Improving information retrieval by semantic embedding

Ye Yuan
University of Twente
PO Box 217, 7500 AE Enschede
the Netherlands
y.yuan-1@student.utwente.nl

Abstract

This research focuses on using semantic embedding to improve the performance of Information Retrieval (IR) for the Covid-19 related tasks. According to previous research, the technology of word embedding can significantly improve the performance of IR. There are many types of semantic embedding models at present. The purpose of this research is not to develop a new one, but to combine multiple popular semantic embedding models and to find a more effective ranking for retrieving a better IR result by a comparative analysis of these semantic embedding technologies. Besides, the current embeddings are mostly based on words, phrases, or documents, not on entities. So, providing the entity-based IR function, which is missing in PubMed or other search engines like Google, is another goal of this research. The expected outcome of this research is an entity-based working prototype focusing on the Covid-19 data, which can visually mark the differences between the search results of different semantic embedding models.

Keywords

word embedding, entity embedding, document embedding, NLP, word2vec, doc2vec, Fasttext, GloVe, cosine similarity

1. INTRODUCTION

The recent (2020) worldwide prevalence of COVID-19 has caused extreme damage in all areas. Due to the limited time, how to build a more sophisticated and intelligent search engine for collecting data to study the virus has become an urgent task for all involved researchers. The two famous medical datasets, MEDLINE[1] and Covid-19[2], contain massive medical data and articles covering a variety of medical cases. However, at present, the most important point we focused on is Covid-19 and the IR [1] method of these two data sets is keyword-based. Although information retrieval with the traditional keyword-based [2] method is workable, there is a large room for improvement in efficiency and accuracy. In the traditional keyword-based way, you can retrieve the accurate results with certain keywords, but the coronavirus involving a lot of technical terms with similar meanings. Searching information with a keyword-based search engine is often time-consuming and returns poor data results, which reduces the efficiency of the process of coronavirus-related research. We hope to have smarter word embedding [3] IR models that considers semantic similarities of words in the matching process, to generate high usable searching results.

By definition, a set of language modeling and feature learning techniques [4] in natural language processing (NLP) are collectively referred to as word embedding, which has been proven being able to improve the performance in NLP tasks such as sentiment classification [5], dependency parsing [6] and sentiment analysis [7]. With word embedding models, preprocessed text or documents are mapped to vectors of real numbers by technologies such as neural networks [8], dimensionality reduction on the word co-occurrence matrix [9], etc. Since it takes the context where the word appears into consideration, it makes the predication of missing words in a document possible. In contrast, the traditional keyword-based search engine cannot solve the problem of high term mismatch and consider the distinct meaning between the semantically similar words in the matching process.

This research project is to develop a search engine based on word embedding models and find a way to optimize the IR concerning coronavirus related data. More specifically, several research issues need to be solved:

1. What is the performance difference between various semantic embedding technologies?
2. How can we combine the results of different semantic embedding to achieve a better retrieval result?
3. How can we compare the results of different semantic embedding more efficiently?

The contribution of this paper is that it shows the precision and the recall of the IR results of Ranking are better than that of all other four technologies to some extent. Although the improvement varies according to different models, it improves the efficiency of IR.

The structure of remaining paper is arranged as follows. Section 2 briefly describes the application of a couple of popular semantic embedding technologies and two studies on the analysis and measurement of the four embeddings involved in the present research. Section 3 presents the algorithm and UI of the feature of 'marking searching results visually'. Section 4 provides the formula, algorithms, and strategies for measurement and ranking. Section 5 presents some influences of training models in this experiment. Section 6 provides the details about the results of the measurement and the involved user evaluation. Section 7 describes the limitations and future works of this research.

2. RELATED WORKS

Semantic embedding

At present, there are two main methods to implement the word embedding: count-based and direct prediction. The popular count-based word embedding frameworks are LSA (Burgess & Lund), COALS, Hellinger-PCA (Rohed etal, Collobert & Lebret) etc. For the word embedding frameworks based on direct prediction, there are Skip-gram/CBOW (Mikolov et al),

[1] https://www.nlm.nih.gov/bsd/medline.html

[2] https://www.kaggle.com/allen-institute-for-ai/CORD-19-research-challenge

NNLM, HLBL, Skip-gram/CBOW (Mikolov et al), NNLM, HLBL, RNN (Bengio et al; Collobert & Weston; Huang et al; Mnih & Hinton) etc.

The main advantages of the count-based word embedding are fast training and efficient usage of statistics, and that of the direct prediction frameworks is being able to capture complex patterns of the context. However, they have various disadvantages, such as only words similarity can be acquired, and disproportionate importance given to large counts for the count-based embedding. By contrast, the limitations of the direct prediction frameworks are scale with corpus size and inefficient usage of statistics.

In this research, four semantic embedding technologies are analyzed: word2vec [10], doc2vec [11], Fasttext [12, 13] and GloVe [14]. The problem of sparsity in word2vec cause the dimension of its vector space higher than other technologies, which causes too much memory resources and low robustness. In contrast, GloVe combined the two schemes (count-based and direct prediction), which can keep its fast training, scalable to huge corpora, and good performance with the small corpus. Fasttext was developed by Facebook in 2016, although there is not much innovation in it, as a shallow neural network, it can get the same precision as the deep neural network and keep a much shorter training time than the deep neural network. For example, with a standard multi-core CPU, it can complete the training of 1 billion words within 10 minutes.

For the document embedding, there are many powerful deep learning strategies. For instance, doc2vec (Le and Mikolov, 2014), lda2vec (Moody, 2016), FastText (Bojanowski et al., 2016), Sent2Vec (Pagliardini et al., 2018), InferSent (Conneau et al., 2017), etc. However, they all cost too much computation resources.

Comparative studies

Since word2vec was published, researchers have proposed a lot of different embedding models. There are many studies compared these embedding methods on various tasks. For example, comparing the precision of results between LDA, word2vec, GloVe, Fasttext, DCH, LSRH, CSDH, SePH, SCM, etc, on the social media dataset like 'InstaCities1M' and 'WebVision' [15], exploring the influence of topic segmentation on information retrieval quality on the on dataset of Wikipedia [16], the comparative analysis of semantic similarity technologies for medical text [17], using word embedding clustering and convolutional neural network to improve the short text classification [18], improving information retrieval in software engineering by document similarities [19], improving textual-visual cross-modal retrieval with generative models [20], aggregating continuous word embeddings for information retrieval [21], etc.

3. RESEARCH OVERVIEW

The procedure of this research consists of four main stages.

Stage 1

The first stage is preparation for the development, such as determining the developing language, set-up the development environment, and designing the user interface of the application.

Stage 2

In the second stage, the first task is pre-processing the corpus. Next, the word embedding models are trained with the pre-processed corpus. Following this, the document embeddings are generated based on the outcome of the word embedding training. Finally, the functions of entity searching are implemented. All the details of this stage are described in Section 4.

Stage 3

In this stage, the first task is implementing the function of calculating cosine similarity [22]. Next, the ranking strategy is implemented. Thus, combining with the other four models, the searching results are marked visually. The details are described in Section 5.

Stage 4

The final stage is about the measurement and user evaluation, which are specified in Section 6.

4. SEMANTIC EMBEDDING

4.1 Preprocess

Based on 100,000 latest medical essays from the dataset of MEDLINE and Covid-19, the stop words in the corpus need to be removed. However, instead of using the common stop word list provided online (like Google English stop words list), a weights list was offered by the company OCLC. Through their project 'Fast and Discriminative Semantic Embedding'[23], this list describes the weights of appearance for each word in an article based on a much larger medical-related training dataset than the current one (the size of training dataset in this research is 100,000). With this weights list, all the words with a value lower than 0.1 are accounted as stop words.

4.2 Word and document embedding

In this research, word2vec, Fasttext and GloVe are word embedding frameworks. They cannot be used to do document searching directly. After the word embedding training complete, each document in the dataset must be converted into a document embedding vector. This process is accomplished by the following algorithm:

Algorithm 1 Generate document embedding

Input: a word embedding model M

Output: a vector list of the documents

```
result ← 0
for each documents d do
    for each entity e in the document d do
        v ← get the vector of e from model M
        v' ← v' + v
    v'' ← the mean value of v
    add v'' to result
return result
```

As the above algorithm, the mean value of the entity list of each document can be calculated by NumPy (a Python library), for instance, which is taken as the final embedding for a document.

coronavirus search

article terms subject author

id	Title [Word2vec]	Score	Title [Doc2vec]	Score	Title [Fasttext]	Score	Title [GloVe]	Score	Title [Ranking]	Score
1	coronavirus	1.0			coronavirus	1.0	coronavirus	1.0	coronavirus	27
2	ncov	0.84			deltacoronavirus	0.97	ncov	0.73	ncov	16
3	sars	0.82			betacoronavirus	0.95	cov	0.71	sars	13
4	cov	0.8			coronaviruses	0.93	sars	0.71	cov	13
5	mers	0.79			carlavirus	0.88	covid	0.69	covid	9
6	covid	0.78			coronaviral	0.88	wuhan	0.64	coronaviruses	9
7	coronaviruses	0.76			avulavirus	0.87	mers	0.61	mers	8
8	chikungunya	0.73			hantavirus	0.87	outbreak	0.59	deltacoronavirus	8
9	wuhan	0.73			ebolavirus	0.87	2019	0.55	betacoronavirus	7
10	outbreak	0.72			arenavirus	0.87	oc43	0.52	wuhan	5

Figure 1: The results of terms-based searching

4.3 Entity embedding

4.3.1 The pattern of entity

Before the preprocess, the corpus needs to be turned into an entity-based pattern. As for entity, it means the words of an essay, the author, the subject, the release date, etc. Here attached an example of the entity-based essay:

7088458715 [title:minimal detectable...][abstract:...] [lang:eng] [author:macdermid jc] [author:nazari g] [issn:...] [doi:10.1519] [type:journal article] [pmid:28731864] [subject:adult]

An entity is presented as the pattern: [‘entity name’: ‘entity content’], and an article consists of a document index and several entities.

4.3.2 The strategy of entity embedding

All the four models split the text by space, but an entity like ‘subject’ is normally a sentence consists of many words and spaces. In the entity-based embedding, we need to take the entire ‘subject’ as an entity. So, all spaces and colon in the text of an article concerning the language, author or subject are replaced with underline. For example, ‘subject: fatty acids omega 3’ is the original pattern of an entity, which should be formatted into ‘subject_fatty_acids_omega_3’. Thus, the latter one will be treated as an entity and vectorized for embedding.

5. EMBEDDING-BASED INFORMATION RETRIEVAL

After the previous embedding steps in Section 4, words, terms, documents, and entities are embedded in the same semantic space, where the cosine similarity can be applied in calculating the similarity/relatedness.

5.1 Cosine similarity

By calculating the cosine similarity [22], we gain a score for input keywords, which indicates how similar it is between two document vectors. Thus, a list of all documents sorted by the score is achieved, by which the most similar ones can be extracted from and returned to the end-users. The formula of the cosine similarity is as follows:

$$cos(\theta) = \frac{\sum_{k=1}^{n} x_{1k} x_{2k}}{\sqrt{\sum_{k=1}^{n} x_{1k}{}^2} \sqrt{\sum_{k=1}^{n} x_{2k}{}^2}} \quad (1)$$

vector 1: [x_{11}, x_{12}, x_{13} …… x_{1n}]

vector 2: [x_{21}, x_{22}, x_{23} …… x_{2n}]

5.2 Entity-based searching

Entity-based searching provides the users or researchers related to the study concerning the Covid-19 a convenient way of searching. For example, with terms-based searching, researchers can get all the terminologies about coronavirus with only one click. However, in the traditional search engine, the users need to collect many results and extract terminologies by themselves.

There are many entities in an article, such as the words in the title and the abstract, release date, ISSN, author, subject, etc. Terms-based searching returns the single term related to the keywords. Subject-based searching gives the subject of an article. Author-based searching displays all the authors related to an article. They are implemented based on word embedding. Figure 1 presents the terms-based searching results. The input keyword is ‘coronavirus’ and the results are all related terminologies. Note that the second column (which is for doc2vec) is invalid, since it is a document embedding model.

5.3 Ranking strategy

The ranking strategy is quite simple. Firstly, based on the results from other four embedding models, a score is assigned to each result. Since those searching results are already sorted, the top one is assigned a score 9, the second one is assigned a score 8. By that analogy, the 10th result is given a zero. Secondly, combining those results, removing the overlapped results, and recalculating the score according to the number of repeats of each result. For instance, the document 789587 appears at the first position of word2vec, second position of doc2vec, third position of Fasttext and ninth position of GloVe, it gets a score 25. (9+8+7+1 = 25)

This is the pseudo code about getting the ranking results:

Algorithm 2 Get ranking results

Input: a results list r1 from word2vec,
a results list r2 from doc2vec,
a results list r3 from Fasttext,
a results list r4 from GloVe

Output: a result list of Ranking

```
rm ← [r1, r2, r3, r4]
s_map ← {} # the key is a document id,
           # the value is an object of a result

for each list l in rm do
    score ← 10
    for each result r in l do
        r.score ← score # assign a score to a result
        score ← score – 1
rm' ← rm sorted by score
rm'' ← rm'

for each result r in rm'' do
    if r existed in score_map then
        # the overlapped results are removed and recalculate the score
        s_map[r.id].score ← s_map[r.id].score + r.score
    else
        score_map[r] ← 1
result ← keyset of s_map sorted by score
return result
```

5.4 Compare results visually

One goal of this research is to investigate a method of marking the similarity or differences of the searching results between the four models visually. The main method is transferring each result as an object, which contains properties like document index, the number of repeats and color. All the same results are marked with same background color on the web UI.

The searching results are visually marked on the web UI as Figure 2. From the first column to the fifth column are the searching results of word2vec, doc2vec, Fasttext, GloVe and the ranking, respectively. The same documents are marked with the same colors. For example, the first result in the first column, the third result in the third column and the first result in the fifth column are set to green color, which means they are the same documents. An item with white background color means there is not the same results exist in any of the five columns.

The pseudo code is as follows:

Algorithm 3 Set the color of each result

Input: a map **m** with document id as the key and the color as the value, a mixed results list **rm** from word2vec, doc2vec, Fasttext and GloVe

Output: a results list with color property for each element

```
color_list = [many colors in 6 bits hexadecimal]
for each object d in rm:
    if repeat number of d > 1 then
        if m does not contain the id of d then
            m[d.id] ← color_list.pop(0)
        else
            d.color ← m[d.id]
return rm
```

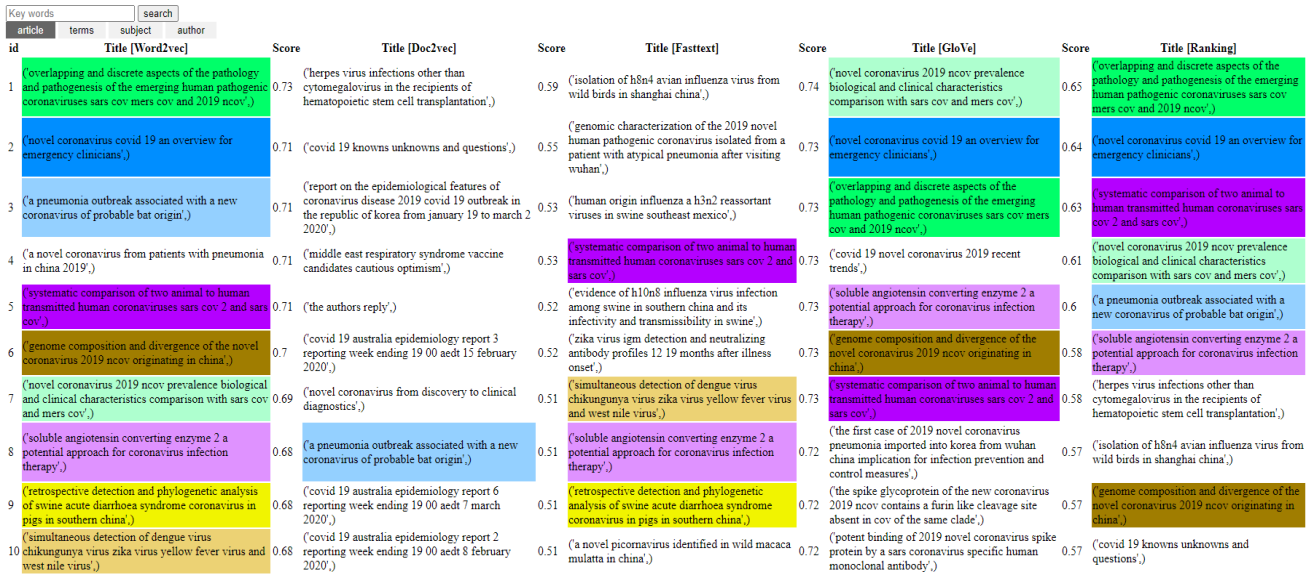

Figure 2: The visually marked results

6. EXPERIMENT SETTING

6.1 Influence factors

The final product of this research is an IR web application, which is implemented with Python. Training with word2vec, doc2vec, Fasttext and GloVe requires many parameter configurations such as vector size, window, min count, etc. One of the most important influential factors is 'min count'. All words with a total frequency lower than this were ignored by the semantic model. According to the dimension of the vector of embedding models, this value varies. With 256 dimensions in this research, this parameter was tested with a value from 2 to 5. Finally, it presents that, with 5 min count, the IR engine can reach its best performance. As mentioned above, the other significant factor is the vector size. With advice from the experts from the company OCLC, the dimension of the vector was set as 256. Besides, the defaults were used for other parameters.

6.2 Measurement time

Table 1 shows the time consumption for the measurement, including preprocess, word embedding training, document embedding training and running testing script. The first line is the time for preprocess of the corpus. The lines from 2 to 5 are the training time for word embedding models. From line 6 to line 8 are the time used for document embedding generation. We can see that there were some differences in the time of training at different models. The training of Fasttext model spent the longest time (28+23=51 minutes). The second time-consuming training is GloVe (33+15=48 minutes), which has only the Linux version. It is training on Google Colab, so the training time is limited. By contrast, word2vec and doc2vec cost less time for training, 29 and 26 minutes, respectively.

Table 1. The time cost for testing/measuring

	Process	Time(minutes)
1.	preprocess	2
2.	word2vec (word embedding)	12
3.	doc2vec (word embedding)	26
4.	Fasttext (word embedding)	28
5.	Glove (word embedding)	33
6.	word2vec (document embedding)	17
7.	Fasttext (document embedding)	23
8.	Glove (document embedding)	15
9.	Total	156

7. EXMPERIMENT RESULTS

In this research, three main properties are required to be measured: precision, recall, and relevance.

7.1 Precision and recall

7.1.1 Testing method

In this section, testing with the keywords from the current dataset for precision and recall. 100 documents are drawn from the dataset, and then extract many words from the title or abstract of these documents as the input keywords for searching. This task was executed for ten rounds, and different lengths of keywords were used for each round. For the first three rounds, the length of the input is around 5 words, then about 50 words for the next three rounds. The last four rounds using the entire title of an essay as input keywords.

7.1.2 Results

Figure 3 and Figure 4 present the percentage of precision and recall out of 1,000 searching for each of the four models, respectively. For example, the first green bar in Figure 3 shows that 791 out 1000 times of searching the first result in the word2vec results set is the target. The first green bar in Figure 4 means that 879 out of 1,000 times of searching found the target in the word2vec results set. Finally, it can be seen from Figure 3 and Figure 4 that both the precision and recall of the ranking results are higher than that of the other semantic embedding technologies. Besides, the error in Figure 3 and Figure 4 show that the ranking is the second stable one for both precision and recall.

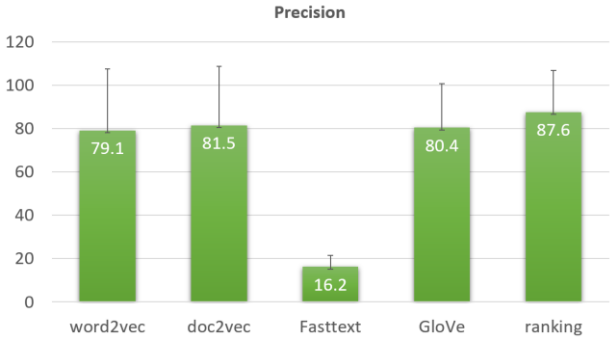

Figure 3: The precision of word2vec, doc2vec, Fasttext, GloVe and ranking

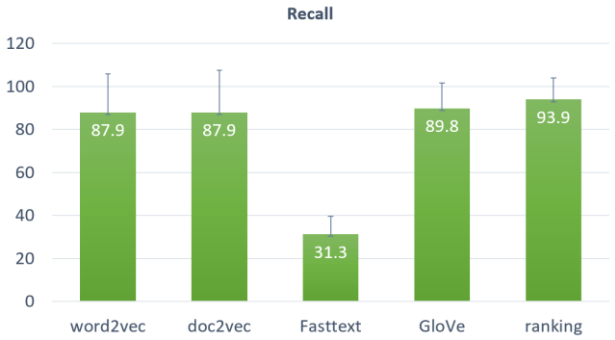

Figure 4: The recall of word2vec, doc2vec, Fasttext, GloVe and ranking

The followings are the definition of precision and recall:

Precision@1: the first result is the target one.

Recall@10: the target result exists in the result set.

7.2 Relevance:

7.2.1 User evaluation

In this user evaluation, three participants are required to do the searching task and fill in the evaluating form (See Table 2 in Appendix). Their feedback is about the relevance between the input keywords and the ten results returned by each semantic embedding model and the ranking algorithm (four models plus a ranking set, 50 results in total). Each of them needs to search the articles in terms of Covid-19 with ten provided keywords. Keywords 1, 2, and 3 contains only one term, respectively. The keywords 4, 5, and 6 are medium size with around 5 words. The keywords 7, 8, 9, and 10 are long size with the abstract of an essay. These keywords are outside the current dataset, collected from PubMed.

Evaluation steps

In the first step, a participant put in the first keyword and clicked the searching button on the web UI. In the second step, the participant read each of the (ten) results displayed on the screen and recorded the number of related results based on two different standards, which is specified in Section 6.2.2 and Section 6.2.3. Then, repeated the above steps for all keywords in Table 2 (See appendix).

Data process

Instead of all detailed data, Figure 5 and Figure 6 describe the average values based on the feedback forms. For example, calculating the average value of relevance for word2vec under the strict standard need to sum all the values of the third columns in Table 2 and divide by 30 (the total number).

7.2.2 Strict standard

Since the relevance is subjective to the participants, two standards are adopted in this user evaluation. Figure 5 shows the data of evaluation in terms of relevance on a 'strict standard'. For example, if the input keyword is 'coronavirus', only the result about 'coronavirus' can be taken as a related article. Even the articles about other highly pathogenic infectious diseases like Sars and Mers are not included. Figure 5 shows that the ranking performs better than word2vec, doc2vec and Fasttext and lower than that of GloVe.

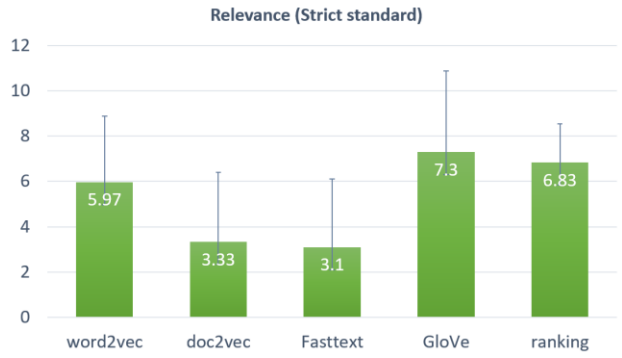

Figure 5: The relevance of word2vec, doc2vec, Fasttext, GloVe and ranking (strict standard)

7.2.3 Loose standard

Figure 6 presents the statistics on a 'loose standard'. By this standard, articles about other epidemic diseases such as influenza virus, apore virus, zika virus, etc. are also counted as correct results.

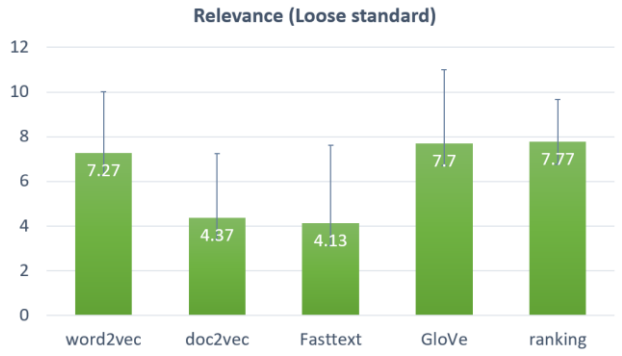

Figure 6: The relevance of word2vec, doc2vec, Fasttext, GloVe and ranking (loose standard)

In conclusion, in the strict standard, compared with the relevance of the ranking results (6.83), that of GloVe is slightly higher standing at 7.3. In the loose standard, the relevance of the ranking results ranks the top arriving at 7.77, which is followed by GloVe and word2vec with 7.70 and 7.27, respectively. Doc2vec and Fattext in stark contrast, occupies the minimal share with merely 4.37 and 4.13, considerably lower than the ranking. Finally, the errors in Figure 5 and Figure 6 show that the ranking results are more stable than other models.

8. CONCLUSION AND FUTURE WORKS

There are three aims in the present research. The first one is to mark the searching results between different semantic models on the web UI. The second aim is to measure the performance of the four models on the dataset related to Covid-19. Finally, investigate a method of ranking which performs better than other embeddings.

The experiment shows that the relevance of word2vec and GloVe are superior to that of doc2vec and Fasttext. However, the results set of doc2vec contains a majority of articles about other viruses, which can only be detected manually instead of the testing scripts. The advantage of GloVe is obvious, it keeps the highest precision, recall, and relevance throughout the entire experiment. The only disadvantage of it is not stable enough. Sometimes it got very low precision and recall nearly to only 10%. In contrast, Fasttext got the lowest performance in this research. The precision, recall, and relevance of Fasttext are always much lower than other models. On the top of that, it costs longer training time than other models.

For the precision and the recall, the ranking results set is higher than all other embedding models. According to the measurement, even searching with the same keywords, the results vary tremendously between different models. In the user evaluation, it presents that the ranking results are better than word2vec, doc2vec and Fasttext, and keeps stable all the time.

For the strategy of mark the searching results visually, this search engine can mark the same document from different models with the same background colors. Thus, users can compare the searching results more intuitively.

In the future, there are more works can be done. Firstly, the ranking algorithm can be improved by taking into consideration the weights of different embedding models. For example, models with better performance get higher weights. They can put more results in the ranking set, and vice versa. Secondly, now, the way of marking the searching results is based on the document index. However, the similarity score is also an important factor. With the similarity scores, the search engine can mark the results with similar content, not just limited to the identical ones. Thirdly, a user evaluation about the strategy of 'mark searching results visually' could be done, to test how effective it can improve the researchers' work. Finally, the weights of the word embedding can be used to train models with better performance if there is more time for the experiment. Besides, the experimental results of Fasttext are much lower than other models, which might be caused by the size of the training dataset or the configuration of the model. With more time, more training and adjustments could be conducted to improve its performance.

9. REFERENCES

[1] Manning, C.; Raghavan, P.; Schutze, H.: Introduction to information retrieval. Natural Language Engineering, 16 (1) (2010), 100–103.

[2] Diego Arroyuelo, Senén González, Mauricio Marin, Mauricio Oyarzún, Luis Valenzuela. To index or not to index: Time–space trade-offs for positional ranking functions in search engines. Information SystemsVolume 89 (March 2020) Article 101466.

[3] Schnabel, T.; Labutov, I.; Mimno, D.; Joachims, T.: Evaluation methods for unsupervised word embeddings. in Proc. of the 2015 Conf. on Empirical Methods in Natural Language Processing (2015), 298–307.

[4] Saiyed Umer, Bibhas Chandra Dhara, Bhabatosh Chanda 2017. A novel cancelable iris recognition system based on feature learning techniques (September 2017), 102-118.

[5] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning sentiment-specific word embedding for twitter sentiment classification," in ACL (1), (2014), 1555–1565.

[6] Anita Balandina, Anastasiya Kostkina, Artem Chernyshov, Valentin V. Klimov. Dependency Parsing of Natural Russian Language with Usage of Semantic Mapping Approach. Procedia Computer ScienceVolume 145 (2018), 77-83.

[7] Ravi, K.; Ravi, V.: A survey on opinion mining and sentiment analysis: tasks, approaches and applications. Knowl. Based. Syst., 89 (2015), 14–46.

[8] Bengio, Y.; Ducharme, R.; Vincent, P.; Janvin, C.: A neural probabilistic language model. J. Mach. Learn. Res., 3 (2003), 1137–1155.

[9] Muskan Garg, Mukesh Kumar. TWCM: Twitter Word Co-occurrence Model for Event Detection. Procedia Computer ScienceVolume 143 (2018), 434-441.

[10] Mikolov, T.; Chen, K.; Corrado, G.; Dean, J.: Efficient estimation of word representations in vector space. CoRR, vol. abs/1301.3781, (2013).

[11] Donghwa Kim, Deokseong Seo, Suhyoun Cho, Pilsung Kang.Multi-co-training for document classification using various document representations: TF–IDF, LDA, and Doc2Vec. Information Sciences, Volume 477, (March 2019), 15-29.

[12] Bojanowski, P.; Grave, E.; Joulin, A.; Mikolov, T.: Enriching word vectors with subword information. Trans. Assoc. Comput. Linguistics, 5 (1) (2017), 135–146.

[13] Joulin, A.; Grave, E.; Bojanowski, P.; Mikolov, T.: Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759, (2016).

[14] Pennington, J.; Socher, R.; Manning, C.: Glove: Global vectors for word representation. in Proc. of the 2014 Conf. on Empirical Methods in Natural Language Processing (EMNLP), (2014), 1532–1543.

[15] Raul Gomez, Lluis Gomez, Jaume Gibert, Dimosthenis Karatzas. Self-Supervised Learning from Web Data for Multimodal Retrieval. Multimodal Scene Understanding (2019), 279-306.

[16] Gennady Shtekh, Polina Kazakova, Nikita Nikitinsky, Nikolay Skachkov. Exploring Influence of Topic Segmentation on Information Retrieval Quality. Internet Science: 5th International Conference (October, 2018), 131-140.

[17] Fakhare Alam, Muhammad Afzal, Khalid Mahmood Malik. Comparative Analysis of Semantic Similarity Techniques for Medical Text. ResearchGate. (March 2020)

[18] Peng Wanga, Bo Xu, Jiaming Xu, Guanhua Tian, Cheng Lin Liu, Hongwei Hao. Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification. Neurocomputing, Volume 174 (January 2016), Pages 806-814.

[19] Xin Ye, Xiao Ma, Xiao Ma, Razvan Bunescu, Chang Liu. From word embeddings to document similarities for improved information retrieval in software engineering. ICSE 16: Proceedings of the 38th International Conference on Software Engineering (May 2016), Pages 404 – 415.

[20] Jiuxiang Gu, Jianfei Cai, Shafiq R. Joty, Li Niu, Gang Wang. Look, Imagine and Match: Improving Textual-Visual Cross-Modal Retrieval With Generative Models. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018) pp. 7181-7189.

[21] Stephane Clinchant, Florent Perronnin. Aggregating Continuous Word Embeddings for Information Retrieval. Conference: Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality (August 2013).

[22] Nora Grieb, Theo Oltrup, Thomas Bende, Martin A. Leitritz. The Cosine Similarity Technique: A New Method for smart EXCIMER laser control. Zeitschrift für Medizinische PhysikIn press, corrected proof, Available online 2 (April 2020).

[23] Rob Koopman, Shenghui Wang, Gwenn Englebienne. Fast and Discriminative Semantic Embedding. Association for Computational Linguistics (May 2019).

10. APPDENDIX

Table 2: The table for user evaluation

#	Keywords	Relevance									
		Strict standard				Loose standard					
		word2vec	doc2vec	Fasttext	GloVe	word2vec	doc2vec	Fasttext	GloVe		
1	coronovirus										
2	sars										
3	mers										
4	Coronavirus envelope protein: current knowledge										
5	Broad-spectrum coronavirus antiviral drug discovery										
6	Host Factors in Coronavirus Replication										
7	https://pubmed.ncbi.nlm.nih.gov/22816037/										
8	https://pubmed.ncbi.nlm.nih.gov/24930446/										
9	https://pubmed.ncbi.nlm.nih.gov/30258004/										
10	https://pubmed.ncbi.nlm.nih.gov/25432065/										

Note: the keywords from 7 to 10 are the abstract of the articles. The addresses of the articles are provided. From the third column to the last column, the participants are asked to fill in the number of searching results related to the keywords for the four models on the strict standard and loose standard, respectively.