/hotel/product) and on the level of its review. While working on the level of entity, specific averaging is used to summarize most influential phrases across all the reviews for the entity. The algorithm for key- phrases retrieval can be logically divided into two steps: retrieval of predictions and scores for each token in each review and aggregation of scores across all the predictions.

Retrieval of scores is the main subject of the experiments. In particular, the experiments were conducted towards two methods: LIME and Attention. As the trained model operated on BPE tokens, which are essentially sub-words, the operation of sub-words merging was implemented. For merged sub-words, corresponding attention scores were summed-up. Main disadvantage of straightforward attention explanation is that its feature scoring gives explanation that is interpreted towards the class with highest probability, although certain features can contribute to increasing of probabilities of other classes. On the other hand, LIME provides explanation that captures contribution of features towards each class. Speaking of LIME, the main disadvantage that we found was disability of using custom tokenizer, which is essential for Ukrainian language. Also, while Attention is an in-built mechanism of model explainability, LIME uses local surrogate models to interpret predictions, which could be not strong enough to understand the data and approximate predictions of much more complex model.

Having obtained the scores for each token and actual predictions for each review, the aggregation of results was done. The aggregation step works for phrases of varying size, supports aggregations relevant to sum and mean and has a functionality for diversification of results based on input tokens. For n-grams other than unigrams, the scores are summed up or averaged, depending on aggregation algorithm's settings. It's worth mentioning that aggregation algorithm is agnostic towards the method used for scoring tokens and is pretty simple in nature, which makes it easier to extend and enhance.

The full pipeline of extraction for key phrases extraction and reviews summarization works in the following way:

1. Process the data in the same way, that was used to process the data for training the models (add spaces between punctuations, remove next-line character, etc.).

2. Tokenize the data using trained BPE tokenizer.

3. Make predictions and explanations based on trained model using LIME or Attention for each review/text.

4. Summarize results using aggregation algorithm.

The aggregation algorithm included logic relevant to generation of scores for n-grams, removal of too similar n-grams and aggregation of results for each possible rating. While creating n-grams, the scores for words forming the n-grams are averaged together. Results are then either summed up or averaged for same n-grams for same labels. It's rather important to use removal of similar n-grams to get better diversification of results. Diversification is done by grouping similar n-grams w.r.t intersection of subphrases on the level of each possible rating. After grouping procedure, only one n-gram in group which has the highest importance score is left. It was empirically found out, that model tended to pay much attention to punctuation relevant to end of sentence. In order to get rid of such bias two options were incorporated:

1. Split paragraph into sentences and finding n-grams inside each sentence separately. This option provided a tolerance towards n-grams with high score consisting of multiple words from different sentences. Despite, the overall score of such n-grams could be pretty high, their construction violates explicit text structure and thus isn't comprehensive.

2. Filtration of punctuation characters while computing n-grams. Although some punctuation characters like exclamation mark or smiles could provide some information inside key-phrases, it was decided to not include them because of intrinsic bias of attention mechanism towards them.

The list of aggregation algorithm's hyper-parameters along with short description for each of them is depicted below:

- n_gram_list – list of n-grams for which generation should be done.
- func_agg_n_gram – function to aggregate scores for n-grams based unigrams, either "mean" or "sum".
- func_agg_overall – function to aggregate scores for same n-grams, either "mean" or "sum".
- diversify – diversification power related to number of tokens to use for duplicates filtering, integer is expected as input. Zero value is relevant to no filtration.

- to_tokenize_sentences – whether to tokenize paragraphs into sentences before getting n-grams.
- to_tokenize_by_punct – whether to additionally tokenize sentences based on punctuation.
- to_del_punct – whether to delete punctuation.
- top_n – top results to return back for each rating.
- min_thr – minimal value of score for returning the phrase.

It's worth mentioning that during comparison of methods in terms of key-phrases retrieval ability, both options were disabled, as they had no impact on comparison.

The experiments to decide which method is more suitable for key-phrases retrieval were conducted. During experiments the aggregation method's parameters were the following: n-gram was set to be equal to 4, both aggregation for phrase scores and overall scores were set to mean, diversification was used with overlap of 2 words, for LIME method top-15 phrases per label were retrieved, whereas for Attention – top-20. The experiments were made towards best model trained for rating estimation. For experiments, a new dataset of diverse entities in terms of their average rating across three domains was constructed. To compare results of LIME and Attention explanation, precision at K metric was used. To retrieve information about global performance of phrases retrieval for specific entity, average precision at K was used:

$$Average\ Precision\ at\ K = \frac{1}{D}\sum\frac{K_r}{K}, \quad (8)$$

where D - number of reviews for specific entity, K- overall number of phrases, K_r – number of relevant phrases.

Precision at K can be used in two setups:

1. With labeled data, in which relevant phrases for each entity are labeled. This setup gives an opportunity to automatically compute the metric.

2. With human evaluation, when results of algorithm are checked by human in terms of their relevance.

To evaluate algorithm proposed in current work, the second approach was applied. The phrase (n-gram) was assumed relevant if it was clear and reflective of predicted category (figure 4.3).

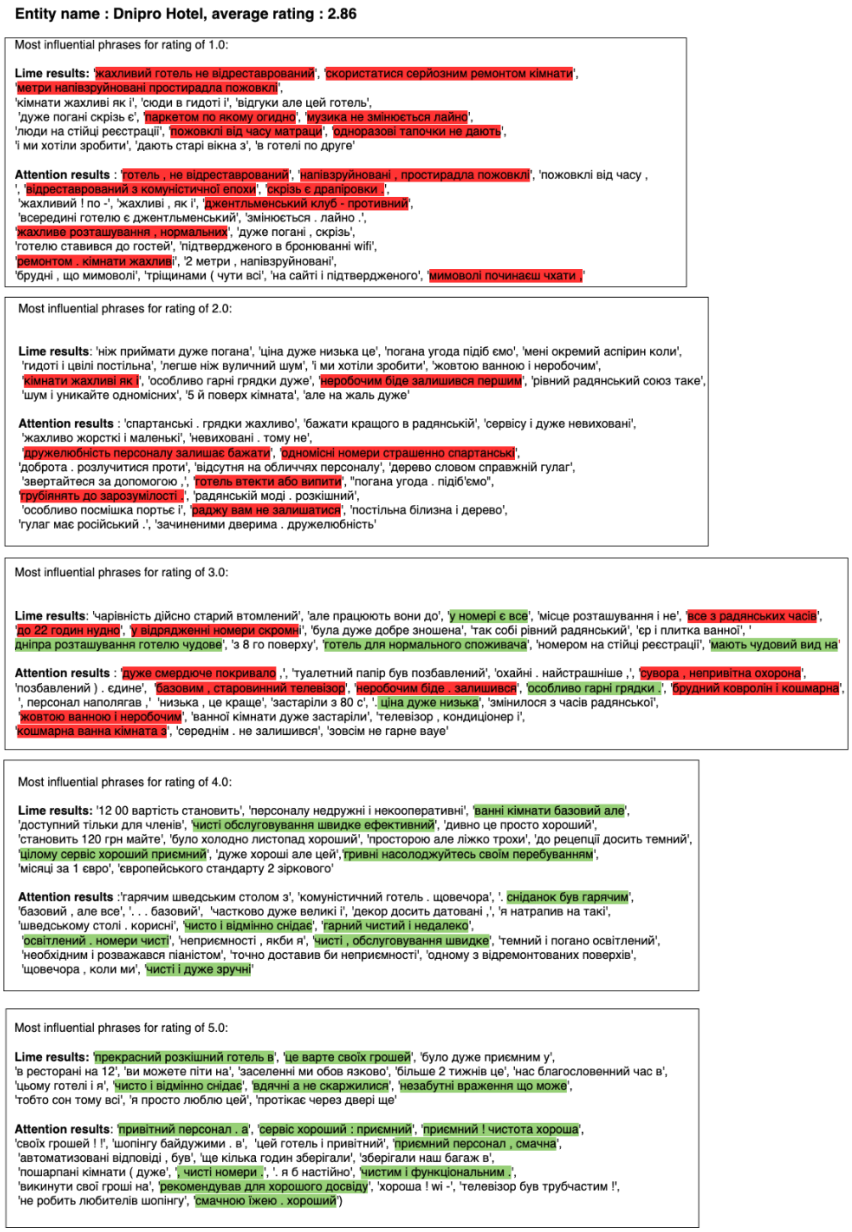


Figure 4.3 - Example of LIME and Attention explanation for one of the entities. In green – positive phrases are shown, in red – negative. For ratings <3 only negative phrases are relevant, for >4 – only positive, for 3 – both negative and positive. Results were validated towards summarization of all reviews w.r.t specific entity and categorized by averaged rating groups (<3, 3 and >=4).

As it can be seen from Table 4.3, Attention method achieves better Precision at K averaged on all rating groups, which was used as main metric. It's worth mentioning that LIME has better coverage in terms of number of phrases, thus it's recommended to use the combination of methods while retrieving key phrases. The algorithm for phrases retrieval can easily be enhanced based on POS tags, which could help to obtain only those phrases which suit specific patterns (e.g. Noun-Adjective, Adjective-Noun-Verb, etc.).

Table 4.3 – Results on problem of key-phrases retrieval

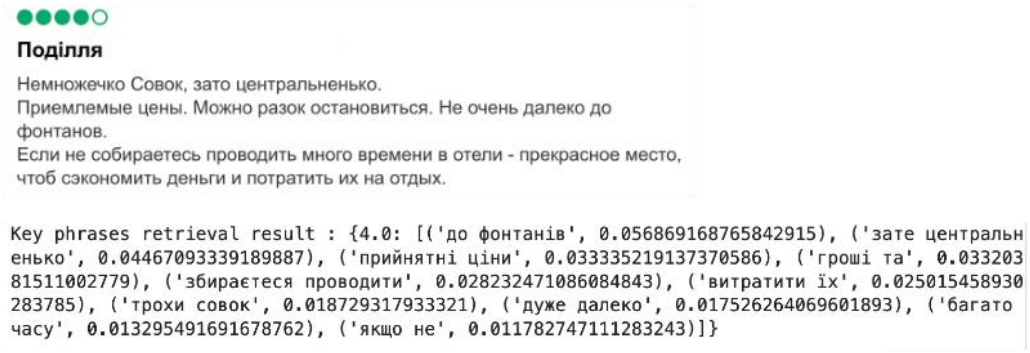
Approach	Precision at K for average rating <3	Precision at K for average rating 3	Precision at K for average rating >=4	Average Precision at K
LIME	0.2806	0.3292	0.221	0.276
Attention	0.308	0.3009	0.266	0.291

Even though the algorithm is constructed to work with pull of reviews w.r.t specific entity (restaurant, hotel, product), it's much easier to make qualitive comparison and analysis by utilizing of key-phrases retrieval for one review. In this scenario, both methods were tried out on reviews from each domain which were unseen by model during training. It's worth mentioning, that the application of methods to one sample of reviews instead of bunch of them results in decrease in performance for key-phrases retrieval thus implying more thorough hyperparameters selection for aggregation stage. The results of such retrieval for Attention mechanism are depicted on figure 4.4, while for LIME on figure 4.5.

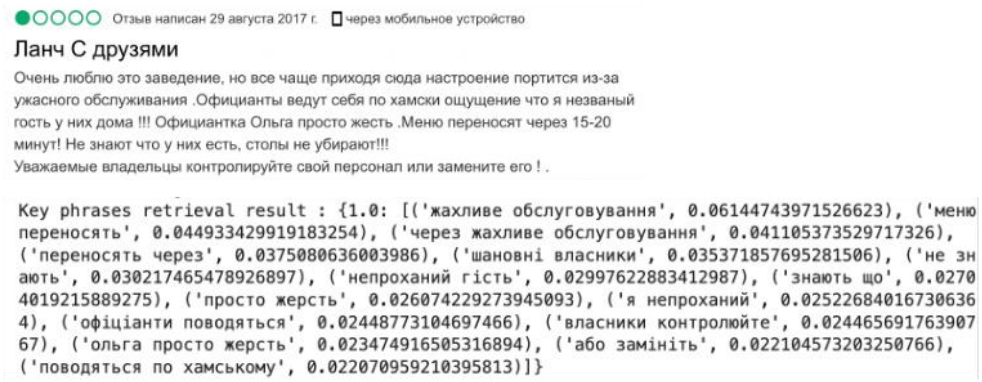
If to compare results of algorithms inference on one review, which are depicted on aforementioned figures, it's pretty clear that for one review LIME provides better categorization of key-phrases which can be useful in terms of ABSA (Aspect-Based-Sentiment-Analysis). Attention at the same time provides general information of most influential key-phrases towards the predicted label. As it was already mentioned, the described method for aggregation has number of hyper-parameters that should be tweaked w.r.t specific use-case. It was empirically found that number

of n-grams should be increased for longer reviews, as it's more probable that such texts contain more complicated language structures.

Hotel example:



Restaurant example:



Product example:

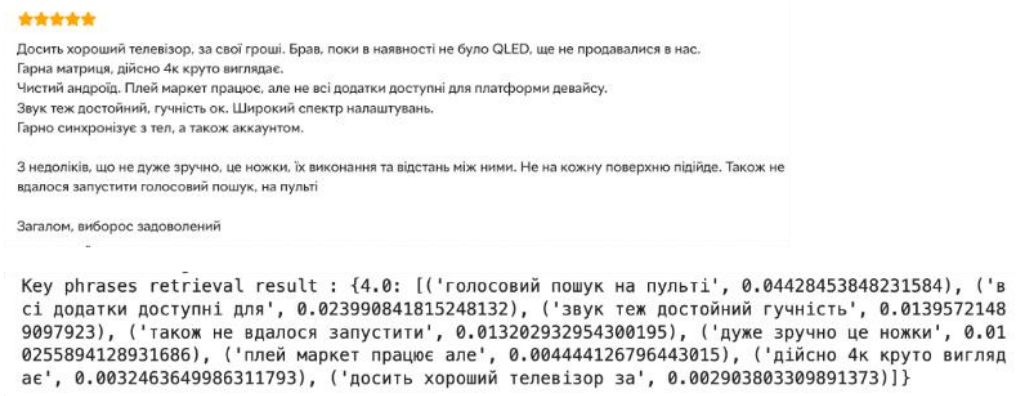
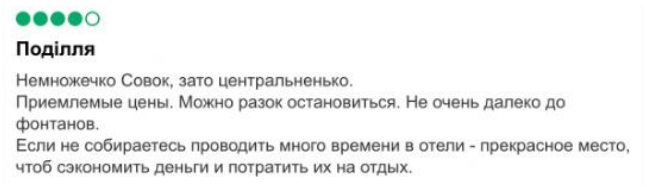


Figure 4.4 – Results of applying Attention-based key-phrases retrieval

Hotel example:

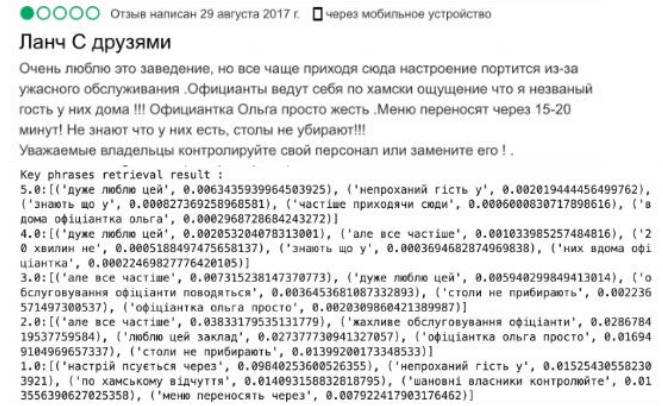


Поділля

Немножечко Совок, зато центральненько.
Приемлемые цены. Можно разок остановиться. Не очень далеко до фонтанов.
Если не собираетесь проводить много времени в отели - прекрасное место, чтоб сэкономить деньги и потратить их на отдых.

```
Key phrases retrieval result :
5.0:[('прийнятні ціни', 0.03638715237521339), ('чудове місце', 0.016164822273017673), ('дуже
далеко', 0.01479356153142479), ('гроші та', 0.007287792589865722), ('збирається проводити',
0.0056115590977661 56), ('на відпочинок', 0.005595461816070104), ('часу в', 0.0020622804854654
54)]
4.0:[('прийнятні ціни', 0.108262919584495 24), ('трохи совок', 0.04133024733240093), ('дуже да
леко', 0.025599348045739184), ('чудове місце', 0.019669584703545153), ('їх на', 0.01116995624
63519), ('не збирається', 0.006782456779058981), ('можна раз', 0.0025734053980415762), ('бага
то часу', 0.0015304405728870304)]
3.0:[('трохи совок', 0.09160484482800949), ('зате центральненько', 0.033737131463451496), ('д
о фонтанів', 0.031247020017061884), ('можна раз', 0.022853114084132653), ('заощадити гроші',
0.022411505839914563), ('витратити їх', 0.019032222112231118), ('якщо не', 0.0153881858093543
04), ('зупинитися не', 0.009284065827133113), ('багато часу', 0.006795296870876809)]
2.0:[('совок зате', 0.0650748170033 3678), ('заощадити гроші', 0.029811743083811185), ('витрат
ити їх', 0.012971767596407387), ('до фонтанів', 0.007159246045093 6885), ('збирається проводит
и', 0.0037397265494417973), ('часу в', 0.003588230092472755), ('якщо не', 0.00339687867407513
07), ('на відпочинок', 0.0020559794095090 26)]
1.0:[('совок зате', 0.022566565159349 87), ('заощадити гроші', 0.014814608464134632), ('не зби
рається', 0.0077115368581598 92), ('та витратити', 0.0060518891369203 55), ('далеко до', 0.0042
39986529142976), ('часу в', 0.00160761194688 7797), ('на відпочинок', 0.0014626155375959027)]
```

Restaurant example:



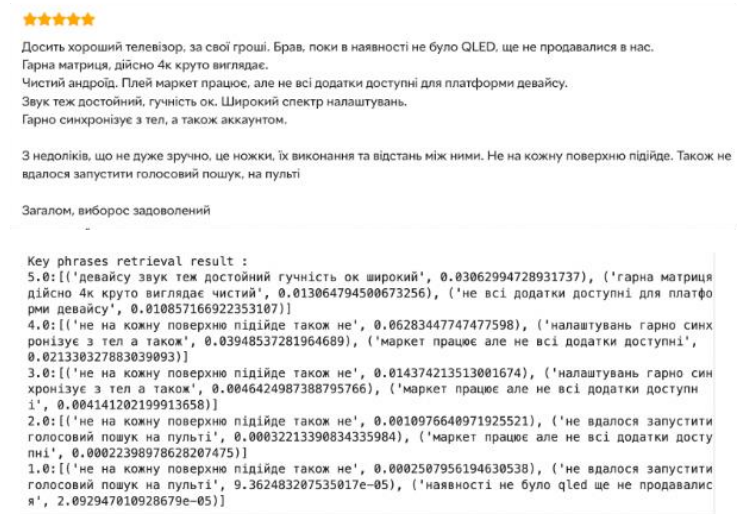
Отзыв написан 29 августа 2017 г. через мобильное устройство

Ланч С друзями

Очень люблю это заведение, но все чаще приходя сюда настроение портится из-за ужасного обслуживания .Официанты ведут себя по хамски ощущение что я незваный гость у них дома !!! Официантка Ольга просто жесть .Меню переносят через 15-20 минут! Не знают что у них есть, столы не убирают!!!
Уважаемые владельцы контролируйте свой персонал или замените его ! .

```
Key phrases retrieval result :
5.0:[('дуже люблю цей', 0.0063435939964503925), ('непроханий гість у', 0.002019444456499762),
('знають що у', 0.000827369258968581), ('частіше приходячи сюди', 0.0006000830717898616), ('в
дома офіціантка ольга', 0.00029687286842432 72)]
4.0:[('дуже люблю цей', 0.002053204078313001), ('але все частіше', 0.001033985257484816), ('2
0 хвилин не', 0.0005188497475658137), ('знають що у', 0.0003694682874969838), ('них вдома офі
ціантка', 0.0002246982777642010 5)]
3.0:[('але все частіше', 0.007315238147370773), ('дуже люблю цей', 0.005940299849413014), ('о
бслуговування офіціанти поводяться', 0.0036453681087332893), ('столи не прибирають', 0.002236
571497300537), ('офіціантка ольга просто', 0.0020309860421389987)]
2.0:[('але все частіше', 0.03833179535131779), ('жахливе обслуговування офіціанти', 0.0286784
19537759584), ('люблю цей заклад', 0.027377730941327057), ('офіціантка ольга просто', 0.01694
9104969657337), ('столи не прибирають', 0.013992001733485 33)]
1.0:[('настрій псується через', 0.09840253600526355), ('непроханий гість у', 0.01525430558230
3921), ('по хамському відчуття', 0.014093158832818795), ('шановні власники контролюйте', 0.01
3556390627025358), ('меню переносять через', 0.007922417903176462)]
```

Product example:



Досить хороший телевізор, за свої гроші. Брав, поки в наявності не було QLED, ще не продавалися в нас.
Гарна матриця, дійсно 4к круто виглядає.
Чистий андроїд. Плей маркет працює, але не всі додатки доступні для платформи девайсу.
Звук теж достойний, гучність ок. Широкий спектр налаштувань.
Гарно синхронізує з тел, а також аккаунтом.

З недоліків, що не дуже зручно, це ножки, їх виконання та відстань між ними. Не на кожну поверхню підійде. Також не вдалося запустити голосовий пошук, на пульті

Загалом, виборос задоволений

```
Key phrases retrieval result :
5.0:[('девайсу звук теж достойний гучність ок широкий', 0.030629947289317 37), ('гарна матриця
дійсно 4к круто виглядає чистий', 0.013064794500673256), ('не всі додатки доступні для платфо
рми девайсу', 0.010857166922353107)]
4.0:[('не на кожну поверхню підійде також не', 0.06283447747477598), ('налаштувань гарно синх
ронізує з тел а також', 0.039485372819646 89), ('маркет працює але не всі додатки доступні',
0.021330327883039093)]
3.0:[('не на кожну поверхню підійде також не', 0.014374213513001674), ('налаштувань гарно син
хронізує з тел а також', 0.004642498738879 5766), ('маркет працює але не всі додатки доступн
і', 0.004141202199913658)]
2.0:[('не на кожну поверхню підійде також не', 0.001097664097192 5521), ('не вдалося запустити
голосовий пошук на пульті', 0.000322133908343359 84), ('маркет працює але не всі додатки досту
пні', 0.000223989786282074 75)]
1.0:[('не на кожну поверхню підійде також не', 0.000250795619463 0538), ('не вдалося запустити
голосовий пошук на пульті', 9.362483207535017e-05), ('наявності не було qled ще не продавалис
я', 2.092947010928679e-05)]
```

Figure 4.5 - Results of applying LIME-based key-phrases retrieval

As it was already mentioned, the approach utilizes BPE tokens and explainable AI allowing it to adopt to data and domains that weren't seen during training. In

the following experiment the key-phrases retrieval is applied to reviews of three different domains: books, attractions and movies reviews. As in the previous experiment, key-phrases retrieval is applied only to one review at the time. In this experiment, both rating estimation and Attention-based key-phrases retrieval are tested. The results are depicted on figure 4.6.

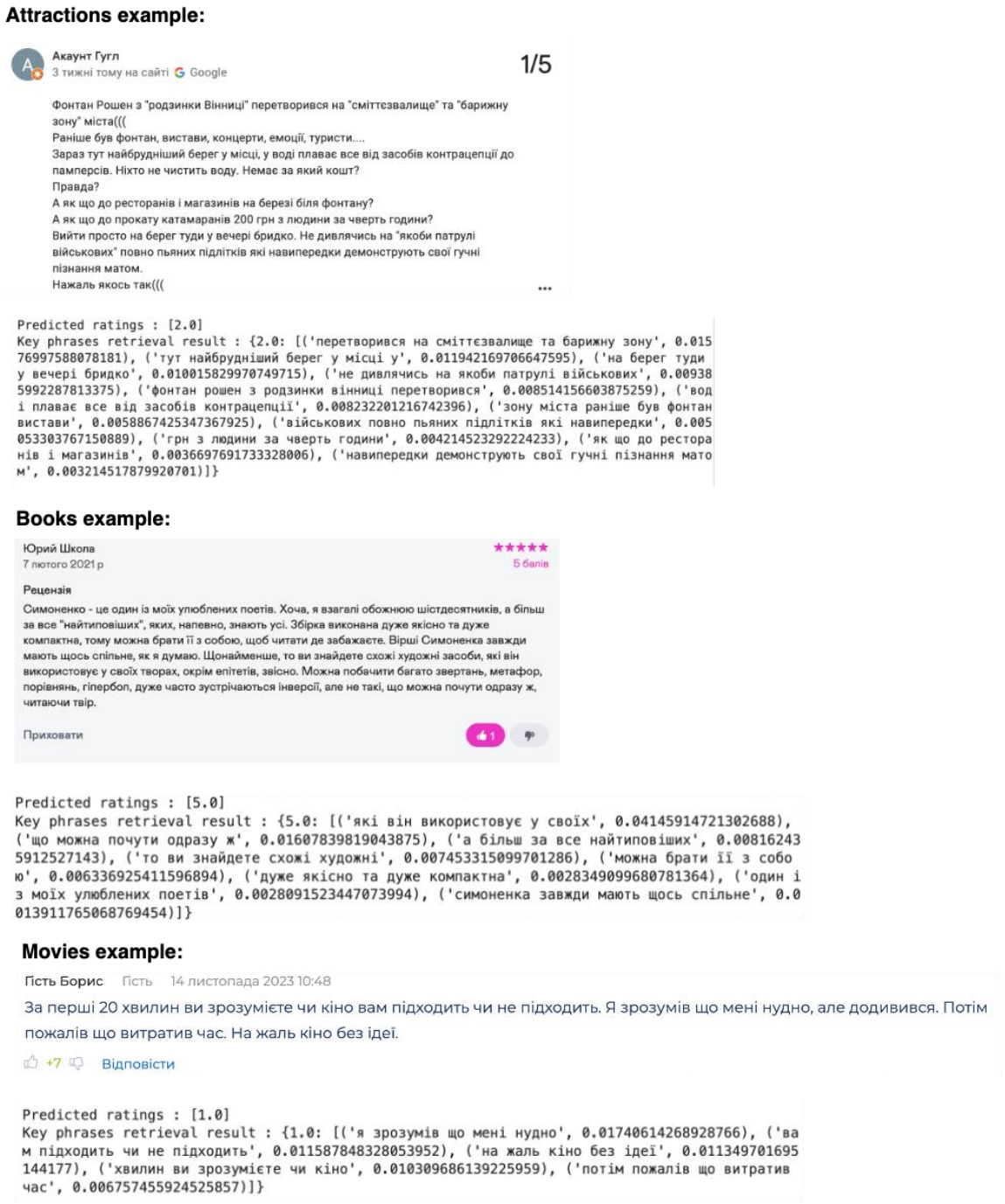


Figure 4.6 – Results of applying Attention-based key-phrases retrieval for unseen domains

As it can be seen from the experiment, trained model does a good job in both predicting rating for unseen domains and retrieving of key-phrases. The empirical results suggest that presented approach can be utilized to some extent for rating estimation and key-phrases retrieval without a need for fine-tuning it to new data.

4.3 Comparison to most similar analog

Finally, it's beneficial to summarize differences between current work and the most similar analog, in particular master thesis of Bobenko. Although no particular comparison in terms of prediction capability of models wasn't provided due to the difference in modeling problems and metrics, the differences in terms of overall approach are highlighted (table 4.4).

Table 4.4 – Comparison of current work with the most similar analog

Characteristic	Current work	Bobenko's work
Number of domains	3	1
Number of data sources	3	2
Number of samples	~671k	~128k
Modeling problems	Sentiment analysis, review rating estimation	Sentiment analysis
Tokenization method	BPE	Word tokenizer
Number of unique tokens	30k	8.238.336
Files needed for inference	Tokenizer, model	Model, table of positive and negative n-grams
Key-phrases retrieval method	LIME/Attention	TF-IDF/PMI
Operating level	Paragraph	Sentences
Handling of OOV tokens	+	-
Data filtering	+	-
Contextualization	+	-

As it can be seen from aforementioned table, the gathered dataset is this work is much more versatile than the one in Bobenko's work and covers 3 different data sources and 3 domains. Logically, models trained on such dataset are capable of understanding and handling more cases. Furthermore, even though the data is human generated meaning that some samples have discrepancy between reviews and their rating, no filtration was done in concurrent work. Speaking of efficiency, by utilizing BPE tokenization current model's embedding matrix needs to store only 30k unique tokens which is in ~274 times lower than in Bobenko's work. By applying BPE tokenization, current model is capable of processing even those words and phrases which were absent in dataset making it more adaptive to new data. For final inference only model itself and tokenizer is needed which is far more memory efficient than storing model with embeddings for each word along with table of positive and negative n-grams. Finally, by utilizing Attention/LIME for key-phrases retrieval presented approach provides more contextualization than retrieval from prebuilt n-grams in table. To summarize, presented approach is more efficient and adaptive to new data.

4.5 Conclusions

The process of ratings estimation and modelling of sentiment based on collected reviews textual data along with creation of technology for key-phrases retrieval is described in this chapter. The evaluation of proposed methodologies to accomplish the task w.r.t predefined metrics is conducted. The analysis of evaluation results for modelling showed that pretraining of word-embeddings, utilization of noise-tolerant objectives and usage of attention mechanism improves results on all the domains on both tasks.

Although, conducted experimenters revealed that in general, attention mechanism provides better results than LIME, its recommended to experiment with both for a specific use-case.

5 THE ECONOMIC SECTION

5.1 The technological audit of the developed system for searching key phrases in Ukrainian-language feedback

In conditions of intense competition, text analysis holds tremendous importance for companies monitoring mass media concerning specific events and businesses. This is vital for generating analytical reports and offering business insights that delve into sentiments regarding particular events or companies within a defined timeframe. Typically, sentiment analysis provides only a general overview of an event or company, but it doesn't address the underlying causes that influenced such outcomes.

To address this, sentiment analysis or reviews rating assessment is required. Considering the challenge of generalizing reviews, it's crucial for analysis methods to account for sarcasm, word order, and other intricate linguistic patterns, subjective nature of reviews, numerous errors, and typos in words, etc., which might lead to a considerable number of diverse tokens that can influence the reliability of drawing conclusions.

Hence, prior to our master's thesis, the task was set to investigate the utilization of deep learning algorithms and classical machine learning to study conditional data distribution and apply artificial intelligence techniques to extract the most significant textual features in the Ukrainian language.

As a result, we selected, trained, and evaluated algorithms for reviewing ratings and sentiment analysis. We chose explanatory AI algorithms for extracting key phrases in the Ukrainian language and constructed an algorithm for key phrase extraction, evaluating it in comparison with the defined explanatory AI algorithms.

To assess the level of commercial potential of our developed system for searching key phrases in Ukrainian-language feedback using artificial intelligence technologies, we conducted a technological audit by inviting three renowned experts: a Doctor of Technical Sciences, a Professor, Oleh BISIKALO Candidate of

Technical Sciences and Associate Professor Maria BARABAN, Candidate of Technical Sciences and Associate Professor Volodymyr HARMASH.

The assessment of the commercial potential of our developed system for searching key phrases in Ukrainian-language feedback using artificial intelligence technologies was conducted based on the criteria summarized in Table 5.1.

Table 5.1 – Assessment criteria for evaluating the commercial potential of any development and their score rating (on a scale of 0 - 1 - 2 - 3 - 4 points)

Evaluation Criteria and Scores (on a 5-point scale)					
Criterion	0	1	2	3	4
Technical Feasibility of the Concept:					
11	The credibility of the concept is not confirmed.	Concept validated by expert opinions	Concept validated by calculations	Concept tested in practice	Product's operational capability verified in real-world conditions
Market Advantages (Disadvantages):					
22	Numerous analogs in a small market	Few analogs in a small market	Several analogs in a large market	One analog in a large market	No analogs for the product in a large market
33	The price of the product is significantly higher than that of analogs	The price of the product is slightly higher than that of analogs	The price of the product is approximately equal to the prices of analogs	The price of the product is slightly lower than that of analogs	The price of the product is significantly lower than that of analogs
44	Technical and consumer properties of the product are significantly worse than those of analogs	The technical and consumer properties of the product are slightly worse than those of analogs	The technical and consumer properties of the product are on par with those of analogs	The technical and consumer properties of the product are slightly better than those of analogs	The technical and consumer properties of the product are significantly better than those of analogs

Continuation of Table 5.1

Evaluation Criteria and Scores (on a 5-point scale)					
Criterion	0	1	2	3	4
Market Prospects					
55	Operational costs are significantly higher than those of analogs	Operational costs are slightly higher than those of analogs	Operational costs are on par with the operational costs of analogs	Operational costs are slightly lower than those of analogs	Operational costs are significantly lower than those of analogs
66	The market is small and lacks positive dynamics	The market is small but shows positive dynamics	Medium-sized market with positive dynamics	Large and stable market	Large market with positive dynamics
77	Active competition from major companies in the market	Active competition	Moderate competition	Slight competition	No competitors
Practical Feasibility					
88	Lack of experts in both technical and commercial implementation of the idea	Requires hiring experts or significant investment of time and money in training existing personnel	Minor training required for staff and slight expansion of the team	Minor training required for staff	Experts available both technically and commerciall y

Continuation of Table 5.1

Evaluation Criteria and Scores (on a 5-point scale)					
Criterion	0	1	2	3	4
99	Significant financial resources needed, but absent. Lack of funding sources for the idea	Slight financial resources needed, but no funding sources available	Substantial financial resources needed, funding sources exist	Slight financial resources needed, funding sources exist	No need for additional funding
110	Requires development of new materials	Materials required are used in military-industrial complex	Expensive materials needed	Accessible and inexpensive materials needed	All materials for idea implementation are well-known and have long been used in production
111	Implementation timeline exceeds 10 years	Implementation timeline exceeds 5 years. Return on investment period exceeds 10 years	Implementation timeline from 3 to 5 years. Return on investment period exceeds 5 years	Implementation timeline is less than 3 years. Return on investment period from 3 to 5 years	Implementation timeline is less than 3 years. Return on investment period is less than 3 years
112	Necessary development of regulatory documents and acquiring numerous permits for production and product implementation	Requires acquiring numerous permits for production and product implementation, demanding significant costs and time	The process of obtaining permits for production and product implementation requires minor costs and time	Only notification to relevant authorities about production and product implementation is necessary	No regulatory constraints on production and product implementation

Invited experts have evaluated our developed system for searching key phrases in Ukrainian-language feedback using artificial intelligence-based technologies quite highly (refer to Table 5.2):

Table 5.2 – Evaluation Results of the Commercial Potential of the Development

Criterias	Last name, initials of the expert		
	Oleh BISIKALO	Maria BARABAN	Volodymyr HARMASH
	Scores given by the experts:		
1	3	4	4
2	4	3	4
3	3	3	3
4	4	4	4
5	3	3	3
6	3	3	4
7	3	4	3
8	4	4	3
9	3	3	4
10	4	4	3
11	4	3	4
12	3	4	4
Sum of grades	41	42	43

The arithmetic mean ($\overline{СБ}$), of the scores assigned by the experts was:

$$\overline{СБ} = \frac{\sum_{1}^{3} Б_i}{3} = \frac{41+42+43}{3} = \frac{126}{3} = 42,00 .$$

The overall level of commercial potential for any development was determined based on the criteria outlined in Table 5.3 [44].

Guided by the recommendations in Table 5.3, it can be concluded that the developed system for searching key phrases in Ukrainian-language feedback using artificial intelligence technologies was evaluated by experts at 42 points, indicating that named development possesses a commercial potential categorized as "high".

Table 5.3 – Levels of Technical and Commercial Potential of the Development

The arithmetic mean of scores calculated based on the experts' conclusions.	The level of technical and commercial potential of the development.
0 – 10	Low
11 – 20	Below average
21 – 30	Average
31 – 40	Above average
41 – 48	High

This is due to the development being based on cross-industry data aggregation and a dataset containing Ukrainian feedback. It addresses the challenge of evaluating contributors' ratings, analyzing their sentiments expressed in Ukrainian, among other aspects. Moreover, it solves the problem of automatic extraction of key phrases and summaries specific to the Ukrainian language.

5.2 The cost estimation for developing a system to search for key phrases in Ukrainian language within feedback using artificial intelligence technologies.

During the work, the following expenses were incurred:

1. Primary salary of executors $З_o$:

$$З_o = \frac{M}{T_p} \cdot t \text{ uah}, \quad (5.1)$$

Where M represents the monthly base salary of a specific executor in UAH;

In 2023, the salary ranges for researchers fall within (6700…26000) UAH/month; T_p – indicates the number of working days in a month; let’s assume T_p = 21 days.

The calculations of the primary salary of the executors will be summarized in Table 5.4:

Table 5.4 – Calculation of the primary salary of executors (developers)

Position title of the executor	Monthly base salary, UAH	Payment per working day (or per hour), UAH	Number of working days	Remuneration costs, UAH	Notes
1. Scientific supervisor of the Master's qualification work	22250	1059,52	20 hrs	≈ 3532	6 hrs per day
2. Student developer - Master's student	6700	319,05	75 days	≈ 23929	
3. Consultant in the economic section	19800	942,86	1,5 hrs	≈ 236	6 hrs per day
4. Other consultants	18500	880,92	3,5 days	≈ 3083	
Total				30 780	

2. Additional remuneration of the executors $З_д$ is calculated as (10…12 of the primary salary of the executors, which means:

$$З_д = (0{,}1…0{,}12) \cdot З_о . \quad (5.2)$$

For our case, we will obtain:

$$З_д = 0{,}113 \times 30780 = 3478{,}14 \approx 3479 \text{ uah.}$$

3. Accruals to the payroll $Н_{зп}$ are calculated by the formula:

$$Н_{зп} = (З_о + З_д) \cdot \frac{\beta}{100}, \quad (5.3)$$

where $З_о$ – primary salary of the executors, UAH;

$З_д$ – additional remuneration of the executors, UAH;

β – The rate of the unified social security contribution for mandatory state social insurance; β = 22%.

Then:

$$Н_{зп} = (30780+3479) \times 0,22 = 7536,99 \approx 7537 \text{ uah.}$$

4. Material expences M are calculated per each material type:

$$М = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i - \sum_{1}^{n} В_i \cdot Ц_в \quad \text{uah,} \tag{5.4}$$

Where $Н_i$ – material expences per *i* designation, kg; $Ц_i$ – cost of a material *i*, uah/kg.; $К_i$ – transportation expences coeficient, $К_i$ = (1,1…1,15); $В_i$ – material disposal mass per material *i*, kg; $Ц_в$ – material waste price per material *i*, uah/kg; n – total number of used materials.

5. The expenses on components К are calculated by the formula:

$$К = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i \text{ uah,} \tag{5.5}$$

Where $Н_i$ – the quantity of components *i*-th type, pcs.; $Ц_i$ – price per component of *i*-th type, uah; $К_i$ – transportation expences coeficient, $К_i$ = (1,1…1,15); n – total number of components.

Following the analogy with other developments, the cost of all utilized material resources is approximately 3000 UAH.

6. Depreciation (А) of equipment, computers, and premises А can be calculated by the formula:

$$A = \frac{Ц \cdot H_a}{100} \cdot \frac{T}{12} \text{ uah}, \quad (5.6)$$

Where Ц – the total book value of fixed assets in UAH;

H_a – the annual depreciation rate: H_a = (2...25)%;

T- is the period of equipment, premises, etc. the usage, in months.

The calculations made have been summarized in Table 5.5:

Table 5.5 – Calculation of depreciation deductions

Equipment, premises, etc	Book value, UAH.	Depreciation rate, %	Period of usage, months.	Depreciation deductions, UAH
1. Personal computers, printers, etc	99900	25	3,25 (70%)	4734,84
2. Department and faculty premises	51600	2,5	3,25 (65%)	222,09
Total				A = 4956,93 ≈ 4957

7. Expenses for electrical power Be are calculated using the formula:

$$B_e = \frac{В \cdot П \cdot Ф \cdot К_п}{К_д}, \quad (5.7)$$

where В – price of 1 kilowatt-hour. Electricity in 2023 W ≈ 4,5 uah/kilowatt;

П – The installed capacity of the equipment kWt; П = 1,05 kWt;

Ф – actual number of equipment operating hours, hours.

Assume, that Ф = 300 hrs;

$К_п$ – power usage coefficient; $К_п$ < 1 = 0,76.

$К_д$ – Useful action coefficient, $К_д$ = 0,61.

Then the expences for electricalpower:

$$В_е = \frac{В \cdot П \cdot Ф \cdot К_п}{К_д} = \frac{4,5 \cdot 1,05 \cdot 300 \cdot 0,76}{0,61} = 1766,96 \approx 1767 \text{ uah.}$$

8. Other expences $В_{ін}$ can be estimated as (50…300)% from the initial salary of the performers:

$$В_{ін} = К_{ін} \times З_о = (0,5..3,0) \times З_о. \quad (5.8)$$

In this case let's assume that $К_{ін}$ = 1,6. Then:

$В_{ін}$ = 1,25 × 30780 = 38 475 uah.

9. Total sum of all the previous expences gives the total expences of the current stage execution by the Student developer - Master's student – В.

In this case:

В = 30780 + 3479 + 7537 + 3000 + 4957 + 1767 + 38475 = 89995 uah.

10. The calculation of the total costs for the development and final refinement of the work we have been done is carried out according to the formula:

$$ЗВ = \frac{В}{\beta}, \quad (5.9)$$

Where β - coefficient characterizing the stage of completion of this work. Since our development still requires refinement, it can be assumed that $\beta \approx 0,65$.

Then: $3B = \frac{89995}{0,65} = 138453,85$ uah or approximately 139 000 uah.

So, the projected total expenses for the system we have developed for searching key phrases in Ukrainian reviews based on artificial intelligence technologies amount to approximately 139,000 hryvnias.

5.3 Calculation of the economic effect from the potential commercialization of the developed system for searching key phrases in Ukrainian language reviews based on artificial intelligence technologies

The market analysis indicates that our developed system for identifying key phrases in Ukrainian language reviews using artificial intelligence technologies will have substantial demand in internet platforms and stores (such as Prom, Rozetka), companies that contain and analyze reviews regarding restaurant businesses and hotels (Booking.com, TripAdvisor), media monitoring companies, and others. Each of these entities will have the ability to receive automated distribution of reviews and automation in the search for factors that most influence the average assessment of various phenomena, events, companies, etc. (for example: regarding positive and negative product qualities, the balance between price and quality, etc.).

Therefore, if our development is implemented starting from January 1, 2024, its results will manifest throughout 2024, 2025, and 2026

The forecast for increasing demand for our development by year is as follows:

a) 2023 - 1 unit (our development);

b) 2024 - +3 units compared to the base year (i.e., 3 clients);

c) 2025 - +6 units compared to the base year (i.e., 6 clients);

d) 2026 - +10 units compared to the base year (i.e., 10 clients).

According to expert conclusions, the potential market price for our development is approximately $26,000, or roughly 1,000,000 Ukrainian Hryvnia. In contrast, similar (though not identical) developments that perform the functions mentioned earlier are priced in the market at up to 800,000 Hryvnia.

The potential increase in net profit $\Delta П_i$, from taking the product to the market will amount to:

$$\Delta П_i = \sum_{1}^{n} (\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\upsilon}{100}), \qquad (5.10)$$

Where $\Delta Ц_o$ – an improvement in the primary qualitative indicator from implementing the outcomes of our development in this year. In our case, this is:

$\Delta Ц_o$ = 1000 – 800 = + 200 000 uah;

N – the main quantitative indicator that defines the scope of activities in the year before the implementation of the development results; N = 1 pcs.;

ΔN – improvement of the main quantitative indicator due to the implementation of the development results.

This improvement will be as follows: in 2024 - +3 units, in 2025 - +6 units, and in 2026 - +10 units.

$Ц_o$ - the primary qualitative indicator (i.e., the price) determining the scope of activity in the year following the implementation of the development results uah; $Ц_o$= 1000 thousand uah;

n – total number of years, during which the positive results from development implementation is expected; in this scenario n = 3;

λ – The coefficient that takes into account the value-added tax (VAT) payment; $\lambda = 0{,}8333$;

ρ – The coefficient that considers the product's profitability. It is recommended to assume ρ = (0,2...0,5); set ρ = 0,5;

υ – the corporate tax rate. In 2023-26 years υ = 18% (assumption).

The potential increase in net profit $\Delta П_1$ for a potential investor during the first year after the possible implementation of our development (2024), it would be:

$$\Delta П_1 = [200 \cdot 1 + 1000 \cdot 3] \cdot 0,8333 \cdot 0,5 \cdot (1 - \frac{18}{100}) \approx 1094 \text{ thousand uah.}$$

For the potential investor during the second year after the possible implementation of our development (2025), it would be calculated similarly:

$$\Delta П_2 = [200 \cdot 1 + 1000 \cdot 6] \cdot 0,8333 \cdot 0,5 \cdot (1 - \frac{18}{100}) \approx 2119 \text{ thousand uah.}$$

The potential increase in net profit $\Delta П_3$ for a potential investor during the first year after the possible implementation of our development during the third year (2026) is totalled:

$$\Delta П_3 = [200 \cdot 1 + 1000 \cdot 10] \cdot 0,8333 \cdot 0,5 \cdot (1 - \frac{18}{100}) \approx 3485 \text{ thousand uah.}$$

The total value of the increased net profits from the potential implementation and commercialization of our development:

$$ПП = \sum_{1}^{т} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.11)$$

Where $\Delta П_i$ – the increase in net profit in each of the years when the results of the completed and implemented work are manifested is as follows;

т – the time period during which the results of the implemented work are manifested is for 3 years, represented by t=3 years.;

τ – the discount rate. Let's assume τ = 0,10 (10%);

t – the period of time from the initiation of the development of the system for searching key phrases in Ukrainian language feedbacks based on artificial intelligence technologies to the moment when potential net profits are obtained by the potential investor can be termed as the "development-to-profit period" or the "investment gestation period.

Then, the present value of the growth of all potential net profits (PP) that a potential investor can gain from the commercialization of our development would be:

$$ПП = \frac{1094}{(1+0,1)^2} + \frac{2119}{(1+0,1)^3} + \frac{3485}{(1+0,1)^4} \approx 904 + 1592 + 2380 = 4876\ 000 \text{ uah.}$$

The present value of investments (PV) that should be allocated towards the implementation of our development would be: PV = (1,0…5,0) × $В_{заг}$.

In our case PV = (1,0…5,0) × 139 = 5 × 139 = 695 thousand uah.

The absolute effect of potential investments made in the implementation of our development would be $Е_{абс}$.

$$Е_{абс} = ПП – PV, \qquad (5.12)$$

The notation " ПП" stands for the present value of the increase in all net profits for the potential investor from the potential commercialization of the development, expressed in currency (hryvnias).

PV – The present value of investments (PV) amounts to 695,000 hryvnias.

The absolute effect from the potential implementation of our development will be:

$$Е_{абс} = 4876 – 695 = 4181\ 000 \text{ uah.}$$

Next, we will calculate the internal rate of return (IRR) of the invested capital:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (5.13)$$

Where $E_{абс}$ – the absolute effect of the invested capital; $E_{абс}$ = 4181 000 uah;

PV – the present value of the initial investments PV = 695 000 uah;

$T_ж$ – development lifecycle, years.

$T_ж$ = 4 years (2023, 2024, 2025, 2026 years)

In this case it would be:

$$E_в = \sqrt[4]{1 + \frac{4181}{695}} - 1 = \sqrt[4]{1 + 6,0158} - 1 = \sqrt[4]{7,0158} - 1 = 1,627 - 1 = 0,627 = 62,7\%.$$

Next, let's determine the minimum profitability, below which it would not be profitable for the potential investor to engage in the commercialization of our development. The minimum profitability or the minimum (barrier) discount rate $\tau_{мін}$ is determined by the formula:

$$\tau_{мін} = d + f, \quad (5.14)$$

where d – the weighted average rate on deposit operations in commercial banks; in 2022-2023 in Ukraine d = (0,10...0,12);

f – indicator characterizing the riskiness of investments;

f = (0,1...0,50). Assume f = 0,30.

In this case:

$$\tau_{мін} = 0,12 + 0,30 = 0,42 \text{ або } \tau_{мін} = 42\%.$$

Given the magnitude $E_в$ = 62,7% > $\tau_{мін}$ = 42%, then a potential investor may indeed be interested in financing and commercializing the development. Next, we

will calculate the payback period for the funds invested in the potential commercialization of the system developed for searching key phrases in Ukrainian language feedback using artificial intelligence technologies.

The payback period $T_{ок}$ is calculated by formula:

$$T_{ок} = \frac{1}{E_{в}}. \quad (5.15)$$

In this case the payback period $T_{ок}$:

$$T_{ок} = \frac{1}{0,627} = 1,59 \text{ years} < 3 \text{ years},$$

this indicates the potential viability of commercializing our developed system for searching key phrases in Ukrainian feedback using artificial intelligence technologies.

Further, a simulation was conducted to model the relationship between the internal rate of return of potential investments and the inflation rate in the country.

If the inflation rate in the country increases to 20%, then:

$$ПП = \frac{1094}{(1+0,2)^2} + \frac{2119}{(1+0,2)^3} + \frac{3485}{(1+0,2)^4} \approx 760 + 1226 + 1680 = 3666\ 000 \text{ uah.}$$

The absolute effect from the potential implementation of our development will amount to:

$$E_{абс} = 3666 - 695 = 2971\ 000 \text{ uah.}$$

Next, we will calculate the internal rate of return (IRR) of the invested investments:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (5.13)$$

where $E_{абс}$ –absolute effect of the invested investments; $E_{абс}$ = 2971 000 uah;

PV – the present value of the initial investments PV = 695 000 uah;

$T_ж$ – development lifecycle, years.

In this case:

$$E_в = \sqrt[4]{1 + \frac{2971}{695}} - 1 = \sqrt[4]{1 + 4,2748} - 1 = \sqrt[4]{5,2748} - 1 = 1,515 - 1 = 0,515 = 51,5\%.$$

Given magnitude $E_в$ = 51,5% > $\tau_{мін}$ = 42%, then the potential investor might be interested in financing and commercializing our development in principle. If the country's inflation rate rises to 30%, then:

$$ПП = \frac{1094}{(1+0,3)^2} + \frac{2119}{(1+0,3)^3} + \frac{3485}{(1+0,3)^4} \approx 647 + 965 + 1220 = 2832\ 000 \text{ uah.}$$

The absolute effect from the potential implementation of our development will be:

$E_{абс}$ = 2832 – 695 = 2137 000 uah.

Next, let's calculate the internal rate of return of the invested investments ($E_в$):

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (5.13)$$

where $E_{абс}$ – the absolute effect from the potential implementation; $E_{абс}$ = 2137 000. uah;

PV – the present value of the initial investments PV = 695 000 uah;

$T_ж$ development lifecycle, years.

In this case:

$$E_в = \sqrt[4]{1+\frac{2137}{695}} - 1 = \sqrt[4]{1+3,0748} - 1 = \sqrt[4]{4,0748} - 1 = 1,42 - 1 = 0,42 = 42,0\%.$$

Given $E_в = 42,0\% \approx \tau_{мін} = 42\%$, then the potential investor might be interested in funding and commercializing our development in principle.

If the inflation rate in the country increases to 40%, then:

$$ІПП = \frac{1094}{(1+0,4)^2} + \frac{2119}{(1+0,4)^3} + \frac{3485}{(1+0,4)^4} \approx 558 + 722 + 907 = 2187\ 000 \text{ uah.}$$

The absolute effect from the potential implementation of our development will be:

$$E_{абс} = 2187 - 695 = 1492 \text{ thousand uah.}$$

Next, let's calculate the internal rate of return of the invested investments ($E_в$):

$$E_в = \sqrt[T_ж]{1+\frac{E_{абс}}{PV}} - 1, \qquad (5.13)$$

where $E_{абс}$ – the absolute effect from the potential implementation; $E_{абс}$ = 1 492 000 uah;

PV – the present value of the initial investments PV = 695 000 uah;

$T_ж$ development lifecycle, years.

In this case:

$$E_в = \sqrt[4]{1+\frac{1492}{695}} - 1 = \sqrt[4]{1+2,1468} - 1 = \sqrt[4]{3,1468} - 1 = 1,332 - 1 = 0,332 = 33,2\%.$$

Given magnitude $E_в = 33,2\% < \tau_{мін} = 42\%$, then a potential investor may not be interested in the commercialization of our development.

The calculations made in the form of graphs are presented in Figure 5.1.

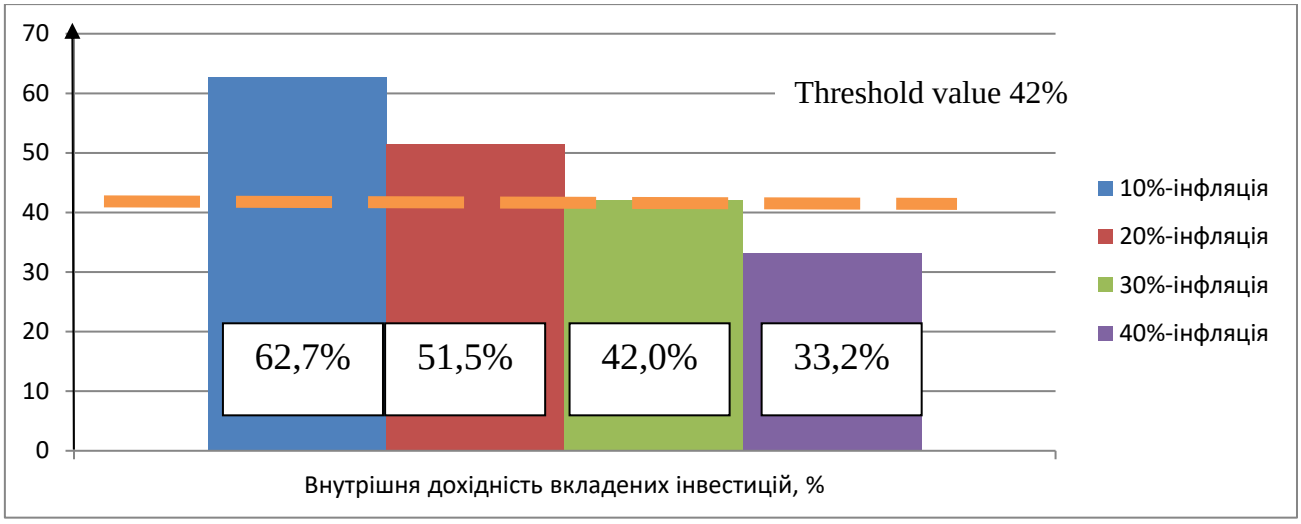


Figure 5.1 – Modeling the relationship between the internal rate of return of potential investments and the inflation rate in the country (10%, 20%, 30%, and 40%).

The analysis of the charts in Figure 5.1 shows that at an inflation rate of 10%, the internal rate of return of investments is Ev = 62.7%, which exceeds the threshold value min = 42%. Therefore, the commercialization of our development may be worthwhile. At an inflation rate of 20%, the internal rate of return of investments is Ev = 51.5%, which also exceeds the threshold value min = 42% The analysis reveals that at an inflation rate of 30%, the internal rate of return of investments is Ev = 42.0%, which equals the threshold value min = 42%. Hence, the commercialization of our development can also be reasonable.

However, at an inflation rate of 40%, the internal rate of return of investments allocated in the commercialization of our development amounts to Ev = 33.2%, which falls below the threshold value min = 42%. Therefore, the potential investor's commitment to commercialize our development might be in question. However, a final decision on this matter requires additional calculations (possibly reducing the investment risk level, increasing the demand for the development, enhancing the selling price of the development, etc.).

The outcomes of the performed economic part of the master's qualification work are summarized in the table 5.6.

Table 5.6 – The outcomes of the performed economic part

Indicators	Defined in Technical task	Attained in the Master's thesis	Conclusion
1. Development expenses	Less than 150 000 uah	139 000 uah.	Achieved
2. Absolute effect from implementing the development, thousands of UAH	More than ≈ 4000 000uah	4 181 000 uah (with 10%-inflation)	Completed
3. Internal Rate of Return on Investments, %	More than 42%	62,7% (with 10%-inflation)	Achieved
4. Payback Period of Investments, years	Less than in 3 yrs	1,59 yrs	Completed

Thus, the key technical and economic indicators of the developed system for searching key phrases in Ukrainian language within feedbacks using artificial intelligence technologies, as defined in the technical task, have been achieved.

CONCLUSIONS

A new approach to key-phrases retrieval for Ukrainian language is presented in this work. A comprehensive analysis of previous work, competitive approaches and problem at hand was conducted, confirming a need for more adaptable and effective approach. In order to tackle the problem, a method based on training discriminative model and reverse-engineering of its decision process based on explainable AI was provided. The creation of technology for key-phrases retrieval required to solve the list of subtasks, including analysis and choosing of programming language and libraries, data collection, processing, modelling and incorporation of explainable AI.

Based on analysis of programming languages and tools, it was decided to utilize Python language along with the task-specific libraries. Due to the requirement of convenient integration and adaptability to unseen data, BPE tokenization was applied along with Word2Vec embeddings.

The data was collected from TripAdvisor and Rozetka by utilizing web-scrapping techniques. Due to incorporated noise, the data was processed and automatically filtered by applying machine learning. As the result a medium-size dataset of Ukrainian reviews and corresponding ratings consisting of 662907 samples was prepared. Based on data analysis, the impactful insights were gathered and incorporated into the modeling stage.

During modeling stage, list of models was tried out, resulting into models trained for sentiment analysis and reviews rating estimation. In order to fight overfitting and noisy data, noise-tolerant objectives were used along with Dropout technique. The conducted experiments w.r.t f1-macro score revealed that architecture consisting of Attention mechanism, two LSTMs, pretrained Word2Vec embeddings and Huber-loss achieves the best result along all the domains. The final model achieves averaged f1-macro of 0.719 for task of sentiment analysis and 0.5526 for task of rating estimation.

During explainability and key-phrases retrieval stage, the experiments included comparison of Attention and LIME techniques. In order to retrieve key-phrases in the form of n-grams, specific aggregation algorithm was developed. Based on human evaluation it was revealed that Attention on average achieves better results, while LIME provides better coverage of phrases. Further experiments revealed that for applying method to standalone reviews, the need of thorough hyper-parameters selection w.r.t aggregation algorithm arises. Final experiment, showed that approach is applicable to other domains without a need for finetuning to new data. The comparison with most similar analog proved that constructed algorithm is more memory efficient and is more capable of adaptation to unseen data and new domains, which fully satisfies the purpose of work.

Despite of the fact that current approach provides a possibility of key-phrases retrieval for cross-domain reviews, there are still points to improve including extension to new domains, data quality, model's tolerance to sarcasm and noise, multi n-gram retrieval process. In the further work, the plans are to enhance current solution in terms of key-phrases retrieval by applying both modeling and algorithmic techniques and adopt it to unsupervised aspect-based sentiment analysis and compare it to other methods in the field. Even though the experiments with key-phrases retrieval algorithm were conducted in Ukrainian language, it can easily be adopted to any other. All the models, code and data are open-sourced for further analysis and enhancements of Ukrainian NLP.

REFERENCES

1. Vikas Yadav and Steven Bethard. A Survey on Recent Advances in Named Entity Recognition from Deep Learning models. In Proceedings of the 27th International Conference on Computational Linguistics, pages 2145–2158, Santa Fe, New Mexico, USA. Association for Computational Linguistics. (2018)

2. Sevgili, Özge & Shelmanov, Artem & Arkhipov, Mikhail & Panchenko, Alexander & Bie-mann, Chris. Neural entity linking: A survey of models based on deep learning. Semantic Web. 13. 1-44. 10.3233/SW-222986. (2022).

3. Qin Zhang, Shangsi Chen, Dongkuan Xu, Qingqing Cao, Xiaojun Chen, Trevor Cohn, and Meng Fang. A Survey for Efficient Open Domain Question Answering. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 14447–14465, Toronto, Canada. Association for Computational Linguistics. (2023).

4. Gianni Brauwers and Flavius Frasincar. A Survey on Aspect-Based Sentiment Classification. ACM Comput. Surv. 55, 4, Article 65, 37 pages. https://doi.org/10.1145/3503044 (2023).

5. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

6. Bahdanau, D., Cho, K. & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate (cite arxiv:1409.0473Comment: Accepted at ICLR 2015 as oral presentation)

7. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15, 1 (January 2014), 1929–1958.

8. TF–IDF. In: Sammut, C., Webb, G.I. (eds) Encyclopedia of Machine Learning. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-30164-8_832 (2011).

9. Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. Annals of Statistics, 1189–1232.

10. Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273–297.

11. Cox, D. R. (1958). The regression analysis of binary sequences. Journal of the Royal Statistical Society: Series B (Methodological), 20(2), 215–232.

12. Tian, Zhenya & Xiao, Jialiang & Feng, Haonan & Wei, Yutian. (2020). Credit Risk Assessment based on Gradient Boosting Decision Tree. Procedia Computer Science. 174. 150-160. 10.1016/j.procs.2020.06.070.

13. Bijoyan Das, Sarit Chakraborty. An Improved Text Sentiment Classification Model Using TF-IDF and Next Word Negation. URL: https://arxiv.org/abs/1806.06407

14. K. S. Kalaivani, R. Felicia Grace, M. Aarthi, M. Boobeash. Classification of Sentiment Reviews using POS based Machine Learning Approach. URL: https://www.ijert.org/research/classification-of-sentiment-reviews-using-pos-based-machine-learning-approach-IJERTCONV6IS04061.pdf

15. Lithgow-Serrano O, Cornelius J, Kanjirangat V, Méndez-Cruz CF, Rinaldi F. Improving classification of low-resource COVID-19 literature by using Named Entity Recognition. Genomics Inform. 2021 Sep;19(3):e22. doi: 10.5808/gi.21018. Epub 2021 Sep 30. PMID: 34638169; PMCID: PMC8510872.

16. Subbaraju Pericherla and E Ilavarasan 2021 IOP Conf. Ser.: Mater. Sci. Eng. 1085 012008

17. Yoon Kim. Convolutional Neural Networks for Sentence Classification. URL: https://arxiv.org/abs/1408.5882

18. Devendra Singh Sachan, Manzil Zaheer, Ruslan Salakhutdinov. Revisiting LSTM Networks for Semi-Supervised Text Classification via Mixed Objective Function. URL: https://arxiv.org/abs/2009.04007

19. Chunting Zhou, Chonglin Sun, Zhiyuan Liu, Francis C.M. Lau. A C-LSTM Neural Network for Text Classification. URL: https://arxiv.org/abs/1511.08630

20. Peng Zhou , Zhenyu Qi , Suncong Zheng, Jiaming Xu, Hongyun Bao, Bo Xu. Text Classification Improved by Integrating Bidirectional LSTM with Two-dimensional Max Pooling. URL: https://arxiv.org/pdf/1611.06639.

21. Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based LSTM for Aspect-level Sentiment Classification. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 606–615, Austin, Texas. Association for Computational Linguistics.

22. Marco Tulio Ribeiro and Sameer Singh and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. Proceedings of the 22nd International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, (2016).

23. Sheikh Rabiul Islam, William Eberle, Sheikh Khaled Ghafoor, Mohiuddin Ahmed. Explainable Artificial Intelligence Approaches: A Survey. URL: https://arxiv.org/pdf/2101.09429

24. Marco Ancona, Enea Ceolini, Cengiz Öztireli, Markus Gross. Towards better understanding of gradient-based attribution methods for Deep Neural Networks. URL: https://arxiv.org/abs/1711.06104

25. Junlin Wang, Jens Tuyls, Eric Wallace, Sameer Singh. Gradient-based Analysis of NLP Models is Manipulable. URL: https://aclanthology.org/2020.findings-emnlp.24.pdf

26. Jianxing Yu, Zheng-Jun Zha, Meng Wang, Tat-Seng Chua. Aspect Ranking: Identifying Important Product Aspects from Online Consumer Reviews. URL: https://aclanthology.org/P11-1150.pdf

27. Hu, Minqing and Liu, Bing 2004. Mining opinion features in customer reviews AAAI.

28. Wu, Yuanbin and Zhang, Qi and Huang, Xuanjing and Wu, Lide 2009. Phrase dependency parsing for opinion mining Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3

29. Aitor Garc´ıa-Pablos, Montse Cuadros, German Rigau. V3: Unsupervised Aspect Based Sentiment Analysis for SemEval-2015 Task 12. URL: https://aclanthology.org/S15-2121.pdf.

30. Hercig, Tomás Brychcín, Tomás Svoboda, Lukás Konkol, Michal Steinberger, Josef. Unsupervised Methods to Improve Aspect-Based Sentiment Analysis in Czech. URL: https://www.redalyc.org/articulo.oa?id=61547469006.

31. Babenko, Dmytro. Determining sentiment and important properties of Ukrainian-language user reviews : Master Thesis : manuscript rights / Dmytro Babenko ; Supervisor Vsevolod Dyomkin ; Ukrainian Catholic University, Department of Computer Sciences. – Lviv : 2020. – 35 p.

32. Mikolov, T., Chen, K., Corrado, G. & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *CoRR*, abs/1301.3781.

33. Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2016). Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606*.

34. Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

35. Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden,

Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

36. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., … Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32* (pp. 8024–8035). Curran Associates, Inc. Retrieved from http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

37. Rehurek, R., & Sojka, P. (2011). Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, *3*(2).

38. Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, et al.. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

39. Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). New York, NY, USA: ACM. https://doi.org/10.1145/2939672.2939785

40. Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, et al. Lightgbm: A highly efficient gradient boosting decision tree. Advances in neural information processing systems. 2017;30:3146–54.

41. Chollet, F., & others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras

42. Korobov M.: Morphological Analyzer and Generator for Russian and Ukrainian Languages // Analysis of Images, Social Networks and Texts, pp 320-332 (2015).

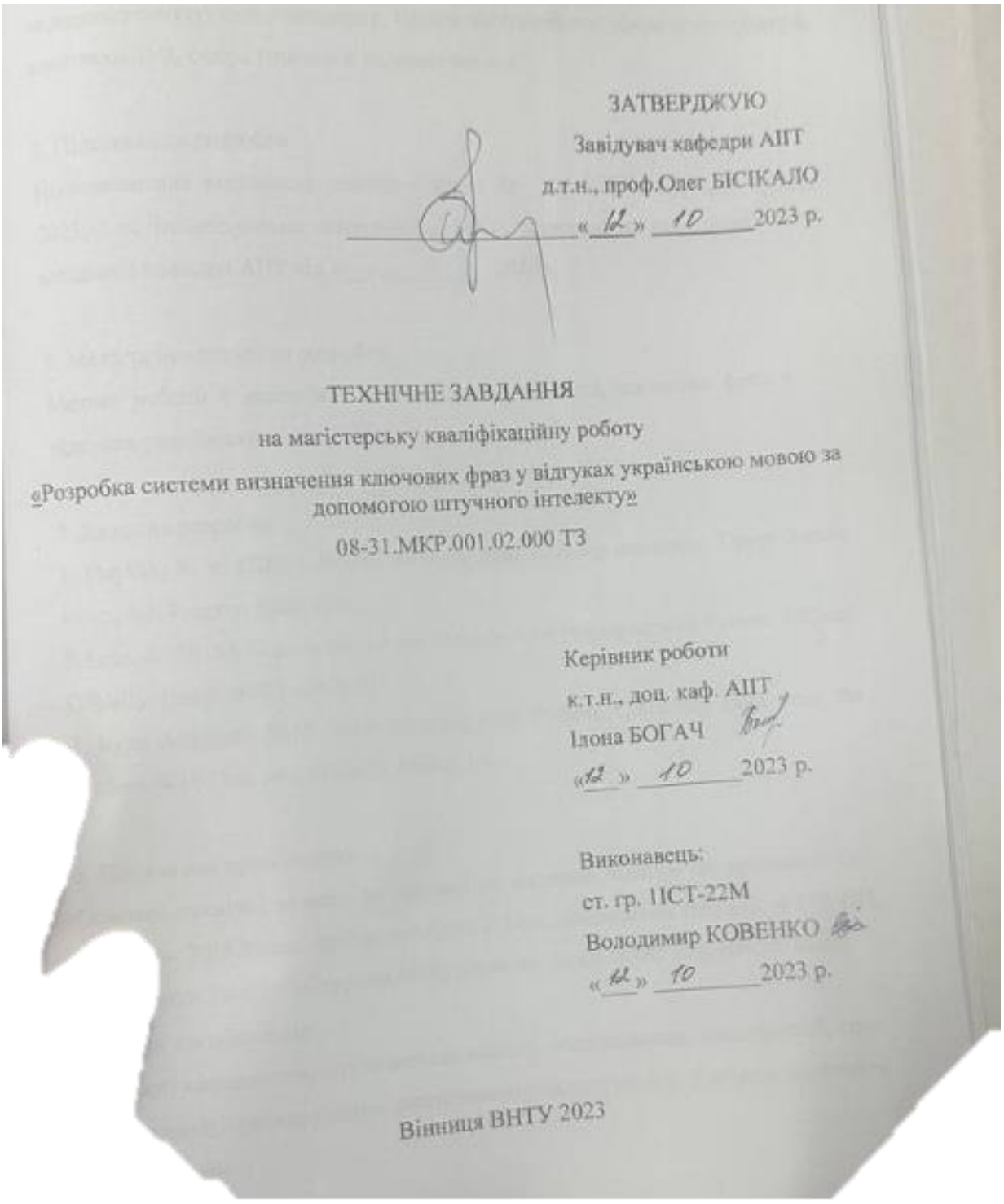
43.Rafael Müller, Simon Kornblith, Geoffrey Hinton. When Does Label Smoothing Help? URL: https://arxiv.org/abs/1906.02629 Author, F.: Contribution title. In: 9th International Proceedings on Proceedings, pp. 1–2. Publisher, Location (2010). LNCS Homepage, http://www.springer.com/lncs, last accessed 2016/11/21.

44. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт. / Укладачі В.О. Козловський, О.Й. Лесько, В.В.Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

APPENDIXES

Appendix A (required)

Technical Task



ЗАТВЕРДЖУЮ
Завідувач кафедри АІІТ
д.т.н., проф.Олег БІСІКАЛО
«12» 10 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на магістерську кваліфікаційну роботу
«Розробка системи визначення ключових фраз у відгуках українською мовою за допомогою штучного інтелекту»
08-31.МКР.001.02.000 ТЗ

Керівник роботи
к.т.н., доц. каф. АІІТ
Ілона БОГАЧ
«12» 10 2023 р.

Виконавець:
ст. гр. 1ІСТ-22М
Володимир КОВЕНКО
«12» 10 2023 р.

Вінниця ВНТУ 2023

1. Назва та галузь застосування

Розробка системи визначення ключових фраз у відгуках українською мовою за допомогою штучного інтелекту. Галузь застосування: сфера моніторингу й аналітики ЗМІ, сфера торгівлі й надання послуг.

2. Підстава для розробки

Підставою для виконання роботи є наказ №247 по ВНТУ від «18» __09___ 2023р., та індивідуальне завдання на МКР, затверджене протоколом №1 засідання кафедри АІІТ від «30» ____08____ 2023р.

3. Мета та призначення розробки

Метою роботи є розробка системи для визначення ключових фраз у відгуках українською.

4. Джерела розробки

1. Haykin, S. S. (2009). *Neural networks and learning machines*. Upper Saddle River, NJ: Pearson Education.

2. Grus, J. (2015). *Data Science from Scratch: First Principles with Python*. Beijing: O'Reilly. ISBN: 978-1-4919-0142-7

3. Ryan Mitchell. 2015. Web Scraping with Python: Collecting Data from the Modern Web (1st. ed.). O'Reilly Media, Inc.

5. Показники призначення

Основні технічні вимоги та мінімальні системні вимоги до програми: ОС: Windows XP/Ubuntu, процесор: Core 2 Duo, оперативна пам’ять: 4 GB ОП, відеокарта: Intel HD Graphics 4000, місце на диску: 2 GB доступного місця.

Методи дослідження:

В роботі використовуються методи аналізу, моделювання, класифікації, спостереження, прогнозування, експерименту та прагматичної моделі наукового дослідження.

Результати роботи програми: визначення оцінки відгуку;визначення тональності відгуку; визначення релевантних фраз відносно передбаченої оцінки відгуку;

6. Економічні показники

До економічних показників входять:

- витрати на розробку – не більше 150 тис. грн;
- абсолютний ефект від впровадження розробки – не менше 4 млн. грн;
- внутрішня дохідність інвестицій – не менше 42%;
- термін окупності – не більше 3 років.

7. Стадії розробки

а) Аналіз предметної області 20.09 –02.10

б) Вибір інструментів розробки, мови програмування й бібліотек 02.10 – 10.10

в) Збір даних й їх обробка 11.10 –23.10

г) Тренування моделей й створення системи для визначення ключових фраз 23.10 – 08.11

д) Економічна частина 08.11 – 11.11

е) Оформлення матеріалів до захисту МКР 11.11 – 20.11

8. Порядок контролю та приймання

Рубіжний контроль провести до «01» грудня 2023 р.

Попередній захист МКР провести до «21» грудня 2023 р.

Захист МКР провести до «18» грудня 2023 р.

Розробив студент групи 1ІСТ-22м _____ Володимир КОВЕНКО

Appendix B (required)

List of graphic materials

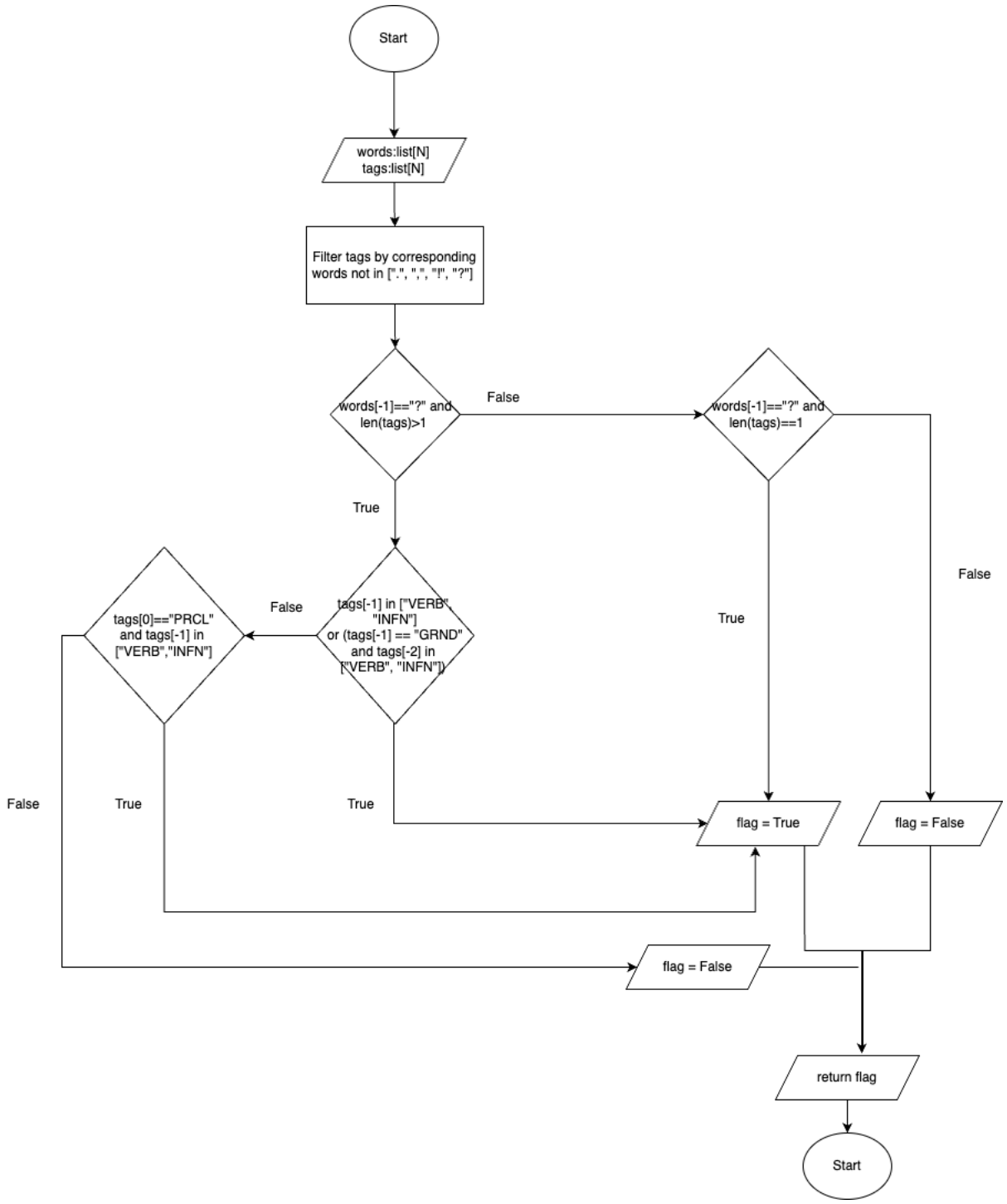


Figure B.1 - Schematic view of an algorithm for questions detection

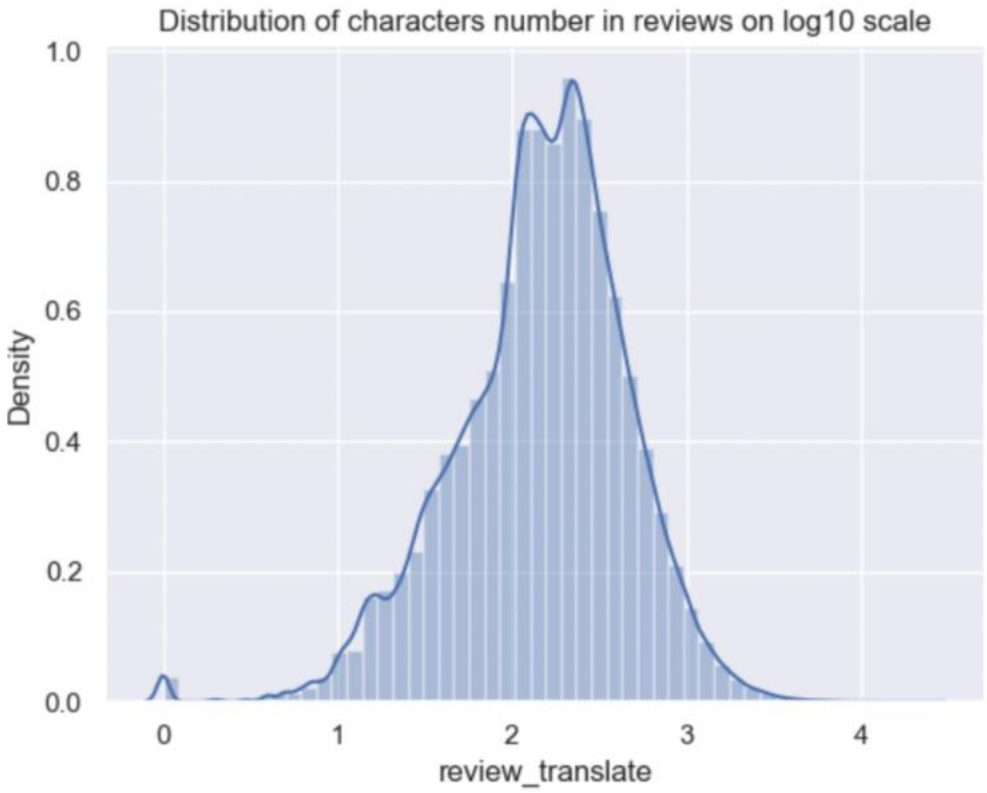


Figure B.2 – Distribution of characters number in reviews

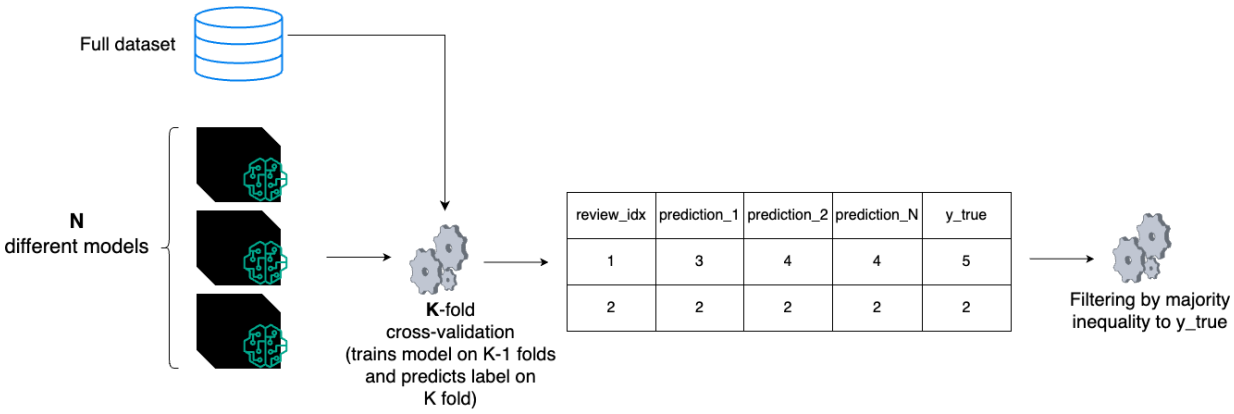


Figure B.3 – Scheme showcasing automated data filtering approach

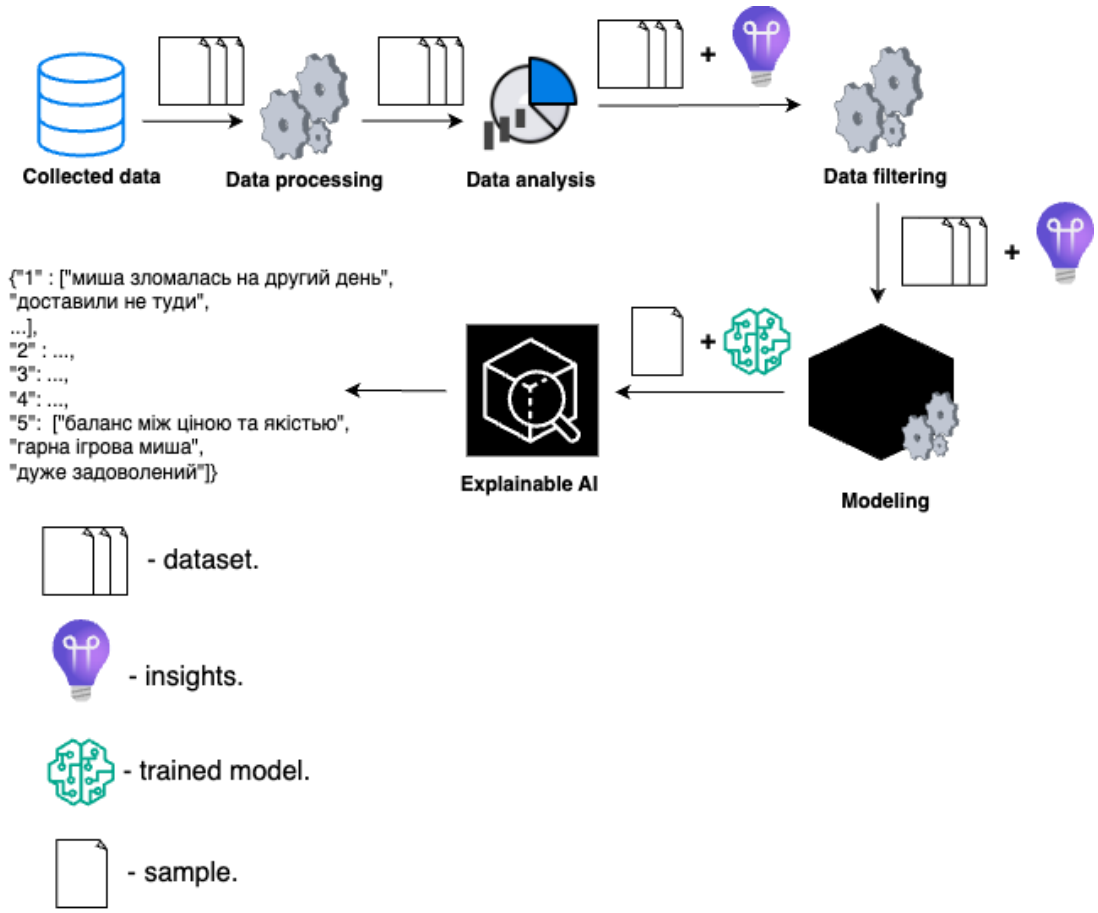


Figure B.4 – Scheme showcasing overall approach to key-phrases retrieval

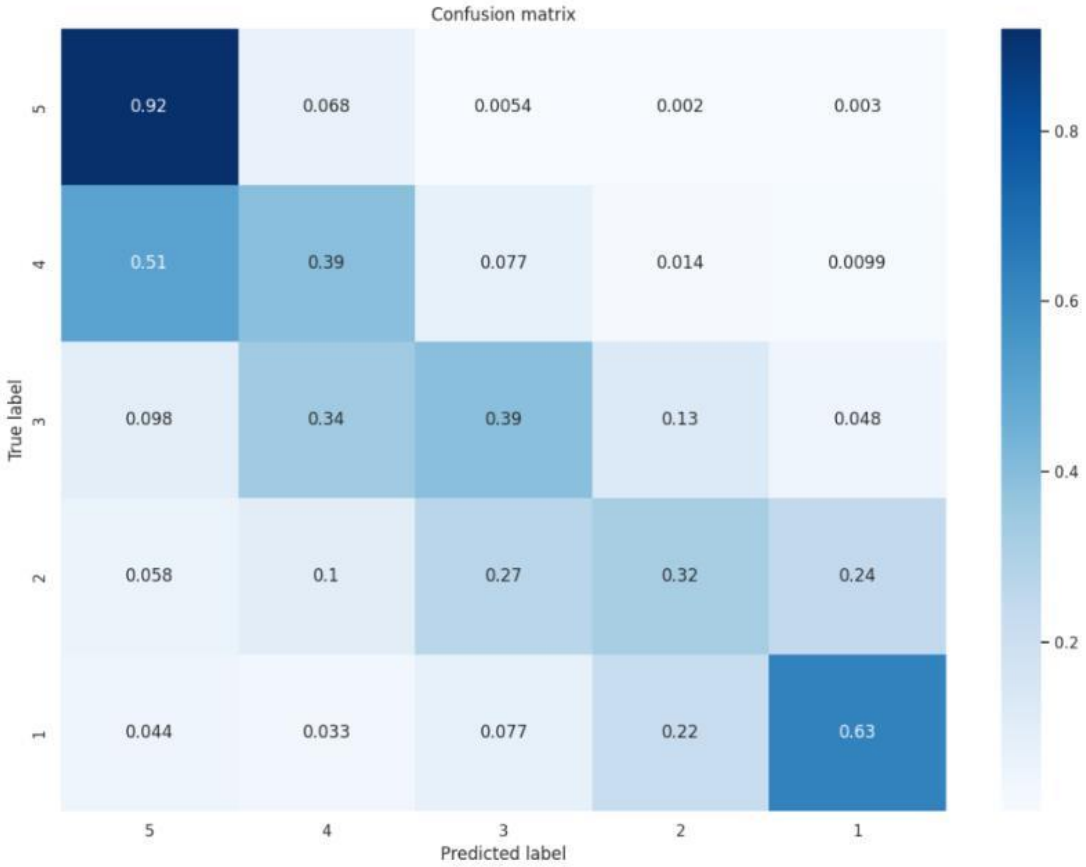


Figure B.5 – Confusion matrix of best model on rating scores estimation

Entity name : Ibiza_Club_Restaurant, average rating : 2.94

Most influencial phrases for rating of 1.0

Lime results : ('місце яке відвернуло нас', 'дуже поганий сервіс за', 'туристична пастка яка не', 'вишибали відмовили у в', 'офіціант був грубим і', 'жахлива музика більше нічого', 'музика це погано не', 'було відмовлено у в', 'сказати що це марна', 'близького сходу такої нісенітниці', 'остерігайтеся клуб це туристична', 'всередині клубу щоб вигнати', 'смішно абсолютне лихо вечері', 'не на літо не', 'в основному говорить що')

Attention results : ('рекомендував це місце .', 'поганий сервіс , за', 'ось такий сервіс .', 'вірте ! ! !', 'ніколи нікому не рекомендував', 'годуйте жахливий народ .', 'воно того не варте', '. ніколи більше ібіца', '. такої нісенітниці бути', 'абсолютно шокує те ,', 'грубі . . і', 'витрачайте гроші кудись ще', 'а вишибали дуже грубі', 'найгірший досвід в цьому', 'йдіть і не витрачайте', 'і ми просимо ,', 'їдьте в одесу в', 'на літо не їдьте', 'випадку . кожен asiat', 'жахлива музика . більше')

Most influencial phrases for rating of 2.0

Lime results : ('офіціант був грубим і', 'музика це погано не', 'них незрозуміло море брудне', 'просто жахлива музика більше', 'занадто дорого для пропонованої', 'кажу воно того не', 'і волю дарувати клієнтам', 'закладі такого рівня дуже', 'але пристойний персонал і', 'або йти кальян взяв', 'брудне тому відпочивайте тільки', 'не їх їжа музика', 'остерігайтеся клуб це туристична', 'сказати що це марна', 'немає сенсу його там') 15

Attention results : ('витрачайте свій час ,', 'погано , не витрачайте', 'брязкіт хамів . оплата', 'йдучи своїми . інакше', 'туристів , відмінний від', 'рахунок « зроби сам', 'від того , який', 'господар своєї справи .', '- річ смішна .', 'інакше україна – прекрасна', 'абсолютно не цікавиться думками', 'країна для подальшого вивчення', 'там немає . брязкіт', 'був грубим і рахунок', 'залишає бажати кращого .', 'офіціанток і барменів .', 'оплата тільки готівкою .', 's дрес - кодом', 'фантастична , але я', 'сам » для туристів') 20

Most influencial phrases for rating of 3.0

Lime results : ('харчування було хорошим сервісом', '80 через його розташування', 'вигляд обслуговування середнє але', 'є місце бути наприклад', 'але пристойний персонал і', 'вони виправлять рівень сервісу', 'привезеш а те що', 'море брудне тому відпочивайте', 'пляж брудний водорості не', 'мені і платіть за', 'всю дорогу до входу', 'такого рівня дуже незручно', 'було антураж закладу непоганий', 'йдучи своїми інакше україна', 'коштував своїх витрат хоча') 15

Attention results : ('рівня дуже незручно і', 'платіть за те ,', 'повірте мені і платіть', 'брудне , тому відпочивайте', 'морських водоростей (вам', 'найближчий банкомат не працював', 'відпочивайте тільки біля басейну', 'утікачів і охоронців більше', 'незрозуміло . море брудне', 'чітким прозорим ціноутворенням .', 'досить грубий і байдужий', 'середнім , але при', 'морською водою плюс дитячий', 'сервісом , було середнім', 'і якось дивно .', 'ще повне шматочків морських', 'підтримує кредитні картки ,', '- ха , пародія', 'дуже зайняте . повірте', 'більше , ніж відпочиваючих') 20

Most influencial phrases for rating of 4.0

Lime results : ('обслуговування середнє але з', 'пляж брудний водорості не', 'хай я з пакистану', 'не встигають загалом хороший', 'рацикт дорожня нудна музика', 'дуже чистий простір ванної', 'хороший клубний будинок але', 'на морі часом рівні', 'він коштував своїх витрат', 'хороша музика смачні коктейлі', 'музика їжа хороша тільки', 'ціни в барі дозволять', 'зрозуміли що це правильне', 'одного з басейнів місце', 'і тому в цілому') 15

Attention results : ('дуже смачно .', 'прибирали . на території', ', водорості не прибирали', 'зовсім не годиться .', 'хороший відпочинок на морі', 'я відвідую цей клуб', 'виправлять рівень сервісу .', 'жертви заплатять їм квиток', '(шезлонг + парасолька', 'плюс сервіс погіршується .', '. я б оцінив', 'довго . багато людей', 'навколо є водорості ,', 'не на належному рівні', 'попереду ще багато роботи', '. велика кількість панянок', 'жахливі , але басейни', 'танцює дуже хороші напої', 'оцінив це дуже добре', 'панянок , які приходять') 20

Most influencial phrases for rating of 5.0

Lime results : ('найвищому рівні заборонено в', 'приємніше сподобалося як змінювалося', 'для такого місця відмінний', 'і релаксу це привід', 'липні 2017 року шезлонг', 'офіціанти на ібіці доставили', 'я поїхав у липні', 'просто одне з найкращих', 'ібіца в одесі це', 'персонал повністю втратив здатність', 'як і слід було', 'приголомшливе місце на морі', 'стали носити яскраві куртки', 'рекомендую це місце дуже', 'прилетіли з усієї дороги') 15

Attention results : ('сервіс для 5 ти', 'залишала бажати кращого ,', "рівні . заборонено в'їзд", '. обслуговування середнє ,', '+ , атмосфера -', 'комфорту та елегантності місця', '. дуже затишно і', 'атмосфера зору і релаксу', '. я гаряче переповідаю', 'рекомендую всім відвідати ,', 'та персонал чудові .', 'величезні постановки , на', 'насолодитеся музикою , чудовим', 'їжа , круті шоу', "в'їзд через мій азіатський", 'добре провести там час', 'крута атмосфера . у', 'передбачають шизлонг і матрац', 'виступи , красиві танцюристи', 'собі дійсний платний квиток') 20

Figure B.6 – Key-phrases retrieval example for restaurant

Appendix C (required)

Excerpt from the protocol of the competition commission

ВІДОМОСТІ
про авторів та наукового керівника наукової роботи
Entropy
(шифр)

Автор 1	Науковий керівник
1. Прізвище *Ковенко*	1. Прізвище *Богач*
2. Ім'я (повністю) *Володимир*	2. Ім'я *Ілона*
3. По батькові (повністю) *Андрійович*	3. По батько *Віталіївна*
4. Повне найменування та місцезнаходження вищого навчального закладу, у якому навчається автор *Вінницький національний технічний університет, м. Вінниця, вул. Хмельницьке шосе, 95*	4. Місце роботи, телефон, email *Вінницький національний технічний університет, +380674305530, ilona.bogach@gmail.com*
5. Факультет *інтелектуальних інформаційних технологій та автоматизації*	5. Посада *доцент кафедри автоматизації та інтелектуальних інформаційних технологій*
6. Курс *2 (магістерський)*	6. Науковий ступень *кандидат технічних наук*
7. Результати роботи *прийняті до друку на конференцію "Mathematical Modeling and Simulation of Systems(MODS'2023)"*	7. Вчене звання *доцент*
8. Результати роботи впроваджено -	
9. Місце проживання, телефон, e-mail *м. Вінниця, Лізи Чайкіної 2, кв 32, (093)141 56 21, volodymyr.kovenko@vntu.edu.ua*	

Автор 2
1. Прізвище *Абдуллаєв*
2. Ім'я (повністю) *Олексій*
3. По батькові (повністю) *Алліжонович*
4. Повне найменування та місцезнаходження вищого навчального закладу, у якому навчається автор *Вінницький національний технічний університет, м. Вінниця, вул. Хмельницьке шосе, 95*
5. Факультет *інтелектуальних інформаційних технологій та автоматизації*
6. Курс *2 (магістерський)*
7. Результати роботи *прийняті до друку на конференцію "Mathematical Modeling and Simulation of Systems(MODS'2023)"*
8. Результати роботи впроваджено -
9. Місце проживання, телефон, e-mail *м. Вінниця, просп Юності 40, кв.66 (098) 742 34 80, oleksii.abdullaev@gmail.com*

Науковий керівник _____ Ілона БОГАЧ

Автор роботи _____ Володимир КОВЕНКО

Автор роботи _____ Олексій АБДУЛЛАЄВ

Рішенням конкурсної комісії Вінницького національного технічного університету студенти Ковенко В.А. та Абдуллаєв О.А. рекомендуються для участі у другому турі Всеукраїнського конкурсу студентських наукових робіт зі штучного інтелекту

Голова конкурсної комісії:
Проректор з науково-педагогічної роботи та організації навчального процесу _____ *Олександр ПЕТРОВ*

М.П. _____ *2023 року*

Appendix D (required)

Conference participation certificate

Appendix E (required)

Code for data collection and processing, model training and inference

```
import bs4
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import requests
from multiprocessing.pool import ThreadPool
from tqdm import tqdm
import time
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import Select
from selenium.common.exceptions import ElementClickInterceptedException,
StaleElementReferenceException
import os
from random_user_agent.user_agent import UserAgent
from random_user_agent.params import SoftwareName, OperatingSystem
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import json
import pickle
import urllib
sns.set()

from multiprocessing.pool import Pool
from contextlib import closing

def multiprocess_func(main_input, func, additional_inputs=None,
                      gather_func=None, to_split=True, gather_func_args=None,
                      chunk_size=100, n_processes=8):
    if not gather_func_args:
        gather_func_args = []
    if not additional_inputs:
        additional_inputs = []
    if not gather_func:
        gather_func = lambda x: [z for i in x for z in i]
    if to_split:
        splitted = [(main_input[i:i + chunk_size], *additional_inputs) if ad-
ditional_inputs else main_input[i:i + chunk_size]\
                    for i in range(0, len(main_input), chunk_size)]
    else:
        splitted = [(i, *additional_inputs) if additional_inputs else i for i
in main_input]
    with closing(Pool(n_processes)) as p:
        result = list(tqdm(p.imap(func, splitted),
                           total=len(splitted)))
    return gather_func(result, *gather_func_args)

"""
# First level parsing
"""

software_names = [SoftwareName.CHROME.value]
operating_systems = [OperatingSystem.LINUX.value, OperatingSystem.WIN-
DOWS.value, OperatingSystem.MACOS.value]
user_agent_rotator = UserAgent(software_names=software_names,
                               operating_systems=operating_systems,
                               limit=100)
main_link = 'https://www.tripadvisor.ru/Hotels-g294473-Ukraine-Ho-
tels.html#LEAF_GEO_LIST'
```

```
main_url = 'https://www.tripadvisor.ru/'

def parse_sites(main_link, user_agent_rotator, max_ex=100):
    first_button_xpath = '//*[@id="component_7"]/div/button'
    next_page_xpath = '//*[@id="taplc_main_pagination_bar_ho-
tels_less_links_v2_0"]/div/div/div/span[2]'

    user = user_agent_rotator.get_random_user_agent()
    custom_options = webdriver.ChromeOptions()
    custom_options.add_argument(f'user_agent={user}')

    driver = webdriver.Chrome(options=custom_options)
    driver.get(main_link)
    WebDriverWait(driver, 90).until(EC.presence_of_element_located((By.XPATH,
first_button_xpath)))
    driver.find_element(by=By.XPATH, value=first_button_xpath).click()

    pages = [driver.page_source]
    ex_counter=0
    while True:
        try:
            WebDriverWait(driver, 90).until(EC.presence_of_element_lo-
cated((By.XPATH, next_page_xpath)))
            driver.find_element(by=By.XPATH, value=next_page_xpath).click()
            ex_counter = 0
        except Exception as ex:
            if not isinstance(ex, (StaleElementReferenceException, Ele-
mentClickInterceptedException)):
                print(ex)
                break
            else:
                ex_counter+=1

            if ex_counter>=max_ex:
                break
        time.sleep(15)
        pages.append(driver.page_source)
    driver.quit()
    return pages

def parse_first_lvl(page):
    soup = bs4.BeautifulSoup(page)
    to_save = []
    for ui_column in soup.find_all('div', {'class':'ui_column is-8 main_col
allowEllipsis'}):
        try:
            bubble_rating_parsed = ui_column.find('a', {'data-
clicksource':'BubbleRating'})

            to_save.append((bubble_rating_parsed.get('alt'), bubble_rat-
ing_parsed.get('href'),
                            ui_column.find('div', {'class':'listing_ti-
tle'}).text))
        except:
            pass
    return to_save

pages = parse_sites(main_link, user_agent_rotator)

hotels_df = pd.DataFrame(multiprocess_func(pages, parse_first_lvl,
                         gather_func=None, to_split=False,
                         n_processes=8), columns=['rating', 'link', 'title'])
```

```
hotels_df = hotels_df.drop_duplicates()

hotels_df['link'] = hotels_df['link'].apply(lambda x: url-
lib.parse.urljoin(main_url, x))
hotels_df['title'] = hotels_df['title'].apply(lambda x:
'.'.join(x.split('.')[1:]).strip())
hotels_df['rating'] = hotels_df['rating'].apply(lambda x:
float(x.split('of')[0].strip().replace(',','.')))
hotels_df['title'] = hotels_df['title'].apply(lambda x: x.replace('/','\\'))

hotels_df['parsed'] = False

hotels_df.to_csv('hotels_links.csv', index=False)



import bs4
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import requests
from multiprocessing.pool import ThreadPool
from tqdm import tqdm
import time
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import Select
from selenium.common.exceptions import ElementClickInterceptedException,
StaleElementReferenceException
import os
from random_user_agent.user_agent import UserAgent
from random_user_agent.params import SoftwareName, OperatingSystem
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import json
import pickle
import urllib
from functools import partial
from selenium.webdriver.common.action_chains import ActionChains
from random_user_agent.user_agent import UserAgent
from random_user_agent.params import OperatingSystem, SoftwareName
import pyautogui
import threading
import multiprocessing
from selenium.webdriver.common.proxy import Proxy, ProxyType
sns.set()

def augment_link(link, num):
    before_link, after_link = link.split('Reviews')
    return before_link+'Reviews-'+f'or{num*5}'+after_link

import stem
from multiprocessing.pool import Pool
from contextlib import closing

def multiprocess_func(main_input, func, additional_inputs=None,
                      gather_func=None, to_split=True, gather_func_args=None,
                      chunk_size=100, n_processes=8):
    if not gather_func_args:
        gather_func_args = []
    if not additional_inputs:
        additional_inputs = []
    if not gather_func:
```

```
        gather_func = lambda x: [z for i in x for z in i]
    if to_split:
        splitted = [(main_input[i:i + chunk_size], *additional_inputs) if ad-
ditional_inputs else main_input[i:i + chunk_size]\
                    for i in range(0, len(main_input), chunk_size)]
    else:
        splitted = [(i, *additional_inputs) if additional_inputs else i for i
in main_input]
    with closing(Pool(n_processes)) as p:
        result = list(tqdm(p.imap(func, splitted),
                           total=len(splitted)))
    return gather_func(result, *gather_func_args)

"""
# Second level parsing with translate + selenium
"""

hotels_df = pd.read_csv('hotels_links.csv')

software_names = [SoftwareName.CHROME.value]
operating_systems = [OperatingSystem.LINUX, OperatingSystem.MACOS.value,
                     OperatingSystem.WINDOWS]

user_agent_rotator = UserAgent(software_names=software_names,
                               operating_systems=operating_systems, limit=ho-
tels_df.shape[0]*2)


def save_html(file, path):
    with open(path+'.html', 'w') as f:
        f.write(file)

def get_driver(user_agent, run_headless=False):
    custom_options = webdriver.ChromeOptions()
    prox = "socks5://localhost:9050"
    custom_options.add_argument('--proxy-server=%s' % prox)

    if run_headless:
        custom_options.add_argument('headless')
    custom_options.add_argument("lang=uk")
    custom_options.add_argument('--ignore-certificate-errors')
    custom_options.add_argument('--disable-dev-shm-usage')
    custom_options.add_argument(f'user-agent={user_agent}')
    driver = webdriver.Chrome(options=custom_options)
    return driver

def check_ip_proxy(address):
    options = webdriver.ChromeOptions()
    options.add_argument('--ignore-certificate-errors')
    options.add_argument('headless')

    prox = "socks5://localhost:9050"
    options.add_argument('--proxy-server=%s' % prox)

    driver = webdriver.Chrome(options=options)
    driver.get('https://api.ipify.org/')
    ip_address = driver.find_element(By.TAG_NAME, "body").text
    driver.quit()

    return ip_address

def check_change_ip(address, default_ip_address, debug=False):
    try:
        ip_address = check_ip_proxy(address)
```

```
    except:
        ip_address = None

    if debug:
        print(f'Old ip: {default_ip_address}, new ip : {ip_address}')

    if default_ip_address!=ip_address and ip_address:
        if debug:
            print('IPs are different')
        return True
    return False

def access_denied_check_with_address(address, url):
    options = webdriver.ChromeOptions()
    options.add_argument('--ignore-certificate-errors')
    options.add_argument('headless')
    prox = "socks5://localhost:9050"
    options.add_argument('--proxy-server=%s' % prox)
    driver = webdriver.Chrome(options=options)

    try:
        driver.get(url)
    except:
        driver.quit()
        return False

    html = driver.page_source
    driver.quit()
    try:
        return bs4.BeautifulSoup(html).find('head').title.text!='Access De-
nied'
    except:
        return True

def access_denied_check_with_page(html):
    try:
        return bs4.BeautifulSoup(html).find('head').title.text!='Access De-
nied'
    except:
        return True

def parse_free_proxies():
    ips = []
    url = 'https://free-proxy-list.net/'
    soup = bs4.BeautifulSoup(requests.get(url).text)
    for i in soup.find('table', {'class':'table table-striped table-bor-
dered'}).find_all('tr'):
        found = i.find_all('td')[:2]
        if found:
            ip, port = found
            ips.append(ip.text+':'+port.text)
    return ips

def wait_and_click_by(driver, value, by, time_sleep=15):
    WebDriverWait(driver, time_sleep).until(EC.presence_of_element_lo-
cated((by, value)))
    driver.find_element(by=by, value=value).click()

"""
## chek proxy
"""

from collections import Counter
from stem import Signal
```

```
from stem.control import Controller

default_ip = check_ip_proxy('')


"""
## parsing itself
"""

import os
import queue

ABS_PATH = 'trip_advisor_data_hotels'
if not os.path.exists(ABS_PATH):
    os.mkdir(ABS_PATH)

for i in hotels_df['title']:
    dir_path = os.path.join(ABS_PATH,i)

    if not os.path.exists(dir_path):
        os.mkdir(dir_path)


def parse_reviews(link, path, abs_path, user_agent,
                  parts_scroll=8, sleep_time_list=None, run_headless=True,
                 max_errors=50):


    # exception handling
    passed = {'got_initial_link': False,
              'see_all_languages': False}
    passed['link'] = link
    passed['hotel_name'] = path

    caught_ex = None

    # overall path
    path_to_save = os.path.join(abs_path, path)

    #check if there are already parsed pages
    n_already_parsed = len(os.listdir(path_to_save))
    if n_already_parsed:
        link = augment_link(link, n_already_parsed)


    # get driver
    try:
        driver = get_driver(user_agent, run_headless)
    except Exception as ex:
        caught_ex = ex

    if caught_ex:
        passed['got_initial_link'] = False
        passed['num_overall'] = 9999
        passed['num_parsed'] = 0
        passed['exception'] = caught_ex
        return passed


    # initial link getting
    try:
        driver.get(link)
        time.sleep(5)
    except Exception as ex:
```

```
        caught_ex = ex


    if caught_ex:
        passed['got_initial_link'] = False
        passed['num_overall'] = 9999
        passed['num_parsed'] = 0
        passed['exception'] = caught_ex
        return passed
    else:
        passed['got_initial_link'] = True

    # check if access denied
    if not access_denied_check_with_page(driver.page_source):
        caught_ex = 'Access dnied'

    if caught_ex:
        passed['got_initial_link'] = False
        passed['num_overall'] = 9999
        passed['num_parsed'] = 0
        passed['exception'] = caught_ex
        return passed

    # see all languages
    try:
        wait_and_click_by(driver, 'Qukvo', By.CLASS_NAME, 30)
        passed['see_all_languages'] = True
        time.sleep(5)
    except:
        passed['see_all_languages'] = False


    c = 0
    errors = 0
    first_page = None

    while True:
        passed['show_more'] = False
        passed['saved_file'] = False
        passed['next_page'] = False


        try:
            # show more
            wait_and_click_by(driver, 'Ignyf', By.CLASS_NAME, 30)
            time.sleep(2)
            passed['show_more'] = True
            # if first page, then save it
            if c == 0:
                first_page = driver.page_source

            # save to txt
            save_html(driver.page_source, os.path.join(path_to_save,
f'page_{str(n_already_parsed+c)}'))
            time.sleep(1)
            passed['saved_file'] = True
            c += 1

            # next page
            WebDriverWait(driver, 30).until(EC.presence_of_element_lo-
cated((By.CLASS_NAME, 'ui_button.nav.next')))
            button_el = driver.find_element(by=By.CLASS_NAME, value='ui_but-
ton.nav.next')
            if button_el.is_enabled() and button_el.is_displayed():
```

```
                button_el.click()
            else:
                break
            passed['next_page'] = True
            errors = 0

        except Exception as ex:
            if not isinstance(ex, (StaleElementReferenceException, Ele-
mentClickInterceptedException)):
                caught_ex = ex
                break
            else:
                errors+=1
            if errors>=max_errors:
                break

        finally:
            time.sleep(np.random.choice(sleep_time_list))


    driver.quit()

    if not caught_ex:
        passed = dict([(k, True) for k in passed.keys()])

    try:
        passed['num_overall'] = int(bs4.BeautifulSoup(first_page) \
                                    .find_all('span', {'data-test-target':
'CC_TAB_Reviews_LABEL'})[0] \
                                    .find('span', {'class': 'iypZC Mc _R
b'}).text)
        passed['got_overall_num'] = True
    except:
        passed['got_overall_num'] = False
        passed['num_overall'] = 0

    passed['num_parsed'] = 5 * (n_already_parsed+c)
    passed['exception'] = caught_ex

    return passed

n_threads = 8
headless = True
sleep_time_list = list(range(3,15))

parse_reviews_partial = partial(parse_reviews,
                                run_headless=headless,
                                sleep_time_list=sleep_time_list,
                               abs_path=ABS_PATH)


user_agents = [user_agent_rotator.get_random_user_agent() for i in range(ho-
tels_df.shape[0])]


sub_df = hotels_df[hotels_df['parsed']==False]
input_tuples = list(zip(sub_df['link'].values.tolist(), sub_df['title'].val-
ues.tolist(), user_agents))

batch_size = 100
sleep_between_batches_time = [120, 180, 300, 600]

batched_input_tuples = [input_tuples[i:i+batch_size] for i in range(0,
len(input_tuples)+batch_size, batch_size)]
```

```
def parse_reviews_multiprocessing(input_tuple):
    link, path, user_agent = input_tuple
    passed_dict = parse_reviews_partial(link, path=path,
user_agent=user_agent)
    return passed_dict

for batch in batched_input_tuples:
    with closing(ThreadPool(n_threads)) as p:
        results = list(tqdm(p.imap(parse_reviews_multiprocessing, batch), to-
tal=len(batch)))

    mask_passed = dict([(i['link'], i['num_parsed']/(i['num_overall']+1)>0.8)
for i in results])
    hotels_df.loc[hotels_df['parsed']==False,'parsed'] = hotels_df.loc[ho-
tels_df['parsed']==False,'link']\
    .apply(lambda x: mask_passed.get(x, False))
    time.sleep(np.random.choice(sleep_between_batches_time))

hotels_df['parsed'].value_counts()

hotels_df.to_csv('hotels_links.csv', index=False)
```

```
import bs4
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import requests
from multiprocessing.pool import ThreadPool
from tqdm import tqdm
import time
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import Select
from selenium.common.exceptions import ElementClickInterceptedException,
StaleElementReferenceException
import os
from random_user_agent.user_agent import UserAgent
from random_user_agent.params import SoftwareName, OperatingSystem
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import json
import pickle
import urllib
sns.set()

from multiprocessing.pool import Pool
from contextlib import closing

from functools import partial

def multiprocess_func(main_input, func, additional_inputs=None,
                      gather_func=None, to_split=True, gather_func_args=None,
                      chunk_size=100, n_processes=8):
    if not gather_func_args:
        gather_func_args = []
    if not additional_inputs:
```

```
        additional_inputs = []
    if not gather_func:
        gather_func = lambda x: [z for i in x for z in i]
    if to_split:
        splitted = [(main_input[i:i + chunk_size], *additional_inputs) if ad-
ditional_inputs else main_input[i:i + chunk_size]\
                    for i in range(0, len(main_input), chunk_size)]
    else:
        splitted = [(i, *additional_inputs) if additional_inputs else i for i
in main_input]
    with closing(Pool(n_processes)) as p:
        result = list(tqdm(p.imap(func, splitted),
                           total=len(splitted)))
    return gather_func(result, *gather_func_args)

def process_buble(x):
    return float('.'.join(x))

def bs4_parse_reviews(input_tuple):
    page_to_parse, hotel_name = input_tuple
    records = []
    try:
        for review_page in bs4.BeautifulSoup(page_to_parse).find_all('div',
{'class':'WAllg _T'}):

            record = {}
            record['overall_rating'] = process_buble(re-
view_page.find('div',{'data-test-target':'review-rating'})\
                                                .span['class'][-1].split('_')[-1])
            per_type_bubble = review_page.find_all('div', {'class':'hemdC S2
H2 WWOoy'})

            if per_type_bubble:
                for j in per_type_bubble:
                    record[j.text+'_rating'] = process_bu-
ble(j.span.span['class'][-1].split('_')[-1])


            record['review'] = review_page.find('div',{'class':'fIrGe
_T'}).text
            record['hotel_name'] = hotel_name

            records.append(record)
    except Exception as ex:
        print(ex)

    return records

def read_file(path):
    with open(path, 'r') as f:
        return f.read()

def parse_reviews_multiproc(name, abs_path):
    path = os.path.join(abs_path,name)
    records = []
    for path_page in os.listdir(path):
        page = read_file(os.path.join(path,path_page))
        records.extend(bs4_parse_reviews((page, name)))
    return records

ABS_PATH = 'trip_advisor_data_hotels'


hotels_df = pd.read_csv('hotels_links.csv')
```

```
hotels_to_load = hotels_df[hotels_df['parsed']==True]['title'].val-
ues.tolist()

partial_parse_reviews_multiproc = partial(parse_reviews_multiproc,
abs_path=ABS_PATH)


reviews = multiprocess_func([i for i in os.listdir(ABS_PATH) if not i.starts-
with('.')],
                  func=partial_parse_reviews_multiproc,
                  to_split=False,
                 n_processes=8)

reviews = pd.DataFrame(reviews)

reviews = reviews.drop_duplicates(['review', 'hotel_name'])

reviews.head()

reviews.shape

reviews['overall_rating'].value_counts().plot.bar()

reviews['overall_rating'].value_counts()

reviews.isna().sum()

reviews[reviews['overall_rating']==3.0].sample()['review'].values[0]
reviews.to_csv('hotel_reviews.csv', index=False)


import os
import pandas as pd
import gc
from multiprocessing.pool import Pool
from contextlib import closing
from tqdm import tqdm


def multiprocess_func(main_input, func, additional_inputs=None,
                      gather_func=None, to_split=True, gather_func_args=None,
                      chunk_size=100, n_processes=8):
    if not gather_func_args:
        gather_func_args = []
    if not additional_inputs:
        additional_inputs = []
    if not gather_func:
        gather_func = lambda x: [z for i in x for z in i]
    if to_split:
        splitted = [(main_input[i:i + chunk_size], *additional_inputs) if ad-
ditional_inputs else main_input[i:i + chunk_size]\
                    for i in range(0, len(main_input), chunk_size)]
    else:
        splitted = [(i, *additional_inputs) if additional_inputs else i for i
in main_input]
    with closing(Pool(n_processes)) as p:
        result = list(tqdm(p.imap(func, splitted),
                           total=len(splitted)))
    return gather_func(result, *gather_func_args)


path = '../../../data_reviews/'
```

```
"""
# Merging the data
"""

df1 = pd.read_csv(os.path.join(path, 'rozetka_ukr.csv'), encoding='windows-
1251',
           sep=';')
df1.shape

df1['entity_name'] = df1['prod_link'].apply(lambda x: x.split('/')[-3])

df1 = df1[['comment','translate', 'rating', 'entity_name']]\
.rename(columns={'comment':'review', 'translate':'review_translate'})
df1['dataset_name'] = 'rozetka'

df2 = pd.read_csv(os.path.join(path, 'rozetka_ru.csv'), encoding='windows-
1251',
           sep=';')
df2.shape

df2 = df2[~df2['prod_link'].isna()]

df2['entity_name'] = df2['prod_link'].apply(lambda x: x.split('/')[-3])

df2 = df2[['comment','translate', 'rating', 'entity_name']]\
.rename(columns={'comment':'review', 'translate':'review_translate'})
df2['dataset_name'] = 'rozetka'

df3 = pd.read_csv(os.path.join(path, 'hotels_final.csv'), encoding='windows-
1251',
           sep=';')
df3.shape

df3 = df3.rename(columns={'hotel_name':'entity_name'})

df3 = df3[['review', 'translate', 'overall_rating', 'entity_name']]\
.rename(columns={'overall_rating' : 'rating', 'translate':'review_trans-
late'})
df3['dataset_name'] = 'tripadvisor_hotels_ukraine'

df4 = pd.read_csv(os.path.join(path, 'restaurants_review_final.csv'), encod-
ing='windows-1251',
           sep=';')
df4.shape

df4 = df4.rename(columns={'name':'entity_name'})

df4 = df4.rename(columns={'overall_rating' : 'rating'})[['review', 'ti-
tle_translate', 'review_translate', 'rating',
                                                        'entity_name']]
df4['dataset_name'] = 'tripadvisor_restaurants_ukraine'

df = pd.concat([df1, df2, df3, df4], axis=0)

df.head()

del df1, df2, df3, df4;
gc.collect();

df = df[~df['rating'].isna()]

df['title_translate'] = df['title_translate'].fillna('')

df = df[~df['review'].isna()]
```

```
df['translated'] = df['review']!=df['review_translate']

df.isna().sum()

df['translated'].value_counts()

df.shape

df[df['rating']==2].sample(1)[['review', 'review_translate']].values

df['entity_name'].nunique()

"""
# Basic data analysis
"""

import nltk
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

sns.set()

from nltk.tokenize import sent_tokenize

"""
## Characters number
"""

print('Max number of characters in translated review : {}'.format(df['re-
view_translate'].apply(len).max()))
print('Min number of characters in translated review : {}'.format(df['re-
view_translate'].apply(len).min()))
print('Mean number of characters in translated review : {}'.format(df['re-
view_translate'].apply(len).mean()))
print('Median number of characters in translated review : {}'.format(df['re-
view_translate'].apply(len).median()))


sns.distplot(np.log10(df['review_translate'].apply(len)))

np.percentile(df['review_translate'].apply(len), q=0.2)

"""
### filter out those reviews which char len is an outlier
"""

df = df[df['review_translate'].apply(len)>np.percentile(df['review_trans-
late'].apply(len), q=0.2)]

"""
### find those reviews which have a lot less characters that real text
"""
df['diff_len'] = df['review'].apply(len)-df['review_translate'].apply(len)
df = df[df['review_translate']!='#ERROR!']
df['diff_len'] = df['diff_len'].apply(abs)
sns.distplot(np.log1p(df['diff_len']))
df = df[df['diff_len']<200]
df[df['translated']==True]['diff_len'].max()

df = df.drop(columns=['diff_len'])

"""
```

```
### deleting empty symbols
"""

df['review_translate'] = df['review_translate'].str.strip()
df = df[df['review_translate'].apply(lambda x: True if x else False)]

"""
### remove \n char
"""

df['translated'].value_counts()

import re
def remove_multy_spaces(text):
    try:
        text = re.sub(r'\s+', ' ', text)
        return text
    except Exception as ex:
        return None

df['review_translate'] = df['review_translate'].str.replace('\n',
'').str.strip()

def spacing_between_chars_text(text):
    text = list(text)
    new_text = []
    for idx_char in range(len(text)):
        if not text[idx_char].isalnum() and text[idx_char]!="'" and
text[idx_char]!=' ':
            new_text.append(' ')
            new_text.append(text[idx_char])
            new_text.append(' ')
        else:
            new_text.append(text[idx_char])

    return ''.join(new_text).strip()

df['review_translate'] = multiprocess_func(df['review_translate'].values,
                  func=spacing_between_chars_text,
                  gather_func=lambda x: x,
                  to_split=False)

df['review_translate'] = multiprocess_func(df['review_translate'].values,
                  func=remove_multy_spaces,
                  gather_func=lambda x: x,
                  to_split=False)

df['review_translate'].values[0]

"""
## Sentence number
"""

sent_tokenized = multiprocess_func(df['review_translate'].values,
                  func=sent_tokenize,
                  gather_func=lambda x: x,
                  to_split=False)

sns.distplot(np.log10([len(i) for i in sent_tokenized]))

df['review_translate_sentences'] = sent_tokenized

"""
# Delete those which are partially translated
```

```
"""

import fasttext
from itertools import chain

model = fasttext.load_model('../../../lid.176.bin')

def detect_lang_sentences(batched_texts, model):
    result = []
    for texts in tqdm(batched_texts):
        lengths = [len(i) for i in texts]
        sentences = list(chain(*texts))
        predicted_langs, _ = model.predict(sentences)
        predicted_langs = list(map(lambda x: x[0].split('__')[-1], pre-
dicted_langs))
        assert sum(lengths)==len(sentences)
        assert len(predicted_langs)==len(sentences)
        batched_langs = []
        start = 0
        end = lengths[0]
        for i in lengths[1:]:
            to_add = predicted_langs[start:end]
            if not to_add:
                break
            batched_langs.append(to_add)
            start = end
            end = end+i

        if predicted_langs[start:end]:
                batched_langs.append(predicted_langs[start:end])
        assert [len(i) for i in batched_langs]==lengths
        result.extend(batched_langs)

    return result

def detect_lang(batched_texts, model):
    result = []
    for texts in tqdm(batched_texts):
        predicted_langs, _ = model.predict(list(texts))
        result.extend(list(map(lambda x: x[0].split('__')[-1], pre-
dicted_langs)))

    return result

batch_size=100
to_detect_lang = df.loc[df['translated']==True, 'review_translate_sentenc-
es'].values
batches = [to_detect_lang[i:i+batch_size] for i in range(0, len(to_de-
tect_lang), batch_size)]

sum([len(i) for i in batches])

result = detect_lang_sentences(batches, model)

batch_size=100
to_detect_lang = df.loc[df['translated']==True, 'review_translate'].values
batches = [to_detect_lang[i:i+batch_size] for i in range(0, len(to_de-
tect_lang), batch_size)]

result = detect_lang(batches, model)

df['language_translated'] = 'uk'
df.loc[df['translated']==True, 'language_translated'] = result
```

```
df = df[df['language_translated']=='uk']

df.drop(columns='language_translated', inplace=True)

"""
# Tokenize texts
"""

from nltk.tokenize import regexp_tokenize

def NLTK_special_chars_excluded_tokenizer(input_text):
    overall_pattern = r"[\w'-]+|[^\w\s'-]+"
    return regexp_tokenize(input_text, pattern=overall_pattern, gaps=False,
discard_empty=True)

def tokenize_sentence_tokens(sentences):
    tokens = []
    for sent in sentences:
        tokens.append(NLTK_special_chars_excluded_tokenizer(sent))
    return tokens

df['review_translate_sentences_tokens'] = multiprocess_func(df['review_trans-
late_sentences'].values,
                  func=tokenize_sentence_tokens,
                  gather_func=lambda x: x,
                  to_split=False)

"""
# Add spaces between chars
"""

# %%
from functools import partial

# %%
def apply_func_sent(sentences, func):
    result = []
    for sent in sentences:
        result.append(func(sent))
    return result

# %%
def spacing_between_chars_tokens(tokens):
    tokens = list(np.hstack([spacing_between_chars(i) for i in tokens]))
    return [i for i in tokens if i]

# %%
def spacing_between_chars(text):
    text = list(text)
    new_text = []
    for idx_char in range(len(text)):
        if not text[idx_char].isalnum() and text[idx_char]!="'":
            new_text.append(' ')
            new_text.append(text[idx_char])
            new_text.append(' ')
        else:
            new_text.append(text[idx_char])

    return ''.join(new_text).strip().split(' ')

# %%
spacing_between_chars_sentences = partial(apply_func_sent, func=spacing_be-
tween_chars_tokens)
```

```
# %%
df['review_translate_sentences_tokens'] = multiprocess_func(df['review_trans-
late_sentences_tokens'].values,
                  func=spacing_between_chars_sentences,
                  gather_func=lambda x: x,
                  to_split=False)

# %%
"""
# Find pos tags
"""

# %%
import pymorphy2

# %%
morph = pymorphy2.MorphAnalyzer(lang='uk')

# %%
def pos_tagging(ent):
    batch, morph = ent
    tags_batch = []
    for sentences in batch:
        tags_sentences = []
        for sentence in sentences:
            tags_sentences.append([morph.parse(word)[0].tag._POS for word in
sentence])
        tags_batch.append(tags_sentences)
    return tags_batch

# %%
df['review_translate_sentences_pos'] = multiprocess_func(df['review_trans-
late_sentences_tokens'].values,
                  func=pos_tagging,
                  gather_func=None,
                  to_split=True,
                  chunk_size=100,
                  n_processes=12,
                  additional_inputs=[morph])

# %%
"""
# Find lemmas
"""

# %%
def lemmatizing(ent):
    batch, morph = ent
    tags_batch = []
    for sentences in batch:
        tags_sentences = []
        for sentence in sentences:
            tags_sentences.append([morph.parse(word)[0].normal_form for word
in sentence])
        tags_batch.append(tags_sentences)
    return tags_batch

# %%
df['review_translate_sentences_lemma'] = multiprocess_func(df['review_trans-
late_sentences_tokens'].values,
                  func=lemmatizing,
                  gather_func=None,
                  to_split=True,
                  chunk_size=100,
```

```
                  n_processes=12,
                  additional_inputs=[morph])

# %%
df.head(1)

# %%
"""
# Delete plain questions
"""

# %%
def is_question_sentences(ent):
    sentences, tags = ent
    is_question_vector = []
    for i in range(len(sentences)):
        is_question_vector.append(is_question(sentences[i], tags[i]))
    return is_question_vector

# %%
def is_question(words, tags):
    tags = [tag for word, tag in list(zip(words, tags))\
            if not word in ['.', ',', '!', '?']]

    # Check if the last character of the sentence is a question mark
    if words[-1] == "?" and len(tags)>1:
        # Check if the sentence ends with a verb or an auxiliary verb
        if tags[-1] in ["VERB", "INFN"] or (tags[-1] == "GRND" and tags[-2]
in ["VERB", "INFN"]):
            return True
        # Check if the sentence starts with an auxiliary verb and ends with a
verb
        elif tags[0] == "PRCL" and tags[-1] in ["VERB", "INFN"]:
            return True
        else:
            return False
    elif words[-1]=='?' and len(tags)==1:
        return True
    else:
        return False


# %%
to_input = list(zip(df['review_translate_sentences_tokens'].values.tolist(),
            df['review_translate_sentences_pos'].values.tolist()))

# %%
questions_mask = multiprocess_func(to_input,
                  func=is_question_sentences,
                  gather_func=lambda x: x,
                  to_split=False,
                  n_processes=12,
                  )

# %%
df['is_question'] = questions_mask

# %%
df = df[~df['is_question'].apply(lambda x: all(x))]

# %%
df.to_csv('processed_data.csv', index=False)
```

```
# %%
!pip install tokenizers
import tensorflow as tf
from tokenizers import Tokenizer, models, pre_tokenizers, trainers, Regex
import tokenizers
import pandas as pd

# %%
model_name = 'deep_lstm_attention_w2v_huber'

# %%
"""
# Load data
"""

# %%
df = pd.read_csv('/home/user/files_for_research_Vova/processed_data.csv',\
                 usecols=['review_translate',
                                                          'dataset_name',
                                                          'rating',
                                                         'translated'])

# %%
df.head()

# %%
subsets = pd.read_csv('/home/user/files_for_research_Vova/train_val_test_in-
dices.csv')

# %%
subsets.head()

# %%
subsets = subsets.merge(df[['dataset_name', 'translated']], left_on='index',
right_index=True)

# %%
"""
# Filter data
"""

# %%
bad_indices = pd.read_csv('/home/user/files_for_re-
search_Vova/files_to_check.csv')

# %%
subsets = subsets[~subsets.index.isin(bad_indices['id'].values)]

# %%
df = df[~df.index.isin(bad_indices['id'].values)]

# %%
df, subsets = df.reset_index().drop(columns='index'), subsets.reset_in-
dex().drop(columns='index')

# %%
"""
# Load tokenizer
"""

# %%
```

```
tokenizer = Tokenizer(models.BPE.from_file(vocab='/home/user/files_for_re-
search_Vova/tokenizer_30k.json',
           merges='/home/user/files_for_research_Vova/merges_tokenizer.txt',
                                              end_of_word_suffix='</w>'))
tokenizer.pre_tokenizer = pre_tokenizers.Split(Regex(r"[\w'-]+|[^\w\s'-
]+"),'removed', True)

# %%
"""
# Encode text
"""

# %%
import seaborn as sns
import numpy as np

# %%
sns.set()

# %%
df['review_translate'] = df['review_translate'].str.lower()

# %%
df['encoded'] = tokenizer.encode_batch(df['review_translate'].values)

# %%
df['encoded'] = df['encoded'].apply(lambda x: x.ids)

# %%
sns.distplot(np.log10(df['encoded'].apply(len)))

# %%
np.percentile(df['encoded'].apply(len), 99)

# %%
encoded_tokens = df['encoded'].values

# %%
from itertools import chain

# %%
padded_tokens = tf.keras.preprocessing.sequence\
.pad_sequences(encoded_tokens, maxlen=300, padding="post")


# %%
padded_tokens.shape

# %%
"""
# Get embeddings
"""

# %%
!pip install gensim

# %%
import gensim

# %%
def load_w2vec(path, vocab, embed_dim=300, glove_backup={}):
    vectors = gensim.models.KeyedVectors.load_word2vec_format(path, bi-
nary=True)
    emb_matrix = np.zeros(shape = (len(vocab) + 1, embed_dim))
```

```
    missed = 0
    for word, idx in vocab.items():
        if idx!=0:
            try:
                emb_matrix[idx,:] = vectors[word]
            except KeyError:
                if glove_backup:
                    try:
                        emb_matrix[idx,:] = glove_backup[word]
                    except:
                        missed+=1
                else:
                    missed+=1
    print(f'Missed words : {missed}')
    return emb_matrix, vectors

# %%
emb_matrix, vectors = load_w2vec('/home/user/files_for_research_Vova/embed-
dings_w2v.bin',
                                 tokenizer.get_vocab())

# %%
"""
# Get labels and split data
"""

# %%
mapping = dict([(i,c) for c,i in enumerate(df['rating'].unique())])

# %%
mapping

# %%
y = df['rating'].map(mapping).values

# %%
num_classes = len(set(y))

# %%
train_indices, val_indices, test_indices = subsets[sub-
sets['split']=='train'].index.tolist(),\
subsets[subsets['split']=='val'].index.tolist(),\
subsets[subsets['split']=='test'].index.tolist()


# %%
train_y, val_y, test_y = y[train_indices], y[val_indices], y[test_indices]

# %%
train_x, val_x, test_x = padded_tokens[train_indices], padded_tokens[val_in-
dices],\
padded_tokens[test_indices]

# %%
train_x.shape

# %%
"""
# Create  model
"""

# %%
class Attention(tf.keras.layers.Layer):
    def __init__(self,
```

```
                 units=128, **kwargs):
        super(Attention,self).__init__(**kwargs)
        self.units = units

    def build(self, input_shape):
        self.W1=self.add_weight(name='attention_weights_1', shape=(in-
put_shape[-1], self.units),
                                initializer='glorot_uniform', trainable=True)

        self.W2=self.add_weight(name='attention_weights_2', shape=(1,
self.units),
                                initializer='glorot_uniform', trainable=True)

        super(Attention, self).build(input_shape)

    def call(self, x):
        x = tf.transpose(x, perm=[0, 2, 1])
        attention = tf.nn.softmax(tf.matmul(self.W2, tf.nn.tanh(tf.mat-
mul(self.W1, x))))
        weighted_context = tf.reduce_sum(x * attention, axis=-1)
        return weighted_context, attention

    def get_config(self):
        config = super().get_config().copy()
        config.update({
            'units': self.units
        })
        return config


# %%
tf.keras.backend.clear_session()
np.random.seed(0)
tf.random.set_seed(0)
# define layers
attention = Attention(units=128, name='attention')
input_layer = tf.keras.layers.Input(shape=(300,), name='input')
word_embedding = tf.keras.layers.Embedding(input_dim=tokenizer.get_vo-
cab_size()+1,
                                                    output_dim=300,
                                                    trainable=True,
                                           name='embedding',
                                           mask_zero=True,
                                                    weights=[emb_matrix])
batch_norm = tf.keras.layers.LayerNormalization(axis=-1)
spatial_dropout = tf.keras.layers.SpatialDropout1D(0.3, name='spatial_drop-
out')
lstm1 = tf.keras.layers.LSTM(256, name='lstm1',
                             return_sequences=True)
lstm2 = tf.keras.layers.LSTM(128, name='lstm2',
                             return_sequences=True, return_state=True)
dense1 = tf.keras.layers.Dense(128, activation='relu', name='dense')
dropout = tf.keras.layers.Dropout(0.5, name='dropout')
logits_layer = tf.keras.layers.Dense(num_classes, activation='softmax',
name='output')

#actual flow
embedded = spatial_dropout(word_embedding(input_layer))
lstm_lvl1 = lstm1(embedded)
normed = batch_norm(lstm_lvl1)
context_vector, state_h, _ = lstm2(normed)
weighted_context, attention_scores = attention(context_vector)
final_attn_output = tf.concat([state_h, weighted_context], axis=1)
x = dense1(final_attn_output)
```

```
x = dropout(x)
x = logits_layer(x)
model = tf.keras.Model(input_layer, x)

# %%
"""
# Compile model
"""

# %%
model.compile(loss=tf.keras.losses.Huber(1.0), \
              optimizer=tf.keras.optimizers.Adam(),
             metrics=['acc'])

# %%
"""
# Early stopping
"""

# %%
import operator
class EarlyStopping:
    def __init__(self, tolerance=5, mode='min'):
        assert mode in ['min','max'], 'Mode should be min or max'
        self.mode = operator.lt if mode=='min' else operator.gt
        self.tolerance = tolerance
        self.counter = 0
        self.early_stop = False
        self.extremum_value = None
        self.best_model = None

    @staticmethod
    def copy_model(model):
        copied_model = tf.keras.models.clone_model(model)
        copied_model.set_weights(model.get_weights())
        return copied_model

    def __call__(self, val, model):
        if self.extremum_value is None:
            self.extremum_value = val
            self.best_model = self.copy_model(model)
        else:
            if not self.mode(val, self.extremum_value):
                self.counter+=1
            else:
                self.extremum_value = val
                self.best_model = self.copy_model(model)
                self.counter = 0

        if self.counter==self.tolerance:
            self.early_stop=True

# %%
"""
# Train model
"""

# %%
from sklearn.metrics import f1_score

# %%
def evaluate_on_datasets(y_true, y_pred, split='val'):
    d = {}
    for dataset_name in subsets['dataset_name'].unique():
```

```
            idx = subsets[subsets['split']==split].copy()
            idx['index'] = list(range(idx.shape[0]))
            idx = idx[(idx['dataset_name']==dataset_name)]\
            ['index'].values.tolist()
            score = f1_score(y_true=y_true[idx], y_pred=y_pred[idx],
                             average='macro')
            print(f'{split} f1 score for dataset {dataset_name} : {score}')
            d[f'{split}_f1_{dataset_name}'] = score

    for flag in [True, False]:
        idx = subsets[subsets['split']==split].copy()
        idx['index'] = list(range(idx.shape[0]))
        idx = idx[idx['translated']==flag]['index'].values.tolist()
        score = f1_score(y_true=y_true[idx], y_pred=y_pred[idx],
                         average='macro')
        print(f'{split} f1 score for translated=={flag} : {score}')
        d[f'{split}_f1_translated=={flag}'] = score
    return d

# %%
def update_history(history, d):
    for key, value in d.items():
        res = history.get(key, [])
        res.append(value)
        history[key] = res

# %%
early_stopping = EarlyStopping(mode='max', tolerance=4)

# %%
def training_loop(model, train_x, train_y, val_x, val_y, epochs=10,
batch_size=128,
                  shuffle=True):
    dict_history = {}
    for i in range(epochs):
        if shuffle and i==0:
            indices = np.arange(len(train_x))
            np.random.shuffle(indices)
            train_x = train_x[indices]
            train_y = train_y[indices]

        #train model
        history = model.fit(train_x,tf.one_hot(train_y,num_classes), \
                            validation_data=(val_x,tf.one_hot(val_y,num_clas-
ses)),
          epochs=1, batch_size=batch_size,
                            verbose=0, shuffle=False)
        train_loss, val_loss = history.history['loss'][-1], history.his-
tory['val_loss'][-1]

        #evaluate model
        train_prediction = np.argmax(model.predict(train_x,
batch_size=batch_size), axis=-1)
        val_prediction = np.argmax(model.predict(val_x,
batch_size=batch_size), axis=-1)
        train_f1 = f1_score(y_true=train_y, y_pred=train_prediction,
                            average='macro')
        val_f1 = f1_score(y_true=val_y, y_pred=val_prediction,
                          average='macro')

        #printing evaluation
        print(f'Epoch {i}')
        print(f'Overall train f1 : {train_f1}, overall val f1: {val_f1}')
        print(f'Train loss : {train_loss}, val loss: {val_loss}')
```

```
        d_train = evaluate_on_datasets(y_true=train_y, y_pred=train_predic-
tion, split='train')
        d_val = evaluate_on_datasets(y_true=val_y, y_pred=val_prediction,
split='val')

        if i!=epochs-1:
            print('-'*30)

        #save history
        update_history(dict_history, d_train)
        update_history(dict_history, d_val)
        update_history(dict_history, {'train_f1': train_f1})
        update_history(dict_history, {'val_f1': val_f1})
        update_history(dict_history, {'train_loss': train_loss})
        update_history(dict_history, {'val_loss': val_loss})
        #early stopping

        early_stopping(val_f1, model)
        if early_stopping.early_stop:
            print('Stopping early')
            model = early_stopping.best_model
            break

    return dict_history, model

# %%
dict_history, model = \
training_loop(model, train_x, train_y,
              val_x, val_y, epochs=20, batch_size=2048, shuffle=True)

# %%
dict_history

# %%
"""
# Show charts
"""

# %%
import seaborn as sns
import matplotlib.pyplot as plt

# %%
def plot_history(dict_history, columns):
    plt.figure(figsize=(12,8))
    for i in columns:
        to_plot = dict_history[i]
        plt.plot(range(len(to_plot)), to_plot, 'o-')
    plt.xticks(range(len(to_plot)), range(len(to_plot)))
    plt.xlabel('Epochs')
    plt.legend(columns)

# %%
plot_history(dict_history, ['val_loss', 'train_loss'])

# %%
plot_history(dict_history, ['val_f1', 'train_f1'])

# %%
"""
# Evaluate model
"""

# %%
```

```
test_predictions = np.argmax(model.predict(test_x, batch_size=2048), axis=-1)

# %%
test_f1 = f1_score(y_true=test_y, y_pred=test_predictions,
                           average='macro')
print(f'Overall test f1-score : {test_f1}')

# %%
test_results = evaluate_on_datasets(y_true=test_y, y_pred=test_predic-
tions,split='test')


# %%
"""
# Confusion matrix
"""

# %%
inverse_mapping = dict([(v,k) for k,v in mapping.items()])

# %%
from sklearn.metrics import confusion_matrix

# %%
np.unique(test_y)

# %%
matrix = confusion_matrix(test_y, test_predictions)
matrix_scaled = matrix.astype('float') / matrix.sum(axis=1)[:, np.newaxis]
plt.figure(figsize=(14,10))
sns.heatmap(matrix_scaled, annot=True, cmap=plt.cm.Blues, xticklabels=[in-
verse_mapping[i] for i in np.unique(test_y)],\
            yticklabels=[inverse_mapping[i] for i in np.unique(test_y)])
plt.title('Confusion matrix')
plt.xlabel('Predicted label')
plt.ylabel('True label')

plt.show()

# %%
test_df = df[subsets['split']=='test'].copy()

# %%
test_df['predicted_rating'] = [inverse_mapping[i] for i in test_predictions]

# %%
"""
# Save history results
"""

# %%
history = pd.DataFrame(dict_history)
for k,v in test_results.items():
    history[k] = v

# %%
history['model'] = model_name

# %%
history.to_csv("/home/user/jupyter_notebooks/Ukranian-SA/notebooks/train-
ing/training_results_filtered.csv", mode='a', header=None, index=None)

# %%
"""
```

```
# Save model
"""

# %%
model.save(f'/home/user/files_for_research_Vova/{model_name}.h5')

# %%
```

Appendix F (required)

Plagiarism check protocol

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка системи визначення ключових фраз у відгуках українською мовою за допомогою штучного інтелекту

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ: кафедра Автоматизації та інтелектуальних інформаційних технологій, факультет інтелектуальних інформаційних технологій та автоматизації
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 98.1% Схожість 1.9%

Аналіз звіту подібності (відмітити потрібне):
☑ 1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
☐ 2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
☐ 3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Роман МАСЛІЙ
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Володимир КОВЕНКО
(підпис) (прізвище, ініціали)

Керівник роботи _____ Ілона БОГАЧ
(підпис) (прізвище, ініціали)