Suleyman Demirel University

UDC 004.8

SULTANOVA NAZERKE ZHOLDYBAYEVNA

Open Vocabulary Model for Kazakh Language using Deep Neural Networks

6D070400 – Computing Systems and Software

Thesis for the Degree Doctor of Philosophy (PhD)

Scientific advisors:

Assoc. Prof., PhD, Kozhakhmet Kanat
PhD, Adjunct Professor, Mateus Mendes

Republic of Kazakhstan

Kaskelen, 2021

CONTENTS

Normative References

This thesis uses references to the following standards:

- ”Instructions for the preparation of a dissertation and author’s abstract” Ministry of education and science of the Republic of Kazakhstan, 377-3 zh.
- GOST 7.32-2001. Report on research work. Structure and design rules.
- GOST 7.1-2003. Bibliographic record. Bibliographic description. General requirements and compilation rules.
- GOST 7.32-2017. System of standards of information, librarianship and publishing. Research report. Structure and design rules.

Symbols and abbreviations

BPTT - Backpropagation through time
DL - Deep Learning
GPU - Graphical processing unit
JSON - JavaScript Object Notation
LD - Local density
LSTM - Long-Short-Term memory
LinearSVC - Linear Support Vector Classifier
NLP - Natural Language Processing
NLTK - Natural language toolkit
NN - Neural networks
NNLM - Neural networks language model
OCR - Optical character recognition
POS - Part of speech
RNN - Recurrent neural networks
SVM - Support vector machines
TF-iDF - Term frequency - Inverse document frequency
kNN - k-Nearest Neighbours

INTRODUCTION

Assessment of the current state of the scientific and technological problem being solved. For the past 25 years there has been a demand for software solutions related to text processing, which has repeatedly experienced periods of growth, related to the emergence of personal computers, and with the rapid development of the Internet, and the rapid development of the Internet, and, In this natural language remains the most important way of communication, be the input of the search query on the miniature screen of the mobile phone, hints of the car navigator or business correspondence. Practically in all such applications such or otherwise the language model is used. So, for a convenient input of texts on a mobile phone, it is necessary to use the predicate input system, which practically corresponds to the direct application of the language model; language model - an indefinite part of the system of speech recognition, including volume and vocal search; Linguistic models are used in machine translation systems, the quality of which at the moment is still far from ideal, but still grows steadily.

Justification of the need for research work. Natural language processing helps computers communicate with people in their native language and scale other language tasks. For example, NLP allows computers to read text, hear speech, interpret it, measure mood, and determine which parts are important.

Modern machines can analyze more language data than humans, without fatigue and in a consistent, unbiased manner. Given the vast amount of unstructured data that is generated every day, from medical records to social media, automation will be critical to efficiently analyzing text and speech data.

While supervised and unsupervised learning, and especially deep learning, are now widely used to model human language, there is also a need for syntactic and semantic understanding and domain expertise that are not necessarily present in these machine learning approaches. NLP is important because it helps disambiguate the language and adds a useful number structure to the data for many downstream applications such as speech recognition or text analysis.

Information about the planned scientific and technical level of development, patent research and conclusions from them are determined by the completeness of the study of the process of developing models and methods for analyzing the effectiveness of open vocabulary language models and comparing them with existing models. The scientific and technical level of the dissertation work will be ensured by the novelty and adequacy of the results obtained, their practical significance and promising use. As a result of the research, models and methods for Kazakh text modelling will be developed.

Information about the metrological support of the thesis. When writing the work, legislative and regulatory documents, official publications on natural language processing, materials of scientific conferences, as well as state and corporate statistical and analytical materials and documents were used.

Relevance of the topic of the dissertation research. The relevance of the considered topic is based on the need to develop an innovative model for Kazakh language to be used in natural language processing tasks that is crucial for advancing digitization of Kazakhstan.

The scientific novelty of the research topic is determined by the fact that the innovative language model has been built. The substantive novelty differs from the previous models based on the application of the neural networks using the graphical processing unit that makes the computation more efficient.

The purpose of the dissertation.

The purpose of the study is the development of models and methods for the analysis of the effectiveness of language modelling for Kazakh language using deep neural networks technologies.

Object of research language models with the application of recurrent neural networks.

Subject of research is the models and methods for creating and improving the efficiency of neural networks for building the character-based model for Kazakh language.

Research tasks, their place in the implementation of research work in general. To achieve the planned results of work, the following tasks have been identified:

- study and analyze the current state of the art of language models for different languages
- to develop a functional diagram and architecture of recurrent neural model
- to develop methods and algorithms for character-based language modeling using recurrent neural networks
- analyze and justify the choice of optimization models for the text generation models
- compare the performance of the developed model with the state of the art

Methodological base of research. The dissertation research used general scientific methods of cognition (analysis, synthesis, etc.), principles of consistency and complexity, comparative analysis and mathematical modeling, methods for analyzing natural language processes, and manually comparing performance of the proposed model.

Provisions for Defense. The following provisions are submitted to the defense:

- methods and algorithms for language modeling and text generation;
- methods and algorithms for Kazakh language model;
- novel network architecture to generate Kazakh text;
- results of experiments and discussion

The structure and scope of the thesis. The dissertation work consists of normative references, list of symbols and abbreviations, an introduction, 5 chapters, a conclusion, a list of references. It is presented on 82 pages of typewritten text, contains 21 figures, 11 tables, a list of used sources of 108 titles.

The main scientific results of the dissertation research, set out in the dissertation, are presented at the international scientific conference in Nigeria, and the results are published in IEEE Xplore proceedings:

Kazakh Language Open Vocabulary Language Model with Deep Neural Networks. 2019 15th International Conference on Electronics, Computer and Computation (ICECCO), Abuja, Nigeria, 2019, pp. 1-4, doi:10.1109/ICECCO48375.2019.9043253.

Within the framework of this dissertation work, 11 research papers on the topic under consideration were prepared and published, of which:

- three article each in a foreign publication and in an international peer-reviewed scientific journals; [1-3]
- four articles were published in publishing houses that meet the requirements of the highest certification commission of the Ministry of Education and Science of the Republic of Kazakhstan; [4-8]
- four articles have been published in the proceedings of international conferences. [9-11]

Acknowledgements. The author thanks her family for continuous support and approval at each step. The author expresses special gratitude to her scientific supervisor PhD Kanat Kozhakhmet for his guidance, the opportunity for professional development, and support during the research. The author expresses sincere gratitude to her external supervisor Adjunct Professor Mateus Mendes for his comprehensive help and attention in the process of writing a dissertation.

1. PRELIMINARIES

Building a language model is a crucial task in computational linguistics. Especially agglutinative languages require attention since the words are formed by attaching a sequence of different morphemes, where each morpheme changes the meaning of the word respectively. For example, with the root оқу (to read), many different words can be formed such as оқулық (a textbook), оқушы (a pupil), and оқытушы (a teacher).

Over the last few years, the usage of Deep Neural Networks in NLP is growing significantly. It overcomes the manual tagging and analysis [12]. Moreover, Deep Learning has high performance in supervised tasks of NLP such as Sentence Classification and Sentiment Analysis [13]. Premjith, Soman, and Kumar [14] were successful in the application of Deep Neural Networks for building the open vocabulary language model for the Malayalam language.

Data sparsity is a major problem in low-resource languages such as Kazakh. There is no Word-Net or other tools that have labeled data. Therefore, before working with complex tasks such as Machine Translation, it is important to learn the model of the language. This work is devoted to build the open vocabulary language model for the Kazakh Language with the use of Deep Neural Networks. Open Vocabulary Language Model in this research, is the generative model that produces all possible correct words within the context given. A word can be treated as a morpheme generated by characters where any possible word type could be generated.

Chahuneau, Smith, and Dyer [15] detailed priors for morphologically dense languages that apply Bayesian language models. The morphological guesser has been developed with the application of finite-state transducers. The stem lexicon was eradicated thoroughly to build a lexicon-free analyzer for Russian that has been preferred for morphological disambiguation. In 2011, Kang, Ng, and Nguen [16] produced the word-character hybrid-input neural network language model(NNLM) for the Mandarin language. Within the specified work the n-gram model with NNLM ended up being in contrast and demonstrated that NNLM has strengths over the n-gram model. The project was completed by merging two independent neural networks: word input NNLM and character input NNLM as a hybrid-input model. On top of that, the linear interpolation of two neural networks may be convenient, yet the hybrid-input model was computationally faster. The outcomes indicate that the error percentage appeared to be diminished around 6.3% as opposed to the n-gram language model.. Mikolov et. al. [17] presented the subword model of a language where all words are splitted into smaller parts such as syllables. Authors proposed recurrent neural network model using subword input model rather that hybrid input language model. The model in this work learns the sequence of words from the set of data. The time complexity was decreased by reducing the number of nodes in neural network compared to hybrid-input neural model. The works of [16] and [17] were investigated by Miyamoto and Cho [18], and reach the optimal method of using the number of entries at the character level and surpass the

model of word input model. Long-short term memory was used as a recurrent neural network language model together with the adaptive gate for detecting absolute blend of the character-based and word-based inputs. Compositional Character to Word (C2W) model has been proposed by Ling et. al. [19] for constructing word analysis without explicit word-lookup table. It is remarkably meaningful for agglutinative languages including Kazakh. The data contains 20K words in five different languages. The test performance accomplish 97% accuracy. Advantages of character-level and word-level neural network models were comprised in hierarchical RNN based language model by Hwang and Sung [9]. Hierarchical neural network consists of two levels: low and high levels. Character-based input is fed into low level, and the output of this network is fed into higher level neural network where the actual word predictions are produced. The outputs are fed into low level model as backpropagation. Word-level recurrent neural network and hierarchical neural network have been compared and analyzed. It was deduced that word based RNN's produce words with less errors, while hierarchical neural model has fewer features and the parallel computation using GPU increases the time efficiency.

Morphological analyzer for Kazakh Language was implemented by Kessikbayeva and Cicekli [21] using rule-based methods. The broad explanation of the rules in Kazakh langauges and application is given current work. The rule-based analyzer was implemented using finite state transducers. Since the language is agglutinative, all morpholocal rules has been written in lexicon files. The efficiency of the work is about 96%. Moreover, morphological analyzers have been developed for different languages including agglutinative languages. Research papers related to morphological analysis of agglutinative languages up to the year 2006 were reviewed by Yuret and Ture [22]. Authors presented new rule-based method for morphologcal disambiguation named Decision Lists. The work itself is composition of three ideas: (1) employing statistical and rule-based techniques, (2) sparseness problem was solved by considering each feature separately, (3) analysing the previous tags. The correct results were picked manually and the error rate was 4%. Statistical language model has been in unsupervised morphological disambiguation tool that was proposed by Yatbaz and Yuret [23]. Authors suggest that it will be more efficient to calculate the probability of the word context, but not the probability of the word. The corpus consists if 1M semi-labelled words, and conceivable substitution was used to calculate the probability. The work has been conducted for Turkish language and the accuracy was 64.5%. Authors suggest that this model can be applied for any language.

2. LANGUAGE MODELS

The task of language modeling is to determine the probable distribution of words over the chains of words in some language. In this chapter, different types of language models will be discussed.

Seq2Seq models are the most commonly used architecture in machine translation and neural network question-answer systems. The largest amount of memory in such models is spent on storing a representation matrix containing a representation of each word from the dictionary. For the word-by-word generation of a consistent answer, it is required to have, along with the standard form of the word, all its word forms. In some languages, words have only a small number of word forms (for example, singular and plural). Nevertheless, in languages such as Russian, many words have a large number of word forms, obtained by changing the gender, number, case and time. Models with dictionaries that sufficiently cover the set of all word forms exceed reasonable limits both in time and memory. Many papers have moved to character-by-character models to work around this problem. In such models, the size of the dictionary matches the size of the alphabet used. Symbol-by-character generation allows you to avoid storing word forms, however, due to the increase in the length of the sequence several times, the model quickly forgets the beginning of the sentence. An alternative solution is to store only the standard form of words in the dictionary. Such a model will generate inconsistent text and cannot be used on a production system. The generator sequentially processes the words from the normalized question and generates the normalized answer.

2.1 Statistical Approach

Informally speaking, the goal of statistical modeling of a language is to distinguish between possible (probable) or impossible (unlikely) chains of words in a given language. This task naturally arises in such practical areas as speech recognition, optical character recognition (OCR), handwriting recognition [24], machine translation [25], spell checking, predictive input, and others. In the latter case, the task appears in its pure form, i.e. the next word is required to be predicted, given the already known left context.

We consider the last statement formally. Let it be required to evaluate the probability of a sequence of words w_1^t in a language L.

$$P(w_1, ..., w_t) = P(w_1, ..., w_{t-1})P(w_t|w_1, ..., w_{t-1}) = \prod_{i=1}^{t} P(w_i|w_1, ..., w_{i-1}) \tag{2.1}$$

Using this model in its pure form, obviously, would require an estimate of the probabilities $P(w_i|w_1,...,w_{i-1})$ for all admissible word sequences as parameters, which is not feasible in practice. Therefore, a certain equivalence class *Cl* is introduced on sequences: i.e. all sequences falling into the class *Cl* appear to be equivalent in this statistical model [6].

$$P(w_t|w_1,...,w_{t-1}) = P(w_i|Cl(w_1,...,w_{i-1})) \tag{2.2}$$

The choice as the equivalence class of the coincidence of the last n − 1 words of the sequence results in the widely known n−gram models:

$$P(w_t|w_1,...,w_{t-1}) = P(w_i|w_{t-n+1},...,w_{i-1}) \tag{2.3}$$

The likelihood assessment using the maximum likelihood method leads to the following obvious formula:

$$P(w_t|w_1,...,w_{t-1}) = \frac{C(w_1,...,w_t)}{C(w_1,...,w_{t-1})} \tag{2.4}$$

where $C(w_1,...,w_t)$is the number of occurrences of the sequence $Cw_1,...,w_t$ in the training set [15]. For $n = 1$ (the unigram model), the probabilities $p(w_t)$ correspond to the frequencies of the words $w \in V$in the corpus. Thus, for the n−gram model, it is necessary to estimate $|V|^n$ parameters, where |V | — size of the dictionary, i.e. the number of different word forms in the training set. So, for a dictionary with a volume of $|V| = 20,000$ word forms in the framework of the bigram model, we would have to evaluate $|V|^2 = 4 \cdot 108$ parameters. Therefore, on a sufficiently large case with a volume of 10 million word usage, no more than 2.5% of the estimated model volume can be estimated. The remaining bigrams will get zero probability. Obviously, as the length of the n−gram increases, the situation will become more complicated. On the other hand, longer n−grams provide a better predictive model [27]. This observation is a special case of the curse of dimensionality, a problem widely known in machine learning [28].

This problem directly affects the results of the speech recognition system. Obviously, assigning zero probability to the true sequence of words W_0 automatically leads to the wrong result, regardless of how clear the pronunciation was. Thus, the task of statistical modeling of a language is to assess the probability of sequences of words in a given language, and no sequence should receive zero probability.

2.2 Neural Language Modeling

In the previous section, it was concluded that the most promising methods of language modeling today are methods based on recurrent neural networks. In this section, these methods will be discussed in detail.

2.2.1. Recurrent Neural Networks

In this section, we will consider the classical architecture of a recurrent neural network, regardless of its application for language modeling.

First of all, it is worth noting that the term "recurrent neural network" hereinafter refers to the recurrent architecture proposed by Elman in 1990 [29]. Strictly speaking, this architecture is not the only one. Its more precise name is the Elman network, the opposite of the earlier Jordan network [30] and the Hopfield network [31]. Below, however, for convenience, a recurrent neural network will be understood solely as Elman's architecture.

Elman's recurrent network is a two-layer neural network in which the hidden layer obtained in step h_t is input to the network in the next *t+1* step. Refer to Figure 2.1

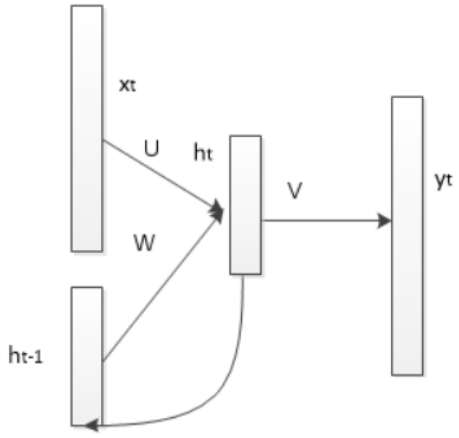

Figure 2.1: General view of Elman's recurrence network

Consider a sample *D* consisting of pairs *(x(t), y(t)),* dependent time series of the same length. Let *(x(t))* and *(y(t))* be defined respectively on the sets *U* and *T*. Then a recurrent neural network is a function approximating the conditional distribution *P(y(t)|x(t))* according to the formulas:

$$h_t = f(W \cdot h_{t-1} + U \cdot x_t + b) \tag{2.5}$$

and

$$y_t = g(V \cdot h_t + d) \tag{2.6}$$

where *W, U, V* - weight matrices, *b, d* - displacements, $x \in U$ - element of the predictor sequence at step *t*, $y \in T$ - probability distribution of elements of an unknown sequence at the same step *t*, $h \in H$ - hidden network layer, and *f* and *g* are activation functions.

The sigmoid function or rectifier linear unit (ReLU) is often taken as *f* [32]. As *g*, a multiclass logit function (softmax) is linear activation.

It should be noted that although formally the sequences *(x(t)), (y(t))* have the same length, at the training stage it is enough to calculate the loss function only at the points of interest to us, therefore *(x(t))* and *(y(t))* can actually have different lengths. This becomes convenient, for example, in the task of assessing the emotional tonality of a text, when an error can be calculated only after reading the entire text [33].

2.2.2 Training recurrent neural networks with back propagation algorithm

Currently, the main optimization method used to select the parameters of the θ model of the neural network is the gradient descent method. In the case of selection of the parameters of the neural network, gradient descent leads to the well-known algorithm of back propagation of error [28].

A similar approach for recurrent neural networks leads to the so-called Backpropagation through time (BPTT) [32].

Consider the output of a recurrent neural network at some step t:

$$\begin{gathered} y_t = g(V \cdot h_t + d) = g(V \cdot f(U \cdot x_t + W \cdot h_{t-1} + b) + d) = \\ g(V \cdot f(U \cdot x_t + W \cdot f(U \cdot x_{t-1} + W \cdot h_{t-2} + b) + b) + d) = \\ g(V \cdot f(U \cdot x_t + W \cdot f(U \cdot x_{t-1} + W \cdot f(...f(U \cdot x_1 + W \cdot h_0)...) + b) + b) + d) \end{gathered} \tag{2.7}$$

We define the error function $\mathcal{L}$ as the sum of errors at each step of a pair of sequences:

$$\mathcal{L}(\theta) = \sum_{1 \le t \le T} \mathcal{L}_t(\theta) \tag{2.8}$$

Obtain expressions for gradients $\frac{\partial \mathcal{L}}{\partial \theta}$:

$$\frac{\partial \mathcal{L}}{\partial V} = \sum_{1 \le t \le T} \frac{\partial \mathcal{L}_t}{\partial y_t} \frac{\partial g(V \cdot h_t + d)}{\partial V} \tag{2.9}$$

$$\frac{\partial \mathcal{L}}{\partial d} = \sum_{1 \le t \le T} \frac{\partial \mathcal{L}_t}{\partial y_t} \frac{\partial g(V \cdot h_t + d)}{\partial d} \tag{2.10}$$

It is seen from Equation (2.7) that the expression for $\frac{\partial \mathcal{L}}{\partial W}$ can be obtained in explicit form if (2.7) is completely written from 1 to *t*. If we return to representation Figure 2.1 in the form of a neural network, then we get the graph shown in Figure 2.2. In fact, we presented the calculation of the output yt as a result of the full cycle of a multilayer neural network with *t−1* layers and the same synapse matrix W. Thus, to calculate $\frac{\partial \mathcal{L}}{\partial W}$, we will use the back propagation algorithm of the error on the neural network obtained by time-sweeping the original recurrent network.

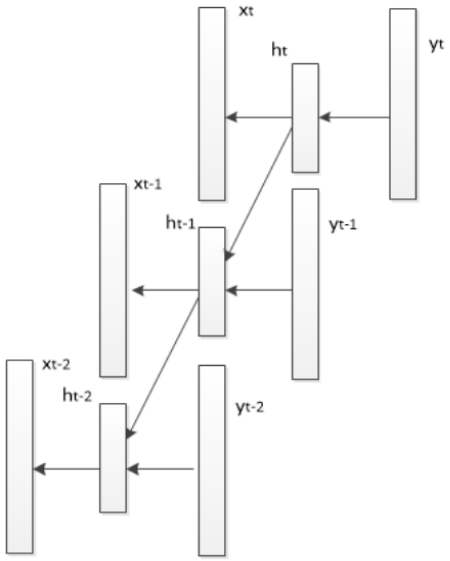

Figure 2.2: The calculation scheme in the error propagation algorithm back in time. The network is shown for 2 steps.

To move on to more formal considerations, for convenience, we introduce the concept of instant derivative.

Definition 1 (Instant derivative). Let h_t be calculated recursively according to the equation $h_t = f(h_{t-1}, \theta)$. The instantaneous partial derivative of θ is the derivative of h_t with respect to θ if it is assumed that $\frac{\partial h_{t-1}}{\partial \theta} = 0$.

$$\frac{\partial \mathcal{L}}{\partial W} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}_t}{\partial h_t} \frac{\partial h_t}{\partial W} =$$

$$= \frac{\partial \mathcal{L}_T}{\partial h_T}\frac{\partial^+ h_T}{\partial W} + \frac{\partial \mathcal{L}_T}{\partial h_T}\frac{\partial h_T}{\partial h_{T-1}}\frac{\partial^+ h_{T-1}}{\partial W} + \sum_{t=1}^{T-2}\frac{\partial \mathcal{L}_T}{\partial h_T}\frac{\partial h_T}{\partial h_t}\frac{\partial^+ h_t}{\partial W} +$$
$$+ \frac{\partial \mathcal{L}_{T-1}}{\partial h_{T-1}}\frac{\partial^+ h_{T-1}}{\partial W} + \sum_{t=1}^{T-2}\frac{\partial \mathcal{L}_{T-1}}{\partial h_{T-1}}\frac{\partial h_{T-1}}{\partial h_t}\frac{\partial^+ h_t}{\partial W} + ... + \frac{\partial \mathcal{L}_t}{\partial h_t}\frac{\partial h_1}{\partial W} =$$
$$= \sum_{t=1}^{T}\frac{\partial \mathcal{L}_t}{\partial h_t}\sum_{k=1}^{t}\frac{\partial h_t}{\partial h_k}\frac{\partial^+ h_k}{\partial W} \tag{2.11}$$

We also note that it follows from the definition of the instantaneous derivative and Equation (2.11) that

$$\frac{\partial^+ h_{T-1}}{\partial W} = \frac{\partial^+ h_T}{\partial W}\frac{\partial h_T}{\partial h_{T-1}} + \frac{\partial \mathcal{L}_{T-1}}{\partial h_{T-1}} \tag{2.12}$$

Using the formulas above, we can propose the following effective learning algorithm for a recurrent neural network [34]. See Table 2.1 for the detailed description of the algorithm.

This algorithm has complexity Θ(T) along the length of the input sequence [34]. Below are formulas for calculating gradients at each step of the network sweep.

$$\mathbf{e}_y = \frac{\partial \mathcal{L}}{\partial y_t} g'(V \cdot h_t) \tag{2.13}$$

$$\frac{\partial \mathcal{L}}{\partial V} = \mathbf{e}_y \cdot h_t^T \tag{2.14}$$

$$\mathbf{e}_h[t] = (V^T \cdot \mathbf{e}_y + W^T \cdot \mathbf{e}_h[t+1]) \cdot diag(f'(h_{t-1})) \tag{2.15}$$

$$\frac{\partial \mathcal{L}}{\partial W} = \mathbf{e}_h[t] \cdot h_{t-1}^T \tag{2.16}$$

$$\frac{\partial \mathcal{L}}{\partial U} = \mathbf{e}_h[t] \cdot \mathbf{u}_t^T \tag{2.17}$$

As shown in Table 2.1, the gradients are summed up at each step.

Table 2.1 Backpropagation Algorithm

Back propogation Algorithm

InitRandomMatrices() - function of random initialization of model matrices
InitZeroMatrices() - function to initialize matrices with zero elements
InitialState() –function of initialization of the hidden state (0 or 0.1)
Input: **{u}, {t}** // sequences of **u** inputs and labels **t**

1. $\mathbf{U}$, $\mathbf{W}$, $\mathbf{V} \leftarrow$ *InitRandomMatrices()*
2. $\mathbf{gW}$, $\mathbf{gV}$, $\mathbf{gU} \leftarrow$ *InitZeroMatrices()*
3. $h_0 \leftarrow$ *InitialState()*
4. **repeat:**
5. **for** $t = 1, ..., N$
6. $h_t \leftarrow f(W \cdot h_{t-1} + U \cdot u_t)$ –feed forward
7. $h_t \leftarrow g(V \cdot h_t)$
8. $\mathbf{gV} \leftarrow \partial\mathcal{L}(y_T, t_T)/\partial\mathbf{V}$
9. $g_h \leftarrow \partial\mathcal{L}(y_T, t_T)/\partial h_T$
10. $\mathbf{gU} \leftarrow g_h \cdot (\partial^+ h_t/\partial\mathbf{U})$
11. $\mathbf{gW} \leftarrow g_h \cdot (\partial^+ h_t/\partial\mathbf{W})$
12. **for** $T-1, T-2, ..., 1$
13. $g_h \leftarrow g_h \cdot (\partial h_{t+1}/\partial h_t + \partial\mathcal{L}(y_t, t_t)/\partial h_t$
14. $\mathbf{gV} \leftarrow \mathbf{gV} + \partial\mathcal{L}(y_t, t_t)/\partial\mathbf{V}$
15. $\mathbf{gU} \leftarrow \mathbf{gU} + g_h \cdot (\partial^+ h_t/\partial\mathbf{U})$
16. $\mathbf{gW} \leftarrow \mathbf{gW} + g_h \cdot (\partial^+ h_t/\partial\mathbf{W})$
17. $U \leftarrow U + \alpha\mathbf{gU}$
18. $V \leftarrow V + \alpha\mathbf{gV}$
19. $W \leftarrow W + \alpha\mathbf{gW}$
20. repeat until convergence

3. CHARACTER-BASED LANGUAGE MODEL

3.1 Introduction

Implementing natural language jobs is extremely important for agglutinative languages. The instances of agglutinative languages are Kazakh, Turkish, and Finnish. The words in these languages are actually built by linking the collection of morphemes. Every single morpheme is capable of turning the word's meaning and also the part of speech. Throughout the last several years, the effective use of Deep Neural Networks in NLP is expanding drastically. It triumphs over the manual labeling and analysis [12]. Additionally, Deep Learning is powerful in supervised tasks related with NLP which includes Sentence Classification and Sentiment Analysis [13]. Premjith, Soman, and Kumar [14] were prosperous with the employing Deep Neural Networks for constructing the open vocabulary language model designed for the Malayalam language. There are plenty of low-resource languages similar to Kazakh. For that reason it is imperative to develop the open vocabulary language model for Kazakh language applying deep neural networks.

The objective of the project is to develop the character-based generative language model for the Kazakh Language. Language model in this research is a sequence to sequence generation task, where an input is a set of the words, and output is the batch of the words which are constructed using characters. A word can be remedied as a morpheme produced by characters in which any attainable word option may be produced. The target is to deliver unseen words which might be easily fit into the particular context. Unseen word is the word that have not been appeared within the train, nevertheless, if the root is identical but the suffix differs, we treat it exactly as unseen word.

3.2 Literature Review

Considerable volume of work continues to be performed related to language modeling for a variety of kinds of languages. At the moment there are two common methodologies for language models: statistical and neural network based.

3.2.1 Statistical approach

Chahuneau, Smith, and Dyer [15] detailed priors for morphologically dense languages that apply Bayesian language models. The morphological guesser has been developed with the application of finite-state transducers. The stem lexicon was eradicated thoroughly to build a lexicon-free analyzer for Russian that has been preferred for morphological disambiguation. In 2011, Kang, Ng, and Nguen [5] produced the word-character hybrid-input neural network language model(NNLM) for the Mandarin language. Within the specified work the n-gram model with NNLM ended up being in contrast and demonstrated that NNLM has strengths over the n-gram model. The project was completed by merging two independent neural networks: word input

NNLM and character input NNLM as a hybrid-input model. On top of that, the linear interpolation of two neural networks may be convenient, yet the hybrid-input model was computationally faster. The outcomes indicate that the error percentage appeared to be diminished around 6.3% as opposed to n-gram language model.

3.2.2 Neural Network approach:

Mikolov et. al. [17] presented the subword model of a language where all words are splitted into smaller parts such as syllables. Authors proposed recurrent neural network model using subword input model rather that hybrid input language model. The model in this work learns the sequence of words from the set of data. The time complexity was decreased by reducing the number of nodes in neural network compared to hybrid-input neural model. The works of [16] and [17] were investigated by Miyamoto and Cho [18], and reach the optimal method of using the number of entries at the character level and surpass the model of word input model. Long-short term memory was used as a recurrent neural network language model together with the adaptive gate for detecting absolute blend of the character-based and word-based inputs. Compositional Character to Word (C2W) model has been proposed by Ling et. al. [19] for constructing word analysis without explicit word-lookup table. It is remarkably meaningful for agglutinative languages including Kazakh. The data contains 20K words in five different languages. The test performance accomplished 97% accuracy. Advantages of character-level and word-level neural network models were comprised in hierarchical RNN based language model by Hwang and Sung [9]. Hierarchical neural network consists of two levels: low and high levels. Character-based input is fed into low level, and the output of this network is fed into higher level neural network where the actual word predictions are produced. The outputs are fed into low level model as backpropagation. Word level recurrent neural network and hierarchical neural network have been compared and analyzed. It was deduced that word based RNN's produce words with less errors, while hierarchical neural model has fewer features and the parallel computation using GPU increases the time efficiency.

Moreover, morphological analyzers have been developed for different languages including agglutinative languages. In this section, we describe the background of the conducted research.

Research papers related to morphological analysis of agglutinative languages up to the year 2006 were reviewed by Yuret and Ture [30]. Authors presented new rule-based method for morphological disambiguation named Decision Lists. The work itself is composition of three ideas: (1) employing statistical and rule-based techniques, (2) sparseness problem was solved by considering each feature separately, (3) analysing the previous tags. The correct results were picked manually and the error rate was 4%.

Morphological analyzer for Kazakh Language was implemented by Kessikbayeva and Cicekli [21] using rule-based methods. The broad explanation of the rules in Kazakh languages and application is given current work. The rule-based analyzer was implemented using finite state transducers. Since the language is agglutinative, all morphological rules have been written in lexicon files. The efficiency of the work is about 96%. Statistical language model has been in an unsupervised morphological disambiguation tool that was proposed by Yatbaz and Yuret [23]. Authors suggest that it will be more efficient to calculate the probability of the word context, but not the probability of the word. The corpus consists if 1M semi-labelled words, and conceivable substitution was used to calculate the probability. The work has been conducted for Turkish language and the accuracy was 64.5%. Authors suggest that this model can be applied for any language.

3.3 Methods and System Architecture

3.3.1 Data Description

Each language including Kazakh has its unique structure of word construction. For text generation purposes and detection of new words, the model needs to understand the word structure and syllable and character sequences. To be successful in this task, the book “Abay Zholy” by Mukhtar Auezov that was originally written in Kazakh was used for training purposes. The dataset contains about 800,000 characters and 110,000 words with approximately 29,000 of them being unique. For training the neural network, the data was tokenized, and all words were transformed into character-based one-hot encoding with the sequence size 256. Stemming methods were not applied to keep the structure of words unchanged.

3.3.2 System Architecture

Deep Neural Networks were applied for constructing generative language model for Kazakh. The language modeling is recognized sequence prediction task. Keeping the long-range dependencies is essential since the aim of the work is to generate valid words according to the context given. Taking that into account, recurrent neural network named long-short term memory was adopted.

LSTMs were presented in [35], subsequently improved and popularized by other researchers, they cope well with many tasks and are still widely used. LSTMs are specifically designed to address long-term dependencies problems. Their specialization is the storage of information for long periods of time, so they practically do not need to be trained. All recurrent neural networks are in the form of a chain of repeating modules of a neural network. In standard RNCs, this repeating module has a simple structure, for example, one tanh layer.

For these tasks, LSTM models were used to generate words because of their ability to remember the previous state. Two different sets of methods were used to compare their accuracies.

Model 1 is presented in Figure 3.1. It consists of one embedding layer, 1 LSTM layer, and dense layer. The model 2 which described in Figure 3.2 consists of two layered LSTM's. Each LSTM layer is followed by Dropout to reduce the overfitting problems. Different parameters were used to train the model to discover which of them will perform better.

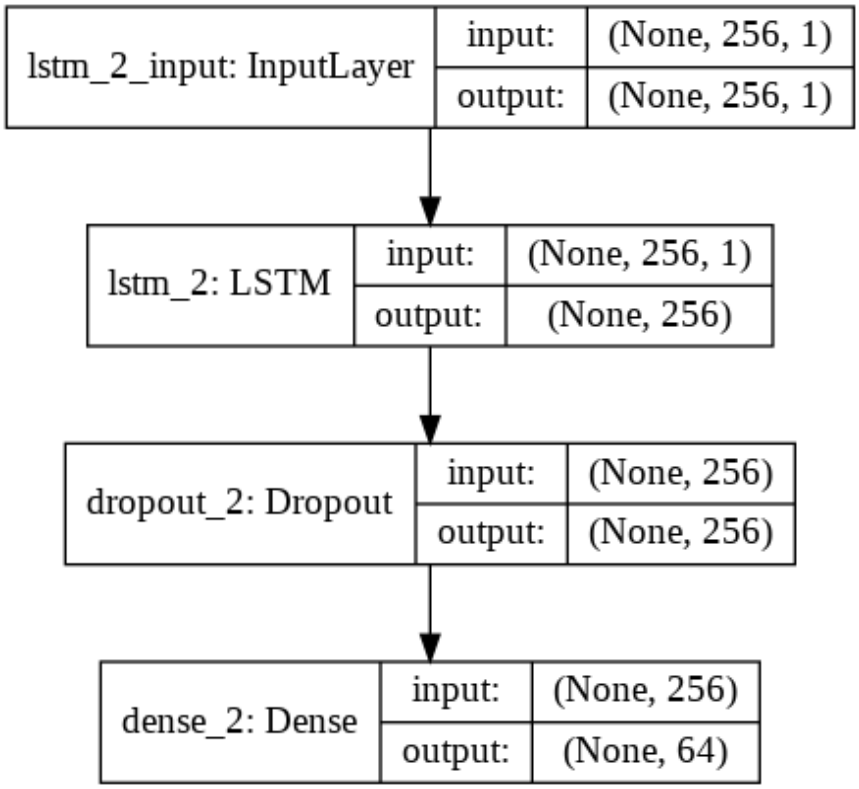

Figure 3.1: Neural network with one layered LSTM

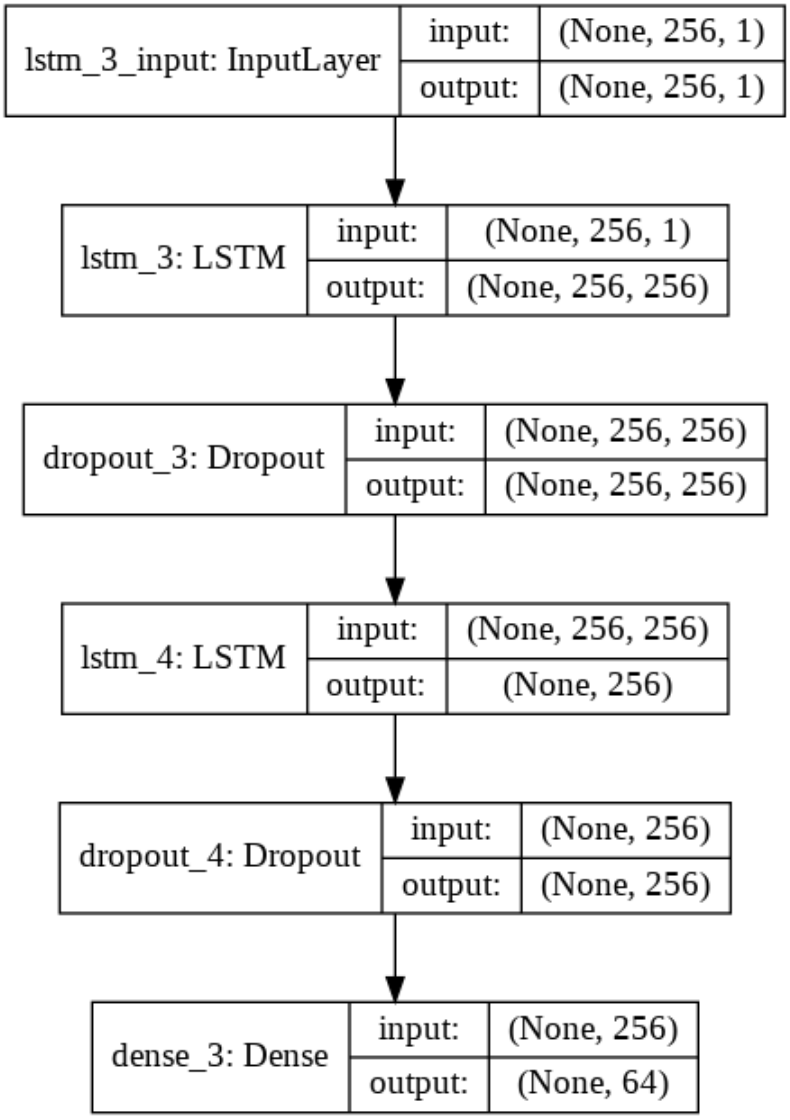

Figure 3.2: Neural Network with multiple layered LSTM

3.4 Evaluation and Test Results

Evaluation of the model is performed by counting the number of correctly generated words over the whole set of words that has been generated. The objective of the work is to generate valid words, however to see that model learns to produce the correct words with different endings that have not been seen during the training.

The parameters of the first model that was shown in Figure 3.1 are:

- batch size - 128
- number of epochs - 20
- dropout - 0.2

On the one hand it turned out to be some kind of nonsense. On the other hand, we see that the understanding of words as a set of characters separated by spaces and even the use of some punctuation marks begins to form on the neural network.

Since the number of epochs is 20 and the temperature was low - the produced results were not complex and contain 1-2 syllables.

The results of the first model showed that about 20% of generated words were valid. And approximately 5% of these valid words were unseen. The testing was done manually.

For the second model that is shown in Figure 3.2, the parameters were:

- batch size - 512
- number of epochs - 50
- dropout - 0.2

Here we see that the words are mainly composed of letters. Dialogs are marked, punctuation marks are well placed, etc. If you look from a far and do not read the text, it looks decent. The words that have been generated have more complex structures with 3-4 syllables. The incorrectly generated words contain fewer errors, and mostly misspell one or two characters.

The results of the second model showed that about 60% of generated words were valid. And approximately 2-3% of these valid words were unseen. The testing was done manually. The results can be improved by adding more informal text - for example, news or forums since the "Abay Zholy" is a literature book and the text is formal. Moreover, Word2Vec (FastText) libraries can be used to add the information from the context of the word.

Character-based input can be extended along with word-based input can be added to improve the accuracy. The word-based input needs to be used with a pre-trained embedding model to enhance the precision of the generated words. Generally, word-based text generation models are not expected to produce new or unseen words, yet it can help to generate more proper words in the context given. The advantage of the character-based model is to produce the words with different endings which is a common task for Kazakh language. Additionally, Named Entities were not added to the model, which also can boost the effectiveness of the model.

3.5 Conclusion

The research towards Open Vocabulary Language Model for Kazakh language has been conducted in this work. The use of neural networks is essential to overcome the sparseness problem and produce relevant results. Moreover, the character-based neural model is suggested to compromise with limitedness of vocabulary. Therefore, the mentioned goal will be achieved by implementing by stage and analyzing the proposed models into a language. Moreover, results shed light on future tasks to emerge the type of words in a context by adding word type information also. This can be very useful in

agglutinative languages as the Kazakh language where the meaning and structure of words are very dependent on word endings.

Even though language models with the character-based input are effective in sequential tasks, we will need to take into account long-range dependencies. Matthews, Neubig, and Dyer propose a combination of three processes: generating words as a character based, generating words as a full word form, and morphological analyzer.[36] They have shown that their experiments outperform pure character-based language models. As a future task, Open Vocabulary Language Model for Kazakh Language supposed to be built as a hybrid input language model with concatenation of character input, word input and morphological knowledge.

4. APPLICATIONS OF LANGUAGE MODELS IN NATURAL LANGUAGE PROCESSING TASKS

4.1 Part of speech analysis

4.1.1 Motivation and related works

Morphological analysis for agglutinative languages can be basically performed in three different ways: (1) rule based; (2) statistical approach; (3) hybrid. The new approach is applying the Neural Networks for this task. However, it could be a challenging task for low-resource languages such as Kazakh. In this section, the state of the art for morphological parser is described, and the focus will be on agglutinative languages.

Rule-based approach: Morphological analyzer for Kazakh Language was implemented by Kessikbayeva and Cicekli [21] using rule-based methods. The broad explanation of the rules in Kazakh languages and application is given current work. The rule-based analyzer was implemented using finite state transducers. Since the language is agglutinative, all morphological rules has been written in lexicon files. The efficiency of the work is about 96%. The context-based disambiguation accuracy was 87%.

Statistical Approach: Makhambetov et. al. [36] developed a data-driven morphological analysis based on statistical analysis which considers both inflectional and derivational morphology. Their method applied Hidden Markov Models and Markov Chains where there was no need to apply formal rules of the language. According to k-fold cross validation score, the performance is 90%.

Bolucu and Can [38] extended Bayesian PoS tagger by Goldwater and Griffiths by applying Hidden Markov Models. The experiments have been done on three languages: Turkish, English, and Finnish.

Ontology-based approach: Bekmanova et. al. [39] developed morphological analyzer for Kazakh and Turkish languages based on the onthologic similarities. Authors developed ontologies for nouns of two languages.

Hybrid approach: Research papers related to morphological analysis of agglutinative languages up to the year 2006 were reviewed by Yuret and Ture [22]. Authors presented a new rule-based method for morphological disambiguation named Decision Lists. The work itself is composition of three ideas: (1) employing statistical and rule-based techniques, (2) sparseness problem was solved by considering each feature separately, (3) analysing the previous tags. The correct results were picked manually and the error rate was 4%.

Assylbekov et al. developed the hybrid morphological disambiguation tool for Kazakh Language. Authors received the data from apertium-kaz that has been tagged using the rules applied by finite transducers by Washington et. al. [40].

Neural Networks approach: Premjith et al.[14] have analysed the internal structure of a word in Malayalam language with the application of Recurrent Neural Networks. Malayalam language is agglutinative language and suffers from the data sparsity, therefore the morphemes borders were identified on the character level. Authors separated the consistent morphemes from the root with the average efficiency of 98%.

Yildiz et al.[41] propose the application of deep neural networks to perform the task of morphological disambiguation for morphologically rich languages such as Turkish. Authors have used the semi-automatically disambiguated corpus by [22]. The efficiency of about 85% was achieved with pre-trained data.

Dhanalakshmi et al. presented the task of morphological analysis as sequence labeling task for Tamil language which is considered as agglutinative languages. Authors developed two models for nouns and verbs respectively using SVMTools of Machine Learning. The morphological analyzer system for verb and noun are trained with the corpus of 130,000 and 70,000 words respectively obtaining the 94.5% in average.

Toleu et al. [42] propose the character-aware morphological disambiguation tool for Kazakh and Turkish Languages. Authors use the LSTM neural network which is one of the Recurrent Neural Networks. The words were analyzed in <root, POS, MC> triplets. This work has made essential improvement to the state of the art of Kazakh Language. The achieved accuracy was 91% per token and 88% accuracy per ambiguous tokens.

4.1.2 Datasets for text processing

Text processing is an essential task in natural language processing. When conducting any experiments or analysis, the most important factor is the dataset. Kazakh language is a low-resource language which does not have a common large-scale database such as Word-net for English. [43]. Word-net is a lexical database which contains the network of words related by meaning. The words are divided into different groups by conceptual-semantic and lexical relations. There are many tools that were created based on the Word-net, for example the tool NLTK in Python [44] that was developed for text pre-processing purposes for languages including English.

In this chapter we will describe the datasets for Kazakh language that were suitable for part of speech tagging task.

Apertium-kaz: tool for analysis

One more tool that is available online is *apertium-kaz* [40] based on Finite State Transducers. Current state of the *apertium-kaz* [40] is

- Number of stems: 36539
- Disambiguation rules: 150
- Coverage: 94.5%

It was initially intended to be compatible with other Turkic languages to compute conversion between languages.

In Kazakh language, one word can have more than one meaning, which makes the disambiguation task more difficult. Different meanings of a word leads to be tagged as

different parts of speech. For example in the sentence *Ауа райы бүгін әбден жақсы, жылы*, the word *жылы* have 5 outputs with:

- "жылы" adj
- "жылы" adv
- "жылы" adj advl
- "жыл" n px3sp nom
- "жылы" adj subst nom

We can notice that the word "жылы"can be a noun, adjective and adverb based on the context given. The most applicable in this context is adj. The outputs of finite-state transducers have been arranged in descendingly depending on the obtained probability.

Kazcorpus dataset

Group of researchers of the Computer Science Lab of the Nazarbayev University Research and Innovation System have developed the Kazakh Language Corpus [37]. This corpus have raw dataset that is available online with about 135 million words. However, for tagging and analysis it cannot be used. The part of speech tagset was developed manually with the students specializing in morphology and syntax. The dataset is in xml format and contains neatly tagged 150 sentences with 1750 words that can be used for Part of Speech tagging. After parsing the xml file, and converting it to .csv format, the data looks as in Table 4.1.

Table 4.1: The part of speech tagset from Kazcorpus

	text	word	root	base	full
0	Біз бейтаныс қызбен жақсы араласа бастадық.	Біз	Біз	SIMZ	SIMZ_N0S0P3C1
1	Біз бейтаныс қызбен жақсы араласа бастадық.	бейтаныс	бей	SE	SE_P3
2	Біз бейтаныс қызбен жақсы араласа бастадық.	бейтаныс	таныс	SE	SE_P3
3	Біз бейтаныс қызбен жақсы араласа бастадық.	қызбен	қыз	ZEP	ZEP_A1N0S0P3C7
4	Біз бейтаныс қызбен жақсы араласа бастадық.	жақсы	жақсы	US	US

Assylbekov's dataset

Assylbekov et al. developed the hybrid morphological disambiguation tool for Kazakh Language [45]. The data was taken from the most visited Kazakh wikipedia sites, and divided into two parts: for training and testing. The authors used different types of articles for testing and training and to make sure that the articles do not overlap, which afterall have been tagged using *apertium-kaz* [40]. The output of *apertium-kaz* contains a few outputs, then the output with the highest probability was taken as the correct label. For this work, the datasets were downloaded from the source and afterall were parsed into one .csv file. The file contains four columns: the word itself, the root,

the part of speech and extended morphological knowledge. The information about the dataset is given in a Table 4.2.

Table 4.2: Train dataset: Most visited wikipedia sites.

Article title	Views	Tokens
Басты бет	1,674,069	-
Жапония	877,693	3,211
Біріккен Ұлттар Ұйымы	807,058	793
CERN	648,464	-
Иран	602,001	2,879
Жапония префектуралары	551,394	-
Футболдан әлем чемпионаты 2014	333,988	257
Жапония Ұлттық футбол құрама командасы	321,249	146
Eurovision ән конкурсы 2010	312,183	101
Абай Құнанбайұлы	242,151	4,083
Радиан	187,225	39
Жасуша	145,010	1,789
Шоқан Шыңғысұлы Уәлиханов	119,780	2,408
		15,706

Table 4.3: Assylbekov et al. merged dataset

Entries	amount
Number of unique words	6908
Number of unique roots	3729
Number of unique full descr classes	538
Number of unique pos tags	34
Total number of entries	17567

4.1.3 Methods for morphological parser

The goal of this work is to develop a system that determines the part of speech of the given word. Neural networks approaches have been used for categorical classification problems.

Neural network is a feed-forward network that contains 2 dense hidden layers. Architecture of the network is shown in the figure below:

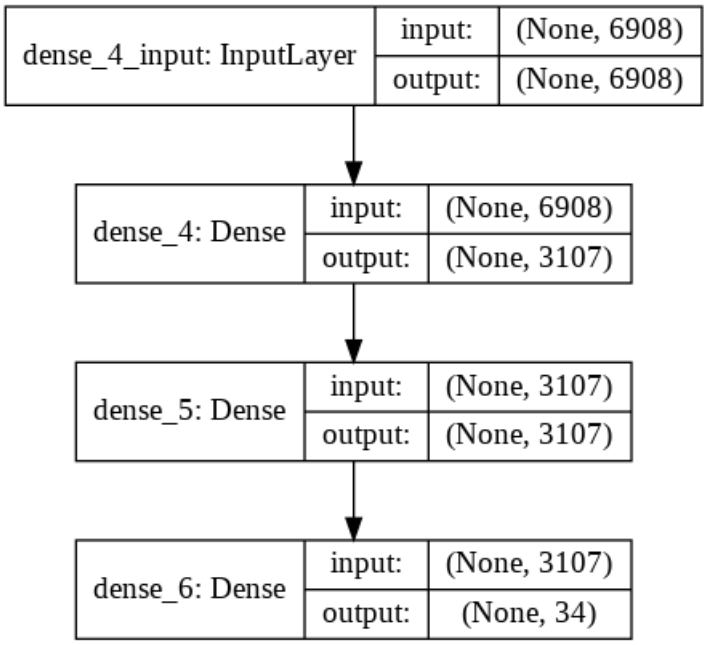

Figure 4.1: Feed-forward neural network for POS tagging

The dataset is described in chapter 3. The words are translated into the dictionary which were converted to one-hot encoded vectors. There are 34 unique classes for different parts of speech. These 34 labels are translated to one-hot encoded vectors as well.

Results. There are two different approaches for input: the word itself and the root of the word. The dataset size and the architecture of the network were the same.
The results of the work is stated in the given table 4.4.

Table 4.4: Results

	dataset size	f1-score word-based	f1-score root-based
train	14931	0.98	0.98
test	2636	0.85	0.88

The objective of the research is part of speech tagging was achieved with 85% accuracy and was increased to 88% when we use the root-based approach. The results imply that the neural network approach has been successful, however the amount of the dataset still needs to be increased.

Moreover, in Kazakh language, there are some words which can be in parts of speeches depending on the context. Therefore, the context of the words needs to be added as a future task.

4.2 Stemming Algorithm

4.2.1 Introduction

In Kazakh language the root is the main meaningful part of the word that expresses the main meaning of the word and the general lexical meaning of all cognates, the rest parts of the words are called suffixes. There are two types of suffixes in Kazakh: inflectional and derivational. Derivational suffixes are used for producing new words from the given root, whereas inflexional suffixes define the word form as: plural, personal, or possessive.

Compared to English languages where the endings of the words are simple and lengths are fixed, in Kazakh language - the word roots can have several inflectional suffixes.

For example, the word оқу - read, can have many different word form such as:

- оқулық - book
- оқушы - pupil
- оқытушы - instructor
- оқыту - teach

It can be noted that there are many words with different meanings with one core root. In text analysis, classification tasks it is very important that these words with common roots have similar vector representations. To overcome this issue, the stemming process is required for any language with rich morphology. Additionally, authors [46] claim that the stemming and lemmatization algorithms enhance the performance of natural language processing tasks. Therefore, in language modeling tasks, stemming is one of the most important parts of the research.

This chapter is to develop a rule-based stemming algorithm for Kazakh language. The rules are pre-defined using regular expressions. The rules are described in detail in the Methods section. Stems of the words will be useful in information retrieval, text classification, and feature extraction that is in high demand in Kazakh language.

5.2 Related Works and Theoretical Framework

Stemming algorithms can be categorized into three types: machine learning (deep learning), statistical methods, and rule-based methods. Machine learning models use pattern recognition to determine the stem of the word, while statistical methods use probabilistic models such as n−gram. Finally, rule-based methods are implemented by pre-defining the rules for the specific language.

Central research is conducted in different languages. Some of these methods are similar to the language of this article or are related to aggressive language. Each of them has its own suggestions, pros and cons, and effectiveness. It is appropriate to discuss, analyze, and discuss tasks related to uniform investigation and the avoidance of duplication.

Turkish language is related to the Kazakh and similar. Both languages are called Turkic. There are many scientific works related to morphological analysis. Based on these arguments, the Turkish language is taken as the first example of this research.

The research paper conducted by T.Kışla and B.Karaoğlan [47] describes two different methods for stemming: rule based and statistical. In order to avoid disambiguation, part of speech analysis is added to this task. The error rate for POS tagging is 8%, additionally, manually tagged corpus reduces the error rate to 6%. The high accuracy of the work has the reason that the suffixes and the dataset are restricted. Authors suggest that the efficiency of algorithms can be enhanced by enlarging the vocabulary of Turkish language.

B. Taner Dinçer and B.Karaoğlan [48] have presented the stemming algorithm which uses probabilistic methods. It was claimed that this algorithm can be applied for any agglutinative language such as Azerbaijani, Tamil, Indonesiand, and of course Kazakh. The time complexity is linear, which overcomes the problem of high computation. The results of the work achieve 96% of accuracy. The algorithm works as follows: the suffixes are deleted one-by-one leaving the stem alone. The pre-processing of the text is an important step of this work where foreign words, abbreviations, and acronyms have been removed from the text. 9,828 words were analyzed correctly out of 10,253.

V.Barakhnin, A.Bakiyeva, T.Batura [49] introduce automatic stemming algorithm for Kazakh language in 2017. The algorithm is found on well-known Porter's algorithm [50] and uses dictionary lookup. Different types of words can be considered as input including verbs, nouns, adjectives and adverbs. Ending combinations are considered from the dictionary table and cut off from the end, and the rest is known as the stem of the word. The research was mainly focused on verbs. Authors believe that presented algorithm is on the right way of development and some amount of errors were revealed in testing phase.

V.Gurusamy, S.Kannan, K.Nandhini [51] proposed the research that has been conducted towards the performance analysis of English language. Three primary stemming algorithms: Porters, Lovins, Paice/Husk were analyzed, evaluated and compared. Since English language has simple morphological structure, all three algorithms used rule-based methodology. Lovins' algorithm is context-sensitive model where the longest matched suffix is eliminated. Similar to Lovins' algorithm, the Porter's algorithm is sensitive to the context of a word. Porter's stemmer is one of most familiar algorithm for English language. Because of the success of this algorithm, this algorithm has been implemented for many different languages. On the other side, the Paice/Husk algorithm for stemming is iterative and conflative. It is similar to Lancaster stemmer, and known as aggressive algorithm which is conveniently developed. 74,450 English words were validated, and the accuracies are:

- Porter - 80.41%
- Lovins - 81.44%
- Paice/Husk - 82.24%.

This work uses a rule-based approach to determine the stem of the word. The morphology and lexical structure of a word can be considered to be successful in building a rule-based stemmer [52] for Kazakh language. Morphological structures of nouns and verbs are visualized in Figure 4.2 and 4.3, respectively.

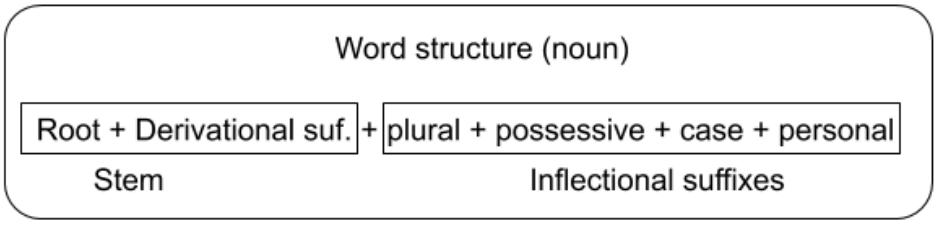

Figure 4.2: Morphological structure of a noun

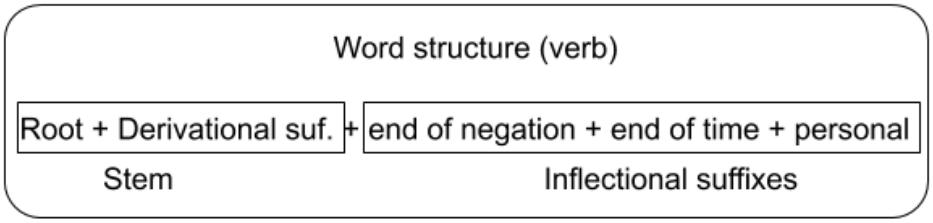

Figure 4.3: Morphological structure of a verb

Since the derivational suffixes moderate the sense of the given word, derivational suffix and the stem together makes the totally new stem. Inflectional suffixes follow the derivational suffixes in a word construction.

For instance, the verb “қала”, which translated to be a ”city” in English and the verb “бар”, which means ”go” in English, The figure 5.3 and 5.4 show the available endings for the words “қала” and “бар” respectively.

There are many algorithms for the stem identification, and can be noted that there are already exists researches that has been conducted earlier. A.M. Fedotov [53] proposed the set of possible suffixes for the verbs and nouns which were used in this paper as a baseline.

Table 4.5: Inflexional suffixes of a noun

Word	Stem	Inflectional suffixes
Қала	Қала	none
Қалалар	Қала	-лар(plural)
Қалаларыңыз	Қала	-лар(plural),-ыңыз(possessive)
Қалаларыңыздан	Қала	-лар(plural),-ыңыз(possessive), -дан(case)
Қалаларыңызданбыз	Қала	-лар(plural),-ыңыз(possessive), -дан(case), -быз(personal)

Table 4.6: Inflexional suffixes of a verb

Word	Stem	Inflectional suffixes
Бар	Бар	none
Барма	Бар	-ма(negation)
Бармаған	Бар	-ма(negation), -ған(end of time)
Бармағансыздар	Бар	-ма(negation), -ған(time), -сыздар(personal)

4.2.3 Methodology

The goal of the work is to correctly determine the stem of the given word. Since the suffixes are sequences one after another, and the Kazakh language follows the rules while constructing the words, rule-based approach was chosen.

The algorithm work using the following principle:

The given word is conditionally separated into two different parts: the root and the suffix. By the word ”conditionally”, it is assumed that the word has not been necessarily correctly divided.

Next, the first syllable is chosen as a root:

```
syllable = re.compile(u"^(.*?[аәеиоөуұүыіэюя])(.*)$")
```

The next step is to diminish the endings of the given word based on the suffix set presented by [53].

Divide Word into Root and endings parts.

The notations for the words were used as [37] Ending analysis:

Let noun endings be:

A1 – plural endings set;
A2 – possessive;
A3 – case;
A4 – personal.
Then, the set of possible ending combinations for noun will be:
A1, A2, A3, A4, A1 + A2, A1 + A3, A1 + A4, A1 + A2 + A3, A1 + A2 + A4, A1 +A2 + A3 + A4, A2 + A3, A2 + A4, A2 + A3 + A4, A3 + A4.
Verb endings:
V1 - negative ending;
V2 - time;
V3 - personal.
Then, the set of possible ending combinations for verb will be:
V1, V2, V3, V1 + V2, V1 + V3, V1 + V2 + V3. [42]

Taking into account the combination rules, the algorithm starts to iterate from the end of the ending. First of all, a temporary variable is created which cuts off the last of the ending type. For example, for nouns it is *P4* - a personal ending and for verbs *P3* is a personal ending.

Then temporary variable gets compared with an ending that have been created at the beginning.

First outcome, if they are not equal, iteration proceeds to the next type of endings and the pattern repeats till the stem has been found, as a result initial ending part changes and gets concatenated to the root.

Second outcome, if they are not equal, function tries to find other types of endings, if one has been found, an iteration proceeds. For example, if P 4 is not found, function searches for P 3 and does the same as the first outcome.

Finally, if there is a match for the derivational suffixes after noun or verb that make adjectives, endings are deleted. Additionally, after noun and verb endings are deleted from the word, derivational endings that form adjectives from noun and verb words are deleted from the word if there is some. The Table 4.5 illustrates the general stemming algorithm for the noun. Since the purpose is not in the grammar structure, algorithm goes through the last priority or the last stage ending presence checking and cutting to the first added ending.

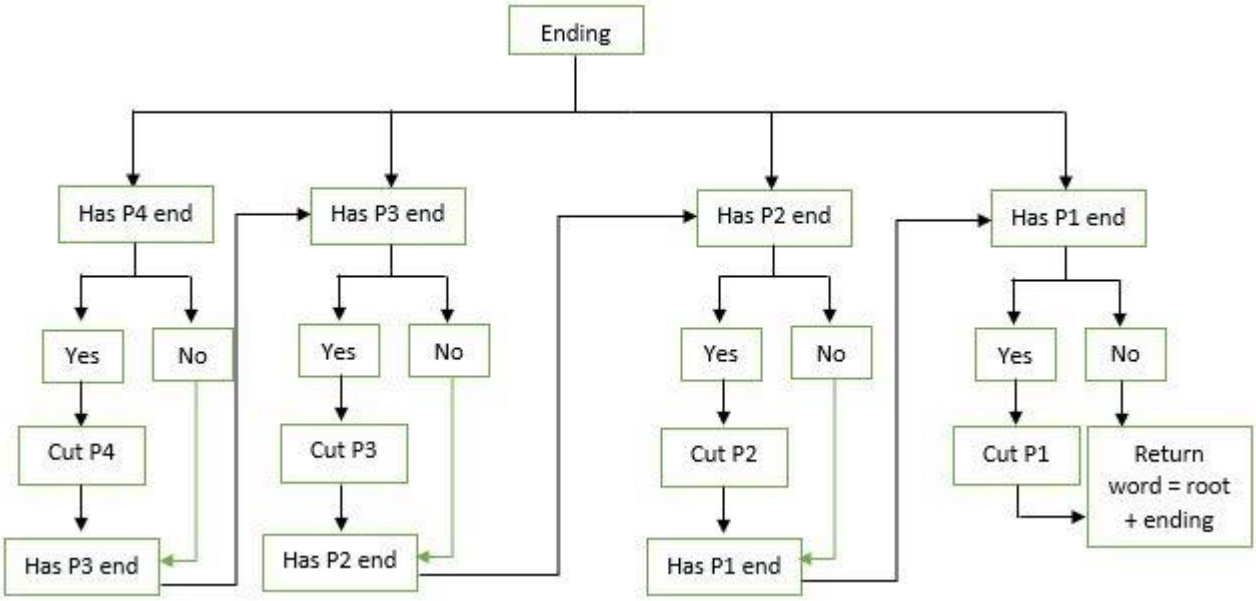

Figure 4.4: Stemming algorithm for the noun

4.2.4 Results

Stemmer algorithm's performance was checked by comparing annotated kazakh language corpus. The corpus is an open Kazakh Language Corpus, which was carried out by philology students from Nur-Sultan(Kazakhstan). The Xml version of the dataset had been further parsed for research. [37]

The results are shown in Table 4.7 Data where all of the existing affixes of the word are annotated, also the part of speech is defined too. As we can see Stemmer works well on stated rules of the algorithm, but since there are many of the possible ending sets, some affixes get deleted too. In addition, there are other parts of speech that have not been considered, therefore differences occur.

Table 4.7: The results of stemming algorithm

	Word	Annotated Root	Stemmer result
0	Біз	Біз	Біз
1	бейтаныс	бейтаныс	бейтаныс
2	қызбен	қыз	қыз
3	жақсы	жақсы	жақ
4	араласа	ара	арала
5	бастадық	баста	баста
6	Лайым	Лайым	Лай
7	осылай	осы	осылай
8	жалғаса	жалға	жалға
9	берсін	бер	бе
10	деп	де	де
11	тілейік	тіле	тілей
12	Еркін	Еркін	Еркі
13	тақырыпта	тақырып	та
14	мақала	мақала	мақала
15	жазуға	жаз	жазу
16	тапсырма	тапсыр	тапсыр
17	берді	бер	бер
18	Оларды	О	О
19	кемінде	кем	кем
20	үш	үш	үш
21	адам	адам	ада
22	лақтырып	лақтыр	лақтыр
23	кеткен	кет	кетке
24	Шынымды	Шын	Шын
25	айтсам	айт	айт
26	соны	со	со
27	мен	мен	ме
28	де	де	де

4.2.5 Conclusion

In conclusion, this paper proposes and describes a rule-based Kazakh language stemming algorithm. Stemming algorithms have been implemented and extensively used in English, Russian and other popular and not so complex word structure languages. Since Kazakh language has more complex and flexible endings and types, which makes its morphology rich, it is more cumbersome to process the rules.

It is important for Kazakh language to intensify the performance of NLP and IR areas by developing applicable tools and algorithms that will certainly influence the performance of processing data.

Algorithm proposed in this paper separates the possible ending combinations fortunately, but in comparison with annotated data, which had been conducted manually by masters, who consciously work with Kazakh language sense of word prepared roots,

admits mistakes. There are several reasons. Firstly, Kazakh language has non-obvious morphology structure, words can have inflectional suffixes that are derivational. Secondly, noun and verb endings can be the same, which makes an algorithm allow mistakes. Finally, named entity recognition algorithms need to be processed first to avoid the issues. The reasons for stemmer accuracy faults may be various. Exception words of Kazakh language which do not follow the common rules, as a future work, exceptions can be added, also the algorithm itself could be enhanced.

4.3 Text Classification Methods

4.3.1 Introduction

There are a large number of industries that manually classify each customer inquiry to which type or class their inquiry belongs to. In such cases it would be very useful to classify those inquiries more efficiently.

This paper proposed methods and solutions to automatically classify those inquiries by using various Machine Learning Algorithms like SVM, Naive Bayes, Logistic Regression by vectorizing texts using TF-IDF. These methods would dramatically decrease the time that would be spent to manually do it. Since all texts are in the Russian language, default text preprocessing methods have to be analyzed to make sure that they are relevant to Russian texts. This analysis would be very helpful for industries that are involved in text classification and categorization of any sort. The organization of this paper is given below: Section 4.3.2 contains Related Works that have been conducted on similar topics and do the review of algorithms that were used in this research. Section 4.3.3 explains methods and techniques that were used during this research. Section 4.3.4 and 4.3.5 demonstrates data description with its properties and Empirical Analysis of our methods and their results. Section 4.3.6 concludes the paper. Section 4.3.7 explains how results can be improved and gives other tips related to Future Work.

4.3.2 Related Works

Text classification methods have become a widely studied field in recent years. Many researchers already found efficient algorithms for text classification and proved its effectiveness in classifying the unstructured text documents for the English language.

Borodkin, Lisin and Strielkowski [54] have presented a way of pre-processing the unstructured text document. Further they mentioned many problems that can occur during pre-processing and provided the solutions for such cases. Additionally, they compared many machine-learning algorithms by estimating the results after training those algorithms on pre-processed data. In this work, authors used Machine learning algorithms such as k-Nearest Neighbours, Naive Bayes, and k-means within the Software Package which is a black box. Their comparative results show that the k-means algorithm outperforms others and has precision of 78,33%.

Mowafy, Rezk and El-bakry [55] provided their sequence of methods to perform pre-processing. Besides that, they profoundly discussed the tf-idf algorithm and its application to their work. Furthermore, they trained various machine learning algorithms and calculated the most suitable ones to work with the format obtained after vectorizing using tf-idf. The authors concluded the superiority of Multinomial Naive Bayes with TF-iDF. Moreover, the proposed model has an application of chi-square which increased the efficiency from 71% to 76%.

Ashwin, Le Minh and Antal [56] discussed text analytics. They reviewed many text related problems that companies tried to solve using text analytics. For each problem they provided the method of solution and the evaluation of that method. It was revealed that most of the problems with text classification in industry require robust algorithms since the data comes from different sources. Additionally, k-means algorithm was proposed as one the convenient algorithms for high clustering quality. Authors suggest to use minimum supervision, instead apply TF-IDF or GloVe.

Tilve and Jain [57] compared three different text classification algorithms(SVM, Naive Bayes, and Standford Tagger) that were trained by two unsimilar datasets(20 Newsgroups and New news). According to their results, Naive Bayes algorithm is a better choice among the others due to its performance and simplicity.

Overall, we analyze all known algorithms and compare their results. It is known that algorithms outperform one another according to given data and its type. However some application identities may affect variations in results. Therefore, analyzing best known algorithms in our case is indispensable to compare their efficiency such as:

- Random Forest Classifier – is one kind of bagging algorithms where a number of decision trees used on a various subsets of dataset and use the averaging to decrease the error and prevent over-fitting. [58]
- Support Vector Machines - are universal learners that are used for classification, regression and outlier detection purposes. It is a very powerful algorithm since it is very effective in high dimensional spaces and has different kernel functions. [59]
- Multinomial Naive Bayes – is a supervised learning method that is suitable for classification of discrete values or fractions such as tf-idf. Naive Bayes algorithms are based on statistical Bayes Theorem where every pair of features are assumed to be independent [60].
- Logistic Regression – is a supervised learning algorithm that uses logit function as its base. The goal of logistic regression is to find the best suitable hypothesis function [61].
- Decision Tree Classifier - is a non-parametric method that is used for supervised learning. This algorithm used a tree-like graph model of decisions [62].

4.3.3 Methods

It is intended to compare all algorithms using TF-IDF as a Vector Space model for text extraction. In order to be able to build a classification model, there are some general steps to perform such as:

- Pre-processing phase
- Training/Testing phase
- Usage phase

In this section, these steps are clarified by giving examples and illustrating in details.

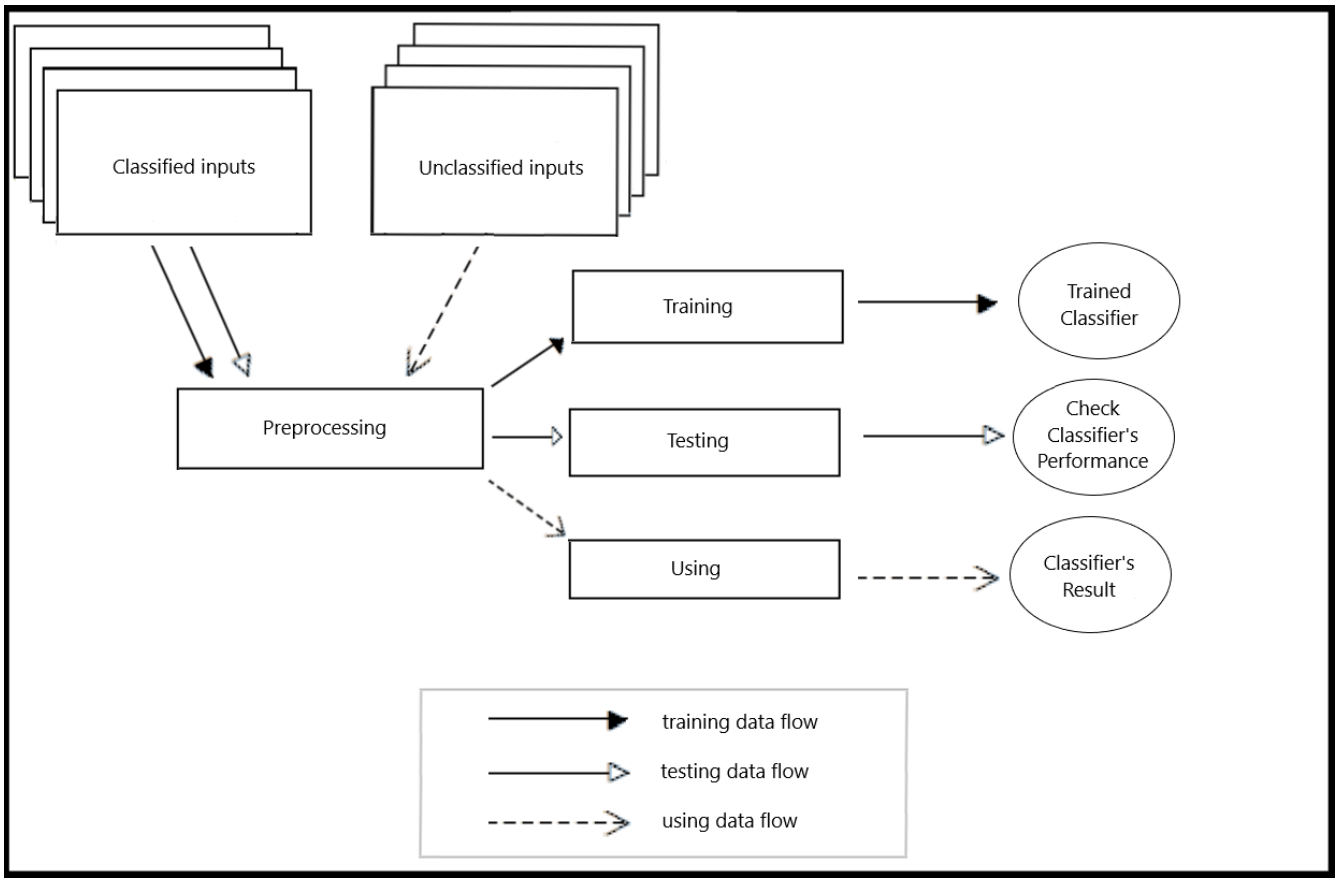

Figure 4.5: The scheme that shows the work process of data

4.3.3.1 Pre-processing phase

Pre-processing phase includes removing stop-words tokenization, removing, stemming.

- Tokenization - is the process of partitioning a string into a list of tokens.
- Removing stop-words - is the process that gets rid of stop-words(e.g.'or','and', 'etc.') that helps to clean the string and use only meaningful words.
- Stemming - is the process that converts different words into similar canonical form.

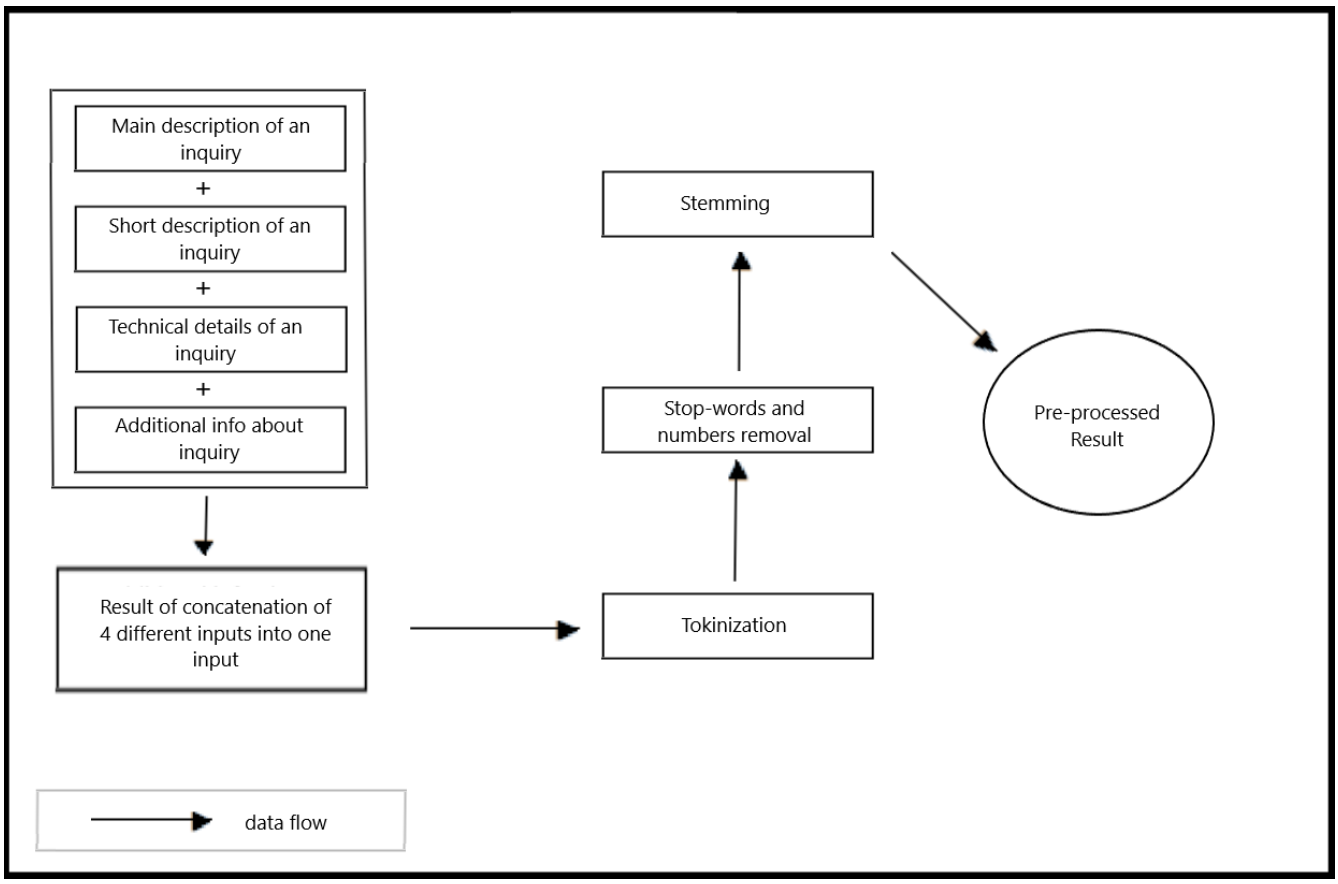

Figure 4.6: The scheme that shows the pre-processing process.

First of all 4 input fields were concatenated to be used as one input. Each of those inputs is treated like a different document. NLTK library's tokenizer [44] is used in order to perform tokenization.

After obtaining the list of tokens, the data must proceed to a stop-words removal process that excludes all stop-words and numbers as well. Stop-words removal process uses NLTK library's default corpus for Russian language[44]. In addition to stop-words and numbers removing, named entities were removed by the library too[44] because they do not have any impact on classifiers. The next and the last step for pre-processing is stemming. In order to perform stemming *pymoprphy2*[63] was used . In this way the pre-processed input is obtained and preserved the text in a clear prepared format for next phases in text classification which is the hanging and testing phase.

4.3.3.2 Training/Testing phase

In order to start the training process the pre-processed input must be translated into more convenient form. One of the best techniques is to convert data into vector form and then use it to work with machine learning algorithms. In this case the data was translated into TF-IDF vectors.TF-IDF - Term Frequency – Inverse Document Frequency, is a numerical statistic that shows how important a word is to a document in

a collection or corpus. Basically, TF-IDF is the product of two statistics, term frequency and inverse document frequency.

- Term Frequency - is a number of times the word occurs in a document
- Inverse Document Frequency - is a measure of how much information the word provides, that is, whether the term is common or rare across all documents.

$$W_{i,j} = tf_{i,j} * \log(N/df_i) \tag{6.1}$$

Where $tf_{i,j}$ = number of occurrences of word i in document j
df_i = number of documents that contains the word i
N = total number of documents

We use this *w's* (weights) in our algorithms in such way:

- Multinomial Naive Bayes: [60]

$$p(c_i) = \frac{N_{c_i}}{N} \tag{6.2}$$

$$p(c_i|w) = \frac{count(w, c_i) + 1}{count(c_i) + |v_{c_i}|} \tag{6.3}$$

Here N_{c_i} is the number of correct words in class *c*
N is total number of words
$p(c_i|w)$ is the probability of correct words in class c_i given the weights *w*.

- Logistic Regression: [61]

$$s(w) = \frac{1}{1 + e^{-w^T x}} \tag{6.4}$$

where *s(w)* is sigmoid activation function
w's are weights and *x*'s are our features

- Linear SVM: [59]

sgn(w): Sign function for Support Vector Classifier:

$w^T x + w_0 \geq 1$ if *y=1*

$w^T x + w_0 < -1$ if *y=-1*

To get TF-IDF vectors scikit-learn's TfIdfVectorizer have been used. This approach uses uni-grams as well as bi-grams. This would output sparse vector which has a feature size of 10459.

$$\begin{pmatrix} & T_1 & T_2 & \dots & T_i \\ D_1 & w_{1,1} & w_{1,1} & \dots & w_{1,i} \\ D_2 & w_{2,1} & w_{2,1} & \dots & w_{2,i} \\ \dots & \dots & \dots & \dots & \dots \\ D_j & w_{j,1} & w_{j,1} & \dots & w_{j,i} \end{pmatrix}$$

Where T_i represent the words and the D_i represent the documents. This phase is in charge for learning the classifier model and the output which is already a trained classifier and is ready for testing phase.

4.3.4 Data Description

Data for this research were retrieved from the website www.kmggs.kz. The data consists of 2 separate databases with slightly different structures. First database consists of 32000 records of customer inquiries. Whereas, the second one is 2000 detailed records of inquiries. All this data is labeled into 20 classes (Figure 7.5) that are different in fields. There are 4 main features that customers would provide and they are labeled into 1 main class and 5 consecutive classes that define the entity of that inquiry.

4 main features are:

1. Main description of an inquiry
2. Short description of an inquiry
3. Technical details of an inquiry
4. Additional info about inquiry

And the classes are as following:

1. Main class: type of a material
2. Group of a material
3. Class of a material
4. Group of a purchase
5. Class of an assessment
6. Quantity of a material

This research mainly focuses on classifying main class because other classes are can be identified by the main class. From Figure 4.7 you can see that records are not equally distributed. Also you can see that there are only 13 classes not 20. This is because other classes are almost nonexistent or can be seen as outliers.

This data is more detailed and contains many classes so in analyzing data this research would mainly focus on this data. Whereas, the second data-set contains a large

number of instances but it is not detailed but more generalized. From Figure 6.4 you can also see that proportions are very similar to the previous one.

This data contains many lexical errors and are not structured. Each instance is written in a heavily technical language with a mixture of random symbols. Additional information and technical details fields has many empty cells.

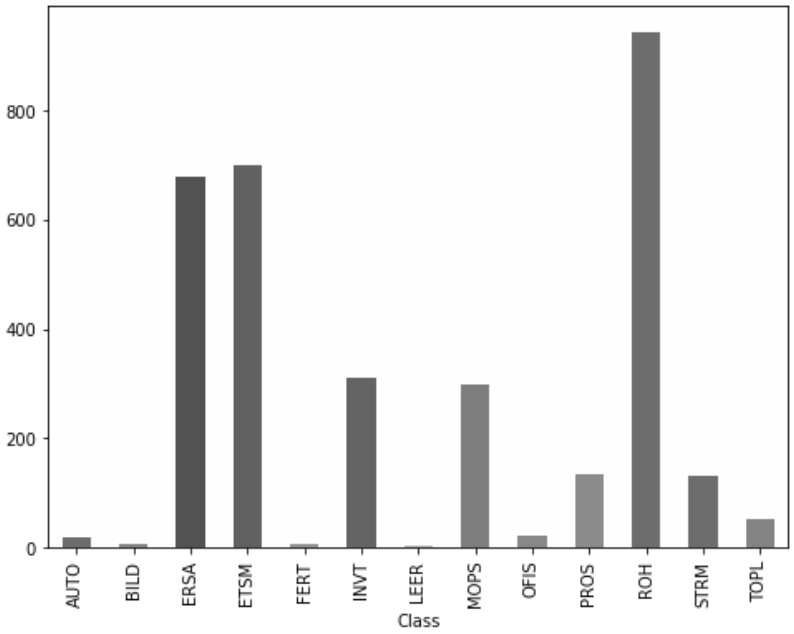

Figure 4.7: Histogram of small data-set.

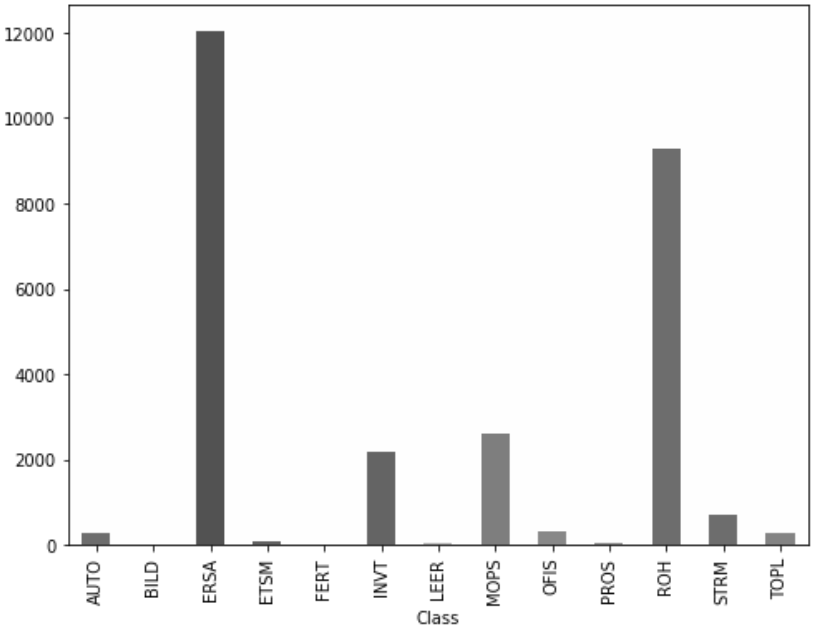

Figure 4.8: Histogram of large data-set.

4.3.5. Empirical Analysis

Since the prime goal of this research paper is to analyze and compare algorithms performance of text classification, main focus would be to analyze their accuracy and effectiveness in real data then compare their productivity. First of all, let's look at TF-IDF. Scikit-learn's TfIdfVectorizer was used to convert raw text into tfidfVectors which produced 10459 dimensional vectors for each document. This also used bi-grams to improve performance. Then cross-validation score was found using scikit-learn's cross val score library. Figure 4.9 shows cross validation scores of 5 algorithms. As you can see from this figure LinearSVC and Naive-bayes algorithms outperformed others. Random forest and Decision tree classifiers are as expected have worse cross validation score in text classification.

As can be seen from Figure 4.9 the best algorithm for now is LinearSVC. After splitting the dataset into test (30%) and train (70%) this model gave the accuracy of 86.33 %. And the heatmap of actual vs predicted can be seen from the Figure 4.10.

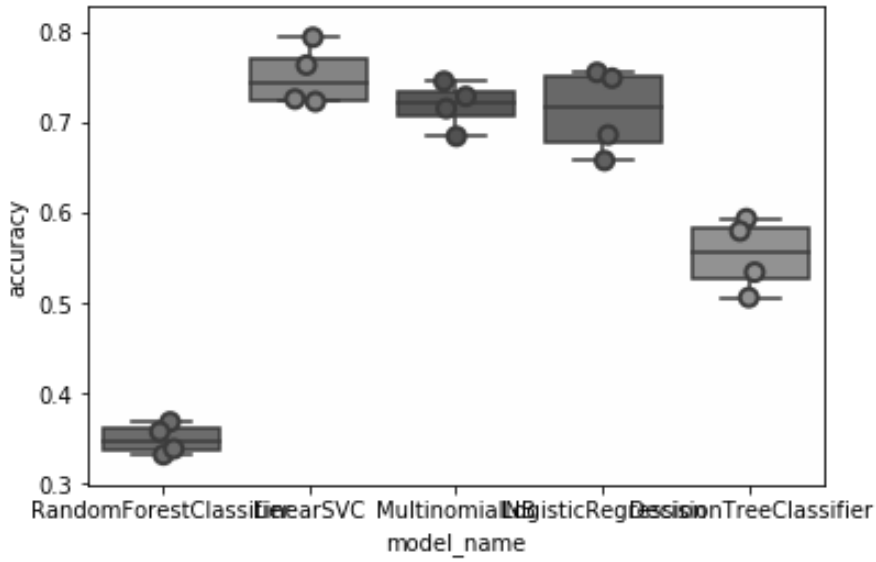

Figure 4.9: Cross validation score.

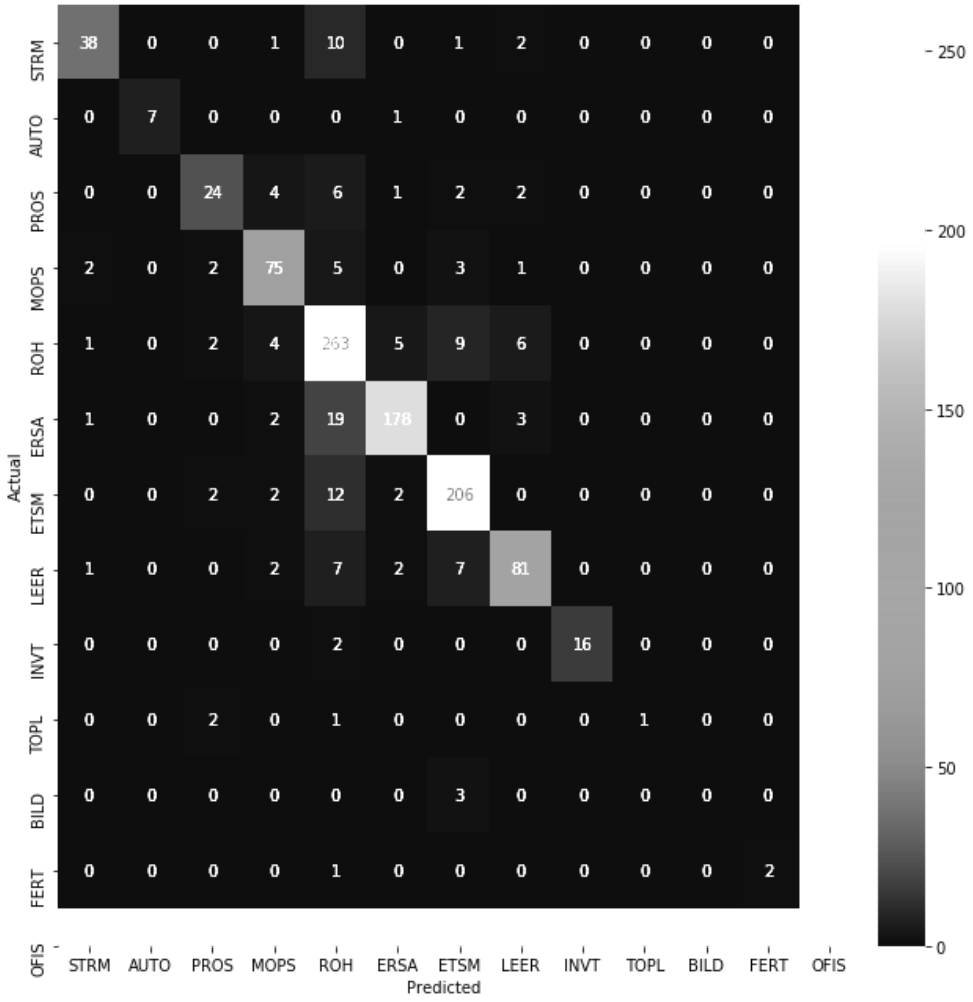

Figure 4.10: Heatmap of actual vs predicted.

As you can observe from the heatmap most misclassifications are due to classes like ”STRM”, ”ERSA” and ”ETSM” being classified as class ”ROH”. The sole reason for this may be due to the fact that class ”ROH” outnumbers other classes very much. That is why classification report of this model is pretty poor in some of the examples with few test cases.

Table 4.8: F1-scores

class	precision	recall	f1-score	support
AUTO	1.00	0.88	0.93	8
PROS	0.75	0.62	0.68	39
MOPS	0.83	0.85	0.84	88
ROH	0.81	0.91	0.85	290
ERSA	0.94	0.88	0.91	203
ETSM	0.89	0.92	0.91	224
LEER	0.85	0.81	0.83	100
INVT	1.00	0.89	0.94	18
TOPL	1.00	0.25	0.40	4
BILD	0.00	0.00	0.00	3
avg / total	0.86	0.86	0.86	1032

4.3.6 Optimization with Neural Networks

This chapter is devoted to evaluate whether the Artificial Neural Networks can be successfully applied to customer inquiry classification task, especially considering the limited number of dataset available for Russian language. Indeed, it seems that it is a valuable solution for this kind of problem. Neural Networks. The literature for applications of Neural Networks is vast and growing. Hassan and Mahmood [64] have compared the linear classifiers and deep neural networks, and authors propose to improve the classification model by using deep learning techniques. Kowsari et. al. [65] proposed the hierarchical deep learning model for text classification which combines a few neural network architectures, and achieved the 86% of accuracy.

In this work the feed-forward fully connected neural network have been implemented with five hidden layers, each consists of 512 nodes. Choosing the optimizer for a neural network is crucial since well trained network need to have properly defined loss function. The first choice for the optimizer in this work was ”adam” optimizer from sklearn library. Unfortunately, the accuracy for this was about 30-40%. Stochastic gradient descent was the optimization for the neural network. Stochastic Gradient Descent (SGD) simply updates the expectation and computes the gradient of the parameters using only a single or a few training examples. The new update is given by:

$$\theta = \theta - \alpha \nabla \theta_j(\theta; x_i, y_i)$$

The results are given in the Table 4.9

Table 4.9: F1-scores for Neural Network

class	precision	recall	f1-score	support
SRTM	0.96	0.50	0.66	52
AUTO	0.22	0.50	0.31	8
PROS	0.54	0.38	0.45	39
MOPS	0.61	0.70	0.65	88
ROH	0.79	0.83	0.81	290
ERSA	0.72	0.85	0.78	203
ETSM	0.82	0.75	0.78	224
INVT	0.92	0.56	0.70	100
TOPL	0.71	0.56	0.63	18
BILD	0.00	0.00	0.00	4
FERT	0.00	0.00	0.00	3
OFIS	0.00	0.00	0.00	3
avg / total	0.76	0.73	0.74	1032

From the results given above, we can see that the F1 score is low on the classes with small amount of data. This happens because we have biased and sparse data and the NN cannot be properly trained.

4.3.7 Conclusion

In conclusion, it can be noted that inquiry classification in russian can be performed pretty good with default TF-IDF and with minimal preprocessing if dataset contains biased class and with limited number of data. It can be seen that LinearSVC and Naive-Bayes performs very good with TF-IDF. Whereas algorithms like Decision tree and Random Forest perform poorly. Moreover, it can be noted that with the sparse data, it is less efficient to apply neural networks. Also, in the process of preprocessing we have observed that libraries like pymorphy2 are very good for stemming russian language. Although, it should be noted that it cannot be used with words with spelling errors. Additional to this Named Entity Removal can be further improved for particular Kazakhstani companies and locations. This work is important since the analysis of real data and it coincides with expected results of algorithms.

4.3.8 Future Works

This model can also be improved by working with outliers. Most outliers are the result of misclassification of the largest class of its closest classes. This can be improved by building a Doc2Vec library and observing whether those outliers can form clusters so we can detect them before classification. Doc2Vec is required in order for us to use the nature of those documents and see if they can be accurately identified. Using this kind of embedding is due to the fact that Doc2Vec gives us a sort of meaning of those documents whereas TF-IDF is just frequencies of words in those documents. Additionally, better Named Entity Removal can be built and applied to this model which can improve accuracy specific for Kazakhstan locations and names. Moreover $\chi 2$

(chi-square) can be added to TF-IDF in order to improve results as it was proposed in[55].

4.4. Summarization Techniques

The aim of the chapter is to consider methods for arguably efficient, cost-effective summarization of texts written in Kazakh Language. This research has a valuable impact on the solution of problems that are indirectly related to evaluation of extracted summaries. The research also considers possible ways of abstracting summarized content to make it possible to use the results as a feed to neural network models. Kazakh Language's sentence construction is observed and precisely described the key points that are unique.

4.4.1 Introduction

Text summarization is a common problem in Deep learning, Natural Language Processing and the related stuff. Generally it refers to methods, techniques of shortening long texts. The purpose here is a brief, coherent summary that contains main points, the gist outlined in the text. Usually, machine learning models are trained to understand texts by feeding samples all over the net. The useful information is distilled and the required summarized text comes in the output. Today, our world is built and consists of pure, unformatted data. The results from the International Data Corporation (IDC) mentions that the total amount of digital data transmitted all over the servers and clients ranges, approximately, from 4.4 zettabytes in 2013 to hit 180 zettabytes in 2025. When it comes to natural language processing, the creation of automatic summarization methods is considered as a very important task. This would allow any student, teacher, or person employed in any other professional field to quickly understand a really huge amount of information at once. Extraction and concatenation of important sentences from the primary text becomes really difficult when we face the uniqueness of every language the text is written in. And Kazakh language is not an exception. The language itself contains a lot of uncommon issues like the "every verb in a sentence written in kazakh is placed at the end". That makes difficulties that should be handled very accurately and with significant attention. The methods that will be used during the research are: extraction and abstraction. The extraction method selects sentences and phrases that have high marks of importance, and combines them into a new short text, without changing the meaning of selected units. When it comes to abstraction, the things are pretty different: the main content (text's gist) is extracted from the source text and paraphrased using linguistic methods that are specific to Kazakh language for semantic analysis and text interpretation. In order to use the mentioned two methods, the sentences should be preprocessed by applying morphological analysis and pronoun resolution techniques. Then, we selectively pick the features to later extract important sentences from the text. All of this collectively gives us properly organized and managed data to be used when we perform text summarization methods.

4.4.2 Literature Review

Text summarization techniques have been considered as an objective of many different researches and have a lot of valuable literature to walk through before proceeding to the rest of the paragraphs. One can get a thorough review of works on the subject at the recent surveys [66].

The research briefly reviews several different text summarization approaches. In order to invest in accuracy we should selectively take text units that have highest scores as a resultant summary. Hidden Markov models [67], conceptual graphs [68], swarm intelligence [69] are arguably efficient proposals that can solve the issue. Kazakh Language's specific rules and considerations have been discussed through some research works [70], [71]. Different summarization algorithms such as "TextRank algorithm", were used in a recent work of researchers [72]. They collected sample data (cyrillic russian and kazakh texts) from the web: mostly online news websites used. The method we use to delete unnecessary noisy indents, punctuation marks and other characters is described in the following researches: [73], [74].

In order to break texts and sentences to different valuable units that are easier to tokenize we used a similar way to a segmentation described at [75]. Pronouns in the sentences located usually at the beginning are removed from the text at the initial stages of summarization. But in fact, they sometimes carry certain significance as they indicate to a subject performing a particular action in a sentence. Therefore some of them were replaced by the concrete names they point to. Other "stopwords" that does not carry any significant value and special meaning are removed from the text. "Stop words" deletion also made along with stemming, which is a common technique in text preprocessing study. There are a lot of other researches that are not directly related but are useful in certain aspects of cyrillic text recognition and analyzing. One of these is [76]. The research uses recurrent neural networks to identify kazakh and russian languages which makes it interesting to learn and walk through. As it comes to specific Kazakh Language letters, the following researchers have done a huge work on factoring out the differences: [70], [71], [73]. They mostly consider the spoken language, involving phonetic transcription of the letters. The methods described there are used in speech recognition projects and robotic IoT control systems.

4.4.3 Methods

There are two methods for summarizing text: extraction-based summarization and abstraction based summarization.

Extraction-based summarization

The extractive summarization is a technique involving pulling the keyphrases or sentences from a text and combining them to create a summary. The extraction is made according to the defined metric without making any changes to the texts. Different types of algorithms could be used to arrange the sentences or keyphrases from most important to the lowest. The final step is to rank the sentences or keyphrases according to their relevance and combine them to generate a summary.

Abstraction-based summarization

Abstractive text summarization aims to compress the lengthy texts to the coherent format which enclose all the most important information and facts from the original document. Summarization techniques that are based on abstraction, develop new keywords and sentences that retrieve the most essential information from the source. Thus, it can be noted that abstraction-based techniques work better rather than extraction-based methods. The abstraction-based summarization can avoid grammar inconsistencies when applied using deep learning architectures. The amount of new phrases which have not arisen in original text measures the level of abstraction. This level remains low in current approaches. Because of the difficulty of developing abstractive models, extractive methods are in most use.

4.4.4 Conclusion

The amount of research on computational linguistics is rapidly increasing. The reasons are the importance and demand of collecting a summary content of a large flow of information. A key point and effectively constructed summary is always easier for public perception. A Sample data, taken mostly from news articles, is effectively extracted and summarized using the methods described in the research. The extractive summarization method in addition with morphological analysis and "stop words" removal and resolution techniques are considered in this work along with the description of abstractive summarization.

4.5 Sentiment Analysis

Natural language processing is considered fundamental for the further development of artificial intelligence. In Kazakhstan, Machine Learning is getting more and more popular and actual because it helps to analyze and classify data better. The goal of this work is to classify Russian texts according to the sentiment, thus, the hypothesis is that the trained model will predict input data correctly and the application for the project implemented will work correctly.

4.5.1 Introduction

Among the most interesting and popular methods of this broad scientific field, there is one that stands alone, called sentiment analysis, which means "analysis of the tonality of texts", in our case the language is Russian. The general definition says that the analysis of the tonality of texts is a class of content analysis methods designed to automatically detect the emotionally colored vocabulary in the text, as well as the author's opinions (emotional assessments) about the objects in the text. In Kazakhstan, Machine Learning is getting more and more popular and actual because it helps to analyze and classify data better. Companies get interested in ML specialists. We have provided with data set, which contain scrapped data from news portals. The problem that they always face is analyzing national usage of tonality in social networks, it helps to

detect suspected people and for many other reasons. Our aim was to use train data to create a model that will predict the sentiment of testing data or input text.

4.5.2 Theoretical framework

Subjective texts are very useful for many applications, for that reason, many researches have shown interest in sentiment analysis tasks. S. Mukherjee [77] has discussed the different Machine Learning models to classify the tonality of English text. Depending on the obtained results, different evaluation metrics have been seen in the work. Soft computing methodologies have been applied for sentiment analysis in a work of Kumar and Jaiswal [78]. Authors displayed how soft computing techniques have been used to overcome the fuzziness such that marked increase in the size, subjectivity, and diversity of social web data, the ambiguity, and uncertainty. Qazi et. al. [79] pointed out how traditional classification problems can be addressed to sentiment analysis problems by adopting improved methods. The work is devoted to review regular, comparative and suggestive reviews and related sentiment analysis techniques.

4.5.3 Problem description and algorithm

The aim of this work is to construct a model that will identify the tonality (neutral, positive, negative) of the text (in the file .json formatted file). To be successful in this task, we need to train the model on current data (train.json). It is worth noting that, according to the rules, the use of a third-party corpus of texts with a marked key is prohibited, but this does not prohibit the use of NOT-labeled corpuses for preprocessing text. The model is trained using machine learning algorithms. The resulting model needs to be able to identify the label (positive, neutral, negative) of testing texts which have not been used for training with high enough accuracy.

The process consists of two parts:

1. Building a model on data from the train.json file;
2. Prediction of tonality, using our algorithm, on the data from the .json file. At the last stage, we send the results of the analysis of tonality to the test data, namely the CSV file containing the id fields (the unique identifier of the document, the tonality of which we determined), sentiment (tonality predicted by the model: negative, positive, neutral).
3. Verification of the results. Scikit-learn accuracy score is used to verify the results.

The model is trained using machine learning algorithms. The resulting model will have to determine the class. We have used Random Forest classifier, imported ensemble models from scikit-learn. We have used the ensembles since with the assumtion that all n base classifiers have the same error rate , we can express the probability of an error of an ensemble can be expressed as a probability mass function of a binomial distribution. That is why the Random Forest algorithm performs better to train the data.

4.5.4 Experimental evaluation

First of all, the data needed huge preprocessing. initial data had been given in JSON format and after cleaning data we converted it to CSV leaving needed data for further usage.

To do this the following processes had been done:

1. Remove non-letters
2. Convert to lower case, split into individual words
3. In Python, searching a set is much faster than searching a list, so convert the stop words to a set
4. Remove stop words
5. Join the words back into one string separated by space, and return the result. First, it fits the model and learns the vocabulary; second, it transforms our training data into feature vectors. The input to fit transform should be a list of strings.

As a visualization of data content, we demonstrate top-10 words:
(They are: (’год’, 34520), (’тенге’, 16268), (’казахстан’, 10545), (’ао’, 10181),(’рк’, 8887), (’области’, 7750), (’млрд’, 7360), (’республики’, 7228), (’развития’, 6215), (’г’, 6176), (’лет’, 6095), (’алматы’, 5666))

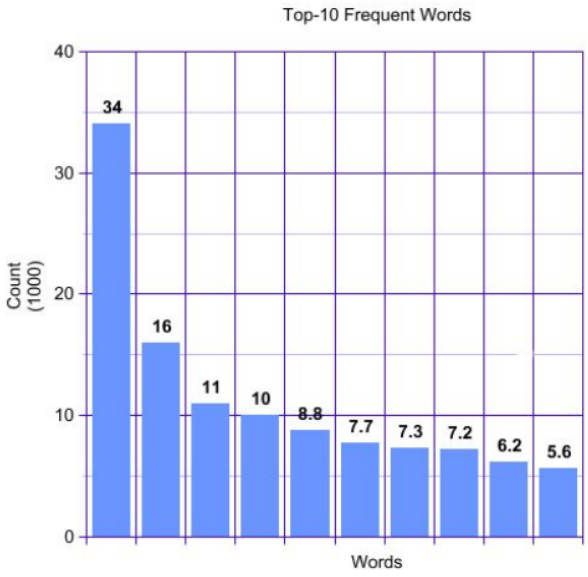

Figure 4.11: Top 10 frequent words

Once data has been preprocessed, Machine Learning algorithm-Random Forest can be trained. Also, in order to get better results, testing data should be cleaned, as it is done in our work.

Without preprocessing we have obtained the effectiveness of algorithm was about 60%. As it was proposed in literature review the application of Support Vector Classifier

and Multinomial Naive Bayes classifiers have increased the performance. The accuracy for SVM is 82.35% and for Naive Bayes is 82.15%.

4.5.5 Application

One of the major project requirement was to build an app for the implemented project. In the process, we have discovered the web application framework - Flask.There are many ways to raise your own web server, which will process HTTP requests from users and return results to browsers. Since we use Python as our main language, we will also choose a library that simplifies the creation of a web server for us from the Python world. Flask is a tool for Python websites. It is a microframework with a built-in web server.

While testing an application we have faced several problems. For example, we need to take into account many flask app features, there was a job done to achieve connection. But the most important case is Memory Error assurance while using preprocessed training data. We could not use all of the prepared data to predict the sentiment of input(test) data. Only the piece of the data, that is why used. Also, we do not have checked all of the data to truthfulness, that is, scrapped data has many words in one line of trained sentiment, do the results may be predicted wrong, it affected the results correctness.

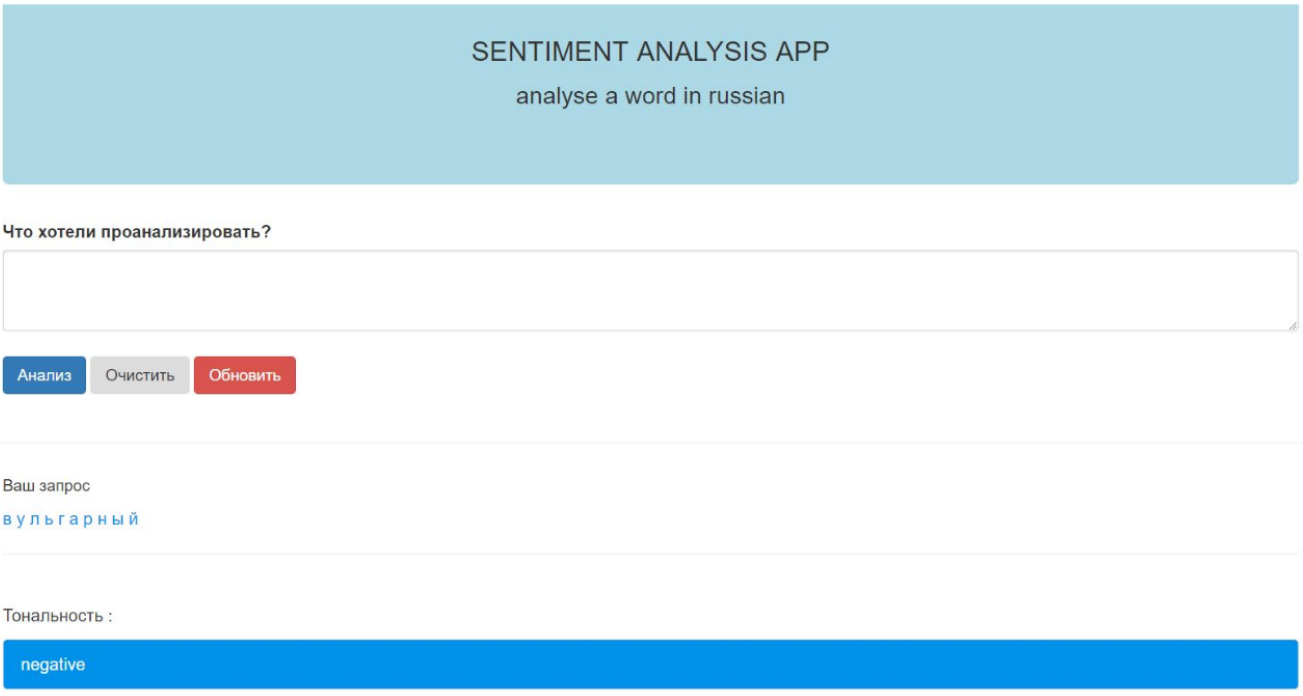

Figure 4.12: Sentiment analysis application

4.5.6 Conclusion and Future Works

As the results show, with the help of this method one can achieve quite high (82%) accuracy on texts of a certain subject. However, a number of unrecoverable errors remain (disregarding errors of external modules, such as errors of morphological and syntactic analyzers). In our opinion, one of the reasons for this kind of error is the limitedness of the emotive space used: some of the vocabulary does not fall (or only partially fall) into our emotive space well-good plus the power of emotiveness. The definition of dimension is an open research question, the solution of which lies in the field of understanding and perception of information by the human brain. Thus, a qualitative improvement in the method of determining tonality that we choose needs further fundamental research, not only in the field of linguistics, but also in the field of cognitive sciences, such as psychology, psycho, and neurolinguistics.

4.6 Anomaly Detection

Anomaly Detection is very important field in data science. There is very high correlation between finding outliers in a dataset and developing good algorithm. The goal of this paper is to review the papers on anomaly detection. Four different approaches are mostly used to deal with outliers: Statistical methods, Distance-based methods, Density-based methods, and Cluster-based methods. We will make a comparative analysis.

4.6.1 Introduction

Anomaly Detection is the problem of finding patterns in data that varies a lot from the most dataset [80]. Additionally, Goldstein and Uchida [81] claimed that detecting abnormal experiments within datasets has always been of great interest in data science. It is essential to develop efficient machine learning algorithms in order to detect anomalies in distributed behavioral databases. The occurrence of data that is distributed across different locations has force the need for anomaly detection techniques in distributed behavioral databases.

The goal of finding outliers from the given dataset is to identify cases that are not usual within data that is mostly homogeneous. We claim that anomaly revealing is extremely important because inconsistencies of data lead to critical information in a wide diversity of application domains. Detecting such deviations from expected behavior in huge dataset is important for ensuring the normal operations of systems across multiple domains such as medicine, ecommerce, computer science and more. Recent studies in this are have been developed in last few decades. For example, there are algorithms based on random forests. According to the research done by Rodin et al. [82], these algorithms were successful in many areas, but due to the weaknesses it is not used now. Recent studies showed that algorithms in deep learning were efficient [83]. However, the use of deep learning for anomaly detection is a research field that is still very unripe. Angermueller et al. (2016) [83] argued that deep learning techniques, in general, require costly training processes. When using dynamic behavior, it is difficult to work with huge training datasets and applying the deep learning techniques successfully.

In order to prevent financial threats to providers and personal threat to clients, it is extremely important to report anomalies in systems. Therefore, we are in need to develop effective machine learning algorithms which were not been previously used in Kazakhstan to reveal outliers in distributed behavioral databases.

4.6.2 Literature review

Research in anomaly detection has become well established in the last decade, whereby significant amount of focus has been given to developing efficient machine learning algorithms. This paper is particularly concerned in the research carried out in behavior based distributed databases. Anomaly detection has been the topic of a number of review articles, surveys, and books. [84] implemented broad article on anomaly detection techniques in machine learning. A deep review of anomaly detection techniques for numeric as well as symbolic data is presented by Agyemang et al. [85]. Patcha and Park [86] and Hofmeyr et. al. [87] present a survey of anomaly detection techniques used specifically for cyberintrusion detection.

A significant amount of research on anomaly revealing was realized in statistics and has been written in several books [88] as well as other survey articles [89]. We can notice that there are a lot of papers were written on outlier identification, however it is surprising that many of them do not obtain the expected result. Supervised machine learning algorithms were used mostly for classification and correlation problems and were succeeded.However it is very important to detect anomalies to build the robust model [83]. Moreover, the anomalous behavior is often dynamic in nature, e.g., new types of anomalies might arise, for which there is no labeled training data. Anomaly detection will be considered as unsupervised machine learning problem. However misdetection may lead to catastrophe in some cases such as aircraft traffic safety. Based on the extent to which the labels are available, anomaly detection techniques can operate in one of the following two modes: Supervised anomaly detection and Unsupervised anomaly detection. In a supervised anomaly detection method we will look for a labeled data, or somehow we need to know which instances are anomalous, however it is not so often in real world problems. Theoretically, supervised methods are believed to obtain better results, however it is very difficult to build model upon given dataset. There are supervised models such as Bayesian method, decision trees, regression, which we can apply directly if we classify the problem as supervised machine learning. On the other side, in unsupervised learning, the dataset do not have the label, and we cannot know whether the given instance is anomalous or not. Of course at first glance we can notice that some data are really far from the others and thus it is outlier, however we need to define an algorithm which will detect this outlier. In other words, they need to build a baseline or foundation of normal behavior. After this step we can build the GOOD model that can detect potentially risky events, without having a priori knowledge of what abnormal behavior looks like, by checking if a new event is dissimilar enough from the definition. Despite the fact that many researchers have significantly contributed to the anomaly detection using machine learning, several important issues remain unsolved.

There is a gap in research of dealing with distributed behavioral databases. In last years, revealing anomalies of behavior based has been gaining broad attention. Hence it is vital to explore new techniques that will efficiently detect the anomalies in behavioral databases. In order to deal with behavioral databases we need to understand what kind of anomalies can be found in this context. Chandola, Banerjee, and Kumar [90] distinguished the simple anomalies from complex anomalies which has two types: contextual and collective. In our case, we will be dealing with contextual anomalies with behavioral attributes. According to Chandola et. al. [90], the behavioral attributes define the non-contextual characteristics of an instance. The notion of a context is induced by the structure in the data set and has to be specified as a part of the problem formulation. Based on the prior research have been made by scholars, we need to use supervised machine learning techniques in order to reveal anomalies in behavioral databases. Additionally, since the databases are distributed, huge dataset is going to be used in our research. Therefore, we will compare different machine learning algorithms in order to use the most efficient one for our behavioral database or develop a new time efficient algorithm.

4.6.3 Methods

Statistical Approach. In this approach, we use classical statistical analysis to determine the probability that this experiment is anomalous. This approach is mostly used in manufacturing to determine whether the object is in working condition or not. This is also the very first choice for data scientists to work with anomalous data. Before doing this analysis, we need to do the preprocessing step, where we determine the most important features in our dataset. Features are selected after learning their correlation and interdependencies. Also, since we are focused on anomalous data, it is better for us to normalize out data. For this, we use Central Limit Theorem from Probability Theory, since it is the best suitable approach. After that we apply statistical approach to determine whether it is an outlier. Statistical approach is one of the well-known algorithms which have been used for outliers identification. Many techniques which can be found in experiments of Barnett and Lewis [80] and Rousseeuw and Leroy [88] are mostly one dimensional. That will cause us problems when our dataset is more than one dimensional because our model will become computationally expensive.

Distance-Based approach. Another important approach is Distance-Based methods. The particular definition was formed by Knorr et al. [92], which they called distance-based (DB) outliers.

The metrics for outliers revealing is Euclidean distance, in other words: An experiment x from a data X is DB(f, D) outlier if the distance of x to X is greater than D. We can use Mahalanobis Distance as well however using the Euclidean Distance make our calculations faster because in Mahalanobis Metrics we use the complex covariance matrices, which will use a lot of time and our model performs slower that is not needed. However, Mahalanobis Metrics can be applied to any dataset, but Euclidean metrics is only available for a dataset with independent features,that is why we cannot rely on Euclidean Distance only because have independent variables is very rare in real

examples. If we ignore that the variables are not independent, we get the inaccurate results in most cases of our observation. Moreover, the given definition above uses f and D which are already given, but it is not usual that we know how far our outliers are located from the majority of data. Again, if we differently choose f and D, we will obtain very different results, consequently, we will misevaluate abnormal data which in other words called anomalies. Because of the problem that we need to calculate all the distances between each pair of points, we will earn a computational problem. Of course this can be solved if we use supercomputers to obtain a faster result. Since we do not have supercomputers in everyday usage the arised question is how to perform the computation.

Density-Based approach. Breunig et al. [93] proposed the alternative way to deal with anomalies, which is called density based approach. In this case we calculate how dense our data points are, and the least dense ones will be specified as outliers. To measure the density we use local outlier factors, which was first presented by Breunig et al. [93]. We say that we use the local outlier factor to measure the density between points. Consequently, if the data has low density it means that it somehow isolated from its neighbors and thus it is local outlier. Firstly, we define how many k-nearest neighbors we need. Using this k value, we measure the local density(LD). LD can be identified as the ratio of the distance between an object and its $k-$ th nearest neighbor and the average distance to $k-$ th nearest neighbor. If we consider a point whoch is far away from its neighbors, and his neighbors are lying on a dense cluster of data which means they have small distances. There are a lot of papers which was written based on the idea of outlying factor and local density. Moreover, many modification were formed while studying the term density based outlier. There are also numereus efficient prunting algorithms which were presented by Jin et al. [94].

Cluster-Based Approach. Another similar appoach to density based outliers is cluster based outlier detection methods. Firstly, lets understand the principle of clusterization. There is one famous clustering algorithm which most data scientists use is k-means algorithm. Basically, we determine the number of clusters, and define their centroids, it happens differently in various algorithms [95]. The Clustering algorithms have the common principle that instances that are far away from the centroid are identified as outliers. Centroid is the center of a cluster, we find it using mean for each feature. The following table shows a Cluster-Based Outlier detection method which Christy et. al. [95] proposed. In the algorithm proposed, data is clustered using a K-Means algorithm and the Euclidean distance of each instance identified. After that the data is sorted is according to their
distance from the centroids of clusters.

Cluster-based approach is very effective from the computational side, but the problem occurs when we need to define the number of clusters. If we will use only one cluster, then this method will be pretty similar to distance based outlier detection method. On the other hand, if we use that each point is a cluster, we will have troubles in

defining outliers. since each cluster is a point and each point is a centroid for each cluster. There are solutions from Deep learning, which we need to apply in the preprocessing step. Huang et al. proposed the alternative way to Deep Learning is Outlier detection algorithm (ROCF) without using top n-factors. ROCF briefly clusters the dataset via constructing Mutual Neighbors Graph, then constructs the Decision Graph. Finally, ROCF detects outliers and outlier clusters though Decision Graphs instead of parameter n or α. The result of this algorithm is 87% accurate.

4.6.4 Conclusion

In order to build a successful machine learning model, we need to put an effort to determine the outliers correctly since it leads to a robust estimation model. In this paper we made an observation for a few approaches that can detect the anomalies in databases. The most significant algorithms from every category are briefly discussed; moreover problems that can be done in future works are also presented in this chapter.

5. PROPOSED MODEL WITH ATTENTION MECHANISM

5.1 Theoretical framework

Attention mechanisms is an approach in machine learning that involves extracting part of the input data (image regions, text fragments) for more detailed processing.

Previously, several models have been proposed that work with text at different levels: symbols, n-grams and words. Boyanovski et al. [96] propose to use a sequence of two recurrent networks: one that works with words, but has a small vocabulary, and a character-by-character model that uses the latent state of the first network as an additional input. Luong et al. [97] propose to generate and read words that are not in the dictionary using a character-by-character network. Mikolov et al. [17] propose to use a small number of the most frequent words, and represent the remaining words as a sequence of syllables. Johansen et al. [98] propose a similar approach: in their model, all words are transformed into a representation vector, which is then fed to the input of the LSTM network. The output sequence in this model is generated character by character.

Often, to solve the problem of image classification, it is not required to process all the pixels of the image: for example, in the classification problem, the background often plays an insignificant role. However, convolutional networks, which are the most popular method for solving this problem, spend the same amount of computational resources on all parts of the image. For example, Spatial Transformer Networks [99] use image transformations (eg affine) to highlight the most important areas. Another approach is the introduction of an agent examining the image [100]. Such an agent at each moment of time processes a small part of the image, and also decides to which position to move the focus of attention. Attention mechanisms can be used to improve the performance of neural networks.

Let us describe the most common approach to implementing attention in neural networks. Initially, a set of vectors $\{h_i\}_{i=1}^{K}$ is selected, over which attention will be carried out. For example, in the problem of generating image headers [90], one can transform the original message using several convolutional layers and use a set of output vectors for each pixel as an object of attention. At the second step, a part of the network generates a key *k*, which will be responsible for which vectors from $\{h_i\}_{i=1}^{K}$ will be used. Using the key, each vector h_i is assigned a weight w_i, which can be interpreted as the probability that attention should be focused on the object h_i.
Most often, these weights are obtained by the formula:

$$w_i = \exp(d(k, h_i))/(\sum_{j=1}^{K} \exp(d(k, h_i))) \quad (5.1)$$

As a measure of proximity *d(k, h)*, the cosine coefficient is usually used: $d(k,h) = \frac{k^T h}{||k|| \cdot ||h||}$. Thus, attention is focused mainly on objects similar to keys. For example, if a vector responsible for the texture of the bark is used as a key, then the parts of the image that show the bark of trees will have the greatest weight.

At the next step, the context vector $\hat{h}$ is calculated - a generalized representation of attention objects, taking into account the key. There are two approaches to its assignment: hard attention and soft attention. With strict attention, $\hat{h}$ is taken to be h_ξ, where $\xi \sim Discrete(w_1, w_2, ...w_K)$. When training the model, a Monte Carlo gradient estimate is used: such an estimate is unbiased, but in practice it has a large variance. Because of this, hard attention models have to use different variance reduction methods such as REINFORCE [91]. With soft attention, $\hat{h}$ is calculated as $\mathbb{E}_{i\sim\xi} = \sum_{i=1}^{K} w_i h_i$. This approach allows the model to be trained deterministically end-to-end. Depending on the weights of attention, information can be either aggregated from the most important parts of the data, or from only one object. Today, it is the soft attention mechanism that is more often used, therefore, only it will be considered below [97] [103] [104].

Attention mechanisms are also commonly used in neural network machine translation. The problem with conventional Seq2Seq models (see section 2.2) is the need to compress all the information in the presentation vector. This problem becomes especially significant when translating long sequences. It has been shown [103] [104] that Seq2Seq with attention significantly improves the performance on long sequences. As an object of attention in such models, the outputs of the last layer of the coding part for each word are used. The output of the last layer of the decoding part is selected as a key. To generate words, the context vector is concatenated with the key and passed through another recurrent layer.

An important application of attention mechanisms is found in the creation of a neurocomputer [105] [106] - a neural network that simulates the structure of ordinary computers. Their distinctive feature is that they have memory and a controller that allows read and write operations to it. Also, attention is used in differentiated versions of data structures such as list, stack, deque, queue [107] [108].

5.2 Types of attention mechanisms

The attention mechanism (attention model) is a technique used in recurrent neural networks (abbreviated RNN) and convolutional neural networks (abbreviated CNN) to find relationships between different parts of input and output data.

Initially, the attention mechanism was presented in the context of recurrent Seq2seq networks for "drawing attention" of decoder blocks to hidden states of RNN for any encoder iteration, not just the last one.

The success of this technique in machine translation has been followed by its implementation in other natural language processing problems and applied to CNN to generate image descriptions [99] and generative adversarial networks [104] (abbreviated GAN).

5.2.1 Generalized attention mechanism

Generalized attention mechanism is a type of attention mechanism, the task of which is to identify patterns between input and output data. Initially, the attention mechanism presented in the original article [106] implied exactly this type of attention.

We will provide an example of using the generalized attention mechanism for a machine translation task.

Seq2seq Basic Architecture

Understanding the attention mechanism in Seq2seq networks requires a basic understanding of the Seq2seq architecture before introducing the attention mechanism.

Seq2seq consists of two RNNs - Encoder and Decoder visualized in Figure 5.1.

Encoder - accepts a sentence in language A and compresses it into a latent state vector.

Decoder - outputs a word in B language, takes the last hidden state of the encoder and the previous predicted word.

Let's consider an example of the Seq2seq network:

- x_i - words in a sentence in language A.
- h_i - hidden state of the encoder.

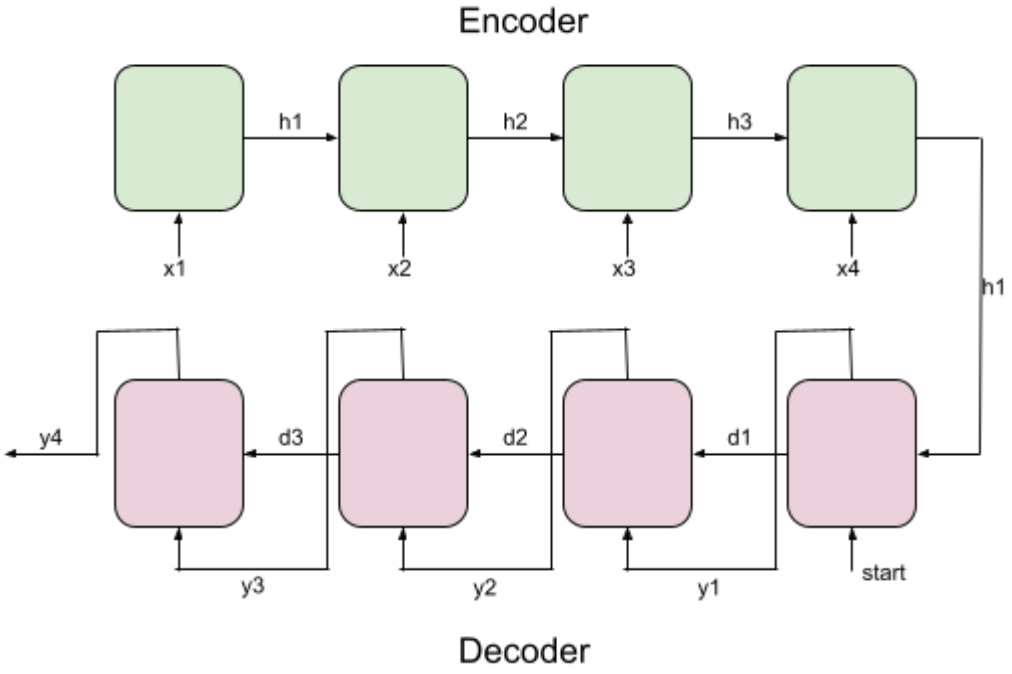

Figure 5.1: Basic principle of Seq2Seq model

Encoder blocks (green) - encoder blocks receiving x_i at the input and transmitting the hidden state hi to the next iteration.

- d_i - hidden state of the decoder.
- y_i - words in a sentence in language B.

Decoder blocks (purple) - decoder blocks that receive as input y_{i-1} or a special token start in the case of the first iteration and return y_i - words in a sentence in B. Send d_i - the hidden state of the decoder to the next iteration. The translation is considered complete when y_i is equal to the special token end.

Using attention mechanism for Seq2seq models

Despite the fact that neural networks are viewed as a "black box" and it is often impossible to interpret their insides in human-understandable terms, nevertheless, an intuitive attention mechanism for humans was able to improve the quality of machine translation of the basic Seq2seq algorithm.

The success of using this approach in a machine translation task is due to the best deduction of patterns between words located at a great distance from each other. Despite the fact that LSTM and GRU blocks are used precisely to improve the transmission of information from previous RNN iterations, their main problem is that the influence of

previous states on the current one decreases exponentially from the distance between words, at the same time, the attention mechanism improves this indicator to linear [107].

RNNs are used when processing data for which consistency is important. In the classic case of using RNN, the result is only the last hidden state h_m, where m is the length of the input data sequence. Using the attention mechanism allows using information obtained not only from the last hidden state, but also from any hidden state h_t for any t.

Attention mechanism layer structure

Generalized attention mechanism in RNN is shown in Figure 5.2.

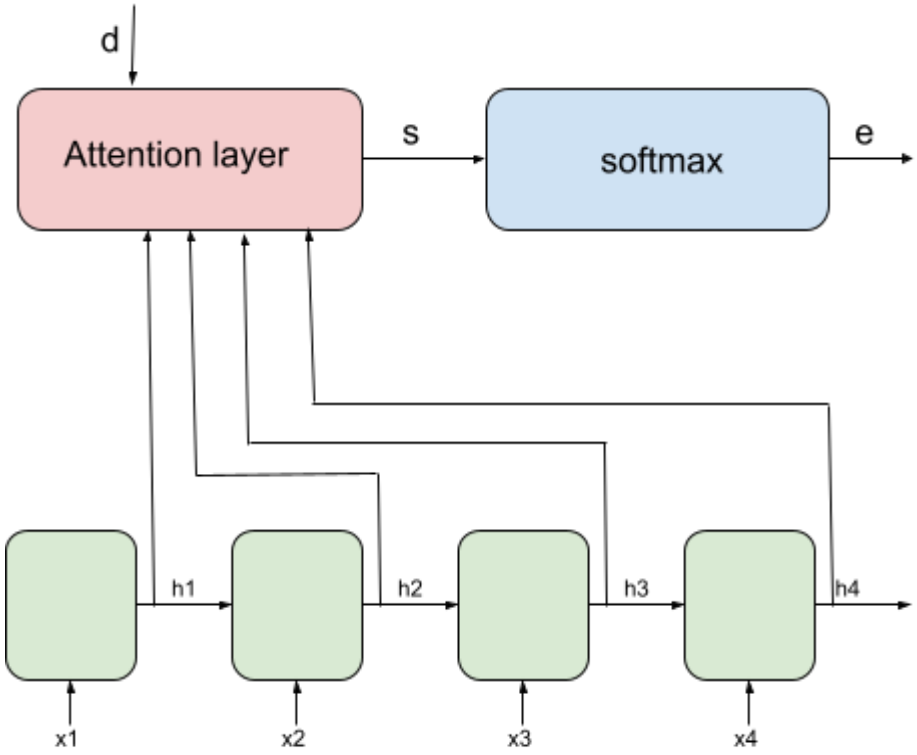

Figure 5.2: Generalized attention mechanism

The layer of the attention mechanism is an ordinary, most often single-layer, neural network at the input of which $h_t, t = 1 \dots m$ are fed, as well as a vector d, which contains a certain context depending on a specific task.

In the case of Seq2seq networks, the vector d will be the latent state d_{i-1} of the previous decoder iteration.

The output of this layer will be the vector s (score) - the estimates based on which the hidden state h_i will be "paid attention".

Then softmax is used to normalize *s* values [7]. Then *e = softmax (s)*.
Softmax is used here due to its properties:

- $\forall s$: $\Sigma^{n}_{i=1}\ softmax(s)_i = 1$
- $\forall s,\ i$: $softmax(s)_i >= 0$

Further, c (context vector) is considered

$$c = \Sigma^{m}_{i=1} e_i h_i$$

The result of the work of the layer of attention is c, which contains information about all hidden states h_i in proportion to the assessment of e

Applying Attention Mechanism to Basic Seq2seq Architecture

When you add a mechanism to this architecture between the RNN Encoder and the Decoder of the attention mechanism layer, you get the following scheme:

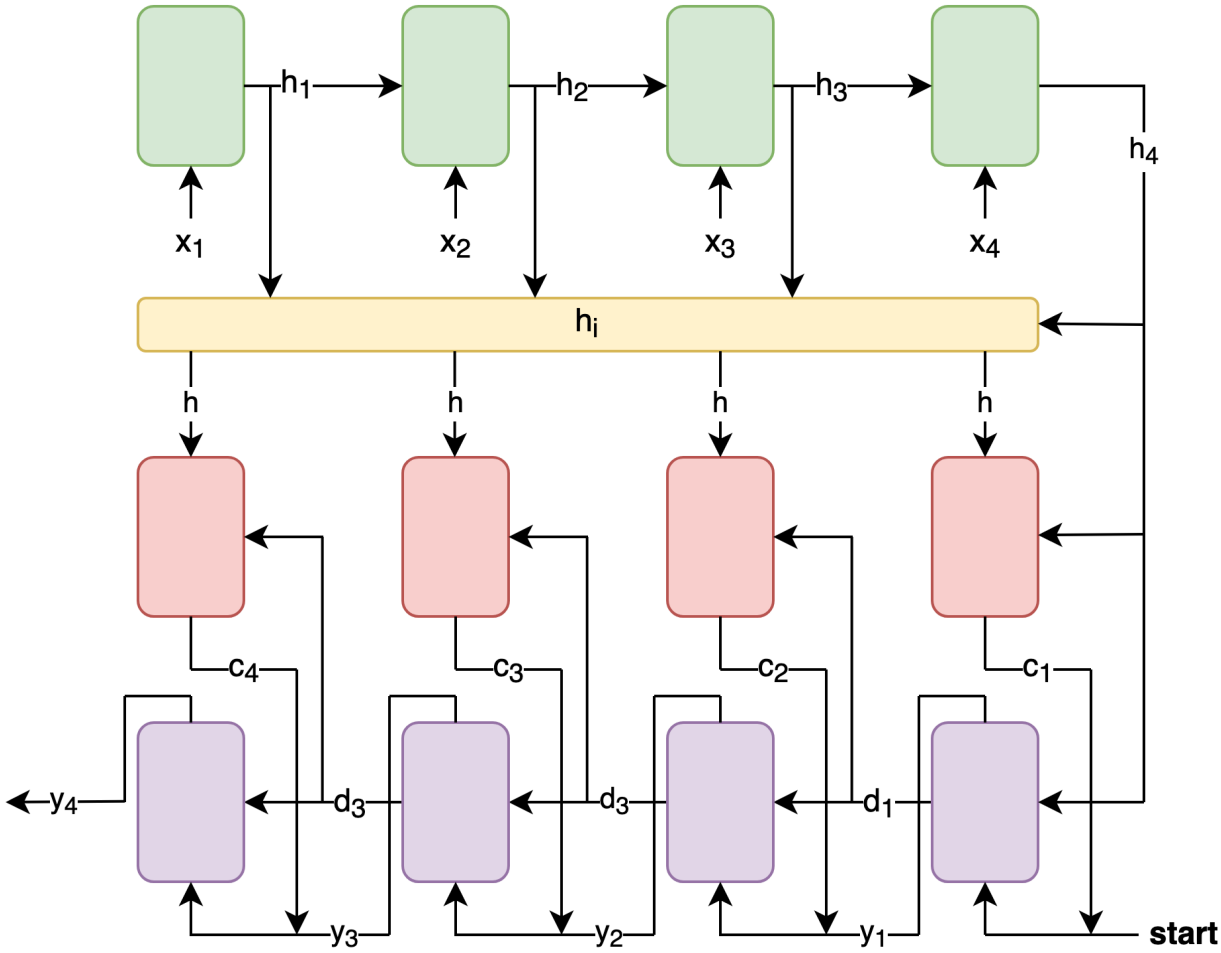

Figure 5.3: Sequence to sequence attention mechanism

Here xi, hi, di, yi have the same purposes as in the variant without attention mechanism.

Encoder hidden states aggregator (yellow) - aggregates all the hi vectors in itself and returns the entire sequence of vectors h

ci is the context vector at iteration i.

Decoder blocks (purple) - input data changes compared to normal Seq2seq network. Now, at iteration i, the input is not yi − 1, but the concatenation of yi − 1 and ci.

Thus, with the help of the attention mechanism, the decoder "focuses" on certain latent states. In machine translation cases, this feature helps the decoder predict which hidden states for certain source words in language A to pay more attention to when translating a given word into language B. That is, which words from the source text to pay attention to when translating a specific word into the target language.

Self-attention

Self-Attention is one type of attention mechanism, the task of which is to identify patterns only between input data.

This technique has shown itself to be so effective in the task of machine translation that it made it possible to abandon the use of RNNs and replace them with conventional neural networks in combination with the Self-attention mechanism in the transformer architecture [108].

This made it possible to speed up the work of the algorithm, since previously the proposal was processed sequentially using RNN. When using a transformer, each word in the sentence being processed can be processed in parallel.

The main difference between Self-Attention and the generalized attention mechanism is that it draws conclusions about dependencies exclusively between input data.

Consider the sentence The animal didn't cross the street because it was too tired and the result of the Self-attention algorithm for the word it. The resulting vector corresponds to the relationship of the word it with all other words in the sentence.

From the visualization of the vector, it can be seen that the Self-attention mechanism found a relationship between the words it and animal. This result can be intuitively explained from a human point of view, which allows machine learning algorithms using this approach to better solve the problem taking into account contextual relationships.

Self-Attention is also successfully used in GAN networks, in particular in the SAGAN algorithm [104].

5.3 Proposed model with attention layer

Generating text is one of the challenges that can be accomplished using deep learning models. This work presents a neural network model for generating natural language text using a recurrent neural network (RNN) and attention mechanism.

Based on an abstract model, a neural network was developed. The vector model of the language is used as a coder, which allows moving towards greater "meaningfulness" of the model and "understanding" of the meaning of words. A recurrent neural network acts as a decoder, since it allows processing information cyclically as it moves from input to output, and the output depends on previous calculations, providing a "memory" effect. The network architecture has three layers: the first layer is the attention mechanism, and two layers in each the input word layer, the projection layer, the recurrent layer, and the softmax layer.

Adding an attention mechanism helps the decoder make better use of the input text information. The mechanism at each generation step determines the so-called attention distribution (the set of weights corresponding to the elements of the original sequence, the sum of the weights is 1, all weights> = 0), and from it it receives the

weighted sum of all hidden states of the encoder, thereby forming a context vector. This vector is concatenated with the embedding of the decoder input word at the stage of calculating the latent state and with the hidden state itself at the stage of determining the next word. So at each step of inference, the model can determine which states of the encoder are most important to it at the moment. In other words, it decides which input words context should be considered the most.

Proposed architecture:

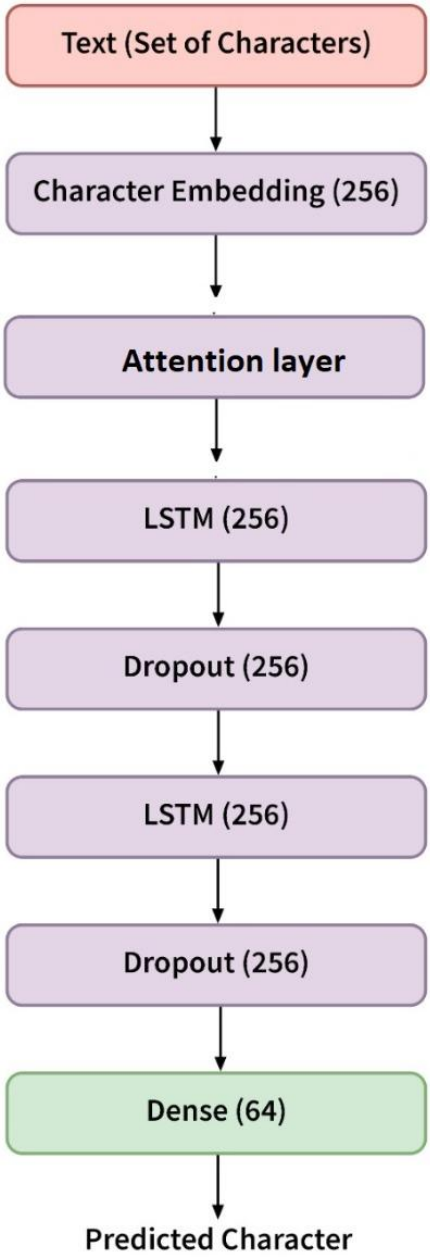

Figure 5.4: Proposed architecture with attention layer

Results:

To obtain the results, random parts of the dataset were taken, on which the network was trained. The data was fed into the network as the initial context, with the direct generation of the text, the previously generated words acted as the context. The network generation results are presented in the table 5.1.

Table 5.1: Results

Input to NN	**Predicted output**
" деп сәлем айтты. бірақ өз топшылауын білдірмеді. осы кезде көрші руларға бірнеше кісіні жай ғана жансыз етіп аттандырған. ол кісілерге тапсыратын сөздерін де бір өзі ғана білді. осындай бұйрықтармен жіберген адамдары да оқшау кісілер. ойда жоқ жандар. мы	нау тон артынан басын көріп келе жатқан жақағы қара қара қатындардың басына қарай тартты.
ік, – деп киіп кетіп ұрыса сөйледі, – албасты да қабаққа қарай басатын. қыры жоқ, қасиеті жоқ басшы болса ыбылыс, жын иектемей нетеді. адал десек, аман десек, жан берейік, ақтайық, ақыретте айыбын өз мойнымызға алайық. бірақ, меңің екі бірдей жаным жоқ. ма	йыр да барып қалдырмай алған соң, байдалы бар ма? ала бір қартаң қараң шығарып, қара бер қайтеді? – деп, қараша қара қара атыр ала берді.
ғады!– деді. бұл желқұйын туралы өз аулындағы бір үлкен аңшының айтқан сөзі еді. жиренше осыны үнемі айта жүретін. абайды сол сөзбен желқұйынның жаланып тұрған түрі қатты қызықтырды. – қоянға шығамысың?- деді. – жүр, атың бар ма? мен сол қоянға шығып барам	ыз? бұл жақын айтқанын айтып берді. бірақ құнанбай ауылдары байдалы бар қара болса, қара қара атыр салып кетті.
а да абай мен ербол бөгелген жоқ. ертең ерте жүреміз деп, ауыл иелеріне	

Table 5.1 continue

рақмет айтып, асқа да қарамай кетіп қалды. келесі күні айтқан сөздерін ақтағандай боп, екі жігіт бақанасқа кетті. қаратайдың аулына барып түстеніп, кеш батқанша сонда болды да, сол күн	дер бастас байлау еді. бірақ бұл жақын айтқан жоқ. қараша қара қара атылдары байдалы бар қара қатты айтып келеді.
жігітек ішіндегі үркімбай аулына түсті. қыстауының жанында отырған алты үйдің барлық иті абалап шығып еді, атшабарлар ақырып, қамшы үйіріп ұмтылып, бездіріп жіберді. әрбір үйдің есігін жамылып, баспағып тұрған балалар да мына тентек қонақтардан қорқып, ін	дерей болды. бірақ құнанбай аулына қарай тартты. байдалы айтқан жоқ.
сөздерін ұстап қалады. әлдекімге қаптап, зіркілдеп сөйлеп отырған әке сөзі кейде бұған бір жортуыл, шабуыл үстіндегі шұбырынды, ұзақ сарын сияқтанады. кейде ұғымсыз сөзден іші пысып, әкесінің пішін тұлғасына қарап, қадалып қалады. тегінде, ертекші, өлеңші	екен. бұл жақында қара қалың қой қойып қалды. қараша қара қара атылдары байдалы айтқан сайын тартып кетті. абай бұл жақында қалған жақып тартылдап кеп, қара қатын айтып келеді.
е жол болатын. абайға ауылдан шығар жерде зере, айғыз сондай киімдерді әзірлетіп ұсынған-ды. кәрі әже әшейінде абайдың дегеніне көне берсе де, бұл тұста ырық бермей, қатты бұйрық еткен: – ата-бабаң жолы осы! барған елің сені кінәламайды, "әкесі күйеу, шеше	сі қарап қалдырмаса, бір қара баласы бар екен. бұл жақын айтқан қалың қойы бар. алдында қара қара атылдар де айтқан салып кетті.
– бәсе, одан да соныңды айтсаңшы! "үлкенмін, ақылым бар, кісіге де ақыл айтам" дейсің! бөрінің артынан бөлтірік ақылды болғандықтан ереді дейсің ғой.	

Table 5.1 continue:

– өй, сен не сандалып келесің осы? үйдегі алжыған әжеңнің ақылынан басқа ақыл жоқ деп алғансың ғой өзің ті	йінді. – деп байлаған екен. бұл жақында байдалы бар қара қатты айтқан жоқ.
ай қарап отырып сөйледі. құнанбай басында мұны аса салқын тыңдады. тек "жігітекті жаныштаудан тоқтамады ғой!", "босаспады ғой!" деген сияқты байдалы дауына жеткен жерде ғана қабағын лезде қатты түйіп ап, абайға тіксіне қарады. баласының өзін барлап: "осыны	ң басына келген жас жағын айтылды бар екен айтарын болса, алған айтқаның бар салын болмаса, мен қарағым, – деп байлаған екен.

As we can see, the generated text coincides with the sentence structure of Kazakh language. The beginning of the output is allocated with the end of the input sentence. Generated words are 99% correct:

Number of words: 592

Number of characters: 3749

Incorrect words: 30

Valid words: 95%

New words/unseen: 26 - 4%

The loss function can be seen from the figure below:

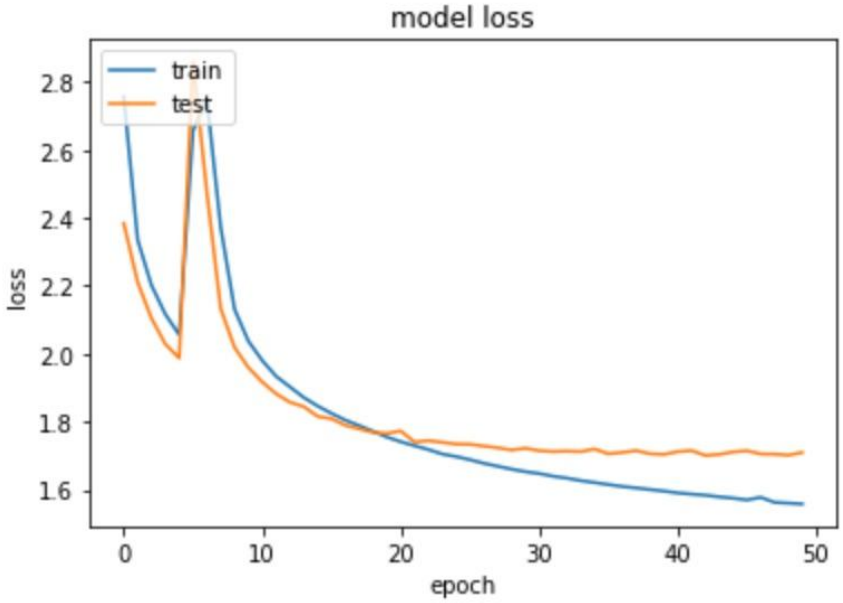

Figure 5.5: Loss function with attention layer

Conclusion

The developed model can preserve the logic of the narrative and build dialogues on relatively short texts (3-4 sentences long), but it lacks a global context and preservation of the structure of the narrative, as in real works of art.

Therefore, further work will be aimed at improving the model in favor of a global understanding of the "meaning" of what it produces.

CONCLUSION

The task of language modeling in the narrow sense is to predict the next word (characters) in the text by looking at the previous words.

The staged work and analysis were performed towards Language Modeling for Kazakh. First of all, Neural network is indispensable to work with sparse tagged data and have meaningful results. Thus, character based neural model seems to deal with limitedness and word based will keep long dependency relations in context. Therefore, mentioned goal will be achieved by stagely implementing and analysing the proposed models into a language. Moreover results shed light on future tasks to emerge the type of words in a context by adding word type information also. This can be very useful in agglutinative languages like Kazakh language where meaning and structure of words very dependent on word endings. Language modeling is the most suitable training ground for applying RNN. The construction of a language model relates to the tasks of learning without a teacher. At the moment, in the scientific literature there is a large amount of information on this topic. One of the main authorities in the field of deep learning, Lecun [2] calls this predictive or predictive learning and considers it a prerequisite for the acquisition of common sense by neural networks.

In this work, firstly, character-based model was built using LSTM. It was successfully implemented with some weaknesses. The model was extended by adding an attention layer, which increased the accuracy of the model. Besides generating correct text, the model generates the words that it did not see during the training process. It concludes that the model learned the Kazakh morphology and able to generate words that are correctly used in the context.

The novelty of the work is that an innovative language model has been built. The substantive novelty differs from the previous models based on the application of the neural networks using the graphical processing unit that makes the computation more efficient.

According to the proposed results, the work contributes to the state of the art of Kazakh NLP in general. The goals that were planned are achieved: the current state of the art of language models for different languages was analyzed; architecture of recurrent neural model was developed and proposed; methods and algorithms for character-based language modeling using recurrent neural networks were realized; the choice of optimization models for the text generation models were justified; the performance of the developed model with the state of the art was compared.

The given work is an important contribution to the Kazakh Natural Language Processing research area and can be integrated and collaborated with other systems such as optical text recognition, speech generation, and machine translation.

REFERENCES

1 Kessikbayeva G., Sultanova N., Amangeldi Y., Yurchenko R. (2021) Algorithms Towards the Automated Customer Inquiry Classification. In: Yalaoui F., Amodeo L., Talbi EG. (eds) Heuristics for Optimization and Learning. Studies in Computational Intelligence, vol 906. Springer, Cham. https://doi.org/10.1007/978-3-030-58930-1_14

2 Shoiynbek A., Kozhakhmet K., Sultanova N., Zhumaliyeva R. (2019) The Robust Spectral Audio Features for Speech Emotion Recognition. Applied Mathematics and Information Sciences. ISSN 1935-0090 (print) ISSN 2325-0399 (online).Volume 13, pp 867-870.

3 Kozhakhmet K., Zhumaliyeva R., Shoiynbek A., Sultanova N. (2020) Speech Emotion Recognition For Kazakh And Russian Languages. Applied Mathematics and Information Sciences. ISSN 1935-0090 (print) ISSN 2325-0399 (online).Volume 14, pp 65-68.

4 Sultanova N., Jantayev R. Outlier detection methods using machine learning algorithms in distributed databases. "Bulletin of KazNRTU" of the National Research Technical University named after K.I. Satpayev. ISSN: 1680 - 9211 No. 4 (128) 195-199p

5 Kozhakhmet K., Sultanova N., Botbayeva A., Kapayeva M. Sentiment analysis of russian texts. "Bulletin of KazNRTU" of the National Research Technical University named after K.I. Satpayev. ISSN: 1680 - 9211 No. 5 (135) 445-449p

6 Issabek A., Suliyev R., Kessikbayeva G., Sultanova N., Bogdanchikov A., Orynbekova K. Modeling of critical letter transmissions in Kazakh alphabet according to big data analysis. "Bulletin of KazNRTU" of the National Research Technical University named after K.I. Satpayev. ISSN: 1680 - 9211 No. 6 (136) 234-236p

7 Yerlanuly A., Sultanova N., Kozhakhmet K. Methods for summarization of the Kazakh text. "Bulletin of KazNRTU" of the National Research Technical University named after K.I. Satpayev. ISSN: 1680 - 9211 # 4 (140)

8 Kozhakhmet K., Sultanova N. Methods for Building Neural Language Models for Agglutinative Languages. Materiály XV Mezinárodní vědecko -praktická konference «Aktuální vymoženosti vědy -2019», Volume 8 : Praha. Publishing House «Education and Science» ISBN 978-966-8736-05-6

9 Sultanova N., Kozhakhmet K., Jantayev R., Botbayeva A. Stemming algorithm for Kazakh Language using rule-based approach. 2019 15th International Conference on Electronics, Computer and Computation (ICECCO), Abuja, Nigeria, 2019, pp. 1-4, doi:10.1109/ICECCO48375.2019.9043253.

10 Sultanova N., Kessikbayeva G., Amangeldi Y. Kazakh Language Open Vocabulary Language Model with Deep Neural Networks. 2019 15th International Conference on Electronics, Computer and Computation (ICECCO), Abuja, Nigeria, 2019, pp. 1-4, doi:10.1109/ICECCO48375.2019.9043253.

11 Meraliyev B., Kongratbayev K., Sultanova N. Content Analysis of Extracted Suicide Texts From Social Media Networks by Using Natural Language Processing and Machine Learning Techniques, 2021 IEEE International Conference on Smart Information Systems and Technologies (SIST), 2021, pp. 1-6, doi: 10.1109/SIST50301.2021.9466001.

12 R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. P. Kuksa, “Natural language processing (almost) from scratch”, CoRR, vol. abs/1103.0398, 2011.

13 Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning”, English (US), Nature, vol. 521, no. 7553, pp. 436–444, May 2015, issn: 0028-0836. doi: 10.1038/nature14539.

14 P. B, S. Kp, and M. Kumar, “A deep learning approach for malayalam morphological analysis at character level”, Procedia Computer Science, vol. 132, pp. 47–54, Jan. 2018. doi: 10.1016/j.procs.2018.05.058.

15 C. Dyer, J. Weese, H. Setiawan, A. Lopez, F. Ture, V. Eidelman, J. Ganitkevitch, P. Blunsom, and P. Resnik, “Cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models”, in Proceedings of the ACL 2010 System Demonstrations, ser. ACLDemos ’10, Uppsala, Sweden: Association for Computational Linguistics, 2010, pp. 7–12. [Online]. Available: http://dl.acm.org/citation.cfm?id=1858933.1858935.

16 M. Kang, T. Ng, and L. Nguyen, “Mandarin word-character hybrid-input neural network language model”, in INTERSPEECH 2011, 12th Annual Conference of the International Speech Communication Association, Florence, Italy, August 27-31, 2011, 2011, pp. 625–628. [Online]. Available: http://www.isca-speech.org/archive/interspeech%5C_2011/i11%5C_0625.html.

17 T. Mikolov, I. Sutskever, A. Deoras, L. Hai Son, S. Kombrink, and J. Cernock, “Subword language modeling with neural networks”, Mar. 2019.

18 Y. Miyamoto and K. Cho, “Gated word-character recurrent language model”, CoRR, vol. abs/1606.01700, 2016. arXiv: 1606.01700. [Online]. Available: http://arxiv.org/abs/1606.01700.

19 W. Ling, T. Luıs, L. Marujo, R. F. Astudillo, S. Amir, C. Dyer, A. W. Black, and I. Trancoso, “Finding function in form: Compositional character models for open vocabulary word representation”, CoRR, vol. abs/1508.02092015. arXiv: 1508.02096. [Online]. Available: http://arxiv.org/abs/1508.02096.

20 K. Hwang and W. Sung, “Character-level language modeling with hierarchical recurrent neural networks”, CoRR, vol. abs/1609.03777, 2016. arXiv: 1609.03777. [Online]. Available: http://arxiv.org/abs/1609.03777.

21 G. Kessikbayeva and I. Cicekli, “A rule based morphological analyzer and a morphological disambiguator for kazakh language”, 2016.

22 D. Yuret and F. Türe, “Learning morphological disambiguation rules for turkish”, in Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational

Linguistics, ser. HLT-NAACL ’06, New York, NewYork: Association for Computational Linguistics, 2006, pp. 328–334. doi: 10.3115/1220835.1220877. [Online]. Available: https://doi.org/10.3115/1220835.1220877.

23 M. Ali Yatbaz and D. Yuret, “Unsupervised morphological disambiguation using statistical language models”, Mar. 2019.

24 R. Smith, “Limits on the application of frequency-based language models to ocr”, in ICDAR, Won Best Industrial Paper Award, 2011, pp. 538–542.

25 T. Brants, A. C. Popat, P. Xu, F. J. Och, and J. Dean, “Large language models in machine translation”, in Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), 2007, pp. 858–867.

26 F. Jelinek, “Self-organized language modeling for speech recognition”, in Readings in Speech Recognition, A. Waibel and K.-F. Lee, Eds., Originally distributed as IBM technical report in 1985, Los Altos: Morgan Kaufmann, 1990, pp. 450–506.

27 S. Abney, “Statistical methods”, in Encyclopedia of Cognitive Science. American Cancer Society, 2006, isbn: 9780470018866. doi: 10.1002 /0470018860.s00283. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/0470018860.s00283. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/0470018860.s00283.

28 C. M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics). Berlin, Heidelberg: Springer-Verlag, 2006, isbn: 0387310738.

29 J. L. Elman, “Finding structure in time”, Cognitive Science, vol. 14, no. 2, pp. 179–211, 1990, issn: 0364-0213. doi: https://doi .org/10.1016/0364-0213(90) 90002 - E. [Online]. Available: http://www.sciencedirect.com/science/article/pii/036402139090002E.

30 H. Cruse, “Neural networks as cybernetic systems - part ii”, Brains, Minds, and Media, no. 1, 2006.

31 J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities”, Proceedings of the National Academy of Sciences, vol. 79, no. 8, pp. 2554–2558, 1982, issn: 0027-8424. Doi: 10.1073/pnas.79.8.2554. eprint: https://www.pnas.org/content/79/8/2554.full.pdf. [Online]. Available: https://www.pnas.org/content/79/8/2554.

32 R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, How to construct deep recurrent neural networks, 2013. arXiv: 1312.6026 [cs.NE].

33 A. Timmaraju and V. Khanna, “Sentiment analysis on movie reviews using recursive and recurrent neural network architectures”, 2015.

34 R. J. Williams and J. Peng, “An efficient gradient-based algorithm for on-line training of recurrent network trajectories”, Neural Computation, vol. 2, pp. 490–501, 1990.

35 S. Hochreiter and J. Schmidhuber, “Long short-term memory”, Neural Comput., vol. 9, no. 8, pp. 1735–1780, Nov. 1997, issn: 0899-7667. Doi:

10.1162/neco.1997.9.8.1735. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

36 A. Matthews, G. Neubig, and C. Dyer, “Using morphological knowledge in open-vocabulary neural language models”, in NAACL-HLT, Association for Computational Linguistics, 2018, pp. 1435–1445.

37 O. Makhambetov, A. Makazhanov, I. Sabyrgaliyev, and Z. Yessenbayev, “Data-driven morphological analysis and disambiguation for kazakh”, in Computational Linguistics and Intelligent Text Processing, A. Gelbukh, Ed., Cham: Springer International Publishing, 2015, pp. 151–163, isbn:978-3-319-18111-0.

38 N. Bölücü and B. Can, “Joint pos tagging and stemming for agglutinative languages”, CoRR, vol. abs/1705.08942, 2017. arXiv: 1705.08942. [Online]. Available: http://arxiv.org/abs/1705.08942.

39 G. Bekmanova, A. Sharipbay, G. Altenbek, E. Adali, L. Zhetkenbay, U. Kamanur, and A. Zulkhazhav, “A uniform morphological analyzer for the kazakh and turkish languages”, in AIST, 2017.

40 J. Washington, I. Salimzyanov, and F. Tyers, “Finite-state morphological transducers for three kypchak languages”, in Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC’14), N. C. (Chair), K. Choukri, T. Declerck, H. Loftsson, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, and S. Piperidis, Eds., Reykjavik, Iceland:European Language Resources Association (ELRA), May 2014, isbn: 978-2-9517408-8-4.

41 E. Yildiz, C. Tirkaz, H. Sahin, M. Eren, and O. Sonmez, A morphology-aware network for morphological disambiguation, 2016. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12370/12034.

42 A. Toleu, G. Tolegen, and A. Makazhanov, “Character-aware neural morphological disambiguation”, in Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers),Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 666–671. doi:10.18653/v1/P17-2105.

43 G. A. Miller, “Wordnet: A lexical database for english”, Commun. ACM, vol. 38, no. 11, pp. 39–41, Nov. 1995, issn: 0001-0782. doi: 10.1145/219717.219748. [Online]. Available: http://doi.acm.org/10.1145/219717.219748.

44 E. Loper and S. Bird, “Nltk: The natural language toolkit”, in Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1, ser. ETMTNLP ’02, Philadelphia, Pennsylvania: Association for Computational Linguistics, 2002, pp. 63–70. doi: 10.3115/1118108.1118117. [Online]. Available: https://doi.org/10.3115/1118108.1118117.

45 Z. Assylbekov, F. Tyers, A. Nurkas, A. Sundetova, A. Karibayeva, B. Abduali, D. Amirova, and J. Washington, “A free/open-source hybrid morphological disambiguation tool for kazakh”, Apr. 2016. doi: 10.13140/RG.2.2.12467.43045.

46 A. Solak and F. Can, “Effects of stemming on turkish text retrieval”, in Proceedings of the Ninth Int. Symp. on Computer and Information Sciences (ISCIS’94), Ankara, Turkey, 1994, pp. 49–56.

47 T. Kisla and B. Karaoğlan, “A hybrid statistical approach to stemming in turkish: An agglutinative language”, Anadolu University Journal of Science and Technology-A Applied Sciences and Engineering, vol. 17, Jul. 2016. doi: 10.18038/btda.31812.

48 B. Dincer and B. Karaoğlan, “Stemming in agglutinative languages: A probabilistic stemmer for turkish”, vol. 2869, Nov. 2003, pp. 244–251. doi: 10.1007/978-3-540-39737-3_31.

49 V. Barakhnin, A. Bakiyeva, and T. Batura, “Stemming and word forms generation in automatic text processing systems in the kazakh language”, Computational technologies, vol. 22, pp. 11–21, 2017.

50 M. Porter, “An algorithm for suffix stripping”, Program: electronic library and information systems, vol. 40, pp. 211–218, 2006. doi: 10.1108/00330330610681286.

51 V. Gurusamy and S. Kannan, “Performance analysis: Stemming algorithm for the english language”, International Journal for Scientific Research and Development, vol. 5, pp. 2321–613, Aug. 2017.

52 S. Bekturov, Қазақ тілі: фонетика, грамматика, морфология, синтаксис , The Science of Microfabrication. 2006.

53 A. Fedotov, J. Tussupov, M. Sambetbayeva, I. Idrisova, and A. Yerimbetova, “Development and implementation of a morphological model of kazakh language”, Eurasian Journal of Mathematical and Computer Applications, vol. 3, no. 3, pp. 69–79, 2015.

54 A. M. Borodkin, E. Lisin, and W. Strielkowski, “Data algorithms for processing and analysis of unstructured text documents”, Applied mathematical sciences, vol. 8, pp. 1213–1222, 2014.

55 M. Mowafy, A. Rezk, and H. El-Bakry, “An efficient classification model for unstructured text document”, American Journal of Computer Science and Information Technology, vol. 06, Jan. 2018. doi: 10.21767/23493917.100016.

56 A. Ittoo, L. M. Nguyen, and A. [den Bosch], “Text analytics in industry: Challenges, desiderata and trends”, Computers in Industry, vol. 78, pp. 96–107, 2016, Natural Language Processing and Text Analytics in Industry, issn: 0166-3615. doi: https://doi.org/10.1016/j.compind.2015.12.001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0166361515300646.

57 A. K. S. Tilve, “Text classification using naïve bayes, vsm and pos tagger”, 2017.

58 L. Breiman, “Machine learning, volume 45, number 1 - springerlink”, Machine Learning, vol. 45, pp. 5–32, Oct. 2001. doi: 10.1023/A:1010933404324.

59 T. Joachims, “Text categorization with support vector machines: Learning with many relevant features”, in Machine Learning: ECML-98, C. Nédellec and C.

Rouveirol, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 137–142, isbn: 978-3-540-69781-7.

60 G. I. Webb, "Naïve bayes", in Encyclopedia of Machine Learning, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, 2010, pp. 713–714, isbn: 978-0-387-30164-8. doi: 10.1007/978-0-387-30164-8_576. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_576.

61 D. G. Kleinbaum, "Introduction to logistic regression", in Logistic Regression: A Self-Learning Text. New York, NY: Springer New York, 1994, pp. 1–38, isbn: 978-1-4757-4108-7. doi: 10.1007/978-1-4757-4108-7_1. [Online]. Available: https://doi.org/10.1007/978-1-4757-4108-7_1.

62 T. Hastie, R. Tibshirani, and J. Friedman, "The elements of statistical learning", Aug, Springer, vol. 1, Jan. 2001. doi: 10.1007/978-0-387-21606-5_7.

63 M. Korobov, "Morphological analyzer and generator for russian and ukrainian languages", in Analysis of Images, Social Networks and Texts, M. Y. Khachay, N. Konstantinova, A. Panchenko, D. Ignatov, and V. G. Labunets, Eds., Cham: Springer International Publishing, 2015, pp. 320–332, isbn: 978-3-319-26123-2.

64 A. Hassan and A. Mahmood, "Deep learning for sentence classification", 2017 IEEE Long Island Systems, Applications and Technology Conference (LISAT), pp. 1–5, 2017.

65 K. Kowsari, D. E. Brown, M. Heidarysafa, K. Jafari Meimandi, M. S. Gerber, and L. E. Barnes, "Hdltex: Hierarchical deep learning for text classification", 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Dec. 2017. doi: 10.1109/icmla.2017.0-134. [Online]. Available: http://dx.doi.org/10.1109/ICMLA.2017.0-134.

66 M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez, and K. Kochut, Text summarization techniques: A brief survey, 2017. arXiv: 1707.02268 [cs.CL].

67 J. M. Conroy and D. P. O'leary, "Text summarization via hidden markov models", in Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ser. SIGIR '01, New Orleans, Louisiana, USA: Association for Computing Machinery, 2001, pp. 406–407, isbn: 1581133316. doi:10.1145/38395.2384042. [Online]. Available: https://doi.org/10.1145/383952.384042.

68 S. Miranda-Jiménez, A. Gelbukh, and G. Sidorov, "Conceptual graphs as framework for summarizing short texts", Int. J. Concept. Struct. Smart Appl., vol. 2, no. 2, pp. 55–75, Jul. 2014, issn: 2166-7292. doi: 10.4018/IJCSSA.2014070104. [Online]. Available: https://doi.org/10.4018/IJCSSA.2014070104.

69 M. Binwahlan, N. Salim, and L. Suanmali, "Fuzzy swarm diversity hybrid model for text summarization", Information Processing Management, vol. 46, pp. 571–588, Sep. 2010. doi: 10.1016/j.ipm.2010.03.004.

70 M. Karabalayeva, Z. Yessenbayev, and Z. Kozhirbayev, “Regarding the impact of kazakh phonetic transcription on the performance of automatic speech recognition systems”, Oct. 2017.

71 Z. Kozhirbayev, B. Erol, A. Sharipbay, and M. Jamshidi, “Speaker recognition for robotic control via an iot device”, Jun. 2018, pp. 1–5. doi:10.23919/WAC.2018.8430295.

72 A. Mussina, S. Aubakirov, D. Ahmed-Zaki, and P. Trigo, “Automatic document summarization based on statistical information”, Journal of Mathematics, Mechanics and Computer Science, vol. 96, no. 4, pp. 76–87, 2019, issn: 2617-4871. [Online]. Available: https://bm.kaznu.kz/index.php/kaznu/article/view/581.

73 Z. Kozhirbayev, Z. Yessenbayev, and A. Makazhanov, “Document and word-level language identification for noisy user generated text”, English, in IEEE 12th International Conference on Application of Information and Communication Technologies, AICT 2018 - Proceedings, ser. IEEE 12th International Conference on Application of Information and Communication Technologies, AICT 2018 - Proceedings, 12th IEEE International Conference on Application of Information and Communication Technologies, AICT 2018 ; Conference date: 17-10-2018 Through 19-10-2018, United States: Institute of Electrical and Electronics Engineers Inc., Oct. 2018. doi: 10.1109/ICAICT.2018.8747138.

74 B. Myrzakhmetov, Z. Yessenbayev, and A. Makazhanov, “Initial normalization of user generated content: Case study in a multilingual setting”, English, in IEEE 12th International Conference on Application of Information and Communication Technologies, AICT 2018 - Proceedings, ser. IEEE 12th International Conference on Application of Information and Communication Technologies, AICT 2018-Proceedings, 12th IEEE International Conference on Application of Information and Communication Technologies, AICT 2018 ; Conference date: 17-10-2018 Through 19-10-2018, United States: Institute of Electrical and Electronics Engineers Inc., Oct. 2018. doi: 10.1109/ICAICT.2018.8747161.

75 B. Myrzakhmetov and A. Makazhanov, “Initial experiments on russian to kazakh smt”, Research in Computing Science, vol. 117, pp. 153–160, Sep.2016. doi: 10.13053/rcs-117-1-13.

76 Z. Kozhirbayev, Z. Yessenbayev, and M. Karabalayeva, “Kazakh and russian languages identification using long short-term memory recurrent neural networks”, 2017 IEEE 11th International Conference on Application of Information and Communication Technologies (AICT), pp. 1–5, 2017.

77 S. Mukherjee and P. Bhattacharyya, “Sentiment analysis : A literature survey”, CoRR, vol. abs/1304.4520, 2013. arXiv: 1304 . 4520. [Online]. Available: http://arxiv.org/abs/1304.4520.

78 A. Kumar and A. Jaiswal, “Systematic literature review of sentiment analysis on twitter using soft computing techniques”, Concurrency and Computation: Practice and Experience, vol. 32, no. 1, e5107, 2020, e5107 CPE18-1167.R1. doi:

10.1002/cpe.5107. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.5107. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.5107.

79 A. Qazi, R. G. Raj, G. Hardaker, and C. Standing, "A systematic literature review on opinion types and sentiment analysis techniques: Tasks and challenges", Internet Res., vol. 27, no. 3, pp. 608–630, 2017.

80 L. Martí, N. Sánchez-Pi, J. Molina, and A. C. Garcia, "Anomaly detection based on sensor data in petroleum industry applications", Sensors, vol. 15, pp. 2774–2797, Feb. 2015. doi: 10.3390/s150202774.

81 M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data", PLOS ONE, vol. 11, no. 4, pp. 1–31, Apr. 2016. doi: 10.1371/journal.pone.0152173. [Online]. Available: https://doi.org/10.1371/journal.pone.0152173.

82 A. Rodin, A. Litvinenko, K. Klos, A. Morrison, T. Woodage, J. Coresh, and E. Boerwinkle, "Use of wrapper algorithms coupled with a random forests classifier for variable selection in large-scale genomic association studies", English (US), Journal of Computational Biology, vol. 16, no. 12, pp. 1705–1718, Dec. 2009, issn:1066-5277. doi:10.1089/cmb.2008.0037.

83 C. Angermueller, T. Pärnamaa, L. Parts, and O. Stegle, "Deep learning for computational biology", Molecular Systems Biology, vol. 12, no. 7, p. 878, 2016. doi: 10.15252/msb.20156651. [Online]. Available: https://www.embopress.org/doi/abs/10.15252/msb.20156651.

84 V. Hodge, "A survey of outlier detection methodologies", Artificial Intelligence Review, vol. 22, pp. 85–126, Oct. 2004. doi: 10.1023/B:AIRE.0000045502.10941.a9.

85 M. Agyemang, K. Barker, and R. S. Alhajj, "Mining web content outliers using structure oriented weighting techniques and n-grams", in Proceedings of the 2005 ACM Symposium on Applied Computing, ser. SAC'05, Santa Fe, New Mexico: Association for Computing Machinery, 2005, pp. 482–487, isbn: 1581139640. doi: 10.1145/1066677.1066788. [Online]. Available: https://doi.org/10.1145/1066677.1066788.

86 A. Patcha and J.-M. (Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends", Computer Networks, vol. 51, pp. 3448–3470, Aug. 2007. doi: 10.1016/j.comnet.2007.02.001.

87 S. A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion detection using sequences of system calls", J. Comput. Secur., vol. 6, no. 3, pp. 151–180, Aug. 1998, issn: 0926-227X.

88 P. J. Rousseeuw and A. M. Leroy, Robust Regression and Outlier Detection. USA: John Wiley Sons, Inc., 1987, isbn: 0471852333.

89 M. Markou and M. Singh, "Novelty detection: A review—part 1: Statistical approaches", Signal Processing, vol. 83, pp. 2481–2497, Dec. 2003. doi: 10.1016/j.sigpro.2003.07.018.

90 V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey”, ACM Comput. Surv., vol. 41, no. 3, Jul. 2009, issn: 0360-0300. Doi: 10.1145/1541880.1541882. [Online]. Available: https://doi.org/10.1145/1541880.1541882.

91 R. Pincus, “Barnett, v., and lewis t.: Outliers in statistical data. 3rd edition. j. wiley & sons 1994, xvii. 582 pp., £49.95”, Biometrical Journal, vol. 37, no. 2, pp. 256–256, 1995. doi: 10.1002/bimj.4710370219. Eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/bimj.4710370219. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/bimj.4710370219.

92 E. M. Knorr, R. T. Ng, and V. Tucakov, “Distance-based outliers: Algorithms and applications”, The VLDB Journal, vol. 8, no. 3–4, pp. 237–253, Feb. 2000, issn: 1066-8888. doi: 10.1007/s007780050006. [Online]. Available: https://doi.org/10.1007/s007780050006.

93 M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, “Lof: Identifying density-based local outliers”, in Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, ser. SIGMOD ’00, Dallas, Texas, USA: Association for Computing Machinery, 2000, pp. 93–104, isbn: 1581132174. doi: 10 . 1145 / 342009 . 335388. [Online]. Available: https://doi.org/10.1145/342009.335388.

94 W. Jin, A. K. H. Tung, and J. Han, “Mining top-n local outliers in large databases”, in Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ser. KDD ’01, San Francisco, California: Association for Computing Machinery, 2001, pp. 293–298, isbn: 158113391X. doi: 10.1145/502512.502554. [Online]. Available: https://doi.org/10.1145/502512.502554.

95 A. Christy, G. M. Gandhi, and S. Vaithyasubramanian, “Cluster based outlier detection algorithm for healthcare data”, Procedia Computer Science, vol. 50, pp. 209–215, 2015, Big Data, Cloud and Computing Challenges, issn: 1877-0509. doi: https://doi.org/10.1016/j.procs.2015.04.058. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050915005591.

96 Piotr Bojanowski, Armand Joulin, and Tomas Mikolov. Alternative structures for character-level rnns. arXiv preprint arXiv:1511.06303, 2015.

97 Minh-Thang Luong and Christopher D Manning. Achieving open vocabulary neural machine translation with hybrid word-character models. arXiv preprint arXiv:1604.00788, 2016.

98 Alexander Rosenberg Johansen, Jonas Meinertz Hansen, Elias Khazen Obeid, Casper Kaae Sønderby, and Ole Winther. Neural machine translation with characters and hierarchical encoding. arXiv preprint arXiv:1610.06550, 2016.

99 Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In Advances in Neural Information Processing Systems, pages 2017–2025, 2015.

100 Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In Advances in neural information processing systems, pages 2204–2212, 2014.

101 Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In ICML, volume 14, pages 77–81, 2015.

102 Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning, 8(3-4):229–256, 1992.

103 Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, 2014.

104 Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025, 2015.

105 Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. arXiv preprint arXiv:1410.5401, 2014.

106 Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. Nature, 538(7626):471–476, 2016.

107 Edward Grefenstette, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. Learning to transduce with unbounded memory. In Advances in Neural Information Processing Systems, pages 1828–1836, 2015.

108 Armand Joulin and Tomas Mikolov. Inferring algorithmic patterns with stack-augmented recurrent nets. In Advances in neural information processing systems, pages 190–198, 2015.