Natural Language Processing Project

Aditya Singh Mahala (CE17B023)[1] and Burhanuddin Sabuwala (BE17B011)[1]

Indian Institute of Technology Madras, Chennai 600036, India

Abstract. Information Retrieval is an essential task with applications in various search engines. We utilized topic modeling, concept modeling, word relation model-based approaches for the task of information retrieval. Additionally, we also tried a deep learning-based encoder-decoder-based system. We tested our methods on the Cranfield dataset. We found that simpler methods perform better than complex methods. Latent Semantic Analysis turns out to be our best-performing model.
The classical methods used for Information Retrieval(IR) have been quite effective in the long run. In this project, we have used different techniques, combined with classical IR techniques and some based on deep learning. The main aim of this study is to compare some of the different techniques and then present the best model for IR.

Keywords: LSA, LDA, GloVe, Information Retrival, Encoder-decoder, NLP, Word2Vec, Cranfield

1 Introduction

Information retrieval (IR) is a class of problems where the goal is to obtain relevant documents from a database with the input as a query. It is the science of finding the most relevant set of k documents. One classic example of Information Retrieval is Google Search. Multiple web pages are indexed by the web crawlers. Based upon a given query (Google search), the search engine tries to retrieve the most relevant set of webpages (or documents) and ranks them. Although web search engines are the most popular IR applications, it has a wide variety of applications. Some of the applications are search engines to find the most relevant book or set of books in a library, search engines to search relevant people on social media, Recommender systems, accessing genomic information, accessing appropriate court proceedings, etc.

There are several methods to approach the information retrieval problem. One of the most common methods for IR in structured databases is SQL-based queries. However, the queries used in IR are usually given in a natural language, and therefore, SQL is not compatible with the task. Moreover, the information Retrieval task is challenging because of the complexity of the language and different styles of using the language, which is reflected in the documents as well as queries. Additionally, SQL does not rank the documents. Therefore, there is a need to address IR using other innovative methods. Most of the existing IR-based systems compute a numeric score for every document in the database with

the given query. This makes the ranking process more straightforward, and the user can then choose to retrieve the corresponding document/documents based on the shown results.

2 Problem Definition

Dataset: We are given the Cranfield dataset. It is one of the standard classical datasets for the purpose of information retrieval. Cranfield dataset contains 1400 documents and 225 queries. Associated with each query is a list of relevant documents with their corresponding relevance score. The relevance score denotes whether a document answers the query, whether it is just an example or if it only refers to a few things relevant to the document.

Task: Our task is to build a complete end-to-end IR system. This IR system has been documented in its database. It takes queries as input and reports k relevant documents as output.

3 Background and Related Work

Several methods have been proposed for information retrieval. The first and foremost step for information retrieval is cleaning. Effective pre-processing technique can greatly improve the performance of an IR system [1] [2]. We used the same preprocessing techniques that we had used in assignment 1a. Further, this preprocessed data is converted into some numerical form. It could either be vectors or probabilities. The same treatment is applied to queries. A similarity score could be obtained to rank the documents based on the queries. Various methods have been used to address the IR task.

Several methods have been tried for Information Retrieval. Semantic Analysis-based methods are quite widely used. Some of the popular approaches are Latent Semantic Analysis, Explicit Semantic Analysis, and Latent Semantic Indexing [7], [8]. Aguilar et al. compared term frequency-inverse document frequency (TF-IDF) with Latent Semantic Analysis, Latent Dirichlet Allocation, BM25, and Doc2Vec on the Cranfield dataset. LSA was the best performing method among all of them, followed by BM25, LDA, and Doc2Vec [8].

4 Motivation

The current TF-IDF model assumes orthogonality between all the words. This assumption is not valid. Moreover, the vectors are very sparse and very large. The size of the document TF-IDF matrix is about 6227 x 1400. This occupies huge RAM, which is unnecessary. The large size of the vector also amounts to the considerable time taken to compute the cosine similarity. Therefore, a topic or concept-based model is required. This would make the vectors dense and small, thus optimizing the memory and time requirements. Additionally, it would also

address the issue of orthogonal words. For this purpose, we used Latent Semantic Analysis and Latent Dirichlet Allocation. Additionally, we also used Word2Vec embeddings and GloVe embeddings, taking into account the meaning of the words.

Recurrent Neural network-based models have seen increasing use and popularity in the field of Natural Language Processing. This happened because of their ability to process sequence data. One of the most popular ideas in the field of NLP has been encoder-decoder ([4]) based models for Neural Machine Translation. It uses two separate RNN models. The first one is an encoder that encodes the input sequences to a vector of fixed length and the second one is a decoder that uses this vector to construct the output sequence. The most important part of this model is compressing the input sequence into a vector which can be easily decoded by the decoder to give meaningful output sequences. We used this idea to develop a model which can compress the documents and queries in the Cranfield dataset to a set of vectors that can be later compared to determine the required metrics.

One major issue with RNN's is that their performance with long sequences is not good. In order to tackle that, we have used bi-directional LSTM's ([3]). Another major issue that arises with long sequences is that the context of the text in different parts of sentences is lost in the case of long sentences. For example, take the following sentence "Breakfast was healthy. I had it with my family." where "it" refers to "Breakfast." But such contextual meanings of some words are not effectively modeled by sequential models. In order to model such long-term dependencies between words, we used the attention mechanism ([5]).

5 Proposed Methodology

5.1 Baseline Model

The baseline model consists of several steps. The steps are given below.

1. Sentence Segmentation: We used Punkt tokenizer present in NLTK package
2. Word Tokenization: We used Penn Treebank Tokenizer
3. Stopword removal: We removed all the stopwords mentioned in nltk package
4. Lemmatization: We lemmatized the words to their root form using the spacy package.
5. Information Retrieval: We converted the documents to their TF-IDF vector forms. The queries were also converted to their similar TF-IDF vector form. Then cosine similarity between the document vector and query vector is used to rank the documents.
6. Evaluation: The evaluation is done on multiple metrics. Precision, Sensitivity, Mean Average Precision, normalized Discounted Cumulative Gains, F1-score, time taken for document retrieval are the metrics that we are using to evaluate our methods. Out of all these metrics, only nDCG accounts utilizes the relevance scores of the documents corresponding to each query.

Most of the methods that we are using in this study follow the same preprocessing steps as the baseline model. The preprocessing steps include Sentence Segmentation, Word Tokenization, Stopword Removal, and Lemmatizaition. These methods are also evaluated on the same metrics as mentioned above.

5.2 Latent Semantic Analysis

Latent Semantic Analysis (LSA) is used for analyzing relationships between a set of documents and the terms that are present in them. This is done by producing a set of concepts related to the documents and terms. LSA works on the assumption that the words that are close in meaning will occur in similar pieces of text.

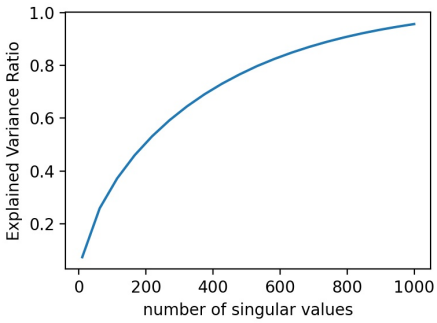

Fig. 1. Explained Variance Ratio for Singular Value Decomposition. This is obtained by Latent Semantic Analysis of documents of Cranfield dataset.

For LSA, we generated the TF-IDF vectors corresponding documents and build a TF-IDF x document matrix. Then, we performed Singular value decomposition to obtain the concepts. The resulting vectors of the documents are linear combinations of individual concepts. This overcomes the assumption that is made in the TF-IDF vector space model that the terms are orthogonal to each other. Further, the query vectors also undergo a similar transformation, and cosine similarity is used to obtain the rankings of the documents. In our study, we retained 500 concepts (or singular values). This amounted to almost 80% of the total explained variance. This is shown in figure 1.

5.3 Latent Dirichlet Allocation

Latent Dirichlet Allocation is another method that takes into account the similarity between words based on documents. LDA is used for topic modeling and topic discovery in Natural Language processing. It is a statistical model that assumes that the documents are made up of a small number of hidden topics.

Most documents will only contain a relatively small number of topics. They follow a probability distribution such that a given document is more likely to contain some topics than others. LDA is a generalized approach of probabilistic Latent Semantic Analysis. LDA assumes that there are θ_i topic distributions for document i and ψ_k word distribution for topic k [9], [10].

For our work, we used the LDA model from the gensim package [11]. The LDA model assumed that each of the documents is made up of 8 topics. The LDA model converts the documents (Bag of word encoded documents) to probabilities of each topic. Further, we used Jensen Shannon Distance (JSD) between the query and the documents to get the corresponding rankings. We used JSD instead of cosine similarity because we are dealing with the probabilities of each topic. Unlike cosine similarity, lower JSD means higher similarity in this case.

5.4 GloVe Embeddings

We also tried GloVe word embeddings [6]. The document is taken as an average of all the words occurring in the document. For similarity ranking, we used cosine similarity.

5.5 Word2Vec Embeddings

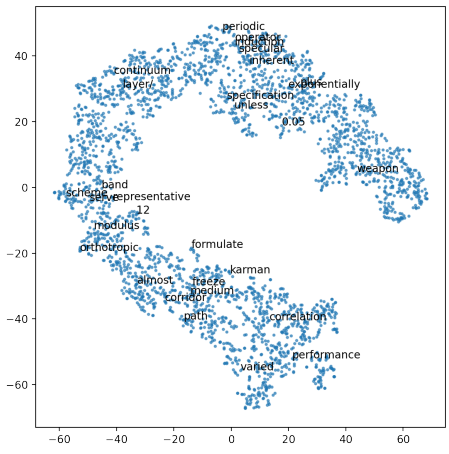

Fig. 2. TSNE visualization of Word2Vec embedding of Cranfield Documents

Word2Vec is a shallow neural network model to learn word associations. Word2Vec is tailored in such a way that cosine similarity between the words would represent their semantic similarity. It assumes that distributionally similar words would also be semantically identical.

Word2Vec was also done using the gensim model [11]. The model was trained on the Cranfield documents. Each of the words in a given document is converted to Word2Vec embeddings. Then the document vector is represented using the average of all of these word vectors. The embeddings are visualized in fig 2. The ranking is obtained by computing cosine similarity between the document vector and the corresponding query vectors.

5.6 Encoder-Decoder base method

Layer (type)	Output Shape	Param
Embedding	Embedding	d
Embedding	(None, 50, 200)	1098200
Encoder_Bidirectional_LSTM$_1$	(None, 50, 128)	135680
Encoder_Bidirectional_LSTM$_1$	(None, 50, 128)	98816
Encoder_Bidirectional_LSTM$_1$	(None, 50, 128)	98816
Encoder_Bidirectional_LSTM$_1$	(None, 50, 128)	98816
attention	(None, 128)	178
RepeatVector	(None, 50, 128)	0
Decoder_LSTM$_1$	(None, 50, 64)	49408
Decoder_LSTM$_2$	(None, 50, 128)	33024
Decoder_LSTM$_3$	(None, 50, 128)	33024
Decoder_LSTM$_4$	(None, 50, 128)	33024
Decoder_LSTM$_5$	(None, 50, 128)	33024
TimeDistributed	(None, 50, 200)	13000
Dense	(None, 50, 200)	0

Table 1. Encoder-Decoder Architecture

1. The first step is data preprocessing, which only includes lemmatization and stop word removal. Once we got the data as tokenized words, we converted them back to sentences by concatenating them. This was done because our model takes input in sequential format and not in the form of word tokens.
2. Then, we used a tokenizer to fix the vocabulary size and create a word index. This also helped us in converting the data from textual data to numeric data, which was later on used for creating a word embedding matrix.
3. After getting the word index for all the words, we created sequences of length 50 to be passed on to our model. These sequences have a word index in place of each individual word. In order to fill sequences of length less than 50, the post padding technique has been used. Vocabulary size is kept to 1000 words.
4. Next step involves creating a word embedding matrix which will be fed to the neural network for creating document tensors. For this model, we have used glove word embeddings([6]). This matrix consists of a row vector representation of each word at its respective index. The dimensions of this matrix are (1000,200) where each row represents a word.

5. Next step involves creating an encoder-decoder model. For this model, we have four encoder layers and four decoder layers. There is an attention layer after the encoder layers. The model uses a self-training mechanism (unsupervised learning) that takes in the sequence and tries to restructure the same sequence. This idea effectively helps us to get a vector representation of the documents. There are 64 hidden nodes for each layer. Input dimensions are (batch_size, 30, 200) where batch_size is 64.
6. For training, we have divided the data into 90 % training set and 10% validation set. We have used Adam optimizer for training with a learning rate of 0.001. After that, we train the model, and once the model is created, we create another model based on this model. This new model takes input in the form of the sequence of the original text and outputs the context vector from the encoder. We repeat this process for both documents and queries to get a list of all the vectors. Once the vector representations are obtained, we used a similar evaluation method for other models.

6 Results

6.1 Information Retrival Performance

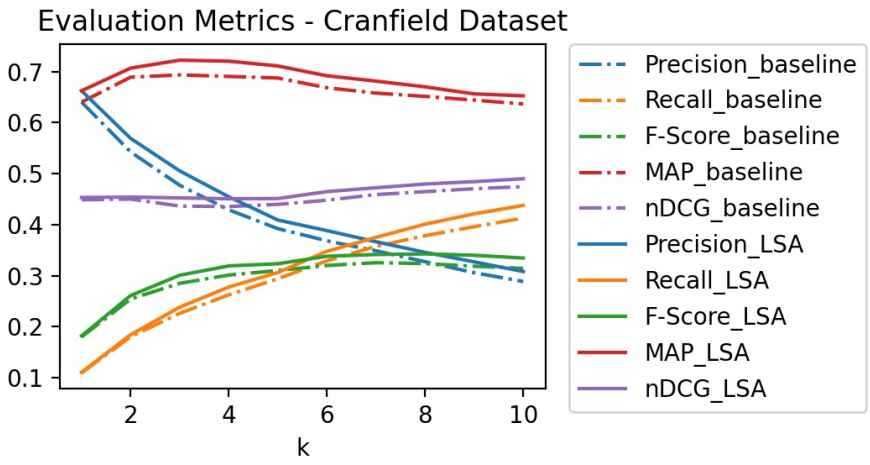

Fig. 3. Latent Semantic Analysis Evaluation score compared with Baseline model

It is seen that Latent Semantic Analysis shows significant improvement compared to the baseline model from figure 3. There is improvement in all the performance metrics that we are considering as compared to the baseline model.

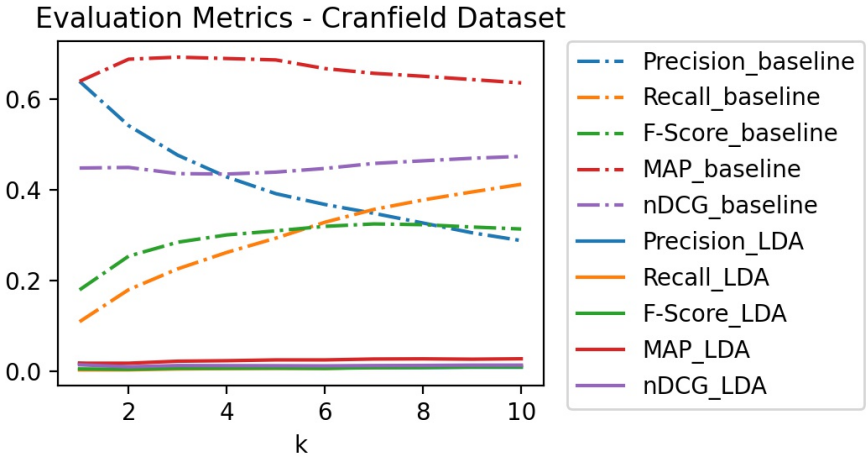

Fig. 4. Latent Dirichlet Allocation Evaluation score compared with Baseline model

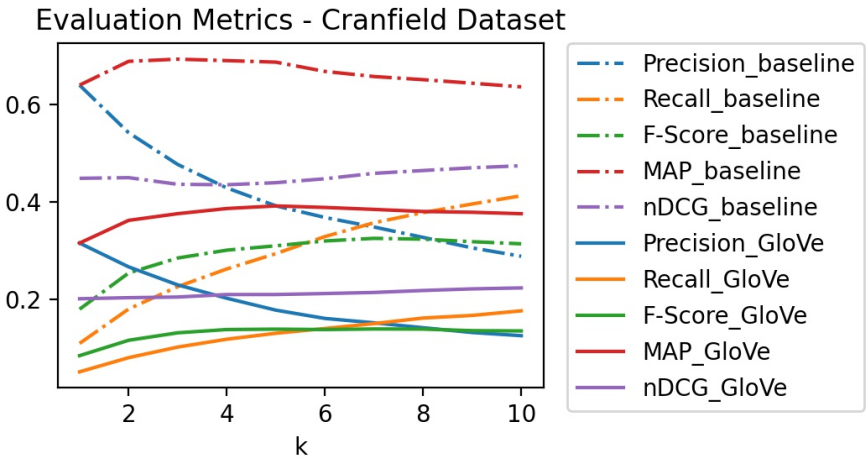

Fig. 5. Glove Embedding Evaluation score compared with Baseline model

From the figure 4, it is seen that Latent Dirichlet Allocation is not performing well. It cannot beat the baseline TF-IDF-based model. All the performance metrics are pretty low, but this model is performing slightly better than random.

GloVe embedding model is performing worse than baseline model in all the performance metrics as seen in figure 5. The GloVe embedding model is able to perform better than random. However, it is not able to beat the baseline model. This suggests that maybe if the GloVe model is trained on a larger dataset than

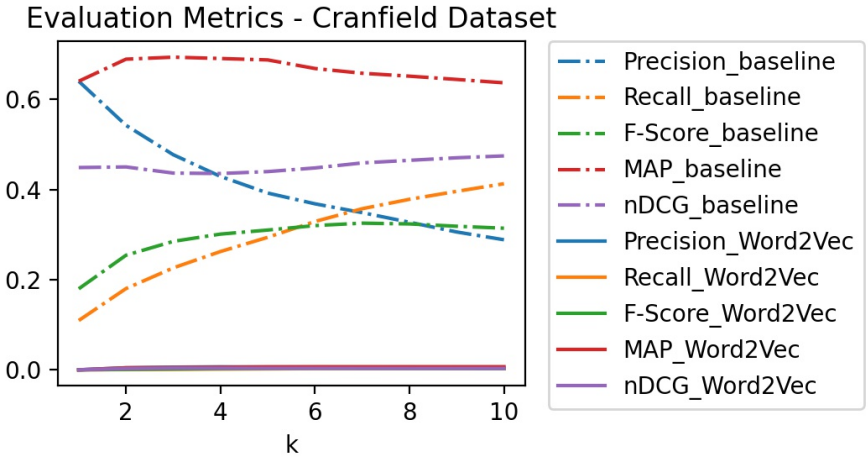

Fig. 6. Word2Vec embeddings Evaluation score compared with Baseline model

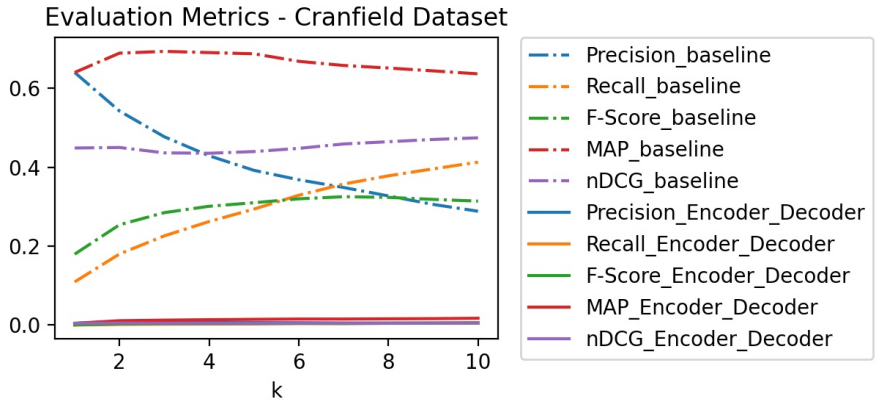

Fig. 7. Ouptut metrics for Encoder-Decoder Model

Cranfield but having similar properties as Cranfield documents and queries, it can potentially beat the baseline model.

Word2Vec model is not performing as expected. It is not able to pick up the relationships between the words correctly. This is clear from the figure 2. The figure 6 shows that the Word2Vec model is not performing as expected. The performance of the Word2Vec model is almost as good as random not. Therefore, it is not entirely helpful in our task. This could potentially be attributed to the

smaller size of the Cranfield dataset compared to the typical corpora used to train the standard Word2Vec models.

As shown in fig. 7, we found out that the encoder-decoder-based model doesn't perform as per expectations on the given dataset.

Therefore overall, LSA performs the best and is the only model that performs better than the baseline model. The second-best model is the baseline model. However, it has a much longer execution time. Then followed by the GloVe model, which also performs well (better than random). This is followed by LDA, Encoder-Decoder-based approach, and Word2Vec embeddings.

6.2 Computational Time

Method	Total time (seconds)	Information Retrival time (seconds)
Baseline model	3545	941
LSA	1336	52.3
LDA	1363	191.3
GloVe	1020	82.3
Word2Vec	1127	62.3
Encoder-decoder	2000.342	447.34

Table 2. Time comparision of different methods

The table 2 shows the amount of time taken to run the complete code and the information retrieval part. It is seen that all the models perform better than the baseline model. This could be attributed to the fact that the vector sizes corresponding to each document are much smaller in any of the models compared to the baseline model. This significantly reduces the time taken to compute similarity scores. It also helps with the storage of the document vectors as these compressed representations would require less memory. The GloVe model is already trained, and it only fetches the word embeddings from the trained model. Therefore, comparing GloVe timings with others might not be a fair comparison.

7 Conclusion

The performance scores that we are getting are consistent with Aguilar et al. [8]. They also had LSA as the best performing model, followed by BM25, LDA, and Doc2Vec on the Cranfield dataset.

The better performance of simpler models such as LSA and the baseline model can be attributed to the fact that their corresponding parameters are estimated much better than the other complex models with multiple parameters since their models are simpler. These other complex models probably require much larger amounts of data for better parameter estimation. In other words, the

complex models like the deep encoder-decoder, LDA, Word2Vec were possibly under-fitted.

On the other hand, LSA and baseline models had an advantage over them as the number of documents in the Cranfield dataset is low. However, GloVe is an exceptional case. We were using pre-trained embeddings of the GloVe model. This might be responsible for giving it the leverage to perform much better than random. However, it was not specific to the peculiar aerospace literature of Cranfield and was not able to beat the baseline model. As for the time complexity, we are reducing the number of components of vector used to represent documents from 6772 (in the baseline TF-IDF model) to a few hundred or a few tens, significantly reduces the information retrieval time.

8 Acknowledgements

We are grateful to Prof Sutanu Chakraborti for providing us with an opportunity to work on this project and equip us with the tools necessary to build an end-to-end Information Retrieval System. We are also grateful to the TAs of the NLP course for helping us with our difficulties. We really had a great time during the course and we learnt a lot. We would also like to acknowledge our classmates for initiating very interesting discussions in the class that helped us in our project.

References

1. Harnani Mat Zin, Norwati Mustapha, Masrah Azrifah Azmi Murad, and Nurfadhlina Mohd Sharef, "The effects of preprocessing strategies in sentiment analysis of online movie reviews", AIP Conference Proceedings 1891, 020089 (2017) https://doi.org/10.1063/1.5005422
2. Haddi, Emma, et al. 'The Role of Text Preprocessing in Sentiment Analysis.' Procedia Computer Science, vol. 17, 2013, pp. 26–32. DOI.org (Crossref), doi:10.1016/j.procs.2013.05.005.
3. Hochreiter, Sepp, and Jürgen Schmidhuber. "Long Short-Term Memory." Neural Computation, vol. 9, no. 8, Nov. 1997, pp. 1735–80. DOI.org (Crossref), doi:10.1162/neco.1997.9.8.1735.
4. Cho, Kyunghyun, et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation." Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, 2014, pp. 1724–34. DOI.org (Crossref), doi:10.3115/v1/D14-1179.
5. Bahdanau, Dzmitry, et al. 'Neural Machine Translation by Jointly Learning to Align and Translate.' ArXiv:1409.0473 [Cs, Stat], May 2016. arXiv.org, http://arxiv.org/abs/1409.0473.
6. Pennington, Jeffrey, et al. "Glove: Global Vectors for Word Representation." Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, 2014, pp. 1532–43. DOI.org (Crossref), doi:10.3115/v1/D14-1162.

7. Egozi, Ofer, et al. 'Concept-Based Information Retrieval Using Explicit Semantic Analysis.' ACM Transactions on Information Systems, vol. 29, no. 2, Apr. 2011, pp. 1–34. DOI.org (Crossref), doi:10.1145/1961209.1961211.
8. Aguilar, J. et al. 'Comparison and Evaluation of Different Methods for the Feature Extraction from Educational Contents.' Computation, Jan. 2020. repository.eafit.edu.co, http://repository.eafit.edu.co/handle/10784/28641.
9. Blei, D., Ng, A., Jordan, M. Latent Dirichlet allocation. J. Mach. Learn. Res. 3 (January 2003), 9931022.
10. Blei, David M.' Probabilistic Topic Models. Communications of the ACM, vol. 55, no. 4, Apr. 2012, pp. 77–84. April 2012, doi:10.1145/2133806.2133826.
11. Gensim–python framework for vector space modeling. NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic, 3(2). 2011
12. Mikolov, Tomas, et al. 'Efficient Estimation of Word Representations in Vector Space.' ArXiv:1301.3781 [Cs], Sept. 2013. arXiv.org, http://arxiv.org/abs/1301.3781.