Journal of Theoretical and Applied Information Technology
30th April 2020. Vol.98. No 08
© 2005 – ongoing JATIT & LLS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

TOWARDS A WORD EMBEDDING BASED APPROACH FOR SEMANTIC INDEXING

[1]ILYAS GHANIMI, [1]ELHABIB BENLAHMAR, [1]ABDERRAHIM TRAGHA, [1]FADOUA GHANIMI
[1]University Hassan II, Department of Mathematics and Computer Sciences, Morocco
E-mail: [1] ghanimiilyas@gmail.com,

ABSTRACT

In this article we present a new approach to semantic indexing of documents using word embedding relaying on representing words as numerical vectors based on the contexts in which they appear. This approach is validated by a set of experiments and a comparison with other approaches. We show that the proposed approach achieves results equivalent or better.

Keywords: *Information retrieval, semantic indexing, word embedding, BERT, NLP.*

1. INTRODUCTION

Currently, due to the steady increase in computational power, the growing availability of open source data and the continuous improvement of machine learning algorithms, the NLP is rapidly developing.

The ultimate objective of NLP is to read, decipher, understand, and make sense of the human languages in a manner that is valuable.

For a long time, the majority of methods used to study NLP problems employed shallow machine learning models and time-consuming, hand-crafted features to derive meaning from human languages.

However, with the recent popularity and success of word embeddings (low dimensional, distributed representations), deep learning based models have achieved superior results on various language-related tasks.

Our work focuses on semantic indexing. In particular, we are working on the phase of extracting information from texts, which is a preliminary step in the process of indexing documents. Semantics refers to the meaning that is conveyed by a text. Semantic analysis is one of the difficult aspects of Natural Language Processing that has not been fully resolved yet. It involves applying computer algorithms to understand the meaning and interpretation of words and how sentences are structured.

We therefore use the statistical method TF-ID to detect the most important terms in the corpus.

These terms are then transformed into input for the deep learning based algorithm to get the indexation terms.

Section 2 presents a state of the art on semantic indexing methods. The steps of our approach as well as the theoretical bases are presented in section 3. In section 4 we present an experimental validation of the proposed method and we end with a conclusion.

.

2. DEEP LEARNING

Deep Learning means the techniques based on neural networks. These networks are inspired by biological neurons: each neuron receives an input signal from several other neurons (or directly signals from the outside world) and performs a very simple operation on these signals; the result of this operation is transmitted to several other neurons.

Traditional neural networks (feedforward neural networks) can achieve near state-of-the-art results on a range of unstructured and structured language processing tasks. They are suitable for the treatment of characteristic vectors of fixed sizes and can therefore be used with TF IDF. Nevertheless it is difficult to adapt for processing unbounded size sequences (for example, a sequence of word vectors obtained with word embedding techniques).

There are two main architectures for processing sequences: convolutional neural networks and recurrent neural networks.

2.1 Convolutional Neural Networks

Convolutional neural networks (CNN) utilize layers with convolving filters that are applied to local features [1].

Originally invented for computer vision to allow successive preprocessing of small parts of an image using the same set of parameters, CNN models have subsequently been shown to be effective for NLP and have achieved excellent results in semantic parsing [2], sentence modeling [3], search query retrieval [4] and various NLP tasks [5].

Instead of processing successive parts of an image, CNNs for NLP are used to process subsequences of words.

training procedure. The fact remains that CNNs perform very well on many tasks and compete or even outperform recurrent neural networks in some situations[13].

2.2 Recurrent neural networks

Recurrent Neural Networks (RNNs) are main stream in numerous Natural Language Processing (NLP) assignments. The thought behind RNNs is to process information by sequences. By construction, RNNs meet two essential criteria for NLP:

- They model the dependencies between words.
- They can be used very for sequences of different sizes.

While the ordinary NN does not respect the temporal order of the input data, the recurrent neural network avoids this problem by having the notion of time built into it.

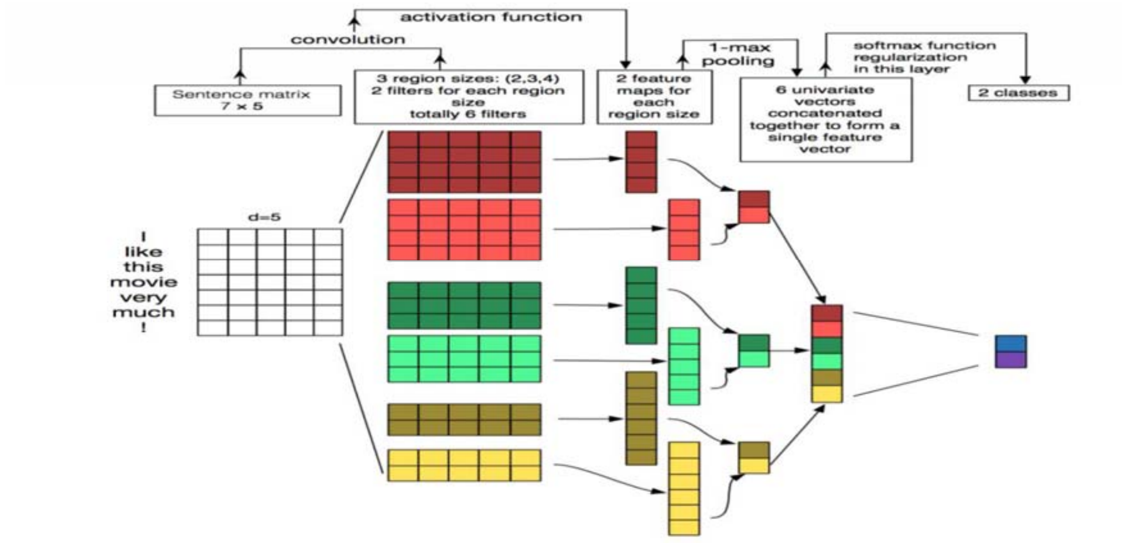

activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
I
like
this
movie
very
much
!

Figure 1: CNN modeling on text [12]

Building a CNN architecture means that there are many hyperparameters to choose from, some of which I presented above: Input represenations (word2vec, GloVe, one-hot), number and sizes of convolution filters, pooling strategies (max, average), and activation functions (ReLU, tanh).

Most CNN architectures learn embeddings (low-dimensional representations) for words and sentences in one way or another as part of their

They perform out a similar assignment for each component in an arrangement, with the yield being reliant on the past calculations. RNNs have a memory which catches data about what has been figured up until this point. To prepare subjective arrangements of data sources RNNs utilizes their inside memory. At each layer, new data is included and that data is passed on for an uncertain number of systems. RNNs get information and create yield at each progression.

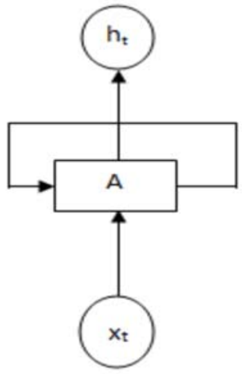


Figure 2: Recurrent Neural Network

- An output layer used to establish network predictions for the current element.

Fi Figure 3 shows what a typical RNN looks like. The diagram shows a RNN being unrolled into a full network. For example, if the input sequence is a four-word sentence, the network would be unrolled into a 4-layer neural network, one layer for each word.

LSTM/GRU:

Al Though RNNs have the right framework to model temporal data, they suffer from some inadequacies. NNs learn by back propagating the error to modify the network weights. If the network is too large or the number of time steps is too long, the distance over which the error must be transmitted becomes infeasible and the network stops learning. This problem is referred to in literature as the vanishing gradient problem.

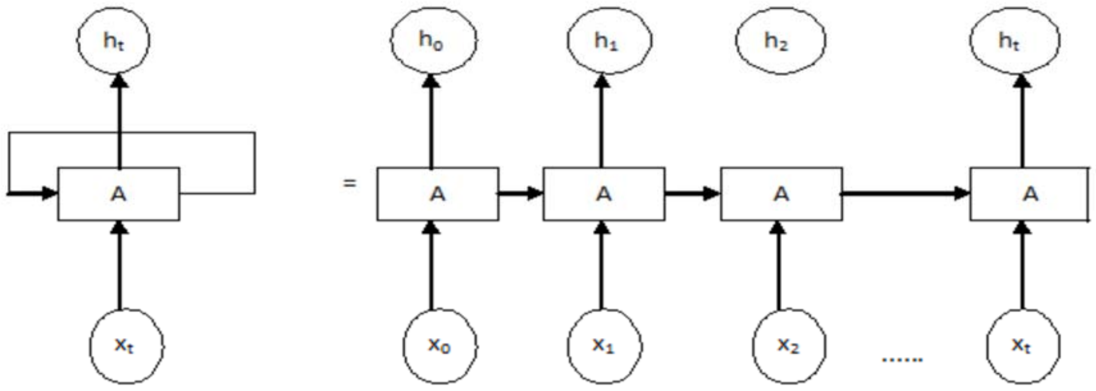


Figure 3: An unrolled Recurrent Neural Network

Classic RNN'S

Recurrent neural networks are generally composed three layers:

- An input layer corresponding to the characteristics of the current element of the sequence,
- A hidden layer having a recurrence relation with itself.

L The LSTM is a modified RNN designed to overcome vanishing gradients [6]. In addition to the hidden state, the LSTM carries a cell state that preserves long term information. The LSTM solves the problem by having a shortcut path to transmit the gradients back. This unique architecture involves three gates. The forget gate decides the contents of the cell state that are relevant to the problem and discards everything else. The input gate's function is to ensure that relevant information in the input is stored in the cell state. Finally, the output gate adds the relevant input to the cell state and passes it on to the next time step. This architecture ensures that the gradients are propagated back into the earlier time steps in the network.

Here, the cells are layered one over the other like the basic architecture. However, in the central layer, the cells are forced to withhold their connection with the layer above them until the last timestep. This ensures that the output of the layer is a summary of the contents of all the input timesteps. This summary vector is then fed as an input to all the timesteps in the layer above.

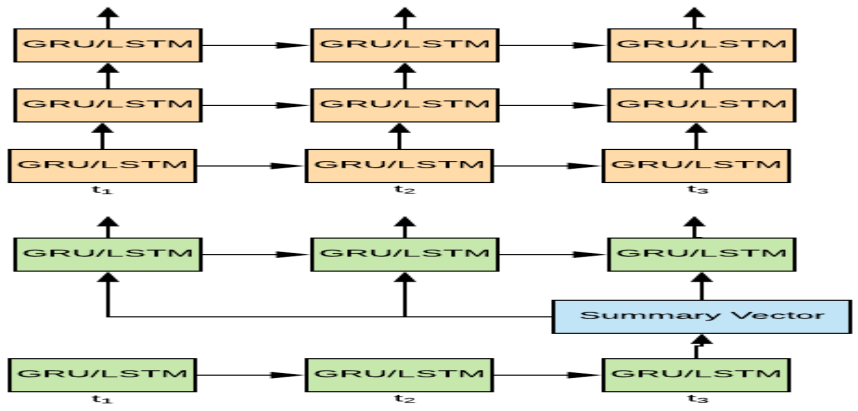


Figure 4: Regular stacked RNN architecture with summary vector.

The GRU is another modified RNN cell that is similar to the LSTM in many ways [7]. It differs from the LSTM in that it has two gates in place of three. The update gate and the reset gate preserve long term information in the data.

While the reset gate determines the information that is important to hold over the long term, the update gate ensures that this information is added to the hidden state.

This architecture ensures that the GRU has fewer parameters to train than an LSTM. It saves memory and takes less time to train. However, on small sequences this typically does not make much of a difference.

Encoder-Decoder (ED) Model

Encoder-decoder neural networks were introduced by Sutskever et al in the context of machine translation [8]. This model is popular in the field of natural language processing(Figure 5).

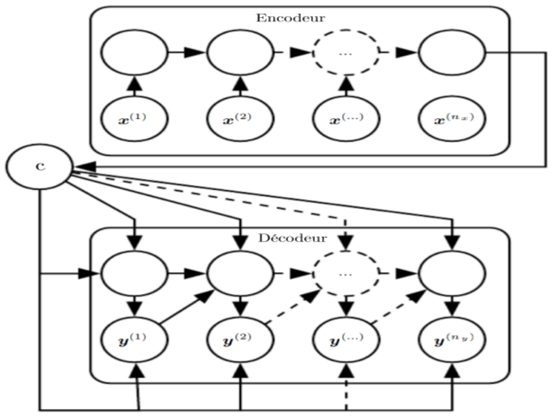


Figure 5: Encoder-Decoder Model

3. Our new semantic indexing approach

As shown in Figure 6 , the approach we propose is composed of two stages: extraction of indexing terms using TF-IDF and extending them semantically using BERT.

Our method for detecting key terms is a preliminary step to a semantic indexing process. Indeed, to build the final representation of potential result, which are generated from the BERT mode.

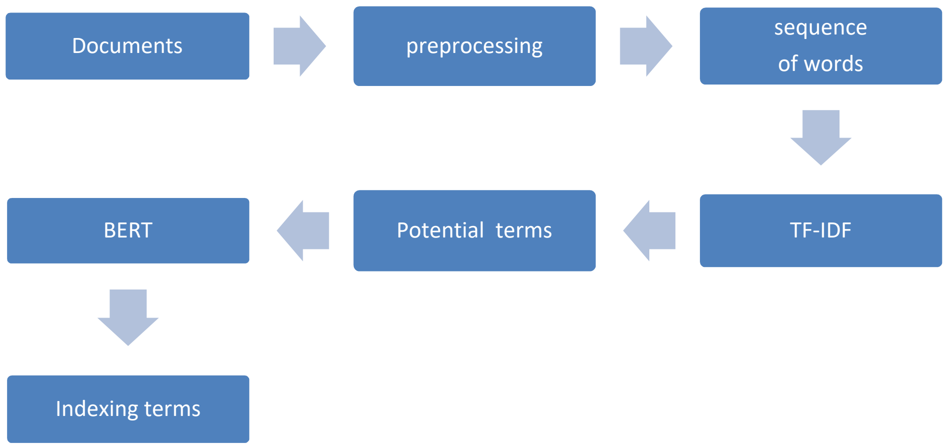


Figure 6: An overview of our proposed method

documents or queries, a semantic indexing method must be completed.

The procedure we use in the first step is Term Frequency Inverse Document Frequency (TF-IDF). This weighing scheme can be categorized as a statistical procedure, though its immediate results are deterministic in nature.

Though TF-IDF is a relatively old weighing scheme, it is simple and effective, making it a popular starting point for other, more recent algorithms [11].

The purpose of our approach is to make use of the strengths of TF-IDF as a starting point for our algorithm. The use of the BERT natural language model is to complement the weakness of the TF-IDF framework in understanding the true semantic meaning of a document.

We consider our document as a sequence of words. The TF-IDF model, gives higher weight to the most frequent terms in the corpus. We use TF-IDF scores to rank our first potential results; then we add to the potential results, the terms the most similar to them.

The similarity is calculated using vectors of the all terms of the corpus and the vectors for each

Because we already are able to get the weight TF–IDF score as the previous section described, the specific approach is to add this two scores together and make a final sort. It is reasonable because the information those two scores represented is orthogonal.

The TF–IDF score represents the scenario that the user input exactly match the documents; the semantic score represents the scenario that user want to search some relevant documents.

Even BERT is the current best model for language representation; it still is a black box model and very sensitive to noise. Moreover, a more common scenario is that user expected search result exactly matches user’s input, and semantic information only is a supplement for traditional keywords searching approach.

On the other hand, TF–IDF model does not take semantic information into consideration at all. It is a compromise approach to combining the two score together.

3.1 TF-IDF

The most basic unsupervised method for keywords extraction is TF-IDF (Term Frequency – Inverse Document Frequency) [11].

Tf-Idf, is a metric for calculating the relevance of terms in documents, very used in Information Retrieval and Text-Mining. Essentially, this technique measures how important a certain word is on a document regarding other documents in the same collection.

Basically, a word gets more important in a certain document the more it occurs in that document. But if that word occurs in other documents, its importance decreases. Words that are very frequent on a single document tend to be more valued than common words that occur on more documents, like articles or prepositions.

Then normalized to prevent word on very long documents to get higher Tf values.

Equation 1 measures the probability that a term i occurs in a document j.

$$TF_{ij} = \frac{n_{ij}}{\sum_k n_{kj}} \qquad (1)$$

where n_{ij} is the number of times the term i occurs in a document j and then it is divided by the total of words in document j. Idf component measures the general relevance of a given term.

Equation 2 consists in the count of the number of documents that a term t_i occurs.

$$Idf_i = \log\frac{|D|}{|\{d_j/t_i \in d_j\}|} \qquad (2)$$

where |D| represents the total number of documents in the collection and { : } j i d j d t ∈ the number of documents where the term ti occurs. Tf-idf (equation 3) is then the multiplication of the two previous equations.

$$TFIdf_{ij} = TF_{ij} \times Idf_{ij} \qquad (3)$$

However, we must consider that TF-IDF select candidate keyphrases based on their statistical frequencies without considering semantic similarity between words. The inconvenient of this method is that it ignores the words which have low frequency.

Generally, the calculation of Tf-Idf is made in separate, calculating the Tf and Idf components separately, and finally multiplying both components to get the final Tf-Idf value. Tf component (term frequency) simply measures the number of times a word occurs on a certain document.

As noted before, a set of terms is typically extracted as potential candidate terms using heuristic rules. These rules are used to keep the number of candidates to a minimum.

In the approach we propose, the following algorithm is used for selecting candidate terms:

- First, a predefined list of stopwords is used to remove stop words [10].

.

- Select as candidate keywords the N potential candidate keywords the most frequent in the document using TF-IDF scores, while N may be set to any fixed value, usually ranging from 5 to 20 keywords.

3.2 Word embedding with BERT

The BERT (Bidirectional Encoder Representations from Transformers) is a natural language representation model [9] that makes use of Transformer, an attention mechanism which reads the entire sequence of words at once to encode input token, making it the best language representations model nowadays.

BERT also uses multilayer network to capture the text meaning. BERT is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. This characteristic allows the model to learn the contextual relations between words in a text based on all of its surroundings.

It has been proved that the BERT model is more capable of understanding the true semantic meaning of the text by presenting which is successfully applied in many NLP tasks,

Pre-trained representations reduce the need for many heavily-engineered task specific architectures. BERT is the first fine tuning based representation model that achieves state-of-the-art performance on a large suite of sentence-level and token-level tasks, outperforming many task-specific architectures.

Inspired by [9], we propose a new method that aims to select from the candidate terms the more representative for the document. As mentioned above, the input used by the BERT model is constituted of a list of candidate terms obtained according to the steps described before.

BERT is a state-of-the-art pre-training language re presentations with which we can generate the word embedding: We use the last layer of BERT hidden layer and sort up each token's output vector used to represent word.

We utilize the advantages of the pre-trained language model BERT to generate better word embedding of the list of candidate terms obtained by TF-IDF. With the word embedding created by BERT, we are able to calculate the similarity of different word and obtain a semantic score, which could be used to introduce semantic information in our semantic retrieval task.

Thus, we use the Cosine similarity to find out the words the most similar to the potential terms and add them to the list of keywords.

The intuition for this new approach is as follows: Apply BERT word embedding to keywords extraction task, and improve the performance of the task by taking advantage of the rich semantic features of BERT word embedding.

4. Evaluation and results

In this section, we start with the presentation of the datasets, then we present the results of our experiments.

.

4.1 Dataset

For our experiments, we tested the proposed approach on a dataset composed of documents 60 documents extracted from:

- Wikipedia.com
- Medium.com
- Google Scholar

4.2 Results

We evaluate our new method by studying the results of our proposed IRS. The typical approach, which is also adopted by the SemEval-2010 shared task on terms extraction[14], is to first create a mapping between the extracted documents in the gold standard and those relevant in the system output map, and then score the output using evaluation metrics the commonly used by researchers. That is precision/recall/F-measure.

$$Precision = \frac{N_{RE}}{N_E} \quad (4)$$

$$Recall = \frac{N_{RE}}{N_R} \quad (5)$$

$$F - Mesure = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (6)$$

Where N_{RE} is the number of relevant extracted document , N_E is the number of extracted documents and N_R number of relevant documents.

We calculated the Mean average precision (MAP) and the precision with 5 documents (P @ 5) with of different requests applied on the same Dataset.

The choice of these two measures is justified by the fact that the Mean Average precision gives a general overview of the effectiveness of our approach.

The 5-document precision gives a judgment of the performance of this approach on the documents most consulted by a user of an IRS (the first 5 documents in the list returned by the IRS).

Figure 8 shows the 10-point Mean average precision curves with our proposed approach method.

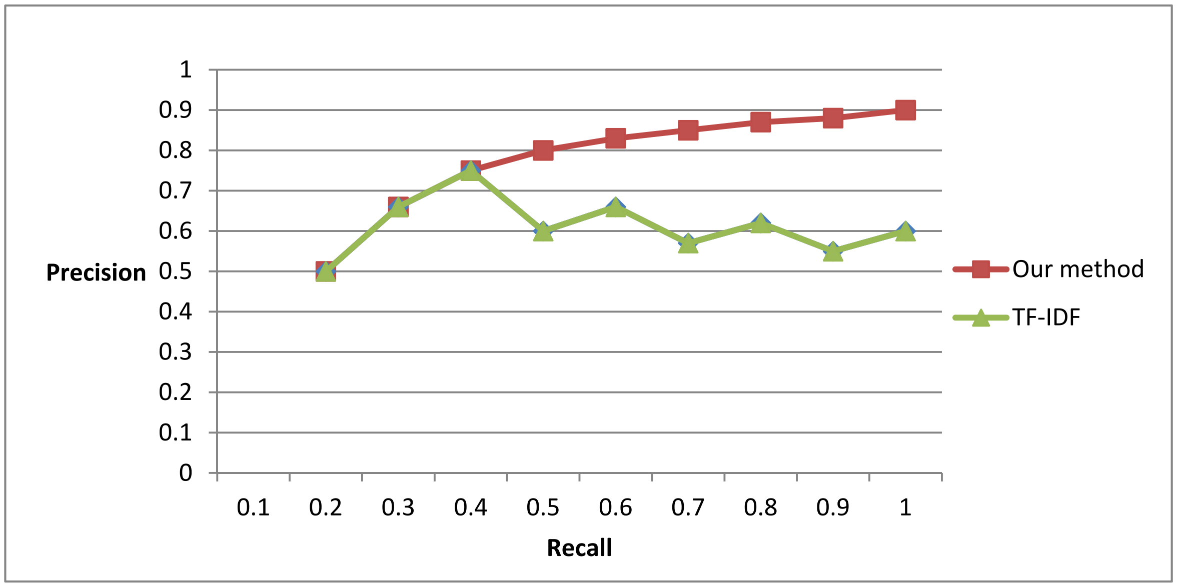


Figure 7. Results of the proposed method

We provide the experiment results of the two metrics mentioned above in Figure 8 and Table 1. The numbers are the mean value computed based on the outputs of the queries using the corresponding experiment setup.

The approach	MAP	P@5
Our approach	0.798	0.83
TF-IDF	0.612	0.66

Table 1: Results for the new approach to the Dataset of evaluation

4.3 Evaluation

As reflected in the Figure 7 and Table 1, our proposed method outperforms the traditional TF–IDF method in the two metrics.

Figure 8 is an example of the outputs given with the same input query. The first 4 documents are generated by our approach.

Figure 7 presents the curves of Mean average precision at 10 points of recall with our proposed method.

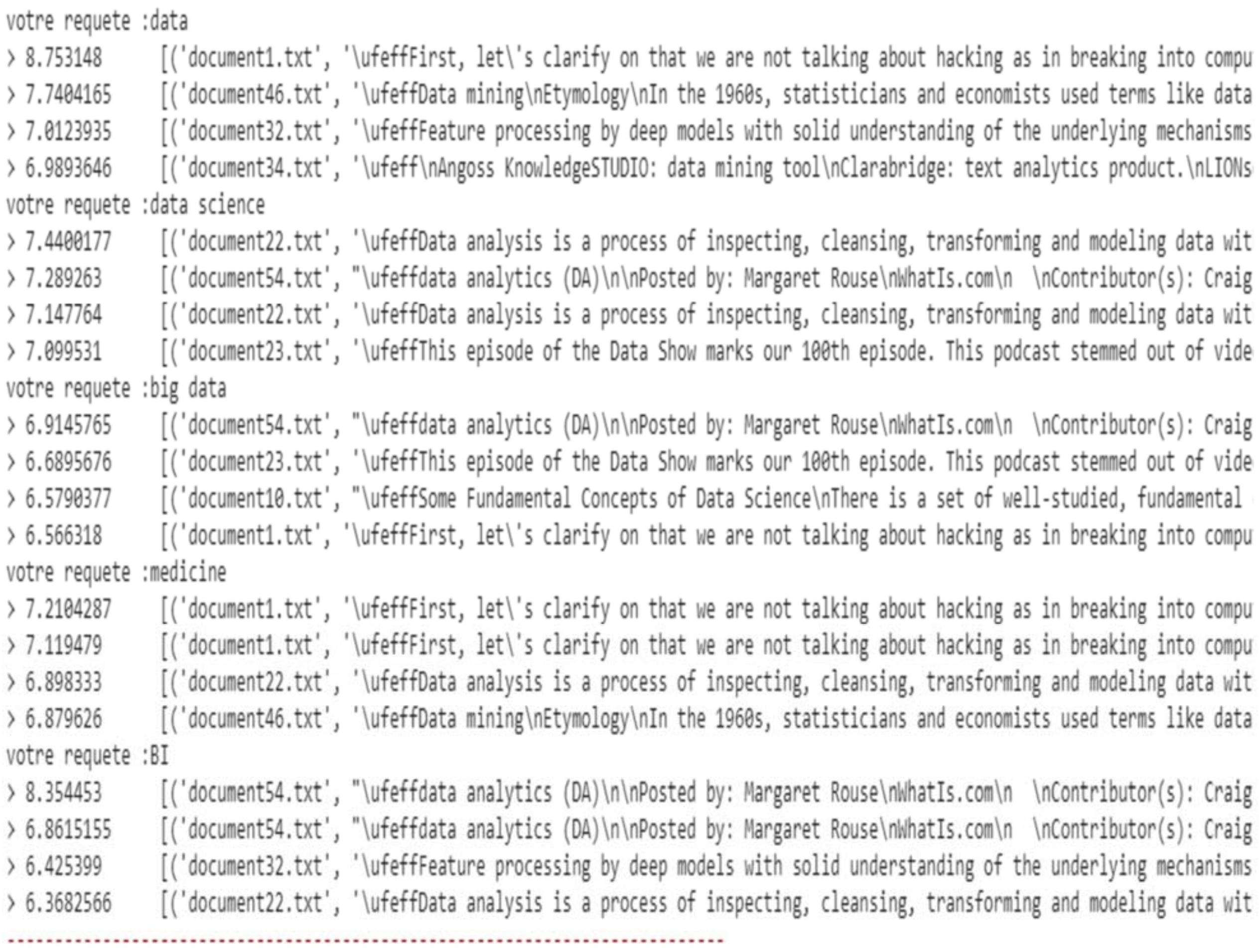
This shows that by using the TF–IDF score, we are more capable of finding the most relevant documents to user queries.

Moreover, the BERT natural language model aids the TF–IDF method in terms of providing semantic relevance measures so as to better compare the true semantics between the user query and potential answers.

As we can see in Table 1, the model of the TF–IDF method together with the BERT model outperforms the traditional TF–IDF method by a significant margin in the two metrics.

This well illustrates that our proposed model indeed improves the semantic search precision as compared to the traditional TF–IDF method.

However, the precision from both our proposed approach and the traditional TF–IDF method is still quite low. We believe that it is caused by the diverse nature and relatively small size of the experiment dataset. There may not be enough semantically similar documents in the dataset.



```
votre requete :data
> 8.753148     [('document1.txt', '\ufeffFirst, let\'s clarify on that we are not talking about hacking as in breaking into compu
> 7.7404165    [('document46.txt', '\ufeffData mining\nEtymology\nIn the 1960s, statisticians and economists used terms like data
> 7.0123935    [('document32.txt', '\ufeffFeature processing by deep models with solid understanding of the underlying mechanisms
> 6.9893646    [('document34.txt', '\ufeff\nAngoss KnowledgeSTUDIO: data mining tool\nClarabridge: text analytics product.\nLIONs
votre requete :data science
> 7.4400177    [('document22.txt', '\ufeffData analysis is a process of inspecting, cleansing, transforming and modeling data wit
> 7.289263     [('document54.txt', "\ufeffdata analytics (DA)\n\nPosted by: Margaret Rouse\nWhatIs.com\n  \nContributor(s): Craig
> 7.147764     [('document22.txt', '\ufeffData analysis is a process of inspecting, cleansing, transforming and modeling data wit
> 7.099531     [('document23.txt', '\ufeffThis episode of the Data Show marks our 100th episode. This podcast stemmed out of vide
votre requete :big data
> 6.9145765    [('document54.txt', "\ufeffdata analytics (DA)\n\nPosted by: Margaret Rouse\nWhatIs.com\n  \nContributor(s): Craig
> 6.6895676    [('document23.txt', '\ufeffThis episode of the Data Show marks our 100th episode. This podcast stemmed out of vide
> 6.5790377    [('document10.txt', "\ufeffSome Fundamental Concepts of Data Science\nThere is a set of well-studied, fundamental
> 6.566318     [('document1.txt', '\ufeffFirst, let\'s clarify on that we are not talking about hacking as in breaking into compu
votre requete :medicine
> 7.2104287    [('document1.txt', '\ufeffFirst, let\'s clarify on that we are not talking about hacking as in breaking into compu
> 7.119479     [('document1.txt', '\ufeffFirst, let\'s clarify on that we are not talking about hacking as in breaking into compu
> 6.898333     [('document22.txt', '\ufeffData analysis is a process of inspecting, cleansing, transforming and modeling data wit
> 6.879626     [('document46.txt', '\ufeffData mining\nEtymology\nIn the 1960s, statisticians and economists used terms like data
votre requete :BI
> 8.354453     [('document54.txt', "\ufeffdata analytics (DA)\n\nPosted by: Margaret Rouse\nWhatIs.com\n  \nContributor(s): Craig
> 6.8615155    [('document54.txt', "\ufeffdata analytics (DA)\n\nPosted by: Margaret Rouse\nWhatIs.com\n  \nContributor(s): Craig
> 6.425399     [('document32.txt', '\ufeffFeature processing by deep models with solid understanding of the underlying mechanisms
> 6.3682566    [('document22.txt', '\ufeffData analysis is a process of inspecting, cleansing, transforming and modeling data wit
-------------------------------------------------------------------------------
```

Figure 8: An example of outputs

5. CONCLUSION

At present, BERT has achieved the most advanced performance in many NLP tasks, but few works combine it with TF-IDF method for semantic indexation.

We proposed a novel approach to combine these two methods into a unified model that has been tested in relevant documents extraction tasks.

It calculates the TF–IDF score and then incorporate the BERT natural language model to complement the TF–IDF framework by providing true semantic relevance measures of potential indexing terms.

Experiments show that the model proposed in this paper achieves good results in the used dataset during a semantic search. Based on our experiment results, our proposed approach outperforms the traditional TF–IDF method by a significant margin in the metrics used for evaluation.

REFERENCES

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278-2324, November.

[2] W. Yih, X. He, C. Meek. 2014. Semantic Parsing for Single-Relation Question Answering. In Proceedings of ACL 2014.

[3] N. Kalchbrenner, E. Grefenstette, P. Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. In Proceedings of ACL 2014.

[4] Y. Shen, X. He, J. Gao, L. Deng, G. Mesnil. 2014. Learning Semantic Representations Using Convolutional Neural Networks for Web Search. In Proceedings of WWW 2014.

[5] Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuglu, P. Kuksa. 2011. Natural Language Processing (Almost) from Scratch. Journal of Machine Learning Research 12:2493–2537.

[6] Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. Neural computation, 9(8), pp.1735-1780.

[7] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y., 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

[8] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. CoRR, abs/1409.3215, 2014. URL http://arxiv.org/abs/1409.3215.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert:
Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018).

[10] Peter Turney. 1999. Learning to extract keyphrases from text. National Research Council Canada, Institute for Information Technology, Technical Report ERB-1057.

[11] Gerard Salton and Christopher Buckley. Termweighting approaches in automatic text retrieval. Information Processing and Management, 24(5):513–523)(1988).

[12] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. CoRR, abs/1708.02709, 2017.

[13] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional neural networks. In ECCV, 2014.

[14] Hasan Kazi Saidul, Ng,Vincent. Automatic keyphrase extraction. 2014. A survey of the state of the art. 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference. Vol. 1 Association for Computational Linguistics (ACL), 2014. p. 1262-1273.

.