SDU University

UDC 20.23.19 Manuscript Copyright

# AYAZBAYEV DAUREN ARYNULY

**Development of algorithms for search engine of text documents**

8D06102 – Computer science

Thesis for the Degree of

Doctor of Philosophy (PhD)

Scientific advisor

doctor PhD,

assoc. professor,

A. Bogdanchikov

# Foreign scientific advisor

doctor PhD,

professor,

I. Varlamis

# Republic of Kazakhstan

# Kaskelen, 2024

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **NORMATIVE REFERENCES** …………………………………………… | 4 | |
| **ABBREVIATIONS** ……………………………………………………….. | 5 | |
| **INTRODUCTION** …………………………………………………………. | 6 | |
| **1 LITERATURE REVIEW** ……………………………………………….. | 9 | |
| 1.1 Google …………………………………………………………….…….. | 9 | |
| 1.2 Microsoft Bing …………………………………………………………. | 9 | |
| 1.3 Baidu, Inc ………………………………………………………………. | 9 | |
| 1.4 Facebook AI Similarity Search…………………………………………. | 9 | |
| 1.5 Elasticsearch ……………………………………………………………. | 11 | |
| 1.6 Works of foreign researchers ……………………………………….…... | 11 | |
| 1.7 Experiments related with the Kazakh language …………………..…... | 15 | |
| Conclusion for literature review …………………………………………... | 20 | |
| **2 EXPERIMENTAL EVALUATION** …………………………………… | 21 | |
| 2.1 Identifying lexical compatibilities of words by vectors of specialized  words ………………………………………………………………….….. | 21 | |
| 2.1.1 Skip gram model ………………………………………………..…... | 22 | |
| 2.1.2 System of defining lexical compatibilities…………………………… | 24 | |
| 2.1.3 Conclusion of the experiment …………………………………..…... | 25 | |
| 2.2 Stemming of the Kazakh language …………………………………….. | 26 | |
| 2.2.1 Conclusion of the experiment …………………………………..…... | 28 | |
| 2.3 Defining semantically close words of Kazakh language by distributed system Apache Spark ……………………………………………………... | 28 | |
| 2.3.1 Defining similarities of words ……………………………………..... | 29 | |
| 2.3.1.1 Edit distance ……………………………………………………..... | 29 | |
| 2.3.1.2 Euclidean distance ……………………………………………..….. | 29 | |
| 2.3.1.3 Cosine similarity ………………………………………………...….. | 30 | |
| 2.3.2 Conclusion of the experiment ……………………………………..... | 31 | |
| 2.4 Classification of Scientific Documents in the Kazakh Language Using Deep Neural Networks and a Fusion of Images and Text……………….… | 31 | |
| 2.4.1 Experiment Dataset ……………………………………………..…... | 31 | |
| 2.4.2 Classification of papers by text ………………………………..……. | 32 | |
| 2.4.3 Neural network …………………………………………………..….. | 34 | |
| 2.4.3.1 Activation function in neural network ………………………..…... | 35 | |
| 2.4.3.2 Stop conditions in neural network …………………………..…… | 36 | |
| 2.4.3.3 Popular neural network architectures…………………………….. | 36 | |
| 2.4.4 Classification of papers by images ……………………………..…… | 37 | |
| 2.4.5 Classification of papers by text and images ……………………..….. | 37 | |
| 2.4.6 Conclusion of the experiment …………………………………..…... | 39 | |
| 2.5 Experiment with FAISS ……………………………………………...... | 39 |
| 2.5.1 Conclusion of the experiment ………………………………………... | 41 |
| 2.6 Experiment with Elasticsearch ……………………………………...…. | 41 |
| Conclusion of the experiment ……………….……………………………... | 42 |
| **3 DESIGNED KOMEKSHY SYSTEM** ………………………………….. | 43 |
| 3.1 Instructions of Komekhy system ……………………………………….. | 43 |
| 3.1.1 User role …………………………………………………………….… | 43 |
| 3.1.2 Admin role ……………………………………………………….…… | 45 |
| 3.2 Architecture of Komekhy system ……………………………………… | 47 |
| 3.3 Django architecture ……………………………………………………... | 51 |
| 3.4 Implementation of Komekshy system ………………………………….. | 52 |
| 3.4.1 morphology\_analyzer application ……………………………………. | 52 |
| 3.4.2 Search\_engine application …………………………………………..… | 53 |
| 3.5 Dataset of Komekshy system ……………………………………….…... | 57 |
| 3.5.1 Reading dictionaries ……………………………………………….….. | 57 |
| 3.5.2 Portal of sozdikqor.kz ……………………………………………….... | 58 |
| 3.6 Limitations of thesis ………………………………………………….…. | 60 |
| 3.6.1 Prefix tree ……………………………………………………………... | 60 |
| 3.6.1.1 Compressed prefix tree …………………………………………….. | 62 |
| 3.6.1.2 Insert operation in prefix tree……………………………………….. | 62 |
| 6.6.1.3 Delete operation in prefix tree ……………………………………… | 63 |
| **5 EXPERIMENTAL TESTING AND RESULTS** ……………………….. | 65 |
| **CONCLUSION** ……………………………………………………………… | 67 |
| **REFERENCES** ………………………………………………………….……. | 70 |

# 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, publishing. Research report. Structure and design rule.

# ABBREVIATIONS

|  |  |
| --- | --- |
| ASA | – Appearance Semantic Attribute |
| BERT | – Bidirectional Encoder Representations from Transformers |
| CBOW | – Continuous Bag of Words |
| CNN | – Convolutional Neural Network |
| CSS | – Cascading Style Sheets |
| CTC | – Connectionist temporal classification |
| CV | – Curriculum Vitae |
| GPT | – Generative Pretrained Transformer |
| GPU | – Graphics processing unit |
| FAISS | – Facebook AI Similarity Search |
| HKM | – Hierarchical k-means |
| HKR | – Handwritten Kazakh and Russian |
| HTML | – Hypertext Markup Language |
| HTTP | – Hypertext Transfer Protocol |
| KazNU | – Kazakh National University |
| LSTM | – Long-Short Term Memory |
| MFCC | – Mel Frequency Cepstral Coefficients |
| MVDR | – Minimum Variance Distortionless Response |
| NLP | – Natural Language Processing |
| PCA | – Principal component analysis |
| RAM | – Random Access Memory |
| ReLU | – Rectified linear unit |
| RNN | – Recurrent neural network |
| URL | – Uniform Resource Locator |
| YII | – Yes it is |

–

# INTRODUCTION

Nowadays searching information is not a new problem. It has been existing from ancient time. Therefore, today there are a lot of methods of organizing information and searching algorithms. For instance, libraries have a lot of books with various genres. To find relevant book in short time books can be classified by authors (efficient when people search books of certain author), genres (efficient when people search books of some genre like math).

People search for information in the Web or other document collections for various reasons. In order to cover their information need parents can search information about taking care of children, tourists about new place, teachers about new lectures, and researchers about new publications in their discipline. Since the information needs differs in each case, the respective query is also different (instructions, maps, lectures, publications). In addition, relational database management systems have tools for retrieving information. In relational database management systems data is stored in tables. However not all data can be stored in tables. Therefore, search engines should enhance their algorithms to satisfy requirements of all people. Nowadays there are various search engines. They can have different searching algorithms, databases. Hence user for the same query can get different results from search engines. In addition, one searching algorithm can show different results for various languages. Emphasizing on languages with limited linguistic resources, the current thesis focuses on the Kazakh language. The Kazakh language is rich by its suffixes, endings, which alter the meaning of the main word (or stem). In addition, it lacks of rich ontologies or pre-trained language models, so the information retrieval tasks are still hard to solve.

Queries can be classified into different types: matching, semantic, question. In matching type people just try to find all documents which contain query words. An example of such query can be searching some messages from email. In semantic type people not just try to find entered words from documents, but related information as well. An example of such query can be when search certain mathematical theorem with other related theorems. In last type people write their questions to find answers.

Taking into account the statements above new search engine decided to be developed. New search engine will focus on the Kazakh language. The Kazakh language is rich by its suffixes, endings. Words in Kazakh language change their meaning when suffixes are joined to the roots. Therefore, the search engine should be able by morphological analysis define root of word, and return relevant information for entered query.

**The research aim.** Developing a search engine for the Kazakh language, which can return relevant Wikipedia articles for the given query.

**Objectives of research.** For the current thesis following objectives were defined:

1. Doing morphological analysis for user’s query.

2. Using word embedding model to define semantically close words.

3. Designing database of Kazakh Wikipedia articles.

4. Doing an experiment by distributed system Apache Spark for reducing computation time.

**The object of research.** The study focuses on searching algorithm, which is able to find out relevant Wikipedia articles for the given query.

**Research methods.** Assigned objectives were solved by carrying out theoretical and empirical research. In current thesis we used word embedding, cosine similarity, morphological analysis for the Kazakh words to reach defined aim.

**The scientific significance of the study.** In current research work to calculate score of each Wikipedia article new formula was designed empirically. The experiment showed that this formula efficiently can be used to rank documents.

**The practical significance of the study.** People can use new search engine which is called Komekshy. Using Komekshy people can find Wikipedia articles which were ranked by new algorithm.

**The scientific novelty of the work.** Novelty of research work is defining semantically close words for the Kazakh language. Semantically close words are words which are used in one context. Semantically close words are not always synonyms. They can have different meanings. Such words can be considered as semantically close, because they are used in one sentence. An example of semantically close words can be printer, monitor, mouse, computer. Even they have different meanings, they are usually used in one context. In current thesis semantically close words are necessary to expand query. It means by semantically close words not only query words will be searched but other words as well. In current thesis semantically close words were defined for 8497 words. They are divided by parts of speech. Each word has 9 semantically close words.

# Publications:

1. Мамандандырылған Сөздердiң Векторлары Арқылы Сөздердiң Лексикалық Тiркесулерiн анықтау // ҚазҰУ хабаршысы. Математика, механика, информатика сериясы. – 2020. – Т. 107, №3. – Б. 67-73.
2. Stemming of the Kazakh language // Bulletin of Abai Kazakh National Pedagogical University. – 2021. – P. 162-166.
3. Defining semantically close words of Kazakh language by distributed system Apache Spark // Big Data and Cognitive Computing. – 2023. – Vol. 7, Issue 4. – P. 1-13.
4. Classification of Scientific Documents in the Kazakh Language Using Deep Neural Networks and a Fusion of Images and Text // Big Data and Cognitive Computing. – 2022. – Vol. 6, Issue 4. – P. 1-12.

**Acknowledgments.** I would like to express my gratitude to Andrey Bogdanchikov, Iraklis Varlamis for their supervision of doing this research. In addition, I would like to express my gratitude to my family and colleagues for their support.

**Structure and scope of the dissertation.** The thesis consists of five chapters. It contains 47 figures and 12 tables. The first chapter introduces the research topic. Particularly it includes aim, objectives, novelties of current thesis. The second chapter makes an overview for the other similar works. Overview was made for the papers, which solved certain task for designing search engine. The third chapter describes experiments of current thesis. The fourth chapter introduces Komekshy search engine, which was designed by the results of experiments. The last chapter contains the test result of Komekshy. Conclusion part summarizes the research work.

# LITERATURE REVIEW

Nowadays there are a lot of search engines. For instance: Google, Microsoft Bing, Yahoo, Baidu, Yandex, DuckDuckGo, Ask.com, Ecosia, Aol.com, Internet Archive. They can be differentiated by the algorithms they use, the dataset they index, the interface language they are using and so on. Searching of documents is not a simple task, and it can run in parallel with other subtasks like: results’ ranking, translation, collection of documents, classification, clustering, deduplication of results. Hence it is difficult to say which search engine is better: one search engine can work better in one direction, whereas it can show low result in other direction. Below search engines discussed deeper. The main search engines available in the Web nowadays are: Google, Baidu, Inc, Microsoft Bing.

# Google

Google is a company with several working directions like search engine technology, online advertising, cloud computing, computer software, quantum computing, consumer electronics [1]. It has Google News service which collects information from various sites and summarizes them. Using Google various kind of information can be searched. For instance, it can be image, site, news, book. Therefore, dataset which it uses is also various.

# Microsoft Bing

Microsoft Bing is a web search engine, which can search videos, images, maps, web sites. It was written on ASP.NET framework. According to [2] in 2018 year Microsoft Bing took the third place by amount of query. The first place Google (77%), the second place Baidu (14.45%) and the third is Microsoft Bing (4.58%).

# Baidu, Inc

Baidu is a company which offers services like: Chinese search engine, mapping (Baidu Maps), online encyclopedia Baidu Bike, a cloud storage service Baidu Wangpan, a keyword discussion forum Baidu Tieba [3]. According to [3] Baidu is the second largest search engine in the world, however the largest in China. Robin Li is one of co-founder of Baidu in 1996 developed RankDex site scoring algorithm. It used hyperlinks to define popularity of web site. If a web site had a lot of hyperlinks to it, it was considered as popular web site.

# Facebook AI Similarity Search

As an alternative software to the search system of current thesis Facebook AI Similarity Search was investigated. Facebook AI Similarity Search(FAISS) is a library which allows developers search similar documents. By FAISS multimedia documents can be searched. FAISS is written on C++. For searching FAISS uses several algorithms some of them use GPU.

FAISS works by vectors. It transforms documents into numeric form, then calculates Euclidean distance between two vectors. After that documents, which are the closest will be returned. FAISS library also can do following operations:

1. Return not only the closest document, but k the closest documents.
2. Search several vectors in parallel. Hence it can work faster rather that search sequentially.
3. Manipulate the precision of result and its speed. It means developer can decrease precision of result, but can get result 10 times faster or use 10 times less memory.
4. Instead of using Euclidean distance can use maximum inner product search.
5. Store the index on disk rather than on RAM.

FAISS can store collection of vectors in matrices. Particularly, it uses 32-bit floating point matrices. FAISS uses two matrices: xb database matrix, which stores collection of vectors of documents. Its size is nb by d. The second matrix xq stores vector of query. Its size is nq by d. Below in figure 2.1 is an example how the library can be used. This example was taken from [4]. Coordinates of vectors defined by uniform distribution.



Figure 2.1 – Defining vectors

In example above the matrices are represented by numpy array. The code snippet above defines coordinates of vectors of database and query. Following code in figure 2.2 is continuation of the example.

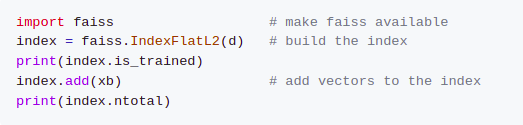


Figure 2.2 – Build FAISS index

In code snippet above the index was defined. is\_trained variable returns true or false value. It defines whether training is necessary. ntotal variable returns number of indexed vectors. Finally, in figure 2.3 searching is performed and its result is printed.

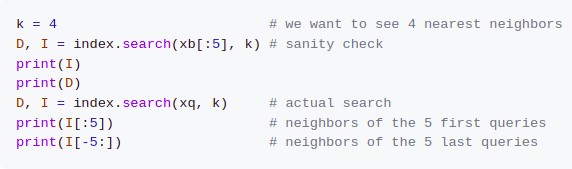


Figure 2.3 – Searching similarity

FAISS was developed based on several researches. For instance, authors of work [5] investigated nearest neighbor search. They considered various algorithms like lloyd’s, hierarchical k-means(HKM). These algorithms have some disadvantages. Therefore, authors of work [5, p. 117-127] proposed their solution. Authors of work [6] in turn, proposed a method of re-ranking of vectors with a limited amount of memory. Advantage of their work is that data can be stored in memory. Hence costly disk accesses can be avoided. In addition, they developed publicly available dataset of one billion vectors with 128 dimensions.

# Elasticsearch

Elasticsearch is an another alternative to current thesis. Elasticsearch is a searching platform with many features. People by Elasticsearch platform can upload their dataset and search them. This platform was developed based on Lucene library. In addition, Elasticsearch has Kibana dashboard which allows draw various graphs. People based on that graphs of returned result can take some decision. The dataset of Elasticsearch can contain several fields. User of platform can define priorities to these fields. Hence manipulate searching result. To expand or reduce searching space Elasticsearch supports operations like “and”, “or”, “not” [7].

Lucene is a search engine library. Using opportunities of Lucene other search engines can be developed. One of them is Elasticsearch. Elasticsearch uses inverted indices [8]. Authors of work [8, p. 101893] replaced inverted index to suffix index.

# Works of foreign researchers

Many works have been done for exploring search engine algorithms, analysis with search engine. The current section makes a survey of such works.

Classification is not only computer science problem, it also exists in other spheres like agriculture. For instance, authors of work [9] classified rice. To classify rice, they used random forest algorithm. In their experiment they had problems related with class member numbers. Some class members were many, some were not. Therefore, the dataset was imbalanced. To solve this issue authors have used random forest algorithm with oversampling. The experiment was done using Google Earth Engine platform. To evaluate the algorithm authors calculated accuracy, precision, recall. The accuracy was 81.46%.

According to [10] approximately 6 billion searches occur per day. To retrieve some information people write queries in search engines. Writing misspelled words in a query can lead to not relevant results. Therefore, to avoid such situation and optimize a search engine authors of work [10, p. 100451-1-100451-5] discussed various kinds of error like transposition, wrong letter, extra letter, missing letter and so on. In paper the authors provided spell check algorithm, which works by representing words in binary form.

Authors of work [11] tried to forecast sales on the basis of search engine data. To do it they have used principal component analysis(PCA), neural network with back propagation model. To represent textual review of some product into numeric form sentiment analysis was done.

To get relevant responses from search engine many factors play role. Authors of work [12] performed semantic analysis for user query and web pages. By this approach ambiguity can be avoided. The work contains the algorithm for ranking results of search engine. The experiment showed that using their algorithm the search engine returned more relevant results than without it. In addition, authors think that analyzing relationship between users can also enhance results.

Search engines can be applied in various spheres like medicine, sport, art. Authors of work [13] decided to develop COVID-19 search engine system which is focused on coronavirus disease.

Authors of work [14] designed model, which can extract keywords from documents. Particularly, they tested the model on patent claims, newsgroup postings, customer reviews. As a model they have used logistic regression model. The size of document can be different. In addition, to properly select keywords from document coefficients of the model can be updated.

Authors of work [15] provided a code search framework for low-resource languages. The proposed approach used web platform for searching various codes. In addition, authors provided retrieval mechanism for given query.

Nowadays search engines can not only search documents, but correct misspelled words in queries as well. To correct misspelled words, the system need to have a dictionary with the right versions of words. However, this technique cannot work if dictionary doesn’t contain entered word. Therefore, authors of work [16] considered typing errors in title searching. Particularly, they have used cosine similarity to define relevant titles.

Authors of paper [17] tried to automatically match European Union Directive provisions with National Implementing Measures by semantic search.

Nowadays web shopping is convenient, because for selling products there is no need to have boutiques. In addition, some online stores have delivery service. However, searching product in web shopping is still needs improvement. Now in web shopping customer writes query to find relevant product. In this case entered query can have several meanings, can contain other words than product name. Hence authors of work [18] proposed methodology for re-ranking products on online store search result. They have used user profile construction to find relevant product. In addition, they have used algorithms like principle component analysis (PCA), k-means clustering for searching.

Searching can be considered as complex task, because documents can contain various kind of content. For instance, it can be text, image, source code. Authors of work [19, p. 100117-1-100117-10] proposed a system which can search for smart contracts on blockhain explorers. Particularly, their system finds smart contracts according to developer’s search preferences.

The format of query can be different. For instance, text can be used to search text, images. Authors of work [20] proposed a retrieval approach for Text-to-Image, Image- to-Text. Particularly, for product description the model had to find out corresponding image among 100 images. The second task was for an image of product the model had to find out corresponding product description among 100 descriptions. To do it their approach uses vision and language transformers. Their architecture separates text and image embedding. The model was tested on FashionGen and DeepFashion-Synthesis datasets.

When people search some document they can have different purposes. For instance, people can search document to find out necessary information like formulas, theorems. Another example is searching without exact answer. This case can occupy when doing literature review. Researcher can search papers, which are similar to his/ her topic. Here researcher doesn’t know which paper exactly to search. Authors of work [21] proposed people searching method by appearance description. This task is very important when people are lost. The proposed method contains two steps: attribute classifier learning, people search process.

The task of finding the best candidate for specific job position can be difficult in case if number of CV is large. The searching the best candidate system can work based on keywords matching. In that case recruiter need to write from general query to specific. This method allows reducing number of CV. Author of work [22] built a model for finding experts based on semantic text searching. To do it the model expands the query and rank the result. In experiment datasets like Wikipedia, BabelNet, WordNet, Stackoverflow, Quora were used. The model uses Elasticsearch as search engine. To avoid duplicate, empty CVs outlier detection analysis like z-score statistical analysis was applied.

Authors of work [23] described neural network architecture for POS tag extraction. To extract POS tags they used Bi-Long Short Term Memory.

Authors of work [24] used Mímir open-source semantic search framework to search text. Mímir supports full text Boolean retrieval, structural annotation graph search.

Authors of work [25] designed a Quranic Semantic Search Tool. The tool consists of three phases. In the first phase Holy Quran is manually annotated. In the second phase by Continuous Bag of Words(CBOW) neural network architecture feature vectors are defined. In the third phase feature vectors of Quranic topic and entered query are retrieved. After that by cosine value between Quranic topic and query vectors the most relevant vector is returned.

Authors of work [26] described difficulties of designing search system. For instance, query can have synonyms or polysemy. Synonyms are words which have the same meaning. Words like movie and film are synonyms. Polysemy are words which have different meanings. Word bank can be considered as polysemy. Because bank can be financial institute or river corner. In addition, authors describe various semantic similarity methods edge counting method, information content, feature-based. In paper authors discuss searching methodology, where in semantic relationship synonyms, semantic neighborhood, hyponyms are considered. To rank documents by relevance tf.idf weighting scheme is used.

Nowadays there are various social networks. Text in social network can be short. For instance, according to [27] number of characters in Sino Weibo messages is usually less than 140. Therefore, it can be difficult to find relevant message in social network. Taking into account it authors of work [27, p. 67-77] proposed a multi-feature probabilistic model for searching in social network. To provide relevant response in social network authors analyzed text, user, timestamp, user location (table 2.1).

Table 2.1 – Summary of papers

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Dataset | Algorithm | Used language |
| 1 | 2 | 3 | 4 |
| Classification of rice | Landsat 8  surface reflectance from Google Earth Engine, Area Sampling Framework data from Statistics Indonesia | Random forest with oversampling | - |
| Spell check | Binary  dictionary | Binary integrated  spell check algorithm | English |
| Forecasting | Baidu index query and analysis tool | Back propagation | Chinese |
| Searching of text | Enwiki1m, SogouQ, gene, noaa | Suffix indexing | Chinese |
| Semantic analysis | Their own dataset | Ranking algorithm | English |
| Searching of  Covid related papers | CORD-19 | TF-IDF | English |
| Keywords extraction | Patent claims, newsgroup postings, customer reviews | Logistic regression model | English |
| Semantic search | Not found | Algorithms of web  platform | Not found |
| Semantic search | Prime video | Cosine similarity | Hindi |
| Dataset collection | EUR-Lex website and  national official journals | Okapi BM25 ranking function,Approximate Nearest Neighbor | Italian, French |
| Re-ranking using semantic segmentation | luxury department store De Bijenkorf | PCA, k-means clustering | Dutch |
| Searching for |  | Label classification | Programming |
| Table continuation 2.1 | | | |
| 1 | 2 | 3 | 4 |
| source code |  | algorithm, Feature extraction algorithm, fastText embedding  algorithm | language |
| Finding relevant image, finding relevant text | FashionGen and DeepFashion- Synthesis  datasets | Image embedding, text embedding, Fast k-NN search | English |
| People searching based on appearance  description | Appearance Semantic Attribute (ASA) | Deep CNN | English |
| Finding the best CV for specific job position | Wikipedia, BabelNet, WordNet, Stackoverflow,  Quora | Algorithm of Elasticsearch engine | English |
| Semantic  search | Not found | Bi-Long Short Term  Memory | Not Found |
| Text search | Linked Open  Data | Full text Boolean  retrieval | Not Found |
| Semantic search in Holy  Quran | Quranic and Classic Arabic  corpus | CBOW, cosine similarity | Arabic |
| Searching  methodology | Not found | tf.idf weighting  scheme | English |
| Searching in social network | Sino Weibo | multi-feature probabilistic graphical model  (MFPGM) | China |
| Note – Compiled from sources [8, p. 101893; 9, p. 668-675; 10, p. 100451-100451-5; 11, p. 1005-1023; 12, p. 2287-2295; 13, p. 100068-1-100068-13; 14, p. 1-24; 15, p. 3-30; 16, p. 761-766; 17, p. 1-9; 18, p. 1-19; 19, p. 100117-1100117-10; 20, p. 126196-126196-13; 21, p. 90-98; 22, p. 2422-2430; 23, p. 388-391; 24, p. 52-67; 25, p. 934-3944; 26, p. 161-168; 27, p. 67-77] | | | |

# Experiments related with the Kazakh language

In many experiments important part is data collection. Authors of work [28] developed new Kazakh text corpora. Authors described 3 different data collection approaches. The first approach was searching existing ready to use aligned texts. The second approach was using bitextor tool for crawling websites. The third approach is similar to the second, but here for crawling websites authors used InterText with integrated hunalign tool. As a result, authors obtained 1 076 927 number of words in Kazakh.

Authors of work [29] developed corpus for the Kazakh language. Their corpus was classified as literary, official, scientific, publicistic, informal language classes.

The format of searching information can be different. People can search text, images, characters. The last type can be used in classification tasks. Authors of work [30] worked with Handwritten Text Recognition task. Particularly, they designed Kazakh offline Handwritten Text dataset. The dataset contains 3000 handwritten exam papers, more than 140135 segmented images and about 922010 symbols. The dataset can be used in other experiments related to natural language processing tasks. In addition, they used text recognition methods as CTC-based and attention-based.

Nowadays the Kazakh language uses Cyrillic symbols to represent letters. However, before Cyrillic symbols Latin, Arabic graphics were used. Therefore, information systems should be able to properly process that graphics as well. Authors of work [31] considered four vowels which are written in Arabic graphics. They proposed a method which allows properly edit, show those four vowels.

Authors of work [32] classified Kazakh and Uyghur text by convolutional neural network(CNN). According to [32, p. 387-1-387-13] both languages belong to agglutinative languages. Because in both languages stemming of word can be connected with suffixes changing sentence meaning. Therefore, authors applied morphological analysis to define stemming. In paper authors compared text classification result with defining stemming and without defining. Their experiment showed that when stemming was defined, convolutional neural network classified text better.

Authors of work [33] did similar as authors of work [32, p. 387-1-387-13]. They by CNN model classified text. Authors of work [33, p. 1903-1905] classified text by two ways: morpheme-based approach and word-based approach. In experiment morpheme-based approach showed better result. To perform morphological analysis m2asr morphological analyzer was used.

Authors of work [34] designed a model for classification of Kazakh question. To classify questions CNN and BiGRU neural networks were used. Output layer of CNN model produced high-dimensional semantic features. Those semantic features were input for BiGRU. BiGRU in turn, by SoftMax function classified questions.

In paper [35] authors introduced handwritten Kazakh and Russian(HKR) dataset. This dataset contains Kazakh and Russian handwritten scanned documents. In dataset letters were written in Cyrillic.Therefore the dataset can be used for text recognition problems. HKR contains approximately 63000 sentences which are written by about 200 people. In experiment authors did text recognition by CTC-based and attention based methods.

Authors of work [36] did sentiment analysis for the Kazakh news articles. To do it they have used Long-Short Term Memory(LSTM). LSTM allows finding out long term dependencies of whole text. This approach allows finding out long term dependencies without knowing language rules.

Sentiment analysis is analysis of emotional tone regarding some topic. It has classes like positive, negative, neutral. It can be useful when there are a lot of comments about certain news. To avoid reading each comment and summarize opinions sentiment analysis can be used. Authors of work [37] did sentiment analysis for the Kazakh text. They designed dictionary of Kazakh sentiment words. In paper authors described rule- based method of sentiment analysis where dictionary of emotional words is used.

Nowadays information technologies can facilitate people’s work. For instance, text easily can be translated to another language, text can be classified into various classes. One application of information technologies is defining topic relatedness to its text. Because topic and its main body can have different meanings. Defining topic relatedness can be useful in diploma projects, theses, papers. Because authors can define not appropriately topic to their work. Authors of work [38] designed an algorithm for keyword search from the Kazakh text. They have used Porter stemmer to define keywords from text and produce dictionary of word stems.

Automatic text classification can be applied in various NLP tasks. It is especially useful when the text is long and number of such text is large. For instance, automatic text classification can be used for comments of certain news. Instead of reading all comments, text can be automatically classified. Authors of work [39] introduced a method of text classification of the Kazakh language. They did it by constructing parse tree. The result of work can be applied in question-answering systems.

According to authors of work [40] pretrained transformer models like Bidirectional Encoder Representations from Transformers (BERT), Generative Pre- trained Transformer (GPT) showed good results in sentiment analysis of dominant languages such as English. In paper authors consider two ways of implementing the transfer learning strategies. The transfer learning strategies are zero-shot learning and fine-tuning. In experiment they compared BERT-based multilingual sentiment analysis model with BERT-based model for Turkish language.

Keyword extraction task can be used in various areas. For instance, to define title of text, classification of text. Authors of work [41] did keyword extraction task for three languages: Uyghur, Kazakh and Kirghiz. All these three languages are agglutinative. Therefore, stems can be connected with affixes. When affixes are connected with stems, meaning of word can be changed. Hence morphological analysis plays important role in keyword extraction task. Their method consists of several steps. In first step the text was crawled from the Internet. Then by rule based spellchecker from the text spelling errors were defined. Then by Bi-LSTM CRF-based Uyghur-Kazakh-Kirghiz morpheme segmentation framework stems were defined. Then to define keywords Doc2vec weighted TextRank algorithm was applied. To evaluate the proposed method authors used metrics like precision, recall, F1 score.

Authors of work [42] studied Kazakh language voice recognition task. They tried recognize voice for voice search engine. To make voice normalization they used MVDR-based Built-in Speaker Normalization, MFCC algorithms.

Authors of work [43] developed useful way for storing Kazakh words. Effectiveness of this approach is that people can understand meaning of words without considering the whole context. In addition, three preprocessing tools were designed: producer word forms, morphological analyzer, word ambiguity resolutioner. Used corpus was placed at corpus.kaznu.kz (was available on 31.10.2023).

The Kazakh language is agglutinative language. It means word consists of morphemes. Word can change its form by various suffixes, endings. Authors of work [44] did morphological analysis for the Kazakh language. Particularly they developed classification system for endings and suffixes. Number of suffixes was 26526 units, whereas number of endings was 3565 units.

People can do various mistakes when type in document. For instance, words can be misspelled. In this case letters are typed wrong. Another type of mistake is word order. Here words are typed correctly, but they are written incorrect place. Authors of work [45] developed spell checking application for the Kazakh language. They considered two types of mistakes. In first type of mistake to check spelling of word the dictionary was used. To solve second type of mistake synthetic rules were applied.

In research important part is data collection. Collected data can be used in various purposes. For example, collected data can be used for training, testing neural network. Authors of work [46] presented a system for collecting speech data. The tool is called “Kazakh recorder” which was available online. It allows collect speech data quickly and conveniently. By that tool authors collected over 50 hours of speech data. In data collection process participated 65 people. Each participant on average pronounced 500 sentences. Authors of work [47] did similar work, but they described three different approaches for collecting parallel text (table 2.2).

Table 2.2 – Summary of papers

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Dataset | Algorithm | Used language |
| 1 | 2 | 3 | 4 |
| Data collection | OPUS, New World Bible, Lab IIS, Akorda, TED | Bitextor, InterText tools | Kazakh, English |
| Data collection | Internet websites, books,  dissertations, articles | Crawler | Kazakh,  Russian |
| Handwritten  Text Recognition | Kazakh offline Handwritten Text | CTC-based and attention- based | Kazakh |
| Font designing | Arabic-Kazakh alphabet | The method which is  consists of three rules | Kazakh, Arabic |
| Text classification | [www.uyghur.people.com.cn](http://www.uyghur.people.com.cn/), [www.kazakh.ts.cn/](http://www.kazakh.ts.cn/) | CNN model | Uyghur and  Kazakh |
| Text  classification | Not found | CNN model | Kazakh |
| Question  classification | Not found | CNN, BiGRU | Kazakh |
| Table continuation 2.2 | | | |
| 1 | 2 | 3 | 4 |
| Handwritten text  Recognition | Handwritten Kazakh and Russian(HKR) database | CTC-based, attention-based | Kazakh, Russian |
| Sentiment analysis | Not found | LSTM, word2vec, GloVes | Kazakh, Russian |
| Sentiment  analysis | Not found | Rule-based | Kazakh |
| Keyword search from the Kazakh  text | Dump of Wikipedia database | Porter algorithm | Kazakh |
| Automatic text  classification | Kazakh sentence | Construction  of parse tree | Kazakh |
| Implementing  transfer learning | Not found | zero-shot learning and fine-tuning | Kazakh, Turkish |
| Keyword extraction | Crawled from the Internet | Bi- LSTM\_CRF  for morpheme seg mentation, Doc2vec and TextRank for keyword extraction | Uyghur, Kazakh, Kirghiz |
| Voice recognition for the Kazakh language | Not found | MVDR-based Built-in Speaker Normalization, MFCC algorithms | Kazakh |
| Producer word forms, morpho logical analy zer, word am biguity resolu tioner tools | 140,000 news in the Kazakh language | Rule based | Kazakh |
| Morphological  analysis | Bilingual parallel corpora | Stemming algorithm | Kazakh |
| Spell checking | Kazakh words and sentences | Strings matching, synthetic rules | Kazakh |
| Collection of  speech data | Books | yii2-framework | Kazakh |
| Note – Compiled from sources [28, p. 501-507; 29, p. 1022-1030; 30, p. 116827-1-116827-12; 31, p. 1-7; 32, p. 387-1-387-13; 33, p. 1903-1905; 34, p. 943-946; 35, p. 33075-33096; 36, p. 537-544; 37, p. 9-14; 38, p. 26-31; 39, p. 13-17; 40, p. 1-5; 41, p. 283-1-283-16; 42, c. 1428-1434; 43, p. 1-14; 44, c. 545-550; 45, p. 132-134] | | | |

# Conclusion for literature review

In literature review section various search engines, papers were analyzed. Reading information about search engines we can understand that they are different. For instance, the size of dataset or ranking algorithm can be different. Therefore, one search engine can have large dataset, but ranking algorithm can return not relevant results. Literature review section was divided into two parts: works of abroad researchers, works of researchers from Kazakhstan. Text searching depending on algorithm involves several steps. Therefore, papers were analyzed for specific step like data collection, text classification. To quickly review papers 2.1, 2.2 tables were provided.

# EXPERIMENTAL EVALUATION

During academic years several experiments were done. In first experiment skip gram model was investigated. By skip gram model word can be represented in vector form. Calculating cosines between two vectors allows to define lexically compatibility of two words. This approach helps to define words which are not properly used in sentence. Here sentence can be user’s query.

In second experiment the system was developed which can define stemming of Kazakh words. Stemming is a process of defining root of word. Knowing root of word user’s query can be properly processed.

In third experiment semantically close words of the Kazakh language were defined. It was necessary to expand searching scope. Two different sentences can have the same meaning. User can write one of them whereas Wikipedia articles (“Көмекші” system as response to user’s query returns Wikipedia articles) can contain other words. To find out that matching semantically close words were used in “Көмекші” system.

The last experiment was about classification of documents. To reduce number of documents to search and make “Көмекші” system faster Wikipedia articles must be classified. In fourth experiment scientific papers from Kazakh National University bulletin were classified. Papers were classified by only its images, only text, fusion of text and images.

# Identifying lexical compatibilities of words by vectors of specialized words

For this experiment paper [48] was written. In first experiment lexical compatibilities were calculated by cosines of vectors. Vectors were developed by Andrey Kutuzov, and they were available at [49]. Lexically compatible words are words, which can be in one sentence. For instance, printer, mouse, cooler, monitor words which are related with computer. Hence they can be considered as lexically compatible words. I think words like caw, paper, ball not often occupy together in one sentence. So that words can be considered as not lexically compatible words. In experiment new words were added to text of constitution of Republic of Kazakhstan. The system needed to find out that added word. To do it cosines between vectors of adjacent words were calculated. If cosine between added word and its adjacent words was minimum among other cosine values it was considered as the system found added word. Table 3.1 contains a list of words which were inserted and their respective accuracy.

Table 3.1 – Accuracies of inserted words

|  |  |
| --- | --- |
| Word | Accuracy(%) |
| Green | 57.14 |
| Cow | 100 |
| Wolf | 100 |
| Pen | 71.43 |
| Computer | 57.14 |
| Cosmonaut | 28.57 |
| Car | 14.29 |
| Airplane | 71.43 |
| Iron | 71.43 |
| Aluminum | 85.71 |

According to table 3.1 some words showed high accuracy (cow, wolf for instance), whereas some words low. It means during the training time of model words cow, wolf were used in sentences which were different from the constitution text. To convert word into vector skip gram model was used. Section 3.1.1 describes how skip gram model works.

# Skip gram model

Skip gram model is necessary to represent word in vector form. Current subsection describes how skip gram model works. Skip gram model to represent word in vector form uses neural network.

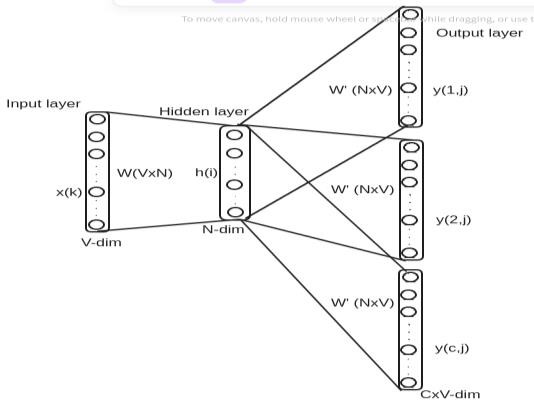
1. From dataset sentences which have word trying to find vector are retrieved. Duplicate words are deleted. As a result, unique words will be input layer of neural network. The structure of neural network can be seen from figure 3.1.

Figure 3.1 – The structure of Skip gram model

In figure 3.1 x is word of input layer. One neuron represents one word. W is weight between neurons. H is neuron of hidden layer. Y is neuron of output layer. V is number of unique words. C is size of window.

1. In input layer all neurons will have 0 values, except neuron of target word. That neuron will have value 1.
2. Weights between input layer and hidden layer, hidden layer and output layer are assigned by random numbers from 0 to 1.
3. Products of input layer and weights of hidden layer(weights between input layer and hidden layer) are calculated by formula (3.1).

|  |  |
| --- | --- |
|  | (3.1) |

1. Output of output layer is calculated by formula (3.2).

|  |  |
| --- | --- |
|  | (3.2) |

Here W is weight between hidden layer and output layer.

6. Softmax function is applied to values of output layer.

|  |  |
| --- | --- |
|  | (3.3) |

where wc,j j-th word of c-th context of output layer.

wo,c c-th word of output layer.

wi target word in input layer.

Number of neighbors of target word defines number of context. One context represent one neighbor. yc,j is probability of being neighbor j-th word of c-th context.

7. After that for each neuron of output layer error is calculated by formula (3.4).

|  |  |
| --- | --- |
|  | (3.4) |

where tc,j can have one or zero value. If j-th word of c-th context is neighbor of c-th context, then it will have value one, otherwise zero.

8. Then all errors of output layer must be summed.

|  |  |
| --- | --- |
|  | (3.5) |

Here c is number of context.

9. In neural network all weights are updated by formula 3.6.

|  |  |
| --- | --- |
|  | (3.6) |

where wi,j (new) is new weight.

wi,j(old) is old weight.

α is learning rate.

hi is value of neuron in hidden layer.

10. Steps 4-9 must be repeated until error will be small enough.

# System of defining lexical compatibilities

Interface of our system looks like in figure 3.2.

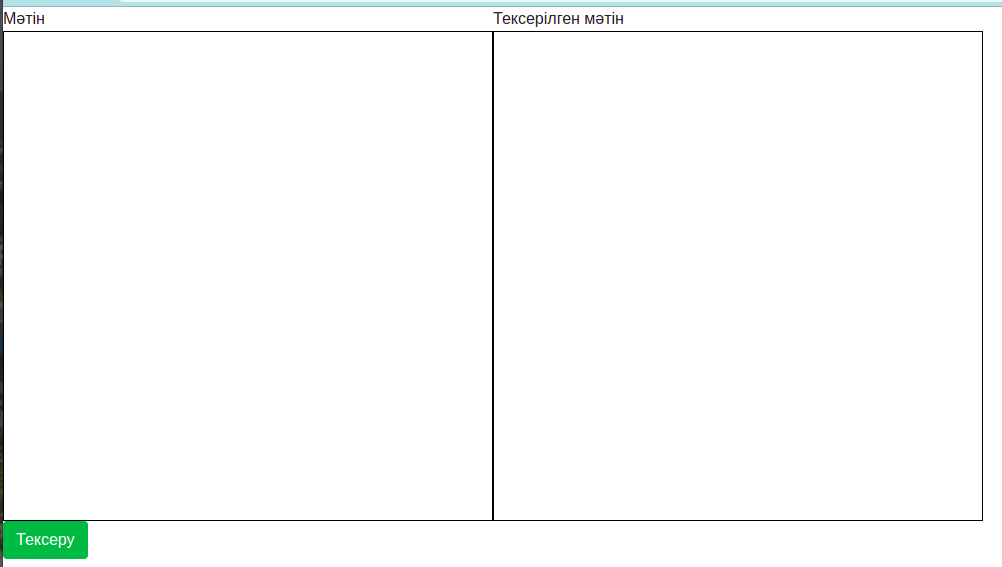


Figure 3.2 – System interface

As shown in figure 3.2 the system contains two sections: “Мәтін”, “Тексерілген мәтін”. In section “Мәтін” user should write text, which wants to check. In “Тексерілген мәтін” section user can see returned result of the system. Words which are defined as not lexical compatibles will be marked as bold text. Figure 3.3 contains an example of result.

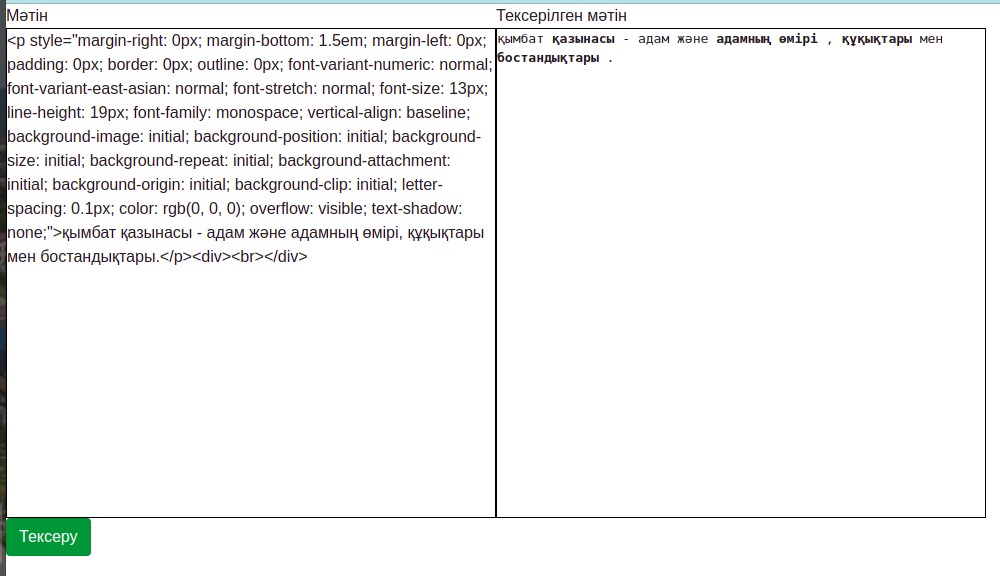


Figure 3.3 – Example of returned result

Figure 3.3 shows that tokens (қымбат, қазынасы), (адамның, өмірі), (өмірі), (құқықтары), (мен, бостандықтары) are not lexically compatibles. After clicking on “Тексеру” button entered text will be processed. All letters will take lowercase forms. The system has predictions.txt file, which contains list of lexically compatible words. Each word has five lexically compatible words. For instance: мемлекеттік: ['асырады', 'істің', 'басқаруды', 'жергілікті', 'атқарушы'];. The system will search entered word from predictions.txt file. If it finds, the next entered word will be searched from the five compatible words. In case of finding, the next entered word will not be marked as bold, otherwise will be marked as bold. In figure 3.3 the first word is “қымбат”. Predictions.txt file for word “қымбат” has lexically compatible words like 'орнықтырады', 'ең', 'зайырлы', 'демократиялық', 'қымбат'. This list doesn’t contain “қазынасы” word(the next entered word), hence “қазынасы” will be considered as not lexically compatible word and will be marked as bold. Table 3.2 contains other examples.

Table 3.2 – The system’s returned result

|  |  |  |
| --- | --- | --- |
| Entered text | Returned result | |
| Қазақстан Республикасының  Парламенті сенат және мәжілістен тұрады | Қазақстан *республикасының парламенті сенат және мәжілістен*  тұрады | |
| президент 7 жылға сайлана алады | президент *7 жылға сайлана*алады | |
| сайлау нәтижесі шықты | | *сайлау*нәтижесі шықты |
| Халық сайлау нәтижесімен таныс  болды | *Халық сайлау* нәтижесімен таныс  болды | |
| адам және қоғам | адам *және*қоғам | |

If predictions.txt file doesn’t contain entered word, then the next word will be considered as compatible.

# Conclusion of the experiment

In current experiment word embedding was analyzed. In word embedding words which are used in one sentence should be located close to each other. For instance, consider ball, apple, banana words. Apple and banana are fruits. Therefore, they can be used in one sentence. So in word embedding apple and banana should be close to each other, but far from ball (because word ball is used in other context). In current experiment word embedding was checked on how well it was trained. To check it in text of constitution of Republic of Kazakhstan extra words were added. That words not related with the constitution. Designed system had to find out that extra words. To find out extra words cosines between corresponding vectors were calculated. As conclusion of the experiment I propose accuracy of the experiment directly related with dataset, on which word embedding was trained. To evaluate the system constitution of Republic of Kazakhstan was used, hence word embedding had to be trained on it.

# Stemming of the Kazakh language

In experiment important step is preprocessing of data. Preprocessing is important because it can influence on final result. Preprocessing can be done by various ways. For instance, words with few frequencies can be deleted from sentences or misspelled words can be corrected and so on. In second experiment to preprocess user’s query stemming was applied. Stemming is a technique of defining root of word. To define root dictionary should be used which contains list of roots. According to stemming technique the last characters will be removed until the root will be found from the dictionary [50]. Root of word is necessary to expand user’s query by semantically close words. Therefore, it was done in second experiment. For this experiment [51] paper was published.

According to [28, p. 501-507] the Kazakh language has 10 parts of speech. In current experiment noun, verb, adjective, numeral parts of speech were analyzed. Figures 3.4, 3.5, 3.6, 3.7 show how suffixes and endings can be added to noun, verb, adjective, numeral respectively.

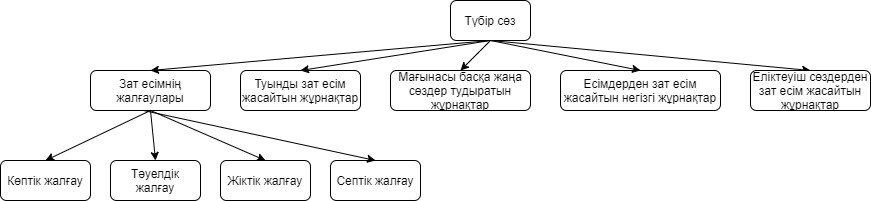


Figure 3.4 – Concatenation of noun with suffixes and endings

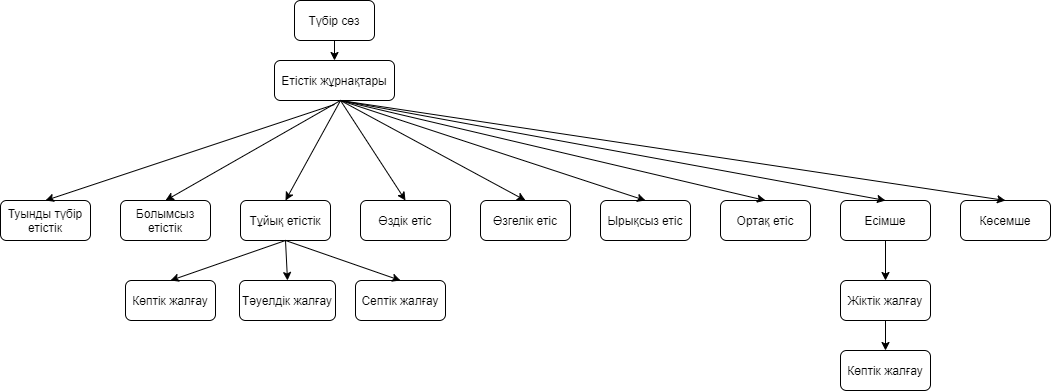


Figure 3.5 – Concatenation of verb with suffixes and endings

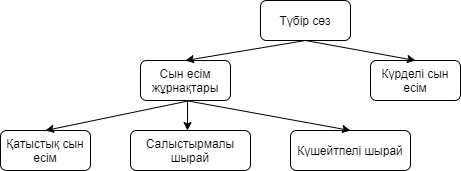


Figure 3.6 – Concatenation of adjective with suffixes



Figure 3.7 – Concatenation of numeral with suffixes

From figures 3.4-3.7 can be seen what kind of suffixes and endings can be added nouns, verbs, adjectives, numerals. However, this is not completely list. For instance root of “ұстаздарға”(to teacher) word should be defined. According to figure 3.4 noun can be added with dative case. Hence ға ending will be removed and the word become ұстаздар. From figure 4 also can be seen that noun can be joined with plural ending. Hence дар ending will be removed and the word become ұстаз. Then from the dictionary of root list ұстаз will be found. One drawback of this approach is that to define root it doesn’t consider neighbour words. Therefore, for one word several roots can be defined. Table 3.3 contains examples of it.

Table 3.3 – Examples of system responses

|  |  |
| --- | --- |
| Entered word | System response |
| 1 | 2 |
| Тастардың | ТАСТА+Verb+Pos+Қатыстық сын есім+Өзгелік етіс+Есімше+Pnon+Gen+A3Sg+Sg  ТАС+Noun+Verb+A3pl+Pnon+Gen |
| Тастармен | ТАСТА+Verb+Neg+Қатыстық сын есім+Өзгелік етіс+Есімше+P1+Dat+A3Sg+Sg  ТАС+Noun+Verb+A3pl+P1pl+Dat |
| Толқын | ТОЛҚЫ+Verb+Pos+Қатыстық сын есім+Өздік етіс+Ырықсыз етіс+Pnon+Sg |
| Table continuation 2.2 | |
| 1 | 2 |
|  | ТОЛҚЫН+Verb+Pos+Қатыстық сын есім+Pnon+Sg  ТОЛҚЫН+Noun+A3sg+Pnon+Nom |
| Тау | ТАУ+Noun+A3sg+Pnon+Nom |
| Notes:  1. Sg – singular Pl – plural  2. A1, A2, A3 – type of ending (жіктік жалғау)  3. P1, P2, P3 – type of ending (тәуелдік жалғау).  4. Nom, Gen, Dat, Acc, Loc, Abl, Ins – respectively seven cases of the Kazakh language.  5. Pnon – non possessive | |

From table 3.3 can be seen that words тастардың, тастармен, толқын had by two roots. For word тастардың roots were таста(throw) and тас(stone). The same roots were defined for word тастармен. However, the word тау was properly defined. So in conclusion the stemming system defined roots properly but with additional roots which are incorrect. To avoid such additional roots neighbor words also should be analyzed.

# Conclusion of the experiment

In current experiment the system, which can define stemming of the Kazakh language was designed. From figures 3.4-3.7 can be seen what kind of suffixes, endings can be connected with root. The Kazakh language is rich with its suffixes, endings. Therefore, for one word root can be defined differently. For instance, for word “таста” root can be тас(stone) or таста(throw). By stemming these two candidates can be defined. However, to find out correct answer adjacent words also should be analyzed. For example, in sentence “Қоқысты жәшікке таста.” (throw trash in box) root will be таста(throw). Because таста in sentence the last word, hence it is verb. In case “Балалар таста отыр.” (children are sitting on the stone). Analyzing adjacent words root can be defined as тас(stone). As conclusion of the experiment I propose performance of stemming depends on language. If language contains words with unique meanings, then stemming can properly identify root of word. In case language has words with several meanings (one word with several roots) then stemming just can identify candidates of roots. To properly define root adjacent words also should be analyzed. This technique is called lemmatization.

# Defining semantically close words of Kazakh language by distributed system Apache Spark

The next experiment was about how semantically close words can be defined. It was done to expand user’s query. Semantically close words, words which can be used in one sentence. However, they can have completely different meaning. For instance, words like monitor, mouse, printer, computer can be used in one sentence hence considered as semantically close words. I think words like window, silk, pen are not so often used in one sentence. Therefore, they can be considered as not semantically close words. However, there is no strict rule which can divide words on semantically close and not close groups. In current experiment words were represented by vectors. Particularly, it was taken from [49]. To define semantically close words cosines between vectors were calculated and chosen ten maximum values. Calculating cosine between pairs of vectors took long time. Therefore, another aim of the experiment was reducing calculation time by distributed system Apache Spark.

# 2.3.1 Defining similarities of words

Similarities can be measured by various metrics like edit distance, Euclidean distance, cosine values.

# Edit distance

Edit distance calculates difference between two strings. By other words edit distance counts minimum number of operations to convert one string to the second string. Possible operations are insertion, deletion, substitution. By these three operations the string has to be converted. For instance, the distance between “code” and “coding” is 3.

Because to get word “coding” letter e should be substituted by letter i. Then letters n, g should be added. So three operations were done (substitution, insertion, insertion), therefore the answer will be 3. Edit distance is useful to fix spelling mistakes.

Edit distance can be implemented recursively. In implementation various test cases should be considered. For example, the first string can be empty, like “”, “cat”. Here the answer should be 3(three times insert operation). The last characters of two strings can be the same, like coding, testing. In that case the distance should be calculated between cod and test.

# Euclidean distance

Euclidean distance is another metric of measuring similarity. Euclidean distance calculates minimum distance between two points. By other words Euclidean distance is the length of straight line which connects two points. To calculate the length of line following formula is used:

|  |  |
| --- | --- |
|  | (3.7) |

where p, q are two points between which distance will be calculated.

pi, qi are coordinates in i-th plane.

n is dimension of space. Figure 3.8 contains an example of how Euclidean distance can be calculated.

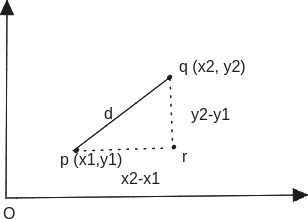


Figure 3.8 – Example of a Euclidean distance

In figure 3.8 d is a Euclidean distance. Point p has coordinates (3,3), whereas point q has coordinates (8, 7). Hence by formula 3.7 d=sqrt((8-3)\*(8-3)+(7-3)\*(7-3))=sqrt(41). Euclidean distance can be good metric for measuring similarity, because in word embedding semantically close words can be close to each other.

# Cosine similarity

The next similarity metric is called cosine similarity. Cosine similarity measures cosine of angle between two vectors. Figure 3.9 shows how cosine can be calculated between two vectors.

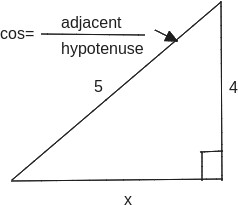


Figure 3.9 – Cosine of two vectors

As shown above cosine is a fraction of adjacent to hypotenuse. Therefore, its range lies from -1 to +1. Cosine takes positive values in range [0o; 90o] and [270o; 360o), negative values in range (90o; 270o). Figure 3.10 shows how cosine function changes in [-2π, 2π] range.

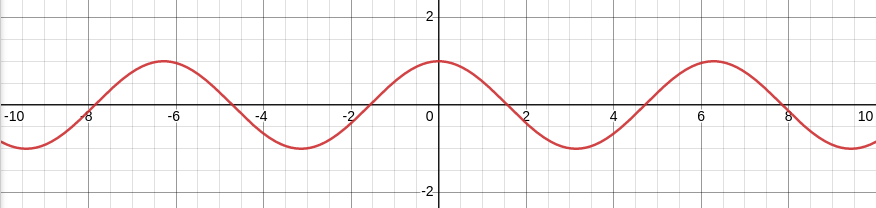


Figure 3.10 – Cosine function

In word embedding semantically close words should be located close to each other (i.e. angle between two vectors is low). Hence cosine value for semantically close words should be high. Figure 3.11 demonstrates it.

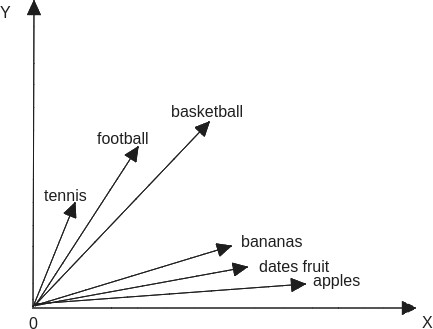


Figure 3.11 – Word embedding of semantically close words

In figure 3.11 points represent words. In addition, figure 3.11 contains two classes: games, fruits. Words tennis, football, basketball are belong to games class. Therefore, that words are located close to each other and have small angles. When angle is small cosine has high value. This technique also works for fruits class.

# Conclusion of the experiment

In current experiment for one word ten semantically close words were defined. Semantically close words were defined by calculating cosines between vectors. From the list of cosine values ten maximum cosines were chosen. The result of this experiment directly depends on coordinates of vectors. Hence word embedding should be correctly trained.

# Classification of Scientific Documents in the Kazakh Language Using Deep Neural Networks and a Fusion of Images and Text

Search engine should return not only relevant result but in short time as well. Therefore, to return the response faster documents should be classified. The next experiment was about classification of scientific documents. As scientific documents were considered papers of KazNU journal. This journal had series like ecology, philology, mathematics, mechanics, computer science, biology, geography. In current experiment papers of the last three series were classified. Particularly they were classified only by text, only by images, fusion of text and images.

# Experiment Dataset

All papers were taken from mathematics, mechanics, computer science, biology, geography series of KazNU journal. Number of papers was 140(40 papers of mathematics, mechanics, computer science series, 50 papers of biology series, and 50 papers of geography series). To classify papers by text only abstracts were used. It was done because papers were written in different languages but all abstracts were written in Kazakh. Each paper had at least one image and at most twenty-two images. In experiment the dataset was divided into training (110 papers) and testing (30 papers) samples using stratified random sampling.

# Classification of papers by text

In current experiment the text was classified by Naive Bayes algorithm. Naive Bayes algorithm worked as following:

1. Each document belongs to one class. In experiment there were three classes: mathematics, mechanics, computer science (it was name of one series, hence it was one class), biology, geography. Each paper was labeled by these three classes.
2. From all papers unique words were defined.
3. Then counted how many times repeated each word in each paper.
4. Probability of each class was computed. As it was mentioned above number of mathematics, mechanics, computer science papers were 40, biology was 50, geography was 50. Hence probability (mathematics, mechanics, computer science)= 40/(40+50+50)=40/140≈0.286 probability(biology)=50/(40+50+50)=50/140≈0.357 probability(geography)=50/(40+50+50)=50/140≈0.357
5. For each class probability of word was calculated.

# *Step 1*

Following example summarizes steps above. Table 3.4 contains papers of three classes.

Table 3.4 – Papers with three classes

|  |  |  |
| --- | --- | --- |
| Paper | Text | Class |
| 1 | Capital of Kazakhstan is Astana. | Geography |
| 2 | In Caspian Sea there are various fishes like  Caspian kutum, Brown trout. | Geography |
| 3 | The Kazakh mountains can be divided as Alpine  and low regions. | Geography |
| 4 | In Kazakhstan there are mountains like Kok-Tobe  Hill, Bolektau, Bektau-Ata. | Geography |
| 5 | The amount of water in human body can be  changed depending on age, gender. | Biology |
| 6 | Bees produce honey, which is useful for human. | Biology |
| 7 | The pomegranate is a fruit which belongs  Lythraceae family. | Biology |
| 8 | Biology studies the structure of cells. | Biology |
| 9 | Minimum Spanning Tree is algorithm for  defining minimum path among all nodes of tree. | Mathematics,mechanics,  computer science |
| 10 | In neural network activation function is used to  define input for the next layer. | Mathematics, mechanics,  computer science |

# *Step 2*

*Unique words:*‘trout', 'for', 'activation', 'Hill', 'amount', 'divided', 'various', 'is', 'which',

'used', 'path', 'Kok-Tobe', 'spanning', 'fishes', 'Caspian', 'age', 'among', 'network', 'Astana',

'define', 'in', 'cells', 'input', 'regions', 'Bektau-Ata', 'tree', 'water', 'the', 'belongs’,

’Lythraceae', 'and', 'kutum', 'next', 'be', 'useful', 'to', 'Sea', 'brown', 'Kazakhstan',

'Bolektau', 'Alpine', 'capital', 'of', 'bees', 'nodes', 'a', 'there', 'gender', 'layer', 'produce',

'human', 'function', 'algorithm', 'minimum', 'body', 'mountains', 'honey', 'biology', 'fruit', 'neural', 'as', 'studies', 'changed', 'family', 'structure', 'are', 'on', 'defining', 'like', 'pomegranate', 'depending', 'all', 'can', 'Kazakh', 'low'. So number of unique words 75.

# *Step 3*

In step 2 unique words were defined. In step 3 that unique words should be counted in each paper. Table 3.5 contains an example of it. In table 3.5 classes were written by 1, 2, 3 numbers which represent geography, biology, mathematics, mechanics, computer science classes respectively. In addition, table 3.5 doesn’t contain all unique words, however according to Naive Bayes algorithm all unique words must be counted.

Table 3.5 – Frequency of words in each paper

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Paper | Trout | For | Activa tion | Hill | Amount | Divi ded | Va rious | Is | Which | Used | Class |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 2 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 3 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 4 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 5 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 2 |
| 6 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 2 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 2 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| 9 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 3 |
| 10 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 3 |

# *Step 4*

probability (mathematics, mechanics, computer science)= 40/(40+50+50)=40/140≈0.286 probability(biology)=50/(40+50+50)=50/140≈0.357 probability(geography)=50/(40+50+50)=50/140≈0.357

# *Step 5*

In step 5 probability of each word in each class should be calculated. To calculate probability following formula should be used:

|  |  |
| --- | --- |
|  | (3.8) |

where n(k) is number of k word in current class. n is number of words in current class. Vocabulary is number of unique words. For instance, below for word “is” probabilities were calculated.

p(w(is)|geography)=(1+1)/(5+10)=2/15 ≈0.13 p(w(is)|biology)=(2+1)/(6+10)=3/16 ≈0.1875

p(w(is)|mathematics,mechanics, computer science )=(2+1)/(6+10)=3/16 ≈0.1875

Table 3.5 contains not all words, however in theory it should contain all words. The above calculation was used table 3.5. In geography class (1+1) was written because in geography class “is” word met one time. 5 is sum of words in geography class. 10 is number of unique words. By this way probabilities of the rest words should be calculated.

# *Step 6*

This step contains testing data. Testing sentence will be “Biology studies cells”, i.e. should be defined to which class this sentence belongs to. As it was mentioned in step 5 for all words probabilities should be calculated. From the list of probabilities, probabilities of words biology, studies, cells should be collected. As an example they will be:

– p(w(biology)|geography)=0.5 p(w(biology)|biology)=0.75;

– p(w(biology)|mathematics,mechanics, computer science )=0.3;

– p(w(studies)|geography)=0.4 p(w(studies)|biology)=0.65;

– p(w(studies)|mathematics,mechanics, computer science )=0.7;

– p(w(cells)|geography)=0.2 p(w(cells)|biology)=0.8;

– p(w(cells)|mathematics,mechanics, computer science )=0.1.

Then probabilities should be multiplied like following: class1=probability(geography)\*p(w(biology)|geography)\*p(w(studies)|geography)\*p (w( cells)| geography) class2=probability(biology)\* p(w(biology)|biology)\* p(w(studies)|biology)\*p(w(cells)|bio logy) class3=probability (mathematics, mechanics, computer science)\*p(w(biology) |mathematics, mechanics, computer science)\*p(w(studies)|mathematics, mechanics, computer science)\* p(w(cells) |mathematics, mechanics, computer science)

Hence class1=0.357\*0.5\*0.4\*0.2=0.01428 class 2=0.357\*0.75\*0.65\*0.8 =0.13923 class3=0.286\*0.3\*0.7\*0.1=0.006006

From these values class2 has maximum value, therefore the testing sentence will be classified as biology class. Steps above explain how Naive Bayes algorithm works and how it was used in current experiment.

# Neural network

Neural network is a kind of network where units represented by neurons. By neural network various problems can be solved. For instance, by neural network classification, image recognition problems can be solved. Neural network consists of layers. Usually the first layer gets some input data like pixels. Therefore, the first layer is called input layer. The last layer is called output layer and it can return answer of problem (for example the name of class). Layers which are located between input and output layers called hidden. Neural networks can have different number of layers and each layer can have different number of neurons. Figure 3.12 shows several architectures of neural network.

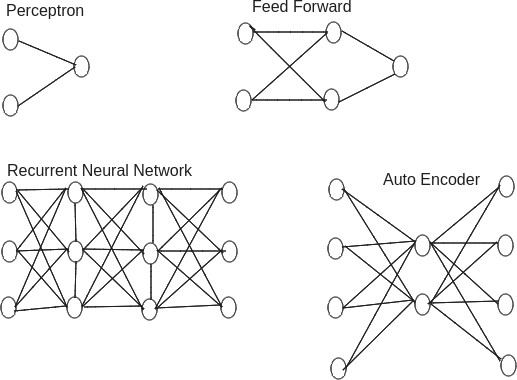


Figure 3.12 – Different architectures of neural network

Neurons of neural network can be differently connected, i.e. some neurons can be strongly connected, whereas some neurons can be weakly connected. Hence to express strongness of connection weights are used. Weight is used between two neurons to show their strongness in numeric form. In neural network neuron gets some input then returns some value. In addition, input value and output value can be different. It depends on activation function. Activation function is function, which determines output value. Figure 3.13 summarizes statements above.



Activatio n function

Input value Output value

Figure 3.13 – Activation function

Nowadays there are various kinds of activation functions like sigmoid, relu, linear. Activation function can influence on performance of model. Therefore, activation function should be properly chosen in solving problem.

# Activation function in neural network

Activation function is a function, which determines value for given input [52]. In neural network each layer can have different activation function. It depends on the task, which needed to be solved. Shapes of graphs of sigmoid and tangent functions are similar. However sigmoid function returns only positive values, whereas tangent returns positive and negative values. For instance, sigmoid function can be used in classification problems. Sigmoid function will return probabilities of classes. Then by certain threshold value the final class will be determined.

Linear function can be used when some process linearly changes. For instance, price of some product can be observed for some period of time.

ReLU function is similar to linear function, however it compares x with 0 and returns maximum value between them. In current experiment ReLU function was used as an activation function.

# Stop conditions in neural network

In neural network connections between neurons can be different. Hence their weights will have different values. In addition, weights are adjusted until model reach sufficient result. This adjustment process is called training of model. After training of model usually the model is tested by other dataset. This process is called testing of model. As it was stated above weights of neural network need to be adjusted until the model reach sufficient result. However sometimes training of model can take long time. Therefore, training process can be stopped by several conditions.

The first condition is number of epochs. Here developer himself / herself defines number of epoch. When number of epochs reaches defined value training process will be stopped. By this way time can be saved.

The second condition is defining threshold for the model. For example, threshold can be number of right answers returned by the model.

In current experiment number of epochs was used to stop training of model.

# Popular neural network architectures

Neural network has various architectures which can be used to solve specific task. Neural network has architectures like perceptron, convolutional neural networks, recurrent neural networks, long/ short term memory [53].

The perceptron architecture is sometimes called feed-forward neural network. The perceptron architecture gets some input, process it by an activation function, and returns an answer. In each epoch the model should try minimize the difference between actual value (the right answer) and returned value. In perceptron each input value is multiplied to corresponding weight. Then their sum is calculated. Then calculated sum will be given to activation function, which produces output value. This value should be compared with the right answer. If they are equal, there is no need to update weights. Otherwise weights should be updated.

The next architecture is called convolutional neural network(CNN). CNN can be used with various kind of inputs like audio signals, images. However, CNN basically is used for classification of images. In current experiment CNN was also used to classify images of papers. In CNN if image has size 100x100 pixels, it doesn’t mean that the input layer will have 10000 neurons. To prevent large amount of neurons in input layer, CNN uses window, which moves through the image. In addition, CNN has filters like max pooling, average pooling, which produce one pixel from set of pixels.

The next popular architecture is called recurrent neural network(RNN). The structure of RNN is similar to perceptron, i.e. it consists of perceptron. However rather than perceptron RNN can store previous states. That’s why in some tasks using RNN can be more efficient.

The next architecture is long/short term memory(LSTM). LSTM can be used in vanishing gradient problems. LSTM has gates like input, output, forget. Input gate is used to add new information to the cell. Output gate decides when to transmit cell’s vector to next hidden layer. Forget gate manages memory cell. It can delete some data from memory cell.

# Classification of papers by images

In current experiment convolutional neural network(CNN) was used to classify images. Each paper had at least one image and at most twenty-two images. Original sizes of images were different. However, because the architecture of neural network needed to have equal lengths, sizes of the images were changed. In input layer sizes of the images were 128x128px. Output layer of CNN model had three neurons, where each neuron represents one class. In CNN model batch size was 64. An activation function was softmax. Softmax function is defined by formula 3.9.

|  |  |
| --- | --- |
|  | (3.9) |

Where yi represents input neuron, n is number of classes. The CNN model trained for 20 epochs.

# Classification of papers by text and images

In current experiment the third type of classification was using text and images. For the classification the same text and images were used as they individually were classified. To classify scientific papers by text and images again CNN was used. However the structure of CNN architecture was different. It had two input layers: one for image, one for text. Then images were concatenated with text. To represent text in numeric form Word2Vec Continuous Skipgram model was used [49]. This model contained 57048825 tokens and its dimension was 100. Figure 3.14 shows the structure of CNN model, which was used in the experiment.

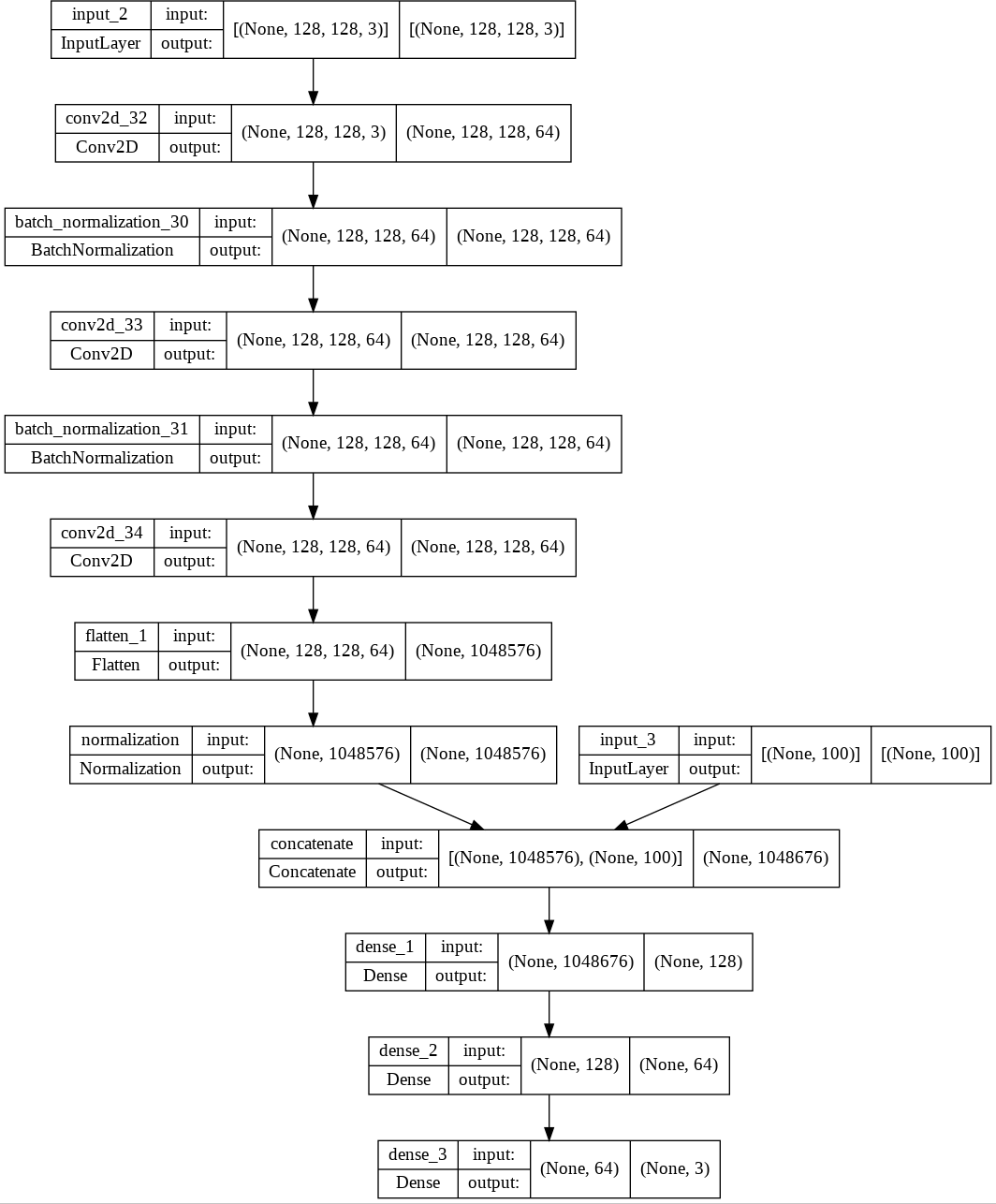


Figure 3.14 – The architecture of CNN model

From figure 3.14 can be seen that the size of images was 128x128 pixels. Output layer had three neurons because there were three classes (mathematics, mechanics, computer science, biology, geography).

Classifying scientific papers by text and images showed higher accuracy rather than classifying papers only by text, or only by images. Table 3.6 shows accuracies of models.

Table 3.6 – Accuracies of three models

|  |  |
| --- | --- |
| Classification | Accuracy(%) |
| Naive Bayes | 83.33 |
| CNN model with pure images | 83.23 |
| CNN model with pure images and text | 87.68 |

According to table 3.6 Naive Bayes and CNN model with pure images approximately showed the same accuracy. In CNN architecture batch size also can influence on the accuracy. Figure 3.15 illustrates how accuracy was changed in different batch sizes.

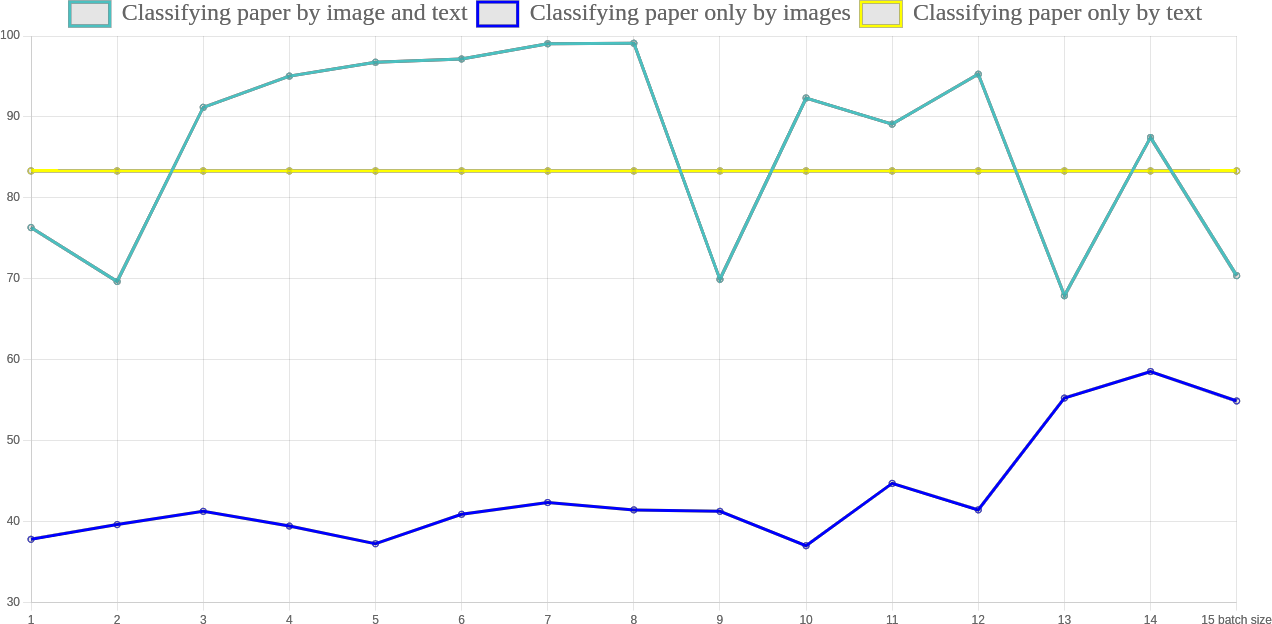


Figure 3.15 - Accuracies in different batch sizes

Batch size is number of images which will be given to the network to train it [54]. For instance, there are 1000 images in dataset. As an example batch size will be 300. In this case [1-300] images will be given to the network and train it. After training the next 300 images (301-600) will be given to the network and so on. At the end 100 images will be given. From figure 3.15 can be seen that batch size is important parameter in classification task. However, from figure 3.15 can be seen that large batch size not always can lead to high accuracy.

# Conclusion of the experiment

Current experiment was about classification of scientific papers. There were three classes: Mathematics, mechanics, computer science(one class), geography, biology. In experiment papers were classified by three methods: only text, only images, combination of text and images. For three methods respectively Naive Bayes, CNN, CNN models were used. Classification by only text and only images approximately showed the same result (83.33%, 83.23% accuracies respectively). Combination of text and images showed slightly better result (87.68% accuracy). Hence as conclusion I propose to raise accuracy of model various kinds of information can be combined. CNN model can have several inputs. Each input can be used for one kind of information. After that they can be concatenated.

# Experiment with FAISS

In current thesis to investigate FAISS library small experiment was done. Particularly by FAISS library for entered query relevant title of Wikipedia article needed be returned. In experiment Colab environment was used. Therefore, to use FAISS, it should be installed in Colab. After installation, dump file of Wikipedia articles was read. However, Wikipedia articles contained extra symbols. To avoid that extra symbols and read the articles properly wiki\_dump\_reader software was used. In experiment 165592 Wikipedia titles were used. The experiment was done on Python programming language. SentenceTransformer class was used as a model. It took kz-transformers/kaz-roberta- conversational string as an argument. Figure 3.16 summarizes the statement above.

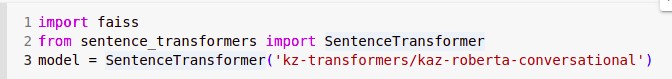


Figure 3.16 - SentenceTransformer model

The dataset of datatransformer was taken from various sources like CC-100: Monolingual Datasets from Web Crawl Data, books, Leipzig Corpora Collection, Open Super-large Crawled Aggregated coRpus, Kazakh News. The dataset had 2 091 564 186 tokens, and 17 802 998 unique tokens [55]. After defining the model titles of Wikipedia articles were encoded by encode method. Queries were encoded by the same way. To evaluate FAISS library various queries were entered. Table 3.7 contains that queries.

Table 3.7 – Returned result by FAISS

|  |  |
| --- | --- |
| Query | Returned five Wikipedia titles |
| Журнал (Journal) | Журнал(Journal), Журнал жүргізу(Managing journal), Газет(Newspaper), Журналистік сауал(Journalist question), Газеталар(Newspapers) |
| Көлік (Transport) | Көлік(Transport), Көлік малы( Cattling transport), Көлік майы(Oil for transport), Көлік құрылысы(Transport structure), Автокөлік(Car) |
| Спорт (Sport) | Спорт(Sport), Спорт классификациясы(Sport classification), Спорт орталығы(Sport center), Спорт газеті(Sport newspaper), Спорт жабдықтары(Sport equipment) |
| Жүзімді өсіру (Growing grapes) | Жүзім шаруашылығы(Viticulture), Гүл өсіру(Breeding flowers), Ұзын өркенді өсіру(Breeding long stem), Жылқы өсіру (Breeding horses), Ит өсіру (Breeding dogs) |
| Ағашты бояу (Painting tree) | Ағашты бояу(Painting tree), Ағаштың безі(Tree gland), Ағаштарды сақиналау(Ringing tree), Ағашты(tree), Ағаш өңдеу(Tree processing) |

From table 3.7 can be seen that FAISS can return titles which don’t contain query words. For instance, for query “transport” it returned “car”, for query “journal” it returned “newspapers”.

# Conclusion of the experiment

From table 3.7 can be seen that FAISS library can do morphological analysis for the Kazakh language. For instance, the last query was Ағашты бояу (Painting tree). For this query FAISS returned titles like Ағаштың безі (Tree gland), Ағаш өңдеу(Tree processing). The titles contain words which are not the same as words of query. This is advantage of FAISS library. In addition, if query contains several words FAISS will search that words separately, not wholly.

# Experiment with Elasticsearch

The experiment with Elasticsearch was similar to the experiment with FAISS. For entered query Elasticsearch needed return relevant title of Wikipedia article. The experiment was performed in Colab environment. To use Elasticsearch it should be installed in Colab environment. After installation and running Elasticsearch by ping method was checked connection between python and Elasticsearch. It was run on localhost and 9200 port number. After that dataset of Wikipedia titles was read. The dataset was in comma-separated values(csv) format. It contained 123185 titles. After reading Wikipedia titles, they needed be imported to Elasticsearch. To do it the format was changed to json format. To import dataset bulk method was used. Various queries were entered in Elasticsearch, and it returned interesting result. Table 3.8 shows entered queries and returned result by Elasticsearch.

Table 3.8 – Returned result by Elasticsearch

|  |  |
| --- | --- |
| Query | Returned five wikipedia titles |
| Журнал (Journal) | Мүғалім(журнал) (Teacher (journal)), Time(журнал) Time (journal), Тайм(журнал) Time(journal), Электронды журнал (Electronic journal), Айгөлек(журнал) Aigolek (journal) |
| Көлік(Transport) | Көлік географиясы(transport geography), Көлік және коммуникация колледжі(College of Transport and Communication), Көлік техникасының энергетикалық қондырғылары(Energetic equipment of transport), Оңтүстік Қазақстан облысының көлік саласы(Transport aspects of south area of Kazakhstan), Батыс Еуропа – Батыс Қытай (көлік жолы) Western Europe – Western China(car road) |
| Спорт(Sport) | Иенген Спорт(Hienghène Sport), Байдарка(спорт) Kayak(sport), Спорт алаңы (Sport field), Спорт құрылысы (Sport construction), Спорт ғимараттары (Sport buildings) |
| Жүзімді өсіру(Growing grapes) | Бақ өсіру(Growing garden), Мақта өсіру(Growing cotton), Нутрия өсіру(Growing nutria), Құс Өсіру (Growing bird), Жүзімді ауылдық округі(Juzimdi rural district) |
| Ағашты бояу(Painting tree) | Бояу(матаны бояу) Painting( painting cloth), Бояу қызыл тамыр (Painting by macrotomіa ugamensіs), Қышқылдық бояу(Acid dye), Бояу қызылтамыры (Painting by macroto mіa ugamensіs), Томар Бояу Кермек(Siberian statice) |

From table 3.8 can be seen that if query has two or more words title of article could contain some of them. Not all words.

# Conclusion of the experiment

From table 3.8 can be seen that Elasticsearch library not defines root of word. For entered query Elasticsearch returned titles which contain query words. It can be seen from table 3.8. However, the most titles were relevant.

# DESIGNED KOMEKSHY SYSTEM

This section describes developed Komekshy system. Komekshy is search engine, which can help people doing research. Therefore, the system is called Komekshy. However, limitation of Komekshy is it can search only Wikipedia articles. The next subsections describe how Komekshy works, demonstrates its interface. Wikipedia articles were chosen as dataset, because Wikipedia provides free dump files. Dump files contains Wikipedia articles sorted by languages. Komekshy is a system, which focuses on the Kazakh language. Chapter three described experiments of this thesis. Komekhsy was developed on the basis of that experiments. Komekshy is web application, which was designed by django framework. Django is framework of Python programming language. Django provides a lot of ready tools which allow avoiding writing code from scratch. The next subsections describe Komekshy system more detaily.

# Instructions of Komekhy system

In Komekshy system there are two roles: admin and user. Komekshy was designed by Django framework. Hence admin role has separate admin panel. In admin panel can be seen list of database tables and manipulate them. User role in first web page has text field where user can write his/her query. Then by algorithm of Komekshy system relevant Wikipedia articles will be defined and returned to user. By this way user can find necessary information.

# User role

To use Komekshy system there is no sign up in system. User freely in first web page can write query and get result. Figure 4.1 demonstrates the first web page of user’s role.



Figure 4.1 – The first web page of user role

As it is shown in figure 4.1 in text field user should enter his/her query. Because the system focuses on the Kazakh language it accepts query in Kazakh. Figure 4.2 shows an example of such query.

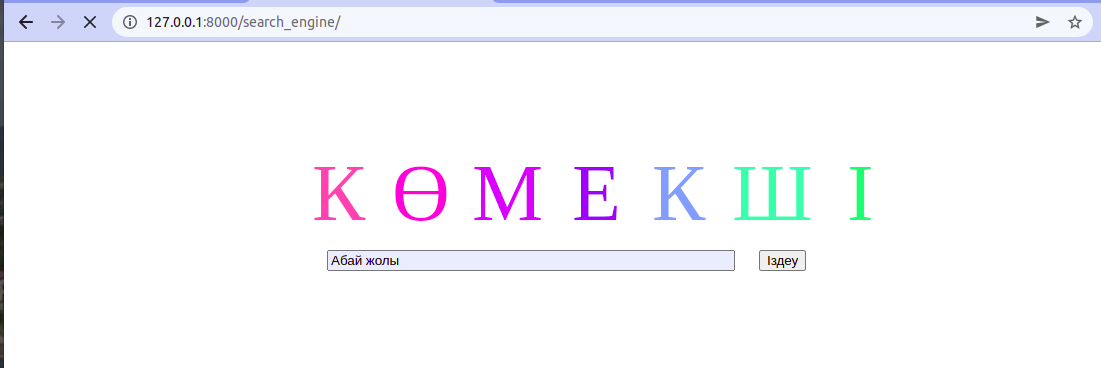


Figure 4.2 – Komekshy with user’s query

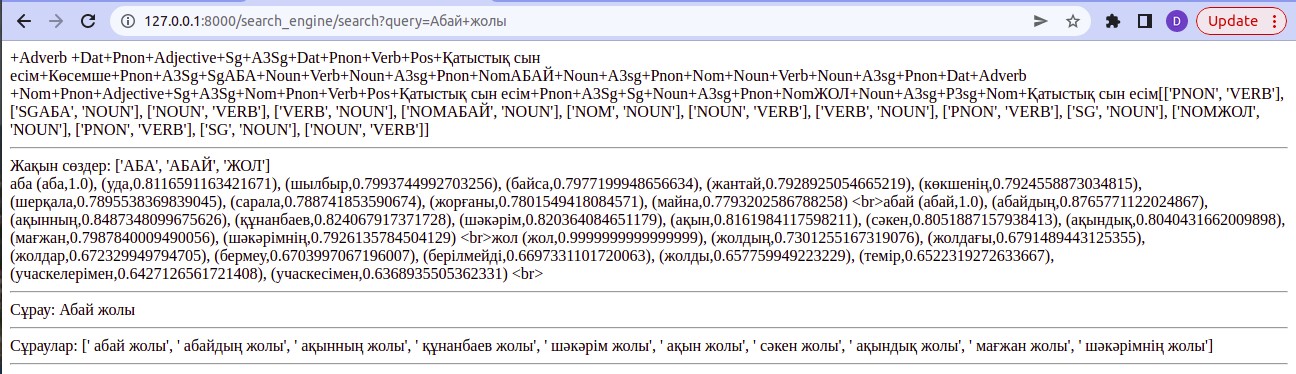
In figure 4.2 user entered “Абай жолы”. It is title of book. Author of this book is Mukhtar Auezov. After writing query user should click on “Іздеу” button. Figure 4.3 shows returned result by Komekshy system.

Figure 4.3 – Returned result for the query “Абай жолы”

From figure 4.3 can be seen that result is divided into several sections. The first section contains morphological analysis of entered query. Particularly the first section contains the list of roots of entered query. Section 3.2 contains more detailed information about morphological analysis. Komekshy defined root of word “Абай” as “Аба”,”Абай”. Root of word “жолы” was defined as “жол”.

From the first section three roots were defined “Аба”,”Абай”,”жол”. After defining roots of query Komekshy will define semantically close words to roots. The second section contains semantically close word and value, which describes on how much root and semantically close word are similar. Similarity was measured by cosine function. Semantically close words are necessary to expand query.

The next section shows original query.

The last section shows how Komekshy expanded original query. It concatenated semantically close words. Finally Komekshy will search not only words “Абай жол”, but “абайдың жолы”, “ақынның жолы”,”құнанбаев жолы”,”шәкәрім жолы”,”ақын жолы”,”сәкен жолы”, “ақындық жолы”,”мағжан жолы”,”шәкәрімнің жолы” as well. Then terminal returns IDs of articles. Komekhsy calculates relevance for each article and returns nine most relevant articles. Figure 4.4 contains IDs of articles for thequery “Абай жолы”.

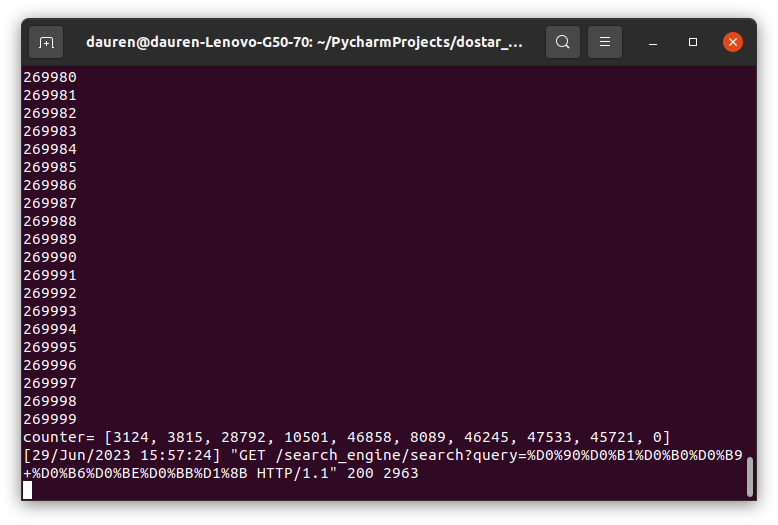


Figure 4.4 – IDs of articles for the query “Абай жолы”

Table 4.1 shows titles of Wikipedia articles and their links on the internet.

Table 4.1 – Titles of Wikipedia articles

|  |  |
| --- | --- |
| ID | Title |
| 3124 | Абай Құнанбайұлы |
| 3815 | Абай Құнанбайұлы |
| 28792 | Әуез Бердіұлы |
| 10501 | Абай жолы (роман) |
| 46858 | Абайдың философиялық көзқарасы |
| 8089 | Сәкен Сейфуллин |
| 46245 | Батыс еуропа әдебиеті |
| 47533 | Абай музыкалық мұрасы |
| 45721 | Орыс әдебиеті |

From table 4.1 can be seen that sometimes articles are repeated. The most relevant article is written first, then second and so on. This is how Komekshy works for user role. The next section describes how Komekshy works for admin role.

# Admin role

Admin role in Komekshy system can manage database. Komekshy system was developed by Django framework. In Komekshy system to open admin panel admin need to authorize. To authorize admin needs, enter username and password. Figure 4.5 demonstrates how it is looks like.

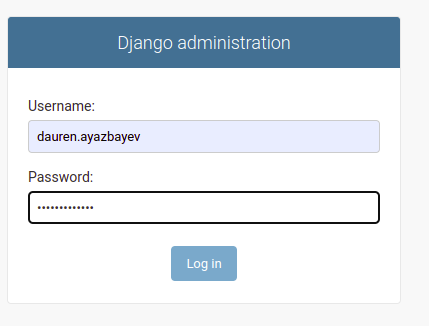


Figure 4.5 – Admin authorization

After successful authorization admin panel should be opened. By default, admin panel contains Groups, Users tables. Users table contains the list of all users in Komekshy system. Admin can edit information about user. Users table has following columns: Username, Password, First name, Last name, Email address, Active, Staff status, Superuser status, Groups, User permissions, Last login, Date joined.

Username column specifies name by which user can enter to the system. This is required column, therefore during the registration user must enter username. Username can have maximum 150 characters. Username can contain letters, digits, “@”, “.” , “+”, “-”, “\_” characters.

Password column is needed to store user’s password. However, this column stores not actual password, but hash string. This allows hide passwords from admin as well. By default, password is encrypted by pbkdf2\_sha256 algorithm.

First name, Last name columns respectively store user’s name, surname. The columns are optional. Hence during the registration user can skip these fields.

Email address column stores user’s email. Several usernames can have the same email address. It means one user can register several times.

Active column has boolean type. Therefore, for this column checkbox is presented. Active column defines whether user is active in system. To delete some account admin just need to unselect the checkbox. Otherwise select.

The next Staff status column also has boolean type and represented by checkbox.

The column defines whether the user can log into to the admin site.

Superuser status is represented by checkbox. If Superuser status has checked(selected) status the user will have all permissions. Otherwise permissions should be chosen explicitly.

Groups column contains available groups in system. By default, it has no any group.

User permissions column contains the list of permissions. By these permissions different roles can be defined. The differences between roles are these permissions. One role has certain permissions, whereas another role has different permissions.

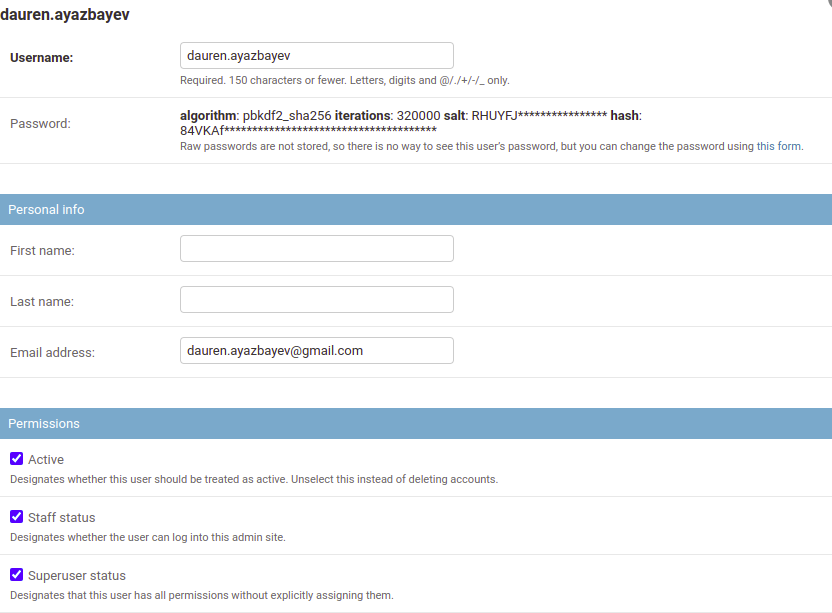
Last login column has datetime type. It stores when the user last time logged in. Date joined column also has datetime type. It stores when the account was built. Figure 4.6 shows users table of Komekshy system.

Figure 4.6 – Users table

# Architecture of Komekhy system

This section describes internal structure of Komekshy system. To explain the architecture of Komekshy system diagrams were used. Figure 4.7 shows use case diagram of Komekshy system.

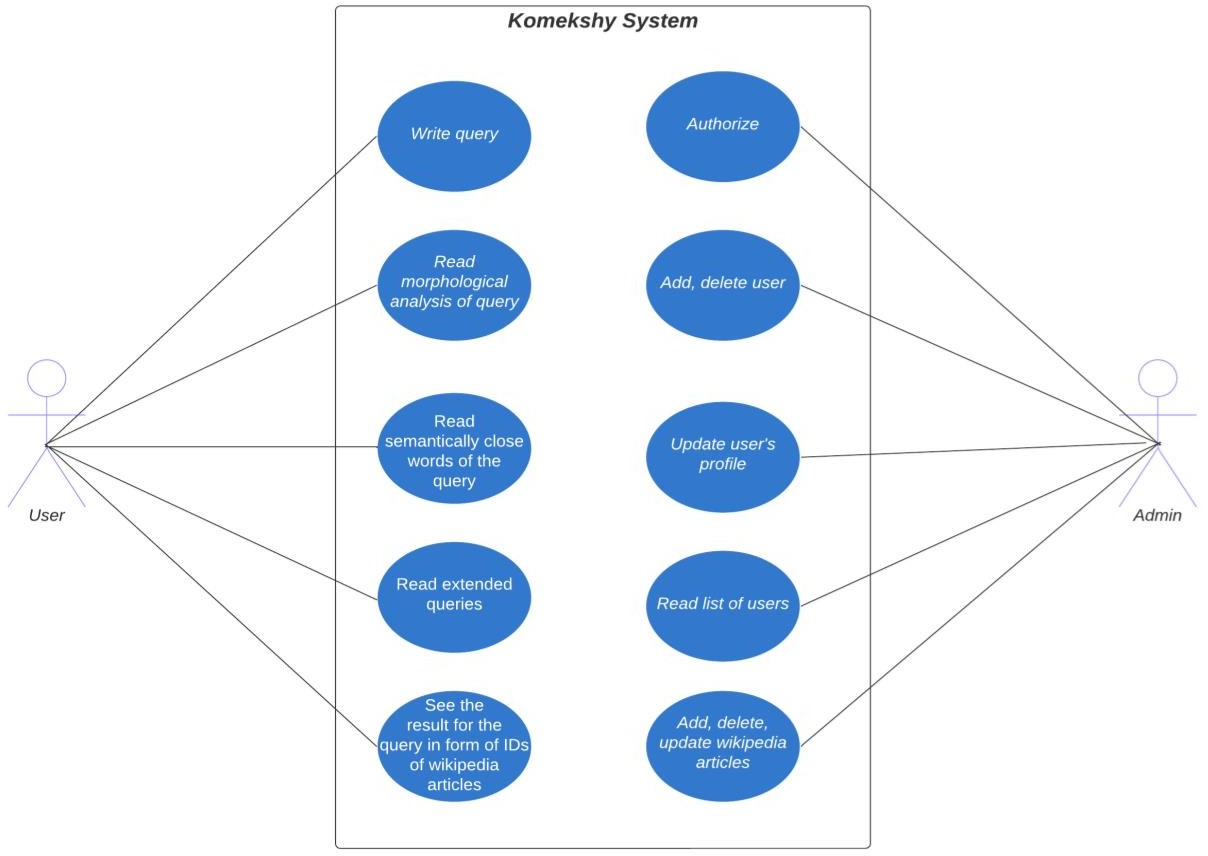


Figure 4.7 – Use case diagram

According to figure 4.7 in Komekshy system there are two roles: user, admin. In addition, the use case diagram shows what can do each role in system. Figure 4.8 shows how data is processed in Komekshy system.

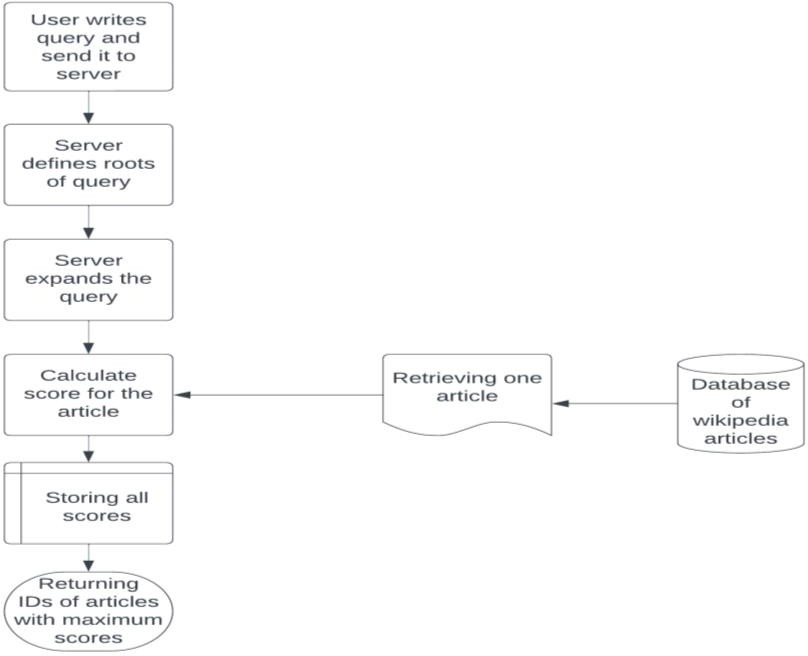


Figure 4.8 – Flowchart of Komekshy system

From figure 4.8 can be seen that server defines roots of words, which were used in query. How Komekshy defines roots is written in 3.2 section. These roots are necessary to expand the query. Particularly in Komekshy system for each root ten semantically close words are defined (if that root exist in dataset of semantically close words). Then the system will replace word with its semantically close words. By this way from one query several queries will appear. After that from database one by one wikipedia articles will be retrieved. By special formula for each article score will be calculated. Wikipedia article has title, body, class. Class is type of wikipedia article. Formula (4.1) contains that formula.

|  |  |
| --- | --- |
|  | (4.1) |

Formula 4.1 calculates score for one article. Here title\_count defines how many times title of article contains words of query. body\_count defines how many times body of article contains words of query.

class\_count defines how many times class of article contains words of query. In Komekshy system as it was mentioned above original query will be expanded. However, in programming it is one string with several queries. Then the system will return IDs of articles with the maximum scores. Figure 4.9 shows the structure of Django project.

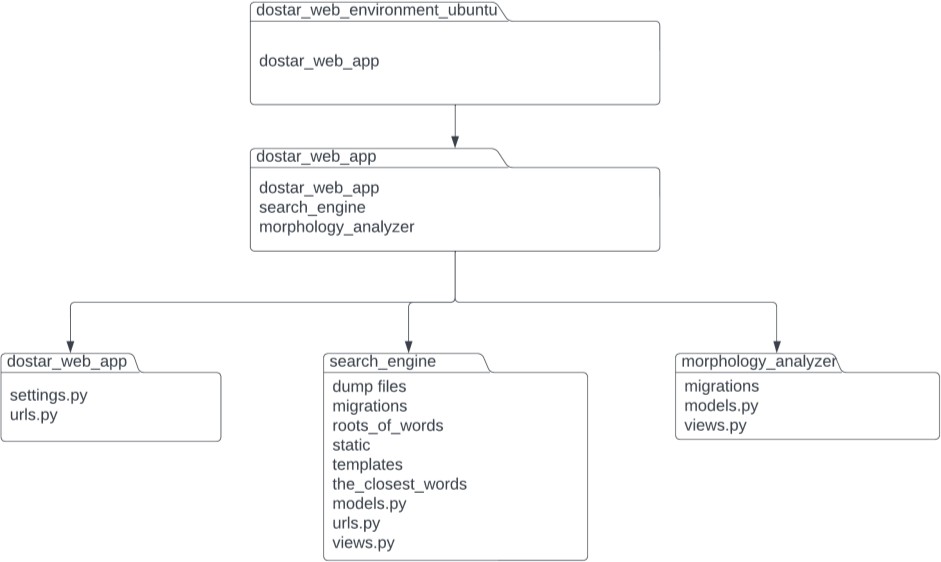


Figure 4.9 – The structure of Django project

In figure 4.9 dostar\_web\_environment\_ubuntu is a name of virtual environment. It contains necessary packages which needs Komekshy system. Therefore, if certain package is needed it should be installed into the environment. In addition, dostar\_web\_environment\_ubuntu environment contains dostar\_web\_app project. dostar\_web\_app project contains dostar\_web\_app, search\_engine, morphology\_analyzer folders. dostar\_web\_app has python files like settings.py, urls.py. settings.py file defines settings of the project. The format of settings is property=value. For instance, the file has DEBUG property, which can have true or false value. This property defines metadata for errors. By metadata developers can identify errors. Another example of property is DATABASES. It defines name of database which is used in django project.

urls.py file defines the list of url pathes and actions. When specific url address is matched the corresponding action will be performed. For instance, figure 4.5 contains window for authorization. To open this window, user in url address field should write “/admin”. This is because urls.py file has path ('admin/', admin.site.urls) path.

search\_engine is a name of application. One django project can have several applications. Each application performs specific task. Then django project will combine all applications to get one web site. Komekshy has two applications: search\_engine, morphology\_analyzer. The last application does morphological analysis for entered query. search\_engine in turn by some algorithms returns relevant Wikipedia articles.

dump\_files is a name of folder which contains xml file of Wikipedia articles. Wikipedia has seperate dump file for one language. New Wikipedia articles can be added into this folder.

migrations folder contains migrations files. Migration files are necessary to track changes in database.They can be used to rollback database and build it again. By this way developer can solve error in database. Another application of migrations files is building database by new team member.

roots\_of\_words folder contains ten txt files. These files contain roots of words of the Kazakh language. The roots were retrieved from books [56-69]. The roots were classified by part of speeches. According to [70] the Kazakh language has 10 part of speeches. Therefore, roots\_of\_words folder has 10 files.

static folder in django framework is devoted for static files like Cascading Style Sheets(CSS), images, JavaScript files.

templates folder contains HTML files. HTML files in Komekshy system were used to interact with user. Komekshy has HTML files like index.html, morphological\_analysis.html. index.html is the first web page of Komekshy system. Particularly it was figure 4.2. morphological\_analysis.html is a web page which shows the result of morphological analysis of written query. Figure 4.3 shows this web page.

the\_closest\_words folder has the list of semantically close words. The list is also classified by part of speeches. The format of list is: word (scw1, cosine value), (scw2, cosine value), (scw3, cosine value), (scw4, cosine value), (scw5, cosine value), (scw6, cosine value), (scw7, cosine value), (scw8, cosine value), (scw9, cosine value), (scw10, cosine value). Here scw is semantically close word. This list is used to expand user’s query.

The next file is models.py. Django framework has a lot file which have specific aim. The file can have several models in it. Each model represents one table in database. Therefore, by model developer can work with database table. Developer for instance can build new table, read, update, delete, insert records. search\_engine application has Wikipedia\_articles model. Therefore, database of Komekshy system also has Wikipedia\_articless table. Wikipedia\_articles model has following properties: title, body, article\_class. They are columns of the table. Title stores titles of wikipedia articles. Its maximum length is 50. body stores text of article. article\_class stores type of article. Its maximum length is also 50.

urls.py file has url adresses and name of function which processes user’s request. By other words when web browser has URL address, that address will be searched in urls.py file. urls.py file for URL address also has name of function which will process that request. For instance urls.py file has path('filling\_database',views.filling\_database). It means if user writes http://127.0.0.1:8000/search\_engine/filling\_database URL address, filling\_database function will be called.

In django framework views.py file is designed to process user’s request. The file has python functions to process requests. In previous example the path was path('filling\_database',views.filling\_database). filling\_database is a name of function which in views.py.

morphology\_analyzer is second application. It is needed to define root, suffixes, endings of word. This application contains migrations folder, models.py and views.py files. Their purpurses are the same as in search\_engine application. Figure 4.10 shows what tables have database and their relationships.

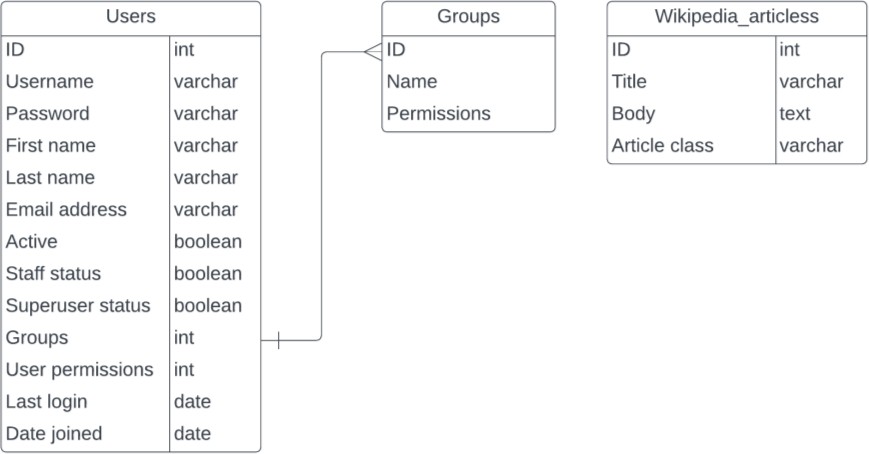


Figure 4.10 – Database diagram of Komekshy system

According to figure 4.10 database has three tables: Users, Groups, Wikipedia\_articless. Django by default has the first two tables. Descriptions of the columns can be read in 4.1.2 section.

# Django architecture

The previous section explained the structure of Komekshy system. This section explains the structure of django framework. As it was shown above django project has a lot of folders, files. To work with django framework developer should know their purposes. Figure 4.11 explains how works django.

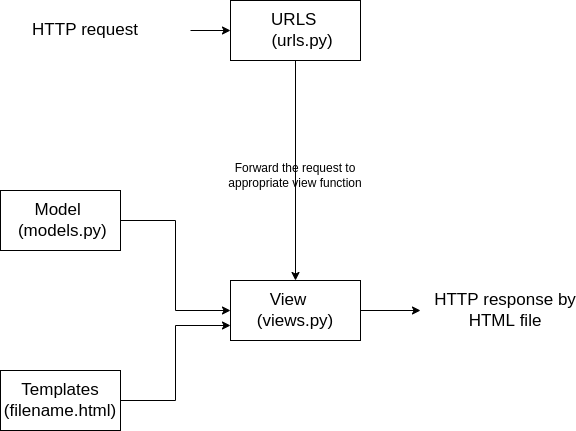


Figure 4.11 – Django architecture

As it is shown in figure 4.11 in django framework user by some action like clicking on button or navigating on some HTML element sends HTTP request. Django has urls.py file which has list of paths. A path consists of url address and a view function. After founding appropriate URL address its corresponding function will be called. Functions are located in views.py file. These functions process user’s request. Here web developer decides how to answer to user’s request. For example, web developer can retrieve some data from database, insert it into template (HTML file) and send it to the user. To retrieve some data model(models.py file) will be used.

To better understand django architecture I will consider internet shopping example. Information about products (like price, description, image) is stored in database. To visit the first page user sends request to server by URL address. From urls.py file corresponding URL will be searched. After finding the URL address its corresponding view function is called. To show the user images, prices, descriptions of products the view function will retrieve information about products from model. Model is like bridge between database and view function. After retrieving information from the model view function will insert it into template. Template is a HTML file. Template just contains structure of web page, but not context. After inserting information into template the response is almost ready. View function need to return it. It was example how django framework works.

# Implementation of Komekshy system

The project of Komekshy system has two applications: morphology\_analyzer, search\_engine. This section describes source code of that applications.

# Morphology\_analyzer application

morphology\_analyzer application contains models.py, views.py files. models.py file of morphology\_analyzer application doesn’t have any model. views.py of morphology\_analyzer application has following functions: morphological\_analysis(request), taueldik\_zhalgau (space, token, nouns,i), get\_space\_for\_taueldik (space,token,nouns,i), get\_space\_for\_esimshe(token,verbs,i), get\_case (space, token,nouns,i), get\_jiktik (space,token,nouns,i), get\_comparative\_power (space, token, adjectives,i), is\_plural (space, token, adjectives,i).

morphological\_analysis(request) function does morphological analysis for given query. Analysis is done if type of query is “get”. The function opens files with the list of roots. After that each root is compared with the fraction of query. If they are matched the function tries to add various kind of suffixes, endings until the end of query is reached. Therefore one query can be classified into several categories. The function returns morphological analysis of given query.

The next function is called taueldik\_zhalgau (space, token,nouns,i). This function defines possessive kind of ending from user’s query and returns it. This type of ending can be concatenated with noun. The function has four parameters: space, token, nouns, i. token parameter is user’s query in string form. space parameter defines empty space in user’s query. Because user’s query can contain several words splitted by empty spaces. nouns parameter stores the list of noun roots. i is index to retrieve specific noun.

The next function is get\_space\_for\_taueldik (space, token,nouns,i). The function returns the length of possessive ending. Particularly for “ЫМЫЗ”,”ІМІЗ”,”ЫҢЫЗ”,”ІҢІЗ” it will return 4. For “МЫЗ”,”МІЗ”,”ҢЫЗ”,”ҢІЗ” endings it will return 3.For ”ЫМ”,”ІМ”,”ЫҢ”,”ІҢ”,”СЫ”,”СІ” endings it will return 2. Finally for “М”,”Ң”,”Ы”,”І” endings it will return 1. The function has four parameters: space, token, nouns, i. Here token is user’s query. Query can contain several words, therefore space parameter defines index of word. nouns parameter contains the list of noun roots, whereas i parameter is needed to retrieve specific noun from the list.

The next function is called get\_space\_for\_esimshe(token,verbs,i). Esimshe is a kind of suffix. It can be concatenated with verbs. Many factors like plurality, negation influence on position of esimshe. The function defines position of esimshe from user’s query and returns it. The function takes three parameters: token, verbs, i. token is user’s query. verbs is list of root verbs. i is index of verb.

get\_case(space,token,nouns,i) function defines which case has given noun. Case is a kind of ending. The Kazakh language has seven cases: gen, abl, dat, loc, acc, ins, nom. The function returns one of them.

Jiktik is a kind of ending in Kazakh language. get\_jiktik(space,token,nouns,i) function returns in which form jiktik is given in user’s query.

get\_comparative\_power(space,token,adjectives,i) function checks if user’s query in comparative form of adjective. If yes the function returns true, otherwise false.

is\_plural(space,token,adjectives,i) function checks if user’s query in plural form.

If yes the function returns true, otherwise false.

# Search\_engine application

Search\_engine application contains files like models.py, views.py, urls.py. models.py is a file which contains models of application. Figure 4.12 illustrates its source code.

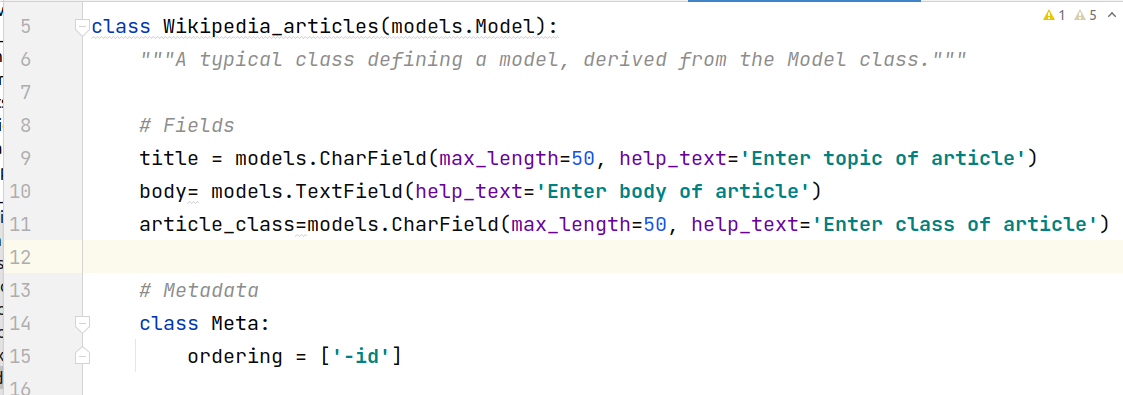


Figure 4.12 – models.py file of search\_engine application

According to figure 4.12 model is called Wikipedia\_articles. This class is a child of Model class. The model has three fields: title, body, article\_class. CharField, TextField, CharField their corresponding types. The model also has Meta subclass. This subclass by “ordering” property sorts Wikipedia articles by id column.

The next considering file is views.py. It has index(request), up(str), search (request), filling\_database (request), get\_part\_of\_speech (text), get\_ranking (query) functions.

index(request) function returns index.html file. index.html file is the first web page of Komekshy system. index(request) function is used to see the first web page.

The next function is called up(str). This function as argument takes some string, then returns the same capital letters. For instance, for “Komekshy” string, the function will return “KOMEKSHY” string. Digits will not be changed.

The next function is search(request). This function firstly checks whether received request is “get”. If the request is get then morphological analysis for it will be done. After defining roots their semantically close words will be searched. To not search semantically close words of the same words duplicates are deleted. Then get\_ranking(query) function is called to calculate score of each article.

filling\_database(request) function as its name says is used to fill database with records. Before filling database, the articles were only in xml file. Therefore filling\_database(request) function opens that xml file, and read it. However, the xml file not only contains the text of article, but additional symbols as well. Hence wiki-dump- reader was used. By it cleaned text can be obtained. In addition, category of article is also written in main body of article. To retrieve category of article, find method was used. find method searches some text from string. If string has that text, find method will return index of first symbol of text. To get category "[[Санат:" and "]]" text will be searched. Then the text which is between "[[Санат:" and "]]" will be retrieved.

get\_part\_of\_speech(text) function gets string as parameter. The function checks whether that string contains “NOUN” or “VERB”. After defining part of speech it returns it in list form. Figure 4.13 summarizes the above explanation.

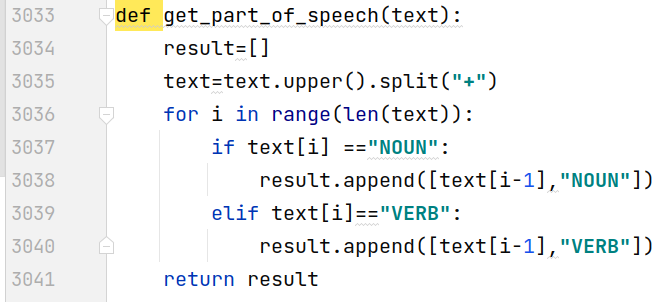


Figure 4.13 – get\_part\_of\_speech(text) function

get\_ranking(query) function gets as parameter user’s query then defines relevant wikipedia articles for it. The argument of function can contain several queries. It can happen if semantically close words of query will be found. get\_ranking(query) function by Wikipedia\_articles model retrieves one article. Then the function counts how many times words of query met in title, body, category of article. After that by formula 4.1 calculates score of current article and store it in data structure. The function prints IDs of articles with the highest score.

search\_engine application has folder templates. This folder contains html files which is used to show the result for user. The first file is called index.html, the second is called morphological\_analysis.html.

index.html the first web page of Komekshy system. Its image is shown in figure 4.1. The file firstly loads static tag. Static tag is necessary to use static files like CSS files, images. Then the letters of word “Көмекші” are written by <svg> tag. At the bottom there is tag <form> to send user’s entered query to view function. Figure 4.14 summarizes stated above.



Figure 4.14 – The source code of index.html file

On line 6 Django loads static tag. Then on line 10 flex value was used. By flex value HTML elements can be aligned. Using flex value is useful because sizes of screens can be different. Flex value will automatically calculate distances between HTML elements depending on size of screen. Font-size CSS property defines size of letters, whereas rgb() function defines color of letter. The figure 4.15 shows tag form.



Figure 4.15 – Tag <form> of index.html

Tag <form> is used to send user’s query to view function. As shown in figure 32 form tag sends query by “get” method. Search is a name of path which is located in urls.py file. This path in turn calls search function of views.py file.

morphological\_analysis.html is a template which shows the result of query. Figure 4.3 shows its image. Figure 4.16 contains source code of morphological\_analysis.html file.

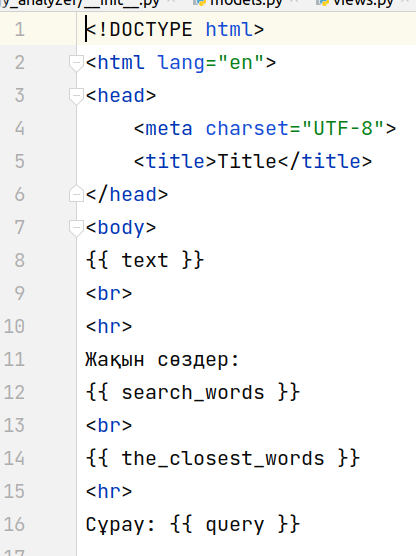


Figure 4.16 – Source code of morphological\_analysis.html

From figure 4.16 can be seen that the source code just defines the structure of the result. The content will be sent by view function. Here {{text}} stores the result of morphological analysis of user’s query. {{search\_words }} stores defined roots for user’s query. These roots are necessary to expand user’s query. {{the\_closest\_words}} stores the list of semantically close words and their cosine values. {{query}} stores user’s original query. After {{query}} there is another keyword {{queries}}. {{queries}} stores the list of expanded queries by Komekshy. For instance, in figure 20 user’s original query was “Абай жолы”. Komekshy from the list of semantically close words found word “Абай”. Its semantically close words were: абай, абайдың, ақынның, құнанбаев, шәкәрім, ақын, сәкен, ақындық, мағжан, шәкәрімнің. Then Komekshy replaced word “Абай” with that ten semantically close words. In this example one word was replaced. However, it is possible to replace several words. For example if query is “Абай ақын” expanded queries will be: [' абай ақын', ' абайдың ақын', ' ақынның ақын', ' құнанбаев ақын', ' шәкәрім ақын', ' ақын ақын', ' сәкен ақын', ' ақындық ақын', ' мағжан ақын', ' шәкәрімнің ақын', ' Абай ақын', ' Абай ақынның', ' Абай қаламгер', ' Абай жыршы', ' Абай жырлаған', ' Абай абай', ' Абай ақынға', ' Абай жазушы', ' Абай көкбай', ' Абай сәкен']. Here words “Абай” and “ақын” were replaced. Expanded queries are necessary for searching algorithm.

search\_engine application has urls.py file. This file contains paths which are shown in figure 4.17.



Figure 4.17 – urls.py file of search\_engine application

The first path is needed to call index function of views.py file. index function returns index.html template. The second path is needed to call search function of views.py file. search function returns morphological\_analysis.html template. The third path is needed to fill database with Wikipedia articles.

# Dataset of Komekshy system

This section describes various datasets which are used in Komekshy. Komekshy is a system which can help people to find relevant Wikipedia article. Therefore, it uses dataset of Wikipedia articles. Wikipedia articles were taken from [71]. Peculiarity of this dataset that it updates its articles. Now Komekshy uses 20191120 version of dump file. Another advantage of this dataset that Wikipedia articles sorted by languages. Komekhsy uses articles which are written in Kazakh, however other languages can be added.

The next dataset is [56, б. 3-750; 57, б. 3-750; 58, б. 3-750; 59, б. 3-740; 60, б. 3-750; 61, б. 3-750; 62, б. 3-750; 63, б. 3-750; 64, б. 3-750; 65, б. 3-740; 66, б. 3-750; 67, б. 3-740; 68, б. 3-740; 69, б. 3-820] dictionaries. These books were used to define semantically close words. Particularly, semantically close words were defined for words, which are included in [56, б. 3-750; 57, б. 3-750; 58, б. 3-750; 59, б. 3-740; 60, б. 3-750; 61, б. 3-750; 62, б. 3-750; 63, б. 3-750; 64, б. 3-750; 65, б. 3-740; 66, б. 3-750; 67, б. 3-740; 68, б. 3-740; 69, б. 3-820] dictionaries. Advantages of this dictionaries are parts of speech are defined, words are written in root form. These dictionaries were available online at [72].

The next dataset is dataset of word embedding [49]. Maintainer of this dataset is Andrey Kutuzov. The dataset was used to define semantically close words. To define semantically close words cosines between vectors were calculated. Section 3.3 in detail describes how semantically close words were defined. The dataset contained coordinates of vectors. It had 57048825 tokens, dimensions of vectors were 100, number of iteration was 2, vocabulary size was 176643.

# Reading dictionaries

As it was mentioned above semantically close words were defined for words which are included in [56, б. 3-750; 57, б. 3-750; 58, б. 3-750; 59, б. 3-740; 60, б. 3-750; 61, б. 3-750; 62, б. 3-750; 63, б. 3-750; 64, б. 3-750; 65, б. 3-740; 66, б. 3-750; 67, б. 3-740; 68, б. 3-740; 69, б. 3-820] dictionaries. These dictionaries contain information like description of word, part of speech. Hence from dictionaries I needed retrieve only roots. To do it python program was written. Content of dictionaries was written in docx files. Hence python program read docx files. In docx files some letters weren’t correctly shown. Therefore, the program replaced that symbols with the correct letters. The program uses re module and docx library. Example of code snippet is shown in figure 4.18.

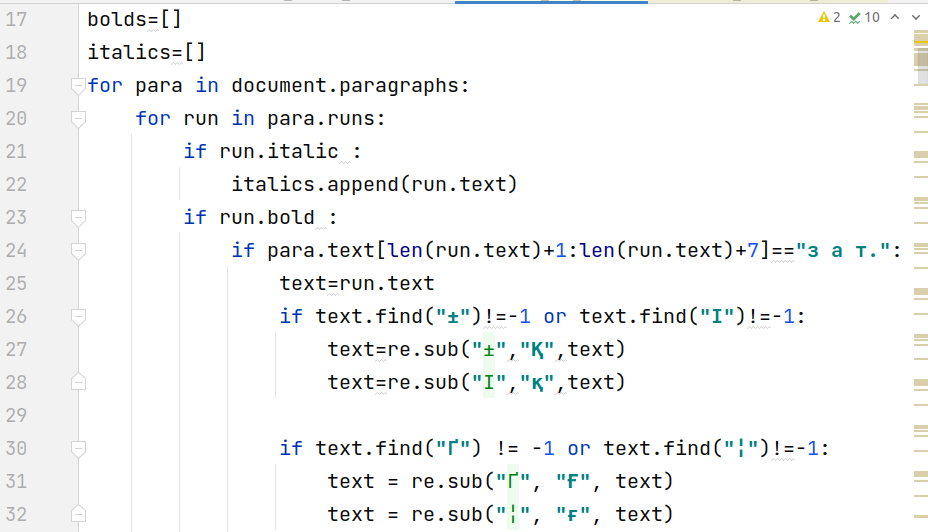


Figure 4.18 – Reading docx file

In [56, б. 3-750; 57, б. 3-750; 58, б. 3-750; 59, б. 3-740; 60, б. 3-750; 61, б. 3-750; 62, б. 3-750; 63, б. 3-750; 64, б. 3-750; 65, б. 3-740; 66, б. 3-750; 67, б. 3-740; 68, б. 3-740; 69, б. 3-820] dictionaries for one word corresponds one paragraph. Therefore, in figure 4.18 the first loop iterates over paragraphs. In 27th,28th lines incorrect symbols are replaced by correct letters. To find root word from text, part of speech was used. The dictionaries have template like “word part of speech”. Hence 24th line checks presence of noun. If part of speech is noun then the program will retrieve the text which is before part of speech (line 25).

# Portal of sozdikqor.kz

The previous section described how roots were retrieved from docx files. However probably because of format of file not all volumes were properly retrieved. Therefore that content was searched with other format. As a result sozdikqor.kz portal was found. The portal contains various dictionaries. People can read meaning of word, synonyms, antonyms, homonyms.

To retrieve roots from portal, volumes which are not properly read needed to be searched. Figure 4.19 shows how sozdikqor.kz portal looks like.

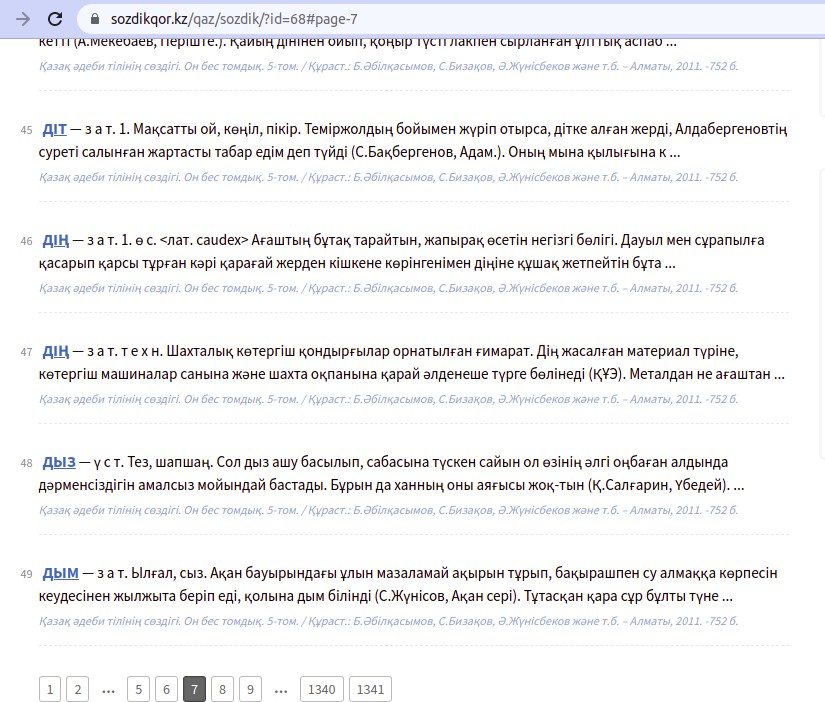


Figure 4.19 – sozdikqor.kz portal

As shown in figure 4.19 one web page contains seven words. User to open new web page needs to click on button below. To retrieve root copying and pasting technique can take long time. Because there are a lot of words. Scraping technique also couldn’t be used. Because when user clicks on button the portal updates current web page. It means user works with one web page, which is updated when user clicks on button.

To solve this issue selenium package was used. By selenium package python program can manipulate web browser. By other words selenium package allows by web browser simulate user’s action. This approach is useful for testing web sites. Especially when web site has a lot of web pages and all of them needed be tested. This approach can work fast. To use selenium package it should be installed and imported. Figure 4.20 shows code snippet of retrieving words from sozdikqor.kz portal.

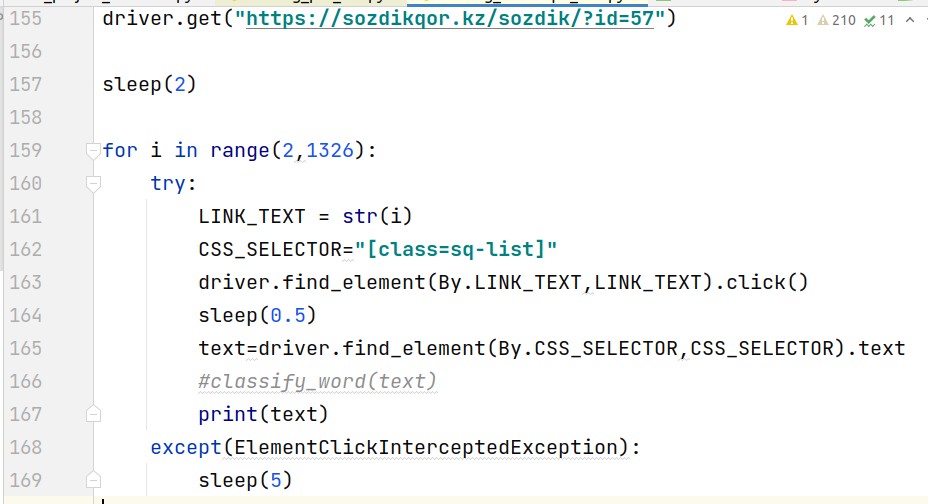


Figure 4.20 – Retrieving words from sozdikqor.kz portal

On line 155 is written URL address from where roots will be retrieved. sleep() method is used to wait response from server of sozdikqor.kz portal. Inside sleep() method number of seconds is written. To change content of web page button on the bottom should be clicked. From figure 4.19 can be seen that inside buttons written numbers. Therefore, on line 163 HTML element is found by text(number). After founding necessary button, click() method is called to update web page. When web page is updated new words are appeared. sleep(0.5) method was called because for updating content web browser needs time. In python code text variable stores one word with definition. classify\_word() function gets text variable. It retrieves only root, classify it by part of speech, and write the root in appropriate file. Finally, roots are written in corresponding files.

# Limitations of thesis

Komekshy search engine has several limitations. One limitation is articles were written in Kazakh language. If user needs an article in other language, he or she will not find it. To solve this limitation database of Komekshy needs to be updated. In Wikimedia articles are divided by languages. To add new language from Wikimedia corresponding dump file should be downloaded. Then reading the dump file the database of Komekhy should be filled by new articles.

Another limitation is number of articles. The database of Komekshy will not be automatically updated if on the Internet new article is appears. To solve this issue the solution for the first limitation should be used.

The next limitation is number of roots in morphological analysis. As it was mentioned previously Komekshy does morphological analysis for entered query. It is necessary to find out root of word. To find out root of word the dictionaries from sozdikqor website were used. Those dictionaries have limited number of roots. Hence if user enters word which doesn’t contain dictionaries, Komekshy will not be able to properly find the root. To solve this issue the list of roots regularly needs to be updated.

Another limitation of current thesis is response time. In search engine the result should be not only relevant but be provided on time as well. In Komekshy the response time depends on number of words which are used in query. As it was mentioned the algorithm of Komekshy from each article will check presence of entered word. If number of words in query is large, then response time will be long. To solve this issue many data structures can be applied. One of such data structure is prefix tree. Prefix tree is data structure which can effectively store strings with the same prefixes. Detailed information about prefix tree provided in next section.

# Prefix tree

Prefix tree is a kind of data structure, where nodes store one letter. Prefix tree as other tries can have parent node, child node, branches. Advantage of prefix tree is it can store the same prefix of several words just one time, rather than saving prefixes several times. For instance, there are three strings “Оқу” (Reading),”Оқушы”(Pupil), ”Оқулық”(Textbook). All three words have the same beginning “Оқу”. In programming allocating seperate memories for each word would be inefficient, because the common part is saved several times. To avoid such case prefix tree can be used. Figure 4.21 shows prefix tree for this example.

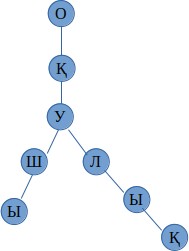


Figure 4.21 – Prefix tree for words “Оқу”,”Оқушы”, ”Оқулық”

From figure 4.21 can be see that each node contains one letter. In addition, for three words just one tree was used without repeating “Оқу” three times. However, in this case end of word is unknown. Hence in prefix tree to know end of word special flag is used. This flag is called “end” and defined for each node. “End” flag has boolean type. All last letters of words have true value. Others are false. Figure 4.22 shows the same prefix tree but with end flag.

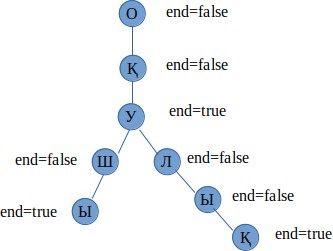


Figure 4.22 – Prefix tree with end flag

Now each node has end parameter. Nodes “У”,”Ы”,”Қ” have true value. Therefore, they will be considered as the last letter of the word. In current prefix tree there are three

words: “Оқу”,”Оқушы”, ”Оқулық”. In addition, common letters(оқу) repeated just one time, rather than repeating three times. In prefix tree insert, delete, search operations can be done.

Prefix tree in Komekshy can be used in various parts. For instance, prefix tree can be used in completing word when user starts to write query. It can be useful when user doesn’t know how to properly write a word. Another application of prefix tree is word searching. In Komekshy search engine response time depends on number of words of query. This is because those words will be counted in each Wikipedia article. To count words faster prefix tree can be used. In counting algorithm words with different prefixes will be skipped.

# Compressed prefix tree

Figure 4.21 demonstrates an example of prefix tree. In that example each letter was described by one node. Searching time in prefix tree depends on searching depth. Searching time is increased as searching depth become deeper. Hence to reduce searching time searching depth should be decreased. With this purpose compressed prefix tree can be used. Compressed prefix tree is modified version of prefix tree, where one node can contain several letters. This approach allows reducing depth of prefix tree. Figure 4.23 shows an example of compressed prefix tree for words “Оқу”(Reading),”Оқушы”(Pupil), ”Оқулық”(Textbook).



Figure 4.23 – Compressed prefix tree with end flag

Now one node contains several letters. Therefore the depth of prefix tree was reduced. In example above three words had the same begining. However words can have different prefixes. For example, prefix tree should have words like кітап, оқу, оқушы, оқулық. Now the first word has different the first letter. In that case the root of prefix tree will be NULL. In prefix tree NULL root is used when words have different prefixes. Figure 4.24 shows such an example.

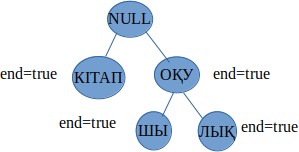


Figure 4.24 – Compressed prefix tree with different prefixes

In figure 4.24 compressed prefix tree contains кітап, оқу, оқушы, оқулық words. In this case words оқу, оқушы, оқулық have the same beginning but word кітап has different beginning. Hence they are divided into two branches. In compressed prefix tree letters are added to the same node until one word has different letter. In that case new branch will be added.

# Insert operation in prefix tree

In prefix tree insert operation is used when new word should be added. In next example prefix tree contains word “Абылай”. The second word is “Абылайхан”, which should be added. To add a new word, the root of prefix tree will be compared with the first letter of second word. They are the same, hence the next letters will be compared. The second letters are again the same, therefore the next letters will be compared. However, if second letters were different new branch would be added. In current example the first word is part of second word, therefore just “хан” part should be added. Figure 4.25 summarizes this explanation.

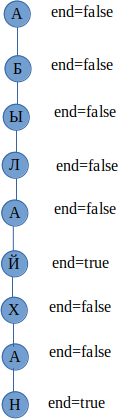


Figure 4.25 – Insert operation with one branch

Figure 4.26 shows another example of prefix tree with several branches.

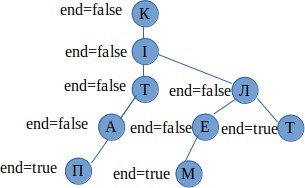


Figure 4.26 – Insert operation with several branches

# Delete operation in prefix tree

Delete operation can be used when a word should be deleted. A word can be deleted in several cases. The first case when the word which should be deleted is subpart of another word. For instance, word “Абылай” is subpart of word “Абылайхан”. To delete word “Абылай” from prefix tree, but at the same time keep word “Абылайхан” end flag of letter “й” should be changed to false value. Figure 4.27 shows how word “Абылай” from prefix tree can be deleted.

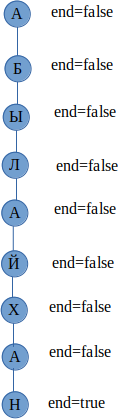


Figure 4.27 – Deleting subword from prefix tree

Now prefix tree has word “Абылайхан”, but doesn’t have word “Абылай”(because end flag is false).

The next case when the resulting prefix tree is subpart of original prefix tree. For instance, original prefix tree contains word “Жолдас”. From this word “дас” part should be deleted. “дас” are the last letters of prefix tree. Therefore, those nodes can be deleted. Figure 4.28 shows how from word “Жолдас” “дас” part can be deleted.

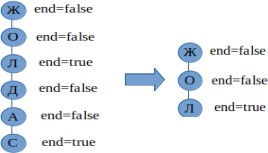


Figure 4.28 – Deleting the last nodes from the prefix tree

After deleting the last three nodes from the prefix tree, it will contain just one word “Жол”.

# 5 EXPERIMENTAL TESTING AND RESULTS

Komekshy search engine was developed on the basis of several experiments which were described in third section. In current thesis I tried that Komekshy returned relevant Wikipedia articles for entered query. In current thesis Wikipedia article was considered as relevant if it contained query word. To test Komekhsy search engine several queries were written. Table 5.1 shows entered queries and returned Wikipedia articles.

Table 5.1 – Obtained titles of Wikipedia articles

|  |  |
| --- | --- |
| Query | Artical title and its ID |
| 1 | 2 |
| Абай Құнанбайұлы | Абай Құнанбайұлы(3124), Абай Құнанбайұлы (3815), Әуез Бердіұлы (28792), Шәкәрім Құдайбердіұлы (935), Шәкәрім Құдайбердіұлы (4671), Сәкен Сейфуллин (8089), Дәстүр және жаңашылдық (47355), Абай жолы (роман) (10501), Абай лирикасы (47279) |
| Ең алғашқы ғарышкер | Фридрих Ницше(2359), Фридрих Ницше(6095), Техас (202), Техас (3247), Техас(3938), Құран (279), Құран (3324), Құран(4015), Queen (824) |
| Түйе | Мал шаруашылығы(14759), Қой шаруашылығы(60265), Көшпелiлер немесе дала өркениетi(240601), Қазақстан жерінде мал шаруашылығының қалыптасуы(72321), Төрт түлік сүті (242117), Қазақ ғылыми-зерттеу институттары (27754), Қазақтың ұлттық тағамдары (236190), Түйелер (7780), Жылқы (қазақ дәстүрі) (87315) |
| Махаббат қызық мол жылдар | Фридрих Ницше (2359), Фридрих Ницше (6095), Техас (202), Техас(3247), Техас(3938), Құран (279), Құран (3324), Құран (4015), Queen(824) |
| Балқаш көлі | Швейцария (263), Швейцария (3308), Швейцария (3999), Шалқар (көл, Батыс Қазақстан облысы) (58740), Баянауыл ұлттық паркі(7636), Балқаш (3015), Балқаш (6751), Зайсан ауданы (9668), Марқакөл қорығы (7632) |
| Жылқы сүті | Төрт түлік сүті (242117), Мал шаруашылығы (14759), Қазақтың ұлттық сүт тағамдары (234889), Қазақтың ұлт тық тағамдары (236190), Жылқы (қазақ дәстүрі) (87315), Қой шаруашылығы (60265), Көшпелiлер немесе дала өрке ниетi (240601), Сүт тағамдары (10114), Жылқы (14356) |
| Ара | Бал арасы (20241), Ара шаруашылығы (19890), Қазақстан ның балды өсімдіктер қоры (236960), Кристаллохимия (11504), Ара өсіру және оның биологиясы (238460), Омар та құрылыстары (236958), Ара балы (10321), Бүкіл әлемдік тартылыс заңы(2688), Бүкіл әлемдік тартылыс заңы (6424) |
| Continuation of table 5.1 | |
| 1 | 2 |
| Жібек жолы | Ұлы Жібек жолы (706), Ұлы Жібек жолы (3751), Ұлы Жібек жолы (4442), Андромеда галактикасы (249398), Андромеда шоқжұлдызы (249372), Электр-ток көздері. Ом заңы (10441), Қазыбек би (444), Қазыбек би (3489),  Қазыбек би (4180) |
| Бидай | Астық дәнді дақыл (235331), 35605, Өсімдік шаруашылығы (68504), Жарма дақылдары (18113), Арпа (12514), Картоп салаты (249598), Жарма дақылы (39589), Қазақтың ұлттық тағамдары (236190), Ресейдің ауыл шаруашылығы және көлік географиясы(68727) |
| Шұбат | Қазақтың ұлттық тағамдары (236190), Ірімшік(10126), Қазақтың ұлттық сүт тағамдары (234889), Төрт түлік сүті (242117), Маргарин (54513), Ақ ірімшік(68312), Ақ тағам дайындау жолдары (150244), Айран(2093), Айран (5829) |

Wikipedia articles were ranked according to formula 4.1. Each article had its score. Table 5.1 contains titles of Wikipedia articles which took the highest scores. Some titles were repeated because in database of current search engine articles are repeated. According to formula 4.1, query word was counted in title, main body, category. After counting each number was multiplied to its corresponding coefficients. For title, main body, category they respectively were 10, 85, 5. Then obtained numbers were summed. The coefficients were chosen empirically.

In current section returned result will be evaluated by accuracy. Accuracy is calculated by formula (5.1):

|  |  |
| --- | --- |
|  | (5.1) |

where TP is number of true positive examples;

TN is number of true negative examples;

FP is number of false positive examples;

FN is number of false negative examples.

In current thesis each article can be classified as relevant or not relevant. As it was mentioned above if returned article contains query word then the article will be considered as relevant, not relevant otherwise. According to formula (5.1) accuracy will be 100%, because Komekshy returns only matched articles.

# CONCLUSION

Searching certain information is still important task for people. This is because information is growing. People can search information with various purposes. For instance, from email some message can be searched by keywords. In this case keywords should be compared with contents of messages. If a message contains those keywords, it means the message is found. However, people can search not only matched documents, but semantically related documents as well. For instance, people can be interested in some mathematical theorems which are related with each other. In search engines they can type name of one theorem. Search engines in turn, should be able to return other related theorems as well. From these examples can be seen that queries of people can be different. In addition, queries can be written in different languages. Each language has its own rules. Therefore, one searching algorithm can show different results for various languages. In current thesis searching algorithm was designed. This searching algorithm was designed for the Kazakh language. The Kazakh language is rich by its suffixes, endings. In Kazakh language word changes its meaning when suffix connected with root. Hence the searching algorithm should be able to properly define roots of words.

In current thesis several aims were defined. Particularly they were: developing searching algorithms for Kazakh text documents, for user’s query returning relevant text documents, designing database of Kazakh Wikipedia articles. To reach those aims in current thesis new search engine, which is called Komekshy was developed. Komekshy uses searching algorithm to define relevant documents. In current thesis document was considered as relevant if it contained words of query. Komekshy was developed by Django framework. Its database contains 276660 Wikipedia articles.

In current thesis several objectives were defined. These objectives are: doing morphological analysis for user’s query, using word embedding model to define semantically close words, doing an experiment by distributed system Apache Spark for reducing computation time. As it was mentioned above in Kazakh language suffixes can change word meaning. Therefore, morphological analysis should be done for each query word to properly define meaning of word. In current thesis morphological analysis was done to define root. To define root Komekshy selects various Kazakh suffixes and endings. Novelty of current thesis is defining semantically close words by word embedding model. In current thesis NLPL repository was chosen as word embedding model. As a result of an experiment of current thesis semantically close words were defined for 8497 words. Semantically close words were defined to search from documents not only query words, but their semantically close words as well. This approach allows finding relevant documents even they don’t contain query words. For the last objective separate experiment was done. In experiment we tried to define semantically close words by several methods. In one of them we used Apache Spark distributed system to make calculation faster. For this experiment [73] paper was published.

The thesis consists of several sections. Introduction part defines problem statement of research. It was about searching relevant documents. In addition, introduction part provides objectives, aims, novelties of current thesis.

In literature review part the survey for the similar works was made. Literature review part list papers, which solve similar tasks. It was done, because it was difficult to find the same work. Therefore, literatures for separate tasks of current thesis were reviewed.

The thesis contains several experiments. Detailed information about them is written in thesis experiment section. The first experiment is called “Identifying lexical compatibilities of words by vectors of specialized words”. For this experiment paper

[48, б. 67-72] was written. The aim of experiment was finding out not lexically compatible words by word embedding. Not lexically compatible words are pair of words which usually not used together. Examples of such words can be plate and wheel, plant and pen, radio and glasses. Because that words usually used in separate sentences. Defining not lexically compatible words allows finding out mistakes in sentences. Particularly mistakes where words correctly written but not properly used. The next experiment is called “Stemming of the Kazakh language”. The Kazakh language is an agglutinative language. Therefore, suffixes can change meaning of word. Endings can change form of word. In current experiment morphological analysis for words was done. It was done to define root of word. For this experiment [51, р. 162-165] paper was published. The third experiment is called “Defining semantically close words of Kazakh language by distributed system”. Semantically close words are words, which used in one sentence. Examples of semantically close words can be synonyms like next to, near, big, huge. However semantically close words are not always synonyms. They can have different meanings as well. For instance, words printer, mouse, monitor can be considered as semantically close words, because they can be used in one sentence. In third experiment for roots of the Kazakh words semantically close words were defined. It was done by word embedding, where words represented by vectors. To define semantically close words angle between two vectors was calculated by cosine. To get the most semantically close words the highest cosine values were chosen. The fourth experiment is called “Classification of Scientific Documents in the Kazakh Language Using Deep Neural Networks and a Fusion of Images and Text”. In fourth experiment scientific papers were classified by three methods. The first method was classifying papers by their text. In experiment as classification algorithm was chosen Naive Bayes. The second method was classifying papers by their images. To classify papers by their images CNN model was used. The third method was classifying papers by images and text. For this experiment [74] paper was published.

As a result of all mentioned above four experiments the new search engine was designed. It is called Komekshy. Komekshy was designed to help people finding relevant documents for entered query. The fourth section of current thesis describes this search engine.

As a conclusion of my thesis I understood that there are lots of searching algorithms. Among them there is no the best algorithm, which satisfies all people requirements. One algorithm can be best in certain condition, whereas another algorithm can show better result in other condition. In addition, the result of search engine not always depends on algorithms, but on entered query as well. Therefore, user of search engine to get relevant document should properly write query. Properly means avoiding words with several meanings, not doing grammar mistakes and so on.

# REFERENCES

1. Google // [https://en.wikipedia.org/wiki/Google.](https://en.wikipedia.org/wiki/Google) 06.12.2023.
2. Microsoft Bing // [https://en.wikipedia.org/wiki/Microsoft.](https://en.wikipedia.org/wiki/Microsoft_Bing) 06.12.2023.
3. Baidu // [https://en.wikipedia.org/wiki/Baidu.](https://en.wikipedia.org/wiki/Baidu) 06.12.2023.
4. Faiss // <https://github.com/facebookresearch/faiss/wiki.> 11.12.2023.
5. Jégou H., Douze M., Schmid C. Product Quantization for Nearest Neighbor Search // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 2011. – Vol. 33, Issue 1. – P. 117-128.
6. Jégou H., Douze M. et al. Searching in one billion vectors: Re-rank with source coding // Procced. IEEE internat. conf. on Acoustics, Speech and Signal Processing (ICASSP). – Prague, 2011. – P. 861-864.
7. Основы Elasticsearch // <https://habr.com/ru/articles/280488/>. 11.12.2023.
8. Xu W., Chen H., Huan Y. et al. Full-text search engine with suffix index for massive heterogeneous data // Information Systems. – 2022. – Vol. 104. – P. 101893.
9. Suryono H., Kuswanto H., Iriawan N. Rice phenology classification based on random forest algorithm for data imbalance using Google Earth engine // Procedia Computer Science. – 2022. – Vol. 197. – P. 668-676.
10. Gowri S., Sathish Kumar P.J., Geetha Rani K. et al. Usage of a binary integrated spell check algorithm for an upgraded search engine optimization // Measurement: Sensors. – 2022. – Vol. 24. – P. 100451-1-100451-6.
11. Zhang C., Tian Y.-X., Fan Z.-P. Forecasting sales using online review and search engine data: A method based on PCA–DSFOA–BPNN // International Journal of Forecasting. – 2022. – Vol. 38, Issue 3. – P. 1005-1024.
12. Fuentes-Lorenzo D., Fernández N., Fisteus J. et al. Luis Sánchez. Improving large-scale search engines with semantic annotations // Expert Systems with Applications. – 2013. – Vol. 40, Issue 6. – P. 2287-2296.
13. Raza S. A COVID-19 Search Engine (CO-SE) with Transformer-based architecture // Healthcare Analytics. – 2022. – Vol. 2. – P. 100068-1-100068-14.
14. Shin H., Lee H.J., Cho S. General-use unsupervised keyword extraction model for keyword analysis // Expert Systems with Applications. – 2023. – Vol. 233. – P. 1-25.
15. Bibi N., Rana T. et al. Reusable Component Retrieval: A Semantic Search Approach for Low-Resource Languages // ACM Transactions on Asian and Low- Resource Language Information Processing. – 2023. – Vol. 22, Issue 5. – P. 1-31.
16. Upadhyay B., Khairnar T., Kotalwar A. Improve retrieval of regional titles in streaming services with dense retrieval // WWW '23 Companion: Companion proceed. of the ACM Web conf. – NY., 2023. – P. 761-767.
17. Ferrod R., Bondarenko D.A., Audrito D. et al. Pairing EU directives and their national implementing measures: A dataset for semantic search // Computer Law & Security Review. – 2023. – Vol. 51. – P. 1-10.
18. Poppink B., Frasincar F., Robal T. An experimental study on re- ranking web shop search results using semantic segmentation of user profiles // Electronic Commerce Research and Applications. – 2023. – Vol. 62. – P. 1-20.
19. Abuhashim A.A., Tan C.C. Improving smart contract search by semantic and structural clustering for source codes // Blockchain: Research and Applications. – 2023. – Vol. 4, Issue 2. – P. 100117-1-100117-11.
20. Moro G., Salvatori S., Frisoni G Efficient text-image semantic search: A multi-modal vision-language approach for fashion retrieval // Neurocomputing. – 2023. – Vol. 538. – P. 126196-1-126196-14.
21. Frikha M., Fendri E., Hammami M. Deep Semantic Attributes for People Search // Procedia Computer Science. – 2021. – Vol. 192. – P. 90-99.
22. Wosiak A. Using semantic enrichment methods in expert search system for recruitment process in IT corporation // Procedia Computer Science. – 2021. – Vol. 192. – P. 2422-2431.
23. Klimov V., Balandina A., Chernyshov A. Application of Long-Short Memory Neural Networks in Semantic Search Engines Development // Procedia Computer Science. – 2020. – Vol. 169. – P. 388-392.
24. Tablan V., Bontcheva K., Roberts I. et al. MMmir: An open-source semantic search framework for interactive information seeking and discovery // Journal of Web Semantics. – 2015. – Vol. 30. – P. 52-68.
25. Mohamed E.H., Shokry E.M. QSST: A Quranic Semantic Search Tool based on word embedding // Journal of King Saud University - Computer and Information Sciences. – 2022. – Vol. 34, Issue 3. – P. 934-945.
26. Khan S., Mustafa J. Effective semantic search using thematic similarity // Journal of King Saud University - Computer and Information Sciences. – 2014. – Vol. 26, Issue 2. – P. 161-169.
27. Kou F., Du J. et al. A multi-feature probabilistic graphical model for social network semantic search // Neurocomputing. – 2019. – Vol. 336. – P. 67-78.
28. Zhumanov Z., Madiyeva A., Rakhimova D. New Kazakh parallel text corpora with on-line access // Procced. Computational Collective Intelligence. – Nicosia, 2017. – P. 501-508.
29. Makhambetov O., Makazhanov A., Yessenbayev Zh. et al. Assembling the Kazakh Language Corpus // Proceed. of the 2013 conf. on Empirical Methods in Natural Language Processing. – Washington, 2013. – P. 1022-1031.
30. Toiganbayeva N., Kasem M., Abdimanap G. et al. KOHTD: Kazakh offline handwritten text dataset // Signal Processing: Image Communication. – 2022. – Vol. 108. – P. 116827-1-116827-13.
31. Dong J. et al. A compromise Arabic-Kazakh coded character processing method based on the OpenType font format // Computer Standards & Interfaces. – 2018. – Vol. 55. – P. 1-8.
32. Parhat S., Ablimit M., Hamdulla A. A Robust Morpheme Sequence and Convolutional Neural Network-Based Uyghur and Kazakh Short Text Classification // Information. – 2019. – Vol. 10, Issue 12. – P. 387-1-387-14.
33. Parhat S., Ting G., Ablimit M. et al. A morpheme sequence and convolutional neural network based Kazakh text classification // Procced. Asia-Pacific Signal and Information Processing Association Annual summit and conf. (APSIPA ASC). – Lamzhou, 2019. – P. 1903-1906.
34. Haisa G., Altenbek G., Aierzhati H. et al. Research on Classification of Kazakh Questions Integrate with Multi-feature Embedding // Procced. 2021 2nd internat. conf. on Electronics, Communications and Information Technology (CECIT). – Sanya, 2021. – P. 943-947.
35. Nurseitov D., Bostanbekov K., Kurmankhojayev D. et al. Handwritten Kazakh and Russian (HKR) database for text recognition // Multimedia Tools and Applications. – 2021. – Vol. 80. – P. 33075-33097.
36. Narynov S.S., Zharmagambetov A.S. On One Approach of Solving Sentiment Analysis Task for Kazakh and Russian Languages Using Deep Learning // Procced. internat. conf. on Computational Collective Intelligence. – Cham: Springer, 2016. – P. 537-545.
37. Yergesh B., Bekmanova G., Sharipbay A. Sentiment Analysis of Kazakh Text and Their Polarity // Web Intelligence. – 2019. – Vol. 17. – P. 9-15.
38. Akanova А., Ospanova N., Kukharenko Y. et al. Development of the Algorithm of Keyword Search in the Kazakh Language Text Corpus // Eastern-European Journal of Enterprise Technologies. – 2019. – Vol. 5. – P. 26-32.
39. Yelibayeva G., Sharipbay A., Mukanova A. et al. Applied Ontology for the Automatic Classification of Simple Sentences of the Kazakh Language // Procced. 5th internat. conf. on Computer Science and Engineering (UBMK). – Diyarbakir, 2020. – P. 13-18.
40. Nugumanova A., Baiburin Y., Alimzhanov Y. Sentiment Analysis of Reviews in Kazakh With Transfer Learning Techniques // Procced. internat. conf. on Smart Information Systems and Technologies (SIST). – Nur-Sultam, 2022. – P. 1-6.
41. Parhat S., Sattar M., Hamdulla A. et al. Uyghur – Kazakh – Kirghiz Text Keyword Extraction Based on Morpheme Segmentation // Information. – 2023. – Vol. 14, Issue 5. – P. 283-1-283-17.
42. Yegemberdi N.T., Akshabayev A. Speech-to-text Recognition System of Kazakh Language for Development of Search Engines // Вопросы устойчивого развития общества. – 2022. – №4. – С. 1428-1435.
43. Akhmed-Zaki D., Mansurova M., Madiyeva G. et al. Development of the information system for the Kazakh language preprocessing // Cogent Engineering. – 2021. – Vol. 8, Issue 1. – Р. 1-15.
44. Рахимова Д.Р., Турганбаева А.О. Задача нормализации слов казахского языка // Научно-технический вестник информационных технологий, механики и оптики. – 2020. – Т. 20, №4. – С. 545-551.
45. Aitimov A.K., Amirgaliyev Y.N. Spell Checking Application in Kazakh Language // Bulletin Suleyman Demirel University. – 2015. – №1. – P. 132-135.
46. Kuanyshbay D., Baimuratov O., Amirgaliyev Y. et al. Speech data collection system for Kazakh language // Procced. 16th internat. conf. on Electronics Computer and Computation (ICECCO). – Kaskelen, 2021. – P. 1-8.
47. Zhumanov Z., Madiyeva A., Rakhimova D. New Kazakh Parallel Text Corpora with On-line Access // In book: Computational Collective Intelligence. (ICCCI 2017). – Cham: Springer, 2017. – P. 501-508.
48. Баймуратов О.А., Аязбаев Д.А. Мамандандырылған сөздердiң векторлары арқылы сөздердiң лексикалық тiркесулерiн анықтау // ҚазҰУ хабаршысы. – 2020. – Т. 107, №3. – Б. 67-73.
49. NLPL word embeddings repository // <http://vectors.nlpl.eu.> 07.12.2023.
50. Stemming // https://searchenterpriseai.techtarget.com/. 07.12.2023.
51. Bogdanchikov A., Baimuratov O., Ayazbayev D. Stemming of the Kazakh language // Bulletin of Abai Kazakh National Pedagogical University. – 2021. – Vol. 1, Issue 73. – Р. 162-166.
52. Function (mathematics) // <https://en.wikipedia.org/wiki.> 07.12.2023.
53. Top Neural Network Architectures For Machine Learning Researchers // [https://www.marktechpost.com/2022/09/23/top-neural-network](https://www.marktechpost.com/2022/09/23/top-neural-network-architectures-for-machine-learning-researchers/). 07.12.2023.
54. What is batch size in neural network? // [https://stats.stackexchange.com.](https://stats.stackexchange.com/questions/153531/what-is-batch-size-in-neural-network) 07.12.2023.
55. Multi-Domain Bilingual Kazakh Dataset // https://huggingface. 11.12.2023.
56. Әбілқасымов Б., Бизақов С., Жүнісбеков Ә. және т.б. Қазақ әдеби тілінің сөздігі. – Алмата: Дәуі, 2011. – Т. 5. – 752 б.
57. Фазылжанова А., Оңғарбаева Н., Ғабитханұлы Қ. және т.б. Қазақ әдеби тілінің сөздігі. – Алмата: Дәуі, 2011. – Т. 13. – 752 б.
58. Қоңыратбаева Ж., Қалиев Ғ., Есенова Қ. және т.б. Қазақ әдеби тілінің сөздігі. – Алмата: Дәуі, 2011. – Т. 6. – 752 б.
59. Күдеринова Қ., Жұбаева О., Жолшаева М. және т.б. Қазақ әдеби тілінің сөздігі. – Алмата: Дәуі, 2011. – Т. 8. – 744 б.
60. Малбақов М., Оңғарбаева Н., Үдербаев А. және т.б. Қазақ әдеби тілінің сөздігі. – Алмата: Дәуі, 2011. – Т. 10. – 752 б.
61. Манкеева Ж., Шойбеков Р., Күдеринова Қ. және т.б. Қазақ әдеби тілінің сөздігі. – Алмата: Дәуі, 2011. – Т. 11. – 752 б.
62. Әшімбаева Н., Ақаев С., Рысбергенова Қ. және т.б. Қазақ әдеби тілінің сөздігі. – Алмата: Дәуі, 2011. – Т. 4. – 752 б.
63. Момынова Б., Сүйерқұлова Б., Фазылжанова А. және т.б. Қазақ әдеби тілінің сөздігі. – Алмата: Дәуі, 2011. – Т. 7. – 752 б.
64. Үдербаев А., Ноқысбеков О., Күдеринова Қ. және т.б. Қазақ әдеби тілінің сөздігі. – Алмата: Дәуі, 2011. – Т. 12. – 752 б.
65. Ыбырайым Ә., Жаңабекова А., Рысбергенова Қ. және т.б. Қазақ әдеби тілінің сөздігі. – Алмата: Дәуі, 2011. – Т. 9. – 744 б.
66. Жанұзақов Т., Омарбеков С., Жұнңсбек Ә. және т.б. Қазақ әдеби тілінің сөздігі. – Алмата: Дәуі, 2006. – Т. 1. – 752 б.
67. Қалиев Ғ., Бизақов С., Нақысбеков О. және т.б. Қазақ әдеби тілінің сөздігі. – Алмата: Дәуі, 2006. – Т. 2. – 744 б.
68. Сүйерқұлова Б., Жапақов С., Жанұзақов Т. және т.б. Қазақ әдеби тілінің сөздігі. – Алмата: Дәуі, 2006. – Т. 3. – 744 б.
69. Жаңабекова А., Жанұзақ Т., Күдеринова Қ. және т.б. Қазақ әдеби тілінің сөздігі. – Алмата: Дәуі, 2011. – Т. 15. – 828 б.
70. Арпабеков С., Р.М. Әбдіқұлова, З.С. Күзекова. Қазақ тілі. – Алматы: Дәуі, 2004. – 120 б.
71. Index of /kkwiki/ // <https://dumps.wikimedia.org/kkwiki/>. 07.12.2023.
72. Sozdikqor // <https://sozdikqor.kz/sozdik/?id=38.> 11.12.2023.
73. Ayazbayev D., Bogdanchikov A., Orynbekova K. et al. Defining semantically close words of Kazakh language by distributed system Apache Spark // Big Data and Cognitive Computing. – 2023. – Vol. 7, Issue 4. – P. 1-13.
74. Bogdanchikov A., Ayazbayev D., Varlamis I. Classification of Scientific Documents in the Kazakh Language Using Deep Neural Networks and a Fusion of Images and Text // Big Data and Cognitive Computing. – 2022. – Vol. 6, Issue 4. – P. 1-12.