Article

Defining Semantically Close Words of Kazakh Language with Distributed System Apache Spark

Dauren Ayazbayev [1,*], Andrey Bogdanchikov [1,*], Kamila Orynbekova [1] and Iraklis Varlamis [2]

[1] Department of Computer Science, Suleyman Demirel University, Kaskelen 040900, Kazakhstan; kamila.orynbekova@sdu.edu.kz
[2] Department of Informatics and Telematics, Harokopio University of Athens, 17779 Athens, Greece; varlamis@hua.gr
* Correspondence: dauren.ayazbayev@sdu.edu.kz (D.A.); andrey.bogdanchikov@sdu.edu.kz (A.B.)

Abstract: This work focuses on determining semantically close words and using semantic similarity in general in order to improve performance in information retrieval tasks. The semantic similarity of words is an important task with many applications from information retrieval to spell checking or even document clustering and classification. Although, in languages with rich linguistic resources, the methods and tools for this task are well established, some languages do not have such tools. The first step in our experiment is to represent the words in a collection in a vector form and then define the semantic similarity of the terms using a vector similarity method. In order to tame the complexity of the task, which relies on the number of word (and, consequently, of the vector) pairs that have to be combined in order to define the semantically closest word pairs, A distributed method that runs on Apache Spark is designed to reduce the calculation time by running comparison tasks in parallel. Three alternative implementations are proposed and tested using a list of target words and seeking the most semantically similar words from a lexicon for each one of them. In a second step, we employ pre-trained multilingual sentence transformers to capture the content semantics at a sentence level and a vector-based semantic index to accelerate the searches. The code is written in MapReduce, and the experiments and results show that the proposed methods can provide an interesting solution for finding similar words or texts in the Kazakh language.

Keywords: cosine; natural language processing; semantically close words; Apache Spark; FAISS; vector

check for updates

Citation: Ayazbayev, D.; Bogdanchikov, A.; Orynbekova, K.; Varlamis, I. Defining Semantically Close Words of Kazakh Language with Distributed System Apache Spark. *Big Data Cogn. Comput.* **2023**, *7*, 160. https://doi.org/10.3390/bdcc7040160

Academic Editor: Giuseppe Maria Luigi Sarnè

Received: 29 August 2023
Revised: 14 September 2023
Accepted: 20 September 2023
Published: 27 September 2023

1. Introduction

The range of natural language processing (NLP) tasks is very wide, comprising tasks from document or collection level (e.g., text summarization, clustering or classification) to sentence or word level (e.g., defining misspelled words, question-answering, etc.) [1]. As in every machine learning task, one of the main challenges is to handle noise and outliers, which in the case of NLP is expressed as misspelled words, out-of-vocabulary terms and, in some cases, mistakes in the use of words with their proper meaning. Typical examples comprise words that are improperly used in a context, words that cannot be found in the dictionary or words with typos that may confuse grammatical, syntactic and semantic parsers. In these cases, semantically close words are proposed by the authors to help reduce the number of mistakes. In the current work, we examine semantic similarity in the context of finding the most similar words for a given word, which can have many applications from information retrieval and question answering to word replacement in paraphrasing tasks.

Words can be semantically close to each other by various features. For instance, two objects can have the same color, shape, mass, etc. Therefore, the words in word embedding space should be close to each other. However, according to [2], word frequency has an influence on the direction of the vector. Even if words have the same meaning but are used at a different frequency, their vectors can have different directions. The authors of work [2] tried to solve this issue. In [3], the authors tried to define word embedding for speech signals. To accomplish this, they used RNN Encoder–Decoder framework, Skip-gram and continuous bag-of-words methodologies.

Nowadays, there are various tools for the English language that can define semantically close words. However, even if they show good results for the English language, they cannot be applied for all languages as well. This is because different languages may have different structures and follow different grammatic and syntactic rules. For instance, prepositions in the English language are used as separate words, whereas in Kazakh, they can be part of a word but not separate words. In order to demonstrate such difficulties, in the current paper, we search for semantically close words in the Kazakh language. This has some intrinsic difficulties mainly because of the use of different suffixes in a word stem that can significantly change its meaning. Our main motivation for this research was to evaluate the usability and performance of semantic search methods and tools and to provide a generic approach that can be easily applied to any language.

The proposed methods aim to be precise and time efficient, and for this purpose, we first represent words as vectors in a multidimensional space by using word embedding and then compare word pairs using the popular similarity measure of cosine (the angle formed by the two vectors). Since the computation time can vary depending on the dimension of the vector and the number of vectors to compare, we propose three different methods that parallel the cosine similarity computation task for multiple pairs. For this purpose, we employ the Apache Spark distributed system, an analytic engine that allows for the computation of tasks in parallel. It also provides interfaces for many popular programming languages, like Python, R, Scala and Java.

In order to evaluate the effectiveness of the proposed approach, we first compile a dataset comprising semantically similar words to several words of the Kazakh language and use popular information retrieval metrics to compare the results. In order to evaluate the time efficiency of the different implementations, we compare the time needed to perform all the necessary comparisons. The contributions of this work can be summarized as the following:

- We develop a new word pair dataset that contains semantically close words of the Kazakh language.
- We propose and implement a method for finding semantically similar words and evaluate its performance in terms of time complexity.
- We improve computation time for many word pairs by developing a parallelized approach.
- We propose a processing pipeline for finding semantically similar documents in languages with limited linguistic resources.

Although the proposed approach can be applied to any language that has the respective resources (e.g., pretrained word embedding or sentence transformers), we decide to test it on the Kazakh language, which is a good example of a language with limited linguistic resources.

In the following section, we perform an overview of works in the Kazakh language that employ word embedding and define semantic similarity in the vector space as well as works that parallelize the vector comparison tasks using a distributed architecture. Section 3 provides the details of the proposed approach and discusses the implemented methods and the materials we employed. Sections 4 and 5 explain the experimental setup and discuss the results achieved so far. Finally, Sections 6 and 7 conclude the paper by summarizing our findings.

2. Literature Review

Among the various works that have been published in the field of NLP in the Kazakh language, we focus on those that relate to our work either in the way they define similarity or in the way they parallelize tasks using distributed architectures. The authors in [4] propose a method for sentiment analysis in excerpts from books written in Kazakh. Their approach was implemented using a distributed architecture on Apache Spark. On a different task, parallelization is used via the Apache Hadoop framework in order to speed up the task of face detection in public images [5].

Back to NLP tasks, it is evident that word-embedding models are frequently employed for mapping words from the same or different languages to a common space of lower dimensionality that allows for better capture of the semantic relatedness between the words. The authors in [6] compared Skip-gram and continuous bag-of-words (CBOW) models and claimed that Skip-gram has better accuracy, whereas the CBOW model can work faster. In their task, the authors tried to map words from one language to another via a shared representation. In particular, they considered Spanish, English, Czech and Vietnamese and described a method in which each word was represented by a vector. In the resulting spaces, the distances between the vectors (i.e., words) from different languages are kept approximately the same as shown in Figure 1, which illustrates an example of how English and Spanish words are projected on a 2-D plane. Therefore, the authors tried to find out the shifts between the vectors by learning a transformation matrix. With the use of this transformation matrix, a word can be translated from one language to another.

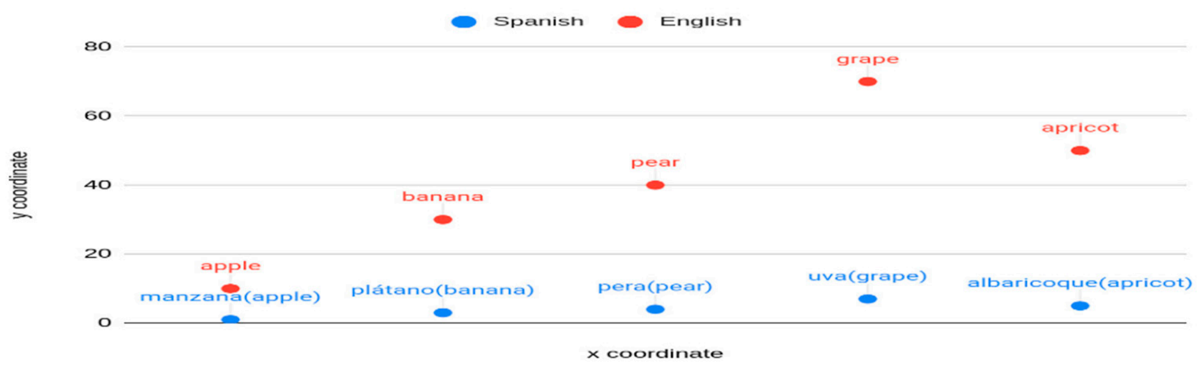


Figure 1. 2-D vectors of English words and the corresponding words in Spanish.

In paper [7], the authors compared the models Skip-gram and CBOW. In their experiment, the Skip-gram model showed higher accuracy, but its training time was longer. Taking this into account, the authors proposed their own method, which combines two models, to achieve high accuracy in a short time. The authors of work [8] showed several methods for detecting similar words. The similarities can be identified by calculating the cosine similarity or with the Euclidean, Manhattan, Hellinger, Bhattacharya and Kullback–Leibler distances. Moreover, the authors showed how the computation time for computing the similarity can be reduced. One method was based on the Hadoop framework, which was used to parallelize the computation of the distances between pairs of vectors. Another method was to neglect the zero coordinates of the vectors and use only the non-zero dimensions in the calculation of similarity, since the product of zero equals zero. The authors in [9] classified comments with the Apache Spark framework. Particularly, they classified comments that can offend people into six types.

Finally, the authors of [10] calculated the closeness of sentences in the Bengali language. They collected sentences from various sources, like newspapers, Facebook comments and posts, text conversations, etc. What is interesting with this work is that they defined the closeness of two sentences using any of the following three methods: (a) cosine similarity with no weighting, (b) IDF weighting and (c) part-of-speech weighting. In addition, they defined word embedding for the Bengali language using Word2Vec, Glove and FastText.

A characteristic of the Kazakh language is that the meaning of a word is defined by the main body of the word (i.e., stem or root word) but can change depending on the suffix. To avoid the additional calculations needed for examining each word separately, researchers can examine only the root word as given by the dictionaries [11–16]. Although most works employ the vector representation model and cosine similarity measure to compare words and find the related ones, there are a few more alternatives for comparing words. The authors in [17] designed a methodology that can be applied to evaluate the accuracy of clustering publications. Their methodology used relatedness measures A, B, where A and B can be citation-based relatedness measures. The author in [18] proposed a model that can measure the relatedness of two sentences. The model had four layers with different similarity features. The experiment showed that using multiple similarity features was better than applying each measure separately. In [19], the authors employ the same lexical thesaurus (i.e., Wordnet) and the list of senses it provides for each word and apply a set of theory principles to define the similarity between the words. In [20], the authors employed formal concept analysis in order to map words to a multidimensional space where each word is evaluated for its relevance to a limited set of features (i.e., dimensions), such as object, color, taste, price, etc., before comparing the resulting vector representations using a fuzzy similarity measure that takes into account the maximum and minimum similarity in each dimension.

Representing words or sentences in a vector-based format is the first step in facilitating information retrieval. The next step is to index these representations in a manner that minimizes comparisons when searching for the most similar content and to optimize the comparisons as much as possible. The subsections that follow describe two popular document indexing and retrieval systems. The first is Facebook AI Similarity Search, which provides a vector-based indexing mechanism for sentences or longer texts, and the second is the Elasticsearch platform, which also provides a vector-based search mechanism. Both of them are used for indexing document collections and performing information retrieval tasks. Since semantically close words can be defined by neighboring vectors, the appropriate indexing of these vectors can dramatically improve the search performance, and the semanting-based results are expected to be better than searching with exact keywords.

2.1. Facebook AI Similarity Search

Facebook AI Similarity Search (FAISS) is a library that allows developers to search for similar documents using a unique indexing mechanism. Using FAISS, both text and multimedia documents can be indexed and searched. FAISS is written in C++, and since it supports GPU, searching can be highly paralleled.

FAISS works with vector representations. Given a collection of text (or multimedia) documents and a transformer mechanism, we are able to convert any document to a multidimensional vector and index it in FAISS. Any query is similarly converted to a vector, and the search for the most similar document is consequently based on the calculation between the query and the document vectors. The FAISS library provides the 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 than search sequentially.
(3) The developer can decrease the precision of the result, but in that case, he/she can obtain the 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 was developed based on several research studies. For instance, the authors of work [21] investigated nearest neighbor search. They considered various algorithms, like Lloyd's and hierarchical k-means (HKM). These algorithms have some disadvantages. Therefore, the authors of work [21] proposed their solution. The authors of work [22], in turn, proposed a method of re-ranking vectors with a limited amount of memory. The advantage of their work is that data can be stored in memory. Hence, costly disk accesses can be avoided. In addition, they developed a publicly available dataset of one billion vectors with 128 dimensions. The authors of work [23] proposed the implementation of the KNN algorithm on a GPU. Their implementation constructed a KNN graph on 95 million images in 35 min and connected 1 billion vectors in less than 12 h on four Maxwell Titan X GPUs [23]. The authors of work [24], in turn, have used FAISS to generate stories. In their system, the user enters some keywords. Then, the system should generate a story using those keywords.

For comparison with our search system, FAISS was applied in our experiment.

2.2. Elasticsearch

Our next search system is called Elasticsearch. Elasticsearch is a searching platform that is based on the Lucene library. People with the Elasticsearch platform can search their documents and analyze them. Particularly, Elasticsearch has a Kibana dashboard that can draw various diagrams. People can make decisions based on those diagrams. In addition, Elasticsearch has analyzers for various languages. Strings can be processed using the analyzers. For instance, the stemming of a word can be identified, HTML tags can be deleted, and synonyms of words can be added. Documents in Elasticsearch can have several fields. The user of the Elasticsearch platform can define a priority to each field by himself/herself. Hence, the user of the Elasticsearch platform can manage the search results. Elasticsearch also supports operations like "and", "or" and "negation", which can expand or reduce the search area [25]. Elasticsearch added a vector search capability that allows for searching of semantically similar terms.

All the aforementioned methods show that word embedding is a popular representation technique for defining word similarity, and text-relatedness metrics can be based on them. In addition, they show that the parallelization of similarity calculations is feasible with the use of a distributed platform and can significantly help in information retrieval tasks. Sentence transformers can also be exploited to facilitate semantic search, especially when they are combined with semantic indices. In the section that follows, we provide more details on the proposed method that bridges the two state-of-the-art practices in text representation and comparison into an integrated approach for defining semantically related words and demonstrates its application in the Kazakh language.

3. Methods and Materials

3.1. Defining Similarities of Words

The vector representation of words can be learned using any of the existing word-embedding techniques that are available in the literature [26]. In the current work, we used the Skip-gram model and the NLPL word-embedding repository, which has a vocabulary that contains 176,643 tokens. Semantically close words were defined as follows:

1. The list of 176,643 vectors contains one vector for each word. As a first step, for a target word, we need to find its coordinates from the list. For example, a vector that corresponds to the word 'silver' in a five-dimensional embedding space would be like (−0.276359, −0.480686, −0.435527, 0.388631, 0.773380). However, the dimension of the vectors we used was 100.
2. The next step is to calculate the cosine similarity between the vector of the target word and the vectors of all words in the list. The cosine similarity is defined by Formula (1):

$$cos(\alpha) = \frac{a_1 b_1 + a_2 b_2 + \cdots + a_{100} b_{100}}{\sqrt{{a_1}^2 + {a_2}^2 + \cdots {a_{100}}^2}\sqrt{{b_1}^2 + {b_2}^2 + \cdots {b_{100}}^2}} \tag{1}$$

   where a_n, b_n are the coordinates of two vectors. As an example, Table 1 contains the cosine similarity of silver and sample words from the list. A higher cosine value (the maximum is 1) corresponds to a higher degree of relation, whereas a lower value (i.e., close to 0) corresponds to less-related words.
3. After computing the cosine similarities for all pairs, we decide to keep the ten most-related words for each target word. This means that, from the 176,643 cosine values calculated for each word, we have to choose the words with the 10 maximum cosine values.

Table 1. Cosine values of silver and other words.

The First Word	The Second Word	Cosine Value
Silver	Iron	0.84
Silver	Banana	0.34
Silver	Lead	0.87
Silver	Apple	0.27
Silver	Tree	0.12

In order to find the best 10 words, we implement three different strategies. In the first strategy, the cosine value for the first pair of words (i.e., <target word, word1>) is set to be the maximum value, and the comparison continues with the remaining words in the list. Every time a word from the list is found to have a similarity value higher than the running maximum value, the new value replaces the running maximum value, and the word is chosen as the running best candidate. At the end of this process, the found best candidate is removed from the list. The process is repeated as many times as needed (10 in our case) to find the best candidates.

In the second method, we first compute the first 10 cosine values and store them in a data structure (e.g., list) in a sorted way. Then, the 11th cosine value is compared with the 10th value, then the 9th value and so on. If it is greater than the x-th value, then the 11th value is inserted into the sorted list and takes the place of element x, shifting all the subsequent elements in the list. Then, the last value will be removed from the list. If the 11th value is smaller than any element in the sorted list, then the next element is compared with the 10 elements in the list and so on.

In the third strategy, we calculate the cosine similarities between the target word and all the words in the list in advance, and we then sort all these similarities using a Python 3 native sorting function. At the end, we keep the top 10 similarities and the respective words. This strategy requires much more memory to store the long list of similarities but processes the list of words once and runs the fastest possible sorting algorithm at the end of this process.

For the evaluation of our results, we decided to compare our results with those of methods that employ different word embedding in order to calculate semantic word similarities. For this purpose, we employ Polyglot [27], a library of word embedding for 137 languages that provides 100,004 words of the Kazakh language, the embedding vectors of dimension 64 each. In our experiments, we use polyglot embedding to calculate the cosine similarities again and use the precision metric to evaluate the results. To calculate the precision of semantically related words for each target word, we rank similar words using their semantic relatedness, and consequently, each word is classified as related or not related. The precision is calculated using Formula (2):

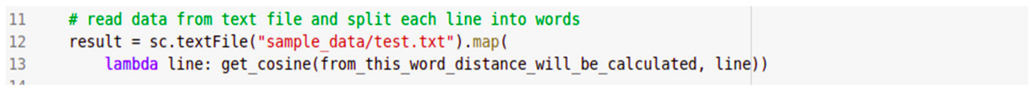
$$\text{Precision} = \text{tp}/(\text{tp} + \text{fp}) \tag{2}$$

where tp corresponds to the true positives (i.e., predicted similar words that are evaluated as similar by humans), and fp corresponds to the false positive examples (i.e., predicted similar words that humans have not classified as similar). The resulting precision for polyglot was equal to 88.61%, whereas the resulting precision for NLPL word embedding was equal to 92.2%.

3.2. Parallelizing the Similarity Calculation Tasks

In order to execute the huge amount of cosine similarity calculations, we decide to parallelize the task on Apache Spark. Apache Spark works on the MapReduce programming pattern, which processes the key value pairs.

In our experiment, the key was the word, and the value was the vector of the word. At each round, the mapper receives key–value pairs one-by-one and calculates the cosine of the target word's vector and the received word's vector. All these cosine values are stored as keys, and the respective word pairs as values. As a result, for each target word and the respective vector, we compute 176,643 cosine values that must be sorted with one of the strategies in order to choose the ten best keys from the list, which correspond to the words that are the nearest words to the target word. To read the coordinates of the embedding vectors, the textFile method of the SparkContext class is used. According to the Apache Spark documentation [28], textFile reads a text file from a Hadoop-supported file system URI and returns a resilient distributed dataset (RDD) of strings. After that, the map method is called, which receives the coordinates of one vector and calculates the respective cosine value. Figure 2 summarizes this process.



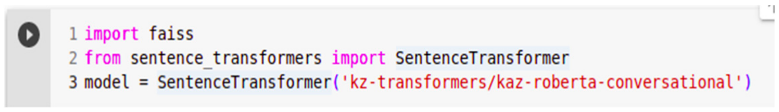
```
# read data from text file and split each line into words
result = sc.textFile("sample_data/test.txt").map(
    lambda line: get_cosine(from_this_word_distance_will_be_calculated, line))
```

Figure 2. Example of code snippet.

3.3. Sentence Transformers and Experiment with FAISS

As it was mentioned above, FAISS is a library for indexing and searching documents based on their vector representations. In order to test the applicability of the proposed vector-based approach in defining semantic similarity, we perform an experiment in the Kazakh language. For this purpose, we transform a large number of documents (titles of Wikipedia articles written in Kazakh) into vectors using a sentence transformer. More specifically, the model is the kaz-roberta-conversational sentence transformer for the Kazakh language. The sentence transformer has been pre-trained on the CC-100 Monolingual Dataset from Web Crawl Data, books, Leipzig Corpora Collection, Open Super-large Crawled Aggregated coRpus and Kazakh News. The training dataset used in this case had over 2 billion tokens and almost 18 million unique tokens.

The dump file of Wikipedia articles written in the Kazakh language contained 165,592 article titles in an XML format. After removing all the unnecessary tags and other information using Python's wiki_dump_reader library, we obtained the sentence vector for each title. Figure 3 summarizes these statements.



```
import faiss
from sentence_transformers import SentenceTransformer
model = SentenceTransformer('kz-transformers/kaz-roberta-conversational')
```

Figure 3. Importing necessary files.

Then, the vectors were indexed with the FAISS library. Table 2 shows the result of searching similar Wikipedia titles.

Table 2. Returned result with 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)

3.4. Experiment with Elasticsearch

The experimental evaluation of Elasticsearch was performed using the same dataset of Wikipedia titles as with the semantic search and FAISS. Once again, we indexed the titles of the Wikipedia articles, this time performing a full-text search, using the default full-text search functionality (i.e., stemming and exact matching) of Elasticsearch. The main objective was to compare the performance of the semantic (vector-based) search against the respective performance of the full-text search. Table 3 shows the result of Elasticsearch.

Table 3. Returned result with Elasticsearch.

Query	Returned Five Wikipedia Titles
Журнал (Journal)	Мұғалім (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 қызылтамыр (Painting by macrotomia ugamensis), Қышқылдық бояу (Acid dye), Бояу қызылтамыры (Painting by macrotomia ugamensis), Томар Бояу Кермек (Siberian statice)

4. Experimental Setup and Results

As mentioned above, the precision was calculated with the polyglot and NLPL word-embedding models. Particularly, we first defined 99 common words, and for each one of them, we defined nine semantically close words. When precision was calculated, words of which we did not know the meaning were neglected. As a result, the polyglot model had 755 true positive and 97 false positive examples, whereas NLPL had 768 true positive and 65 false positive examples. The totals were different because we did not know the meaning of some words and, therefore, how to classify those words. The resulting precision was 88.61% and 92.2% for polyglot and NLPL, respectively. To verify our result, semantically close words were examined by other people as well. Their average precision was 84%.

A default Google Colab machine was used for the experiments. The free version with the CPU processor was employed.

In the current paper, the time for computation of semantically close words was measured. Figures 4 and 5 show how much time was spent to identify semantically close words. Figure 4 shows that the computation time was almost linearly related to the number of target words. However, in some experiments, we had some slight variations. For instance, to define semantically close words for six target words, it took approximately 345 s, whereas for seven words, it took 334.37 s. Similarly, the computation time for eight words was also not linear and took approximately 358.06 s.

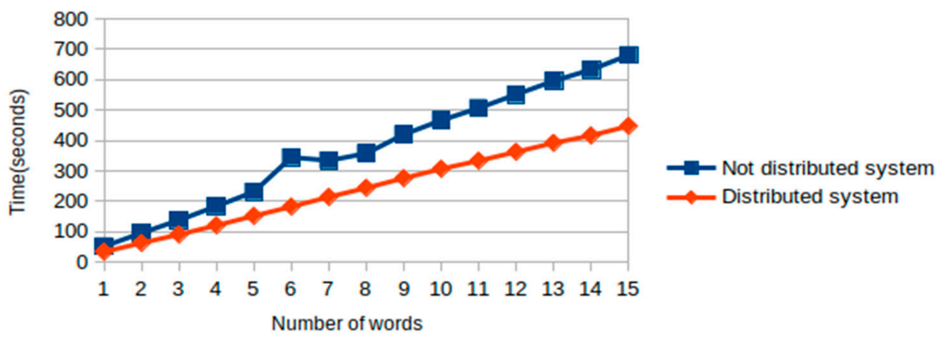


Figure 4. Time to define semantically close words.

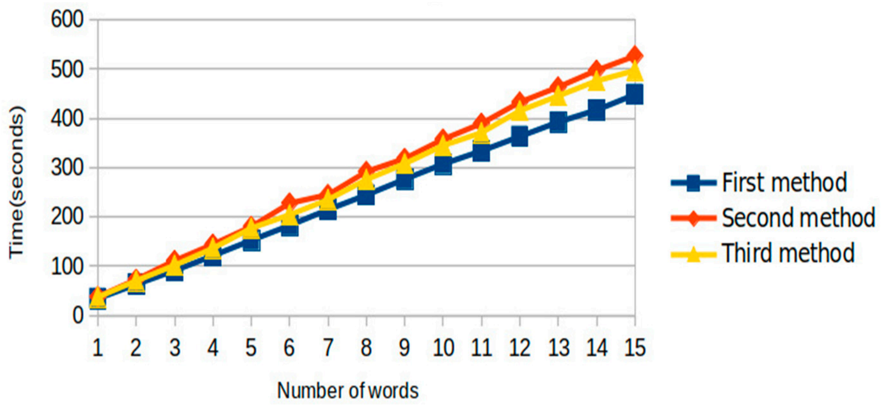


Figure 5. Time to define semantically close words with three methods.

Table 4 shows examples of defined semantically close words with NLPL word embedding.

Table 4. Semantically close words with NLPL word embedding.

Word	Semantically Close Word 1	Semantically Close Word 2	Semantically Close Word 3	Semantically Close Word 4	Semantically Close Word 5
Brave	Brave	By braveness	By justice	Heroes	Of heroes
Scientific	Scientific	Of thesis	Of thesis (plural form)	Of monographs	Most scientific
Big	Big	Little	Huge	Immense	Very
Ancient	Ancient	In the past	Old	Middle ages	Middle ages
Clean	Clean	From diamond	Additional	Because of not allowed	To gasoline

Table 5 shows examples of defined semantically close words with polyglot word embedding.

Table 5. Semantically close words with polyglot word embedding.

Word	Semantically Close Word 1	Semantically Close Word 2	Semantically Close Word 3	Semantically Close Word 4	Semantically Close Word 5
Constant	All	General	Exact	Open	Regularly
Scientific	Exact	Former	Individually	Some	First
Big	High	Constant	World	All	Individually
Ancient	Bottom	Natural	Last	First	All
Clean	Neighbors	Next	Wide	Square	Far

From Tables 4 and 5, we observed that NLPL and polyglot word embeddings showed good results. For instance, in NLPL word embedding for the word brave, the semantically close words were by justice and heroes. In polyglot, for the word big, the semantically close words were high and world. In polyglot, we also noticed that word-embedding words that belong to the same part of speech are located closer than words of different parts of speech.

In the case of the document (i.e., title) search, all the experiments were performed on Google Colab too; therefore, we had to install and import the respective libraries (FAISS, Transformer, Elasticsearch 7.0.0) and then use it in our search task.

5. Discussion

Nowadays, there are various tools for defining semantically close words. However, an algorithm of a certain tool cannot be applied for all languages because each language has its own rules. In the current paper, semantically close words of the Kazakh language were defined with three strategies.

According to Figure 5, among these three strategies, the first method was the fastest. However, the running time of the first strategy depends on the number of vectors. Therefore, if the dataset size is too large, it can lead to a long computation time.

The second strategy is more stable for dataset size because here, for comparison cosines, we use just one iteration. The number of iterations does not depend on the dataset size. The best case of this strategy is when maximum cosines are located at the beginning of the dataset, whereas the worst case is when cosines are sorted in an opposite order than the initial ten cosines.

The performance of the third strategy depends on the algorithm that is used in the Python native sorted function. In addition, it requires more memory to store the result. However, the advantage of this strategy is that more than ten semantically close words can be defined.

In our experiment, to define semantically close words, the NLPL and polyglot word-embedding models were used. The NLPL model showed slightly higher precision. However, the polyglot model located words with the same part of speech closer than other parts of speech. Therefore, the application of the model depends on the task.

6. Application of the Experiment

The authors of work [29] had similar aims as the current paper. However, they used continuous bag-of-words and Skip-gram models. The authors used word embedding to find relevant tweets. The authors of work [30], in turn, performed sentiment analysis of COVID-19 tweets. The authors of work [31] implemented three techniques to extract features from news texts, and one of them was word embedding.

After the current experiment, semantically close words appeared. For one word, ten semantically close words were defined. These semantically close words were used in the search engine called Komekshy. Komekshy is a search engine that can be used to find relevant Wikipedia articles. Komekshy uses semantically close words to expand the user's query. Particularly, the search engine will search not only entered words but semantically close words as well. The result of the current experiment can directly influence the accuracy of the search engine. For instance, from Table 4, we can observe that the semantically close words of the word ancient were in the past, old and middle ages. If the user uses the word ancient in a query, then Komekshy will also search that word from Wikipedia articles. In the case where semantically close words are too general, not relevant Wikipedia articles can be returned. Figure 6 shows how the search engine works.

To expand the user's query, Komekshy will replace the original word with a semantically close word. As a result, from one query, several queries will appear.

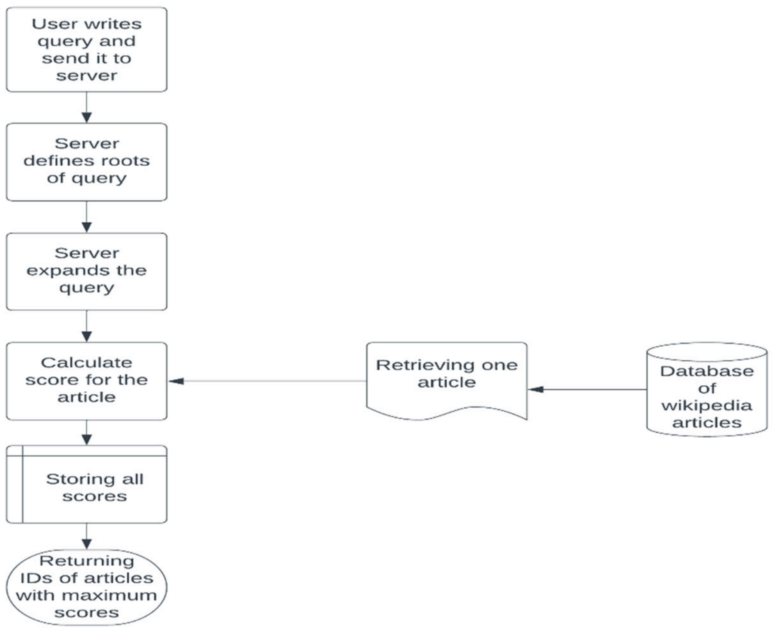

User writes query and send it to server
Server defines roots of query
Server expands the query
Calculate score for the article
Retrieving one article
Database of wikipedia articles
Storing all scores
Returning IDs of articles with maximum scores

Figure 6. Flowchart of the Komekshy search engine.

7. Conclusions

In the experiment of the current paper, vectors from the NLPL repository and polyglot word embeddings were used. Those vectors were used to define semantically close words of the Kazakh language. Semantically close words were defined by calculating the cosines between the vectors and finding the 10 maximum cosines. If the number of vectors is large, it can lead to a long calculation running time. Hence, to reduce that time, Apache Spark was used. Apache Spark allows for the computation of the cosines of vectors on a distributed system. Moreover, to avoid extra calculation, semantically close words were defined only for lemmas. In the experiment, lemmas were taken from [11–16] dictionaries. As a result of the experiment, semantically close words of 8497 words were defined. The semantically close words were defined with three strategies. Among the three strategies, the first strategy was the fastest. However, each method has its own peculiarity. Therefore, the application of these strategies depends on the task. The precision of semantically close words, respectively, was 88.61% and 92.2% for the NLPL and polyglot word-embedding models. This was calculated by doing a survey. From Tables 4 and 5, we can observe the semantically close words of the NLPL and polyglot word-embedding models.

Author Contributions: Conceptualization, A.B., I.V., D.A. and K.O.; methodology, A.B., I.V., D.A. and K.O.; software, A.B., I.V., D.A. and K.O.; validation, A.B., I.V., D.A. and K.O.; formal analysis, A.B., I.V., D.A. and K.O.; investigation, A.B., I.V., D.A. and K.O.; resources, A.B., I.V., D.A. and K.O.; data curation, A.B., I.V., D.A. and K.O.; writing—original draft preparation, A.B., I.V., D.A. and K.O.; writing—review and editing, A.B., I.V., D.A. and K.O.; visualization, A.B., I.V., D.A. and K.O.; supervision, A.B. and I.V.; project administration, A.B.; funding acquisition, this research received no external funding. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Polyglot word embedding was available at https://sites.google.com/site/rmyeid/projects/polyglot (accessed on 3 August 2023). NLPL word embedding was available at http://vectors.nlpl.eu/repository/ (accessed on 3 August 2023). Information about Kazakh transformer was taken from https://huggingface.co/datasets/kz-transformers/multidomain-kazakh-dataset (accessed on 28 August 2023).

Acknowledgments: The authors would like to acknowledge the support and facilities provided by the Suleyman Demirel University, Kazakhstan and Darmen Kariboz, Mukhtar Amirkumar and Abylaikhan Baidullayev for their help conducting the experiment of the current paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Abacha, A.B.; Pierre, Z. Means: A medical question-answering system combining NLP techniques and semantic Web technologies. *Inf. Process. Manag.* **2015**, *51*, 570–594. [CrossRef]
2. Gong, C.; He, D.; Tan, X.; Qin, T.; Wang, L.; Liu, T.-Y. FRAGE: Frequency-Agnostic Word Representation. *Adv. Neural Inf. Process. Syst.* **2018**, 1341–1352.
3. Chung, Y.; Glass, J. Speech2Vec: A Sequence-to-Sequence Framework for Learning Word Embeddings from Speech. *arXiv* **2018**, arXiv:1803.08976.
4. Serek, A.; Issabek, A.; Bogdanchikov, A. Distributed sentiment analysis of an agglutinative language via spark by applying machine learning methods. In Proceedings of the 15th International Conference on Electronics, Computer and Computation (ICECCO), Abuja, Nigeria, 10–12 December 2019; pp. 1–4.
5. Bogdanchikov, A.; Kariboz, D.; Meraliyev, M. Face extraction and recognition from public images using hipi. In Proceedings of the 14th International Conference on Electronics Computer and Computation (ICECCO), Kaskelen, Kazakhstan, 29 November–1 December 2018; pp. 206–212.
6. Mikolov, T.; Le, Q.; Sutskever, I. Exploiting similarities among languages for machine translation. *arXiv* **2018**, arXiv:1309.4168.
7. Onishi, T.; Shiina, H. Distributed Representation Computation Using CBOW Model and Skip–gram Model. In Proceedings of the 9th International Congress on Advanced Applied Informatics (IIAI-AAI), Kitakyushu, Japan, 1–15 September 2020; pp. 845–846.
8. Turney, P.; Pantel, P. From frequency to meaning: Vector space models of semantics. *J. Artif. Intell. Res.* **2010**, *37*, 141–188. [CrossRef]
9. Carta, S.; Corriga, A.; Mulas, R.; Recupero, D.R.; Saia, R. A Supervised Multi-class Multi-label Word Embeddings Approach for Toxic Comment Classification. KDIR 2019—11th International Conference on Knowledge Discovery and Information Retrieval, Vienna, Austria, 17–19 September 2019; pp. 105–112.
10. Iqbal, M.A.; Sharif, O.; Hoque, M.M.; Sarker, I.H. Word embedding based textual semantic similarity measure in bengali. *Procedia Comput. Sci.* **2021**, *193*, 92–101. [CrossRef]
11. Abilkasymov, B.; Bizakov, S.; ZHynisbekov, A.; Malbakov, M.; Konyratbaeva, Z.H.; Nakysbekov, O. *Qazaq Ädebi Tılınıñ Sözdıgı*; Dauir: Almaty, Kazakhstan, 2011; pp. 1–752.
12. Fazylzhanova, A.; Ongarbaeva, N.; Gabithanyly, K.; SHojbekov, R.; Kyderinova, K.; ZHybaeva, O.; Malbakov, M. *Qazaq Ädebi Tılınıñ Sözdıgı*; Dauir: Almaty, Kazakhstan, 2011; pp. 1–752.
13. Konyratbaeva, Z.H.; Kaliev, G.; Esenova, K.; ZHanyzak, T.; Momynova, B.; Syjerkylova, B. *Qazaq Ädebi Tılınıñ Sözdıgı*; Dauir: Almaty, Kazakhstan, 2011; pp. 1–752.
14. Kyderinova, K.; ZHybaeva, O.; ZHolshaeva, M.; Gabithanyly, K.; Ashimbaeva, N.; Yderbaev, A.; Imangazina, A. *Qazaq Ädebi Tılınıñ Sözdıgı*; Dauir: Almaty, Kazakhstan, 2011; pp. 1–744.
15. Malbakov, M.; Ongarbaeva, N.; Yderbaev, A.; Imanberdieva, S.; SHojbekov, R.; Fazylzhanova, A.; Smagylova, G.; Kyderinova, K.; ZHanabekova, A.; Halykova, G.; et al. *Qazaq Ädebi Tılınıñ Sözdıgı*; Dauir: Almaty, Kazakhstan, 2011; pp. 1–752.
16. Mankeeva, Z.H.; SHojbekov, R.; Kyderinova, K.; Fazylzhanova, A.; Bizakov, S.; ZHynisbek, A.; ZHanabekova, A.; Yderbaev, A.; Kaliev, G. *Qazaq Ädebi Tılınıñ Sözdıgı*; Dauir: Almaty, Kazakhstan, 2011; p. 752.
17. Waltman, L.; Boyack, K.W.; Colavizza, G.; Jan van Eck, N. A principled methodology for comparing relatedness measures for clustering publications. *Quant. Sci. Stud.* **2020**, *1*, 691–713. [CrossRef]
18. Gomaa, W.H. A multi-layer system for semantic relatedness evaluation. *J. Theor. Appl. Inf. Technol.* **2019**, 3536–3544.
19. Ezzikouri, H.; Madani, Y.; Erritali, M.; Oukessou, M. A new approach for calculating semantic similarity between words using wordnet and set theory. *Procedia Comput. Sci.* **2019**, *15*, 1261–1265. [CrossRef]
20. Jain, S.; Seeja, K.; Jindal, R. A new methodology for computing semantic relatedness: Modified latent semantic analysis by fuzzy formal concept analysis. *Procedia Comput. Sci.* **2020**, *167*, 1102–1109. [CrossRef]
21. Jégou, H.; Douze, M.; Schmid, C. Product Quantization for Nearest Neighbor Search. *IEEE Trans. Pattern Anal. Mach. Intell.* **2011**, *33*, 117–128. [CrossRef] [PubMed]
22. Jégou, H.; Tavenard, R.; Douze, M.; Amsaleg, L. SEARCHING IN ONE BILLION VECTORS: RE-RANK WITH SOURCE CODING. *arXiv* **2011**, arXiv:1102.3828.
23. Johnson, J.; Douze, M.; Jégou, H. Billion-scale similarity search with GPUs. *arXiv* **2017**, arXiv:1702.08734v1. [CrossRef]
24. George, G.; Rajan, R. A FAISS-based Search for Story Generation. In Proceedings of the 2022 IEEE 19th India Council International Conference (INDICON), Kochi, India, 24–26 November 2022; pp. 1–6.
25. Basics of Elasticsearch. Available online: https://habr.com/ru/articles/280488/ (accessed on 28 August 2023).
26. Li, Y.; Yang, T. Word embedding for understanding natural language: A survey. In *Guide to Big Data Applications*; Springer International Publishing: New York, NY, USA, 2018; pp. 83–104.

27. Al-Rfou, R.; Perozzi, B.; Skiena, S. Polyglot: Distributed Word Representations for Multilingual NLP. In Proceedings of the Seventeenth Conference on Computational Natural Language Learning, Sofia, Bulgaria, 8–9 August 2013; pp. 183–192.
28. Pyspark. SparkContext.textFile—PySpark 3.1.2 Documentation. Available online: https://spark.apache.org/docs/3.1.2/api/python/reference/api/pyspark.SparkContext.textFile.html (accessed on 20 February 2023).
29. Biggers, F.B.; Mohanty, S.D.; Manda, P. A deep semantic matching approach for identifying relevant messages for social media analysis. *Sci. Rep.* **2023**, *13*, 12005. [CrossRef] [PubMed]
30. Javad, H.J.; Sadiq, H.; Mohammad Ali, N.; Rouhollah, B.; Fatemeh, F.; Roohallah, A.; Reza, L.; Ashis, T. BERT-deep CNN: State of the art for sentiment analysis of COVID-19 tweets. *Soc. Netw. Anal. Min.* **2023**, *13*, 99.
31. Dana, S.; Rebwar, M.N. Kurdish Fake News Detection Based on Machine Learning Approaches. *Passer J. Basic Appl. Sci.* **2023**, *5*, 262–271. [CrossRef]