SEMANTIC SEARCH USING WORD EMBEDDINGS

Aishwarya Sahani
Computer Science
University of Illinois at Chicago
Chicago, IL, USA
asahan2@uic.edu

ABSTRACT

Conventional Search engines use traditional Term-Frequency and Inverse Document Frequency (tf-idf) based retrieval techniques to fetch webpages. These models retrieve pages which exactly match the term by checking the similarity between the query & the webpages. The order of these retrieved webpages is in the descending order of similarity. There are various similarity measures, but the cosine similarity works well. However, this technique fails to capture the semantics or the meaning of the query. Documents which do not have the query term have low chance of retrieval. Thus, the modern state-of-the-art search engines transitioned to Deep Learning Approaches by using word embeddings and neural models for document retrieval. The purpose of the project is to develop a search engine which searches semantically by combining the word embeddings with tf-idf. The search results have been further improved by using PageRank algorithm to rank the results. This leads to important & relevant being ranked higher. This simplistic approach gathers positive results.

KEYWORDS

Information Retrieval, Search Engine, Word Embeddings, Term frequency – Document frequency, Ranking, Similarity, word2vec, glove, PageRank

1 INTRODUCTION

Recent work in Information Retrieval have shown the use of Word Embeddings yields better results. Word embeddings are representation of the word in a vector space model across fixed dimensions. These representations are good at representing the syntactic and semantic structure of the word. This helps Word Embeddings to identify similar words easily. Word Embeddings have already improved on language modeling and feature learning techniques in natural language processing. There are multiple word embedding models like word2vec and Glove. These embeddings have been trained on different techniques and on different datasets. A lot of information retrieval techniques have exploited the success of the word2vec vectors. These word embeddings can be trained on the corpus or we can use the pre-trained embeddings. Such embeddings are trained on the web corpus or the Wikipedia corpus. We will rely on the pre-trained embeddings as our corpus is not large enough to train these representations.

Earlier, traditional weighting schemes like Term Frequency – Inverse Document Frequency (tf-idf) techniques would extract the most descriptive terms in a document and measure the similarity across queries and documents. This simplistic approach worked effectively. However, such bag-of-words models do not capture the positional information. Also, they rely on exact keyword matching and thus, they are unavailable to capture the semantics of the word. So, documents with similar words and not the exact words will not be fetched.

This is where earlier mentioned idea of word embeddings come into the picture. We will combine the strengths of the tf-idf techniques and overcome the drawbacks by incorporating word embeddings. This combination will be the dot product multiplication of the tf-idf of the word in the document and the fixed dimensional vector representation of the word given by the embedding. The average of all these word vectors across the document will represent each document and queries. The most similar documents against the query will be the result of the search.

There are multiple similarity metrics to evaluate the similarity between the documents and the query. We will use the cosine similarity metric. Since, it is a reliable evaluation metric. Now, to improve the results we use the PageRank algorithm to find the importance of the retrieved webpages. The PageRank algorithm works on the assumption that important websites are likely to be referenced by other important websites. The more important websites of the lot will be ranked higher. The most important pages will be authentic and trusted. Thereby, relevant to the user. These PageRank results will help us rank the search results. The PageRank algorithm has multiple factors to evaluate the importance which have not been revealed for commercial purposes. These multiple factors of the PageRank have been attributed for the success of Google and other search engines.

2 LITERATURE REVIEW

Recent work in Information Retrieval have been shifting from the weighting schemes like tf-idf to the word embedding approaches. In the conference, Word Embedding based Generalized Language Model for Information Retrieval [14], researchers discussed about the Neural Information Retrieval (Neu-IR) research. Out of the 19 papers that were presented at the conference, 10 papers were about Word embeddings. The Neu-IR papers were about

full-text search using word embeddings [10, 11, 12, 13]. This showed the promise in the up and coming Neural Network & Word embeddings research in the field of Information Retrieval. The conference was the culmination of the impressive work being conducted since the introduction of the word embedding schemes like word2vec [24] & Glove [25]. The success of such word embeddings in Natural Language Processing tasks have led them to be used in Information Retrieval tasks as well [2, 4, 8, 10]. A lot of blogs have also been explaining the simplicity and effectiveness of these models [16, 18, 20, 21, 22]. They have also been highlighting the approaches to combine both the weighting scheme along with vector space representations for effective search results. Research has been ongoing with different techniques for combining them [27, 28, 29]. This paper aims to contribute to this research as well. Various research papers have tweaked the algorithm based on the user personalization. The user habits like clicks have modelled and quantified which results in a weighted multiplication of the tf-idf & word embeddings [10].
In the recent few years, language modelling and word embedding tasks have been at a pinnacle. The models like ELMO [22] & BERT [5] are state-of-the-art and these models have also been incorporated for information retrieval tasks. These types of embeddings are contextual embeddings. Thus, yielding better & appropriate results. However, these models rely on heavy training & a large corpus to learn & differentiate contexts. Thus, we cannot effectively use them in this paper. We will perform a juxtaposition with Glove embeddings & word2vec embeddings for the scope of the paper. A lot of the existing research have used word2vec. However, very few have incorporated Glove. This motivation of the paper will be to compare the results to understand and review which embeddings works the best for our task.
We will also be focusing on the PageRank algorithm to rank the results of the search. The PageRank algorithm has been a key reason for the success of Google and works on a simple idea of ranking important pages higher [23, 32]. There have been a lot of variations on this PageRank algorithm. They have used weights to represent the different factors. However, we will stick with the simple introductory research paper about the PageRank algorithm [31] for the purpose of our research.

3 SYSTEM ARCHITECTURE

The proposed model consists of 3 parts: data collection, the weighing-based information retrieval technique and PageRank computation to fetch the relevant data from the collected data.

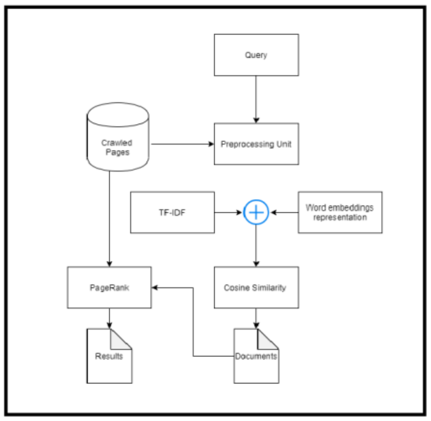

Figure 1: Figure represents the System architecture

The system architecture is divided into the following modules:

1.1 Dataset

We will be crawling the web to create our own Dataset of webpages. These webpages will belong to UIC domain ("uic.edu"). We will start from our seed site https://cs.uic.edu and explore within the "uic.edu" domain by performing a breadth first search using the Queue data structure. Whenever, we are crawling a website, we will capture 2 things viz. all the links the page refers to and all the html content of the page. We will save this in a CSV against the field "URL" to identify the records. To manage the space effectively, we will perform some steps of preprocessing while saving the data. We will perform tokenization, followed by stop word removal to cut down the size of the page. Also, to make sure that the page is relevant, we will exclude the content in the script & style tags.
The CSV acts as a repository for our search. We will perform the search by retrieving the relevant URLs from this dataset. This dataset is a collection of almost 3.5 K webpages.
The textual data would be important for the weighting scheme & word embedding. While the link structure would be required for PageRank computation.
Packages like request & urllib have helped us in requesting & handling of the URLs and prettifying the text content using the Beautiful Soup package. To concurrently perform this process, we have used multithreading. This helped us in improving the speed of the collection.
You can find these datasets in link shared below [33].

1.2 Information Retrieval

The system loads the data in the form of documents (web pages) & their content (html). We had already performed tokenization, followed by stop word removal while storing. We complete the preprocessing by performing lemmatization & word cleansing. We compute the inverted index from the preprocessed data and calculate the tf-idf from the inverted indices over the corpus. The tf-idf is a weighting factor representation of how important a term is to a document & also to the corpus.

Now, we load the word embeddings. The words of the documents are now represented by fixed 300 dimensional vectors. We average these vectors to get one single document vector i.e. the vector which syntactically & semantically represents the document. Now, for every word vector in the document, we perform the multiplication with the tf-idf value. The result will be a vector of the same dimensions with probability of the terms along with the vector representation of the Glove embeddings. This combined value will now be represented by the document in the document vector.

We will now perform similar steps with the query and then compare the query vector & the document vector. For the comparison, we will use cosine similarity as an evaluation metric. We fetch the 50 most similar documents based on cosine similarity for each query. These documents are then ordered for the user based on the PageRank algorithm described below in chunks of 10.

1.3 PageRank

From the crawled pages and the information of the links that they are pointing to, we will compute the PageRank. PageRank works by counting the number and the importance of in-links to a page. If the given page is referring to another page, it indicates that would increase the importance of the latter page. We imagine the World wide web or the crawled data in our case as a graph with the webpages acting as nodes & the out-links from one page to another acting as edges between nodes.

We will store & represent this graph as a matrix to aid for mathematical computations. We will perform matrix multiplication to compute the PageRank values of all pages. After convergence, the results will be a numpy array of floats representing importance. Higher number indicates higher importance. This array can be mapped across pages.

This importance array will be used to rank the relevant results acquired from the second module i.e Information Retrieval by ordering higher quality results over lower quality content. The results retrieved from the earlier module will now be ordered based on their corresponding PageRank values for the relevancy of the user in the chunks of 10.

4 RESULTS & EVALUATIONS

We have used evaluation metrics like Precision, Recall & F1- score to evaluate the system as that is the general evaluation practice. We have 5 pre-defined queries and their 10 gold-standards of results. We have computed the precision & recall values at the 5th document and at the 10th document. The precision & recall values at the end were the same as we have evaluated the system on 10 queries & 10 relevant documents. We will compare the performance of the system for Glove embeddings & word2vec embeddings. For the Glove embeddings, we get a score of 0.76 as our average precision, average recall & macro-average f1-score. The system performed better for current affairs as the system crawled more recent pages. The recent pages outweighed the old pages and the results were clear when we evaluated for historic records. Also, if we look at the precision values at the 5th document, we get a good precision score. Highlighting the idea of PageRank which shows how important pages are more relevant & trusted to the user. The precision decreases & recall increases as we gather less important pages. Surprisingly, the model suffered when we used Word2Vec embeddings. We got a low performance of 0.54 as our average precision, average recall & macro-average f1-score. The highest precision recorded at 0.68 at 5th document. It is down to the principle difference between the embeddings. The Glove matrix is built on a co-occurrence matrix reduction technique and thus works well with a weighted matrix model like tf-idf. The word vectors of the Glove are modelled in a way that they also model the probability of the word. This perhaps explains why Glove embeddings perform better than the Word2Vec embeddings which captures the words in similar contexts.

	Embeddings	
Measure	Glove	Word2Vec
F-score	0.76	0.54

Table 1: The table shows the evaluation results

5 CONCLUSION & FUTURE WORK

We would like to conclude by stating that we have created a dataset of link with 3.5 K preprocessed webpages and their links. The Glove word embeddings have been effectively combined with the weighting scheme of tf-idf. The multiplication of these 2 entities leads to a model which exploits the advantages of both the techniques. Tf-idf techniques helps in matching against the exact words and the semantic structure of the Glove embeddings helps in matching against the similar words as well. Thereby, resulting in a semantic search. The PageRank although an add-on to rank the records proves to be pivotal in showcasing relevant, important & trusted information as high as possible. Due to this, the user does not have to go deep to click on the relevant page. The strength of the model is its simplicity of combining simple retrieval techniques for better results. Thereby, this simple approach has led to a semantic search engine.

For future work, we can probably index the records like PageRank & tf-idf scores of the corpus into a Database or a CSV file. So that, whenever the user inputs a query, rather than computing these values again, the system will fetch from indexed file or table. This will improve the response time of the system. Also, we can set weights for multiplication of tf-idf and embeddings. We can model the user behavior to find & assign weights. These weights will personalize the search results for the user. We can build a larger dataset than the existing crawled dataset. However, indexing will be necessary if we add on in the existing dataset. With a larger dataset, we could effectively train modern contextual groundbreaking word embeddings like ELMO & BERT. ELMO & BERT have worked wonders in a lot of Natural Language Processing Tasks and it would be fascinating to see how these models work in the Information Retrieval Domain with respect to the traditional models.

REFERENCES

[1] Lewandowski, Dirk. "Web searching, search engines and Information Retrieval." Inf. Services and Use 25 (2005): 137-147.

[2] Landthaler, Jörg, Bernhard Waltl, Patrick Holl and Florian Matthes. "Extending Full Text Search for Legal Document Collections Using Word Embeddings." JURIX (2016).

[3] Mitra, Bhaskar and Nick Craswell. "An Introduction to Neural Information Retrieval." Found. Trends Inf. Retr. 13 (2018): 1-126.

[4] Zuccon, Guido et al. "Integrating and Evaluating Neural Word Embeddings in Information Retrieval." ADCS (2015).

[5] Patel, Manish. "TinySearch - Semantics based Search Engine using Bert Embeddings." ArXiv abs/1908.02451 (2019): n. pag.

[6] Lewandowski, Dirk. "The Retrieval Effectiveness of Web Search Engines: Considering Results Descriptions." ArXiv abs/1511.05800 (2008): n. pag.

[7] Bast, H. et al. "Semantic Search on Text and Knowledge Bases." Found. Trends Inf. Retr. 10 (2016): 119-271.

[8] Diaz, Fernando et al. "Query Expansion with Locally-Trained Word Embeddings." ArXiv abs/1605.07891 (2016): n. pag.

[9] Craswell, Nick et al. "Report on the SIGIR 2016 Workshop on Neural Information Retrieval (Neu-IR)." ACM SIGIR Forum 50 (2016): 103 - 96.

[10] Amer, Nawal Ould et al. "Toward Word Embedding for Personalized Information Retrieval." ArXiv abs/1606.06991 (2016): n. pag.

[11] Rekabsaz, Navid et al. "Uncertainty in Neural Network Word Embedding: Exploration of Threshold for Similarity." ArXiv abs/1606.06086 (2016): n. pag.

[12] Roy, Dwaipayan et al. "Representing Documents and Queries as Sets of Word Embedded Vectors for Information Retrieval." ArXiv abs/1606.07869 (2016): n. pag.

[13] Cohen, Daniel et al. "Adaptability of Neural Networks on Varying Granularity IR Tasks." ArXiv abs/1606.07565 (2016): n. pag.

[14] Ganguly, Debasis et al. "Word Embedding based Generalized Language Model for Information Retrieval." Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (2015): n. pag.

[15] Wang, Shenghui and Rob Koopman. "Semantic Embedding for Information Retrieval." BIR@ECIR (2017).

[16] Chawla, Japneet Singh. "Word Vectorization (TFIDF/Word2Vec)." Medium, Medium, 31 July 2018, medium.com/@japneet121/introduction-713b3d976323.

[17] Edmund Martin. "Concurrent Crawling in Python." *Edmund Martin*, 1 July 2017, edmundmartin.com/concurrent-crawling-in-python/.

[18] Egg, Alex. "Search Query Embeddings Using query2vec." *Medium*, Grubhub Bytes, 4 Nov. 2019, bytes.grubhub.com/search-query-embeddings-using-query2vec-f5931df27d79.

[19] Frankie. "How to Create a Web Crawler From Scratch in Python." *DEV Community*, DEV Community, 10 Apr. 2020, dev.to/fprime/how-to-create-a-web-crawler-from-scratch-in-python-2p46.

[20] Husain, Hamel. "How To Create Natural Language Semantic Search For Arbitrary Objects With Deep Learning." Medium, Towards Data Science, 18 Sept. 2018, towardsdatascience.com/semantic-code-search-3cd6d244a39c?gi=50a3e4e1f365.

[21] Taylor, Josh. "Supercharging Word Vectors." *Medium*, Towards Data Science, 31 Dec. 2018, towardsdatascience.com/supercharging-word-vectors-be80ee5513d.

[22] Taylor, Josh. "ELMo: Contextual Language Embedding." *Medium*, Towards Data Science, 6 Jan. 2019, towardsdatascience.com/elmo-contextual-language-embedding-335de2268604.

[23] "The Anatomy of a Large-Scale Hypertextual Web Search Engine." *The Anatomy of a Search Engine*, infolab.stanford.edu/~backrub/google.html.

[24] Mikolov, Tomas et al. "Efficient Estimation of Word Representations in Vector Space." CoRR abs/1301.3781 (2013): n. pag.

[25] Pennington, Jeffrey et al. "Glove: Global Vectors for Word Representation." EMNLP (2014).

[26] Mikolov, Tomas et al. "Distributed Representations of Words and Phrases and their Compositionality." ArXiv abs/1310.4546 (2013): n. pag.

[27] A. Guo and T. Yang, "Research and improvement of feature words weight based on TFIDF algorithm," 2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference, Chongqing, 2016, pp. 415-419, doi: 10.1109/ITNEC.2016.7560393.

[28] Yang, Xinli et al. "Combining Word Embedding with Information Retrieval to Recommend Similar Bug Reports." 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE) (2016): 127-137.

[29] Boom, Cedric De et al. "Representation learning for very short texts using weighted word embedding aggregation." Pattern Recognit. Lett. 80 (2016): 150-156.

[30] Shahmirzadi, Omid et al. "Text Similarity in Vector Space Models: A Comparative Study." ArXiv abs/1810.00664 (2018): n. pag.

[31] Page, Lawrence et al. "The PageRank Citation Ranking: Bringing Order to the Web." WWW 1999 (1999).

[32] Blanco, Roi and Christina Lioma. "Graph-based term weighting for information retrieval." Information Retrieval 15 (2011): 54-92.

[33] "Links." Google Drive, Google, drive.google.com/file/d/115DwigYoGrdAtuodVFNOZlWfAC_Fl977/view?usp=sharing.

.