Improvements on Vector Space Model
Team no: 13

Abhishek Kumar (CS21M002)
Ganesh Jatavath (CS21M019)
Mohammed Safi Ur Rahman Khan (CS21M035)

Jan-May 2022

1 Introduction

Information retrieval (IR) is defined to be the science of advancing the effectiveness of word-based document retrieval. It can also be characterised as "an activity of obtaining content (documents) of an unstructured type (text) from big collections (typically kept on corpus) to suit an information demand" or "an activity of obtaining material that can usually be documented on an unstructured nature i.e., usually text which satisfies an information need from within large collections which is stored on the computers."

The most simple method of performing this information retrieval task is by using Vector Space Model. Vector space model (VSM) or term vector model is an algebraic model for representing text documents (and any objects, in general) as vectors of identifiers (such as index terms). It is basically a statistical model.

But, even though its the most simple and easy way, there are a lot of limitations.

1.1 Limitations of Vector Space Model

- This model fails to take the order of words into account. Although this issue may be solved by using the tri-grams approach (in general n-grams) or other preprocessing techniques like phrase collection.
- Only some keywords which occur frequently are given importance over others. Long documents are poorly represented because they have poor similarity values.
- This model assumes an orthogonal relationship between words. i.e., it assumes that words are not related at all. This is a grave mistake as in the real world, there are many relationships that occur between words (like polysemy, synonymy, hyponymy, etc). Because of this assumption, we miss out on many related documents

- Syntactical (grammar) contexts are not taken into account.
- The words in the query must precisely match the words in the documents. This limitation is resolved to some extent by using lemmatization but it still exists.
- It relies on blind matching of words if query and words of documents and never consider the semantic closeness and similarity of words. Documents with similar context but different term vocabulary won't be associated,resulting in a false negative match.
- It relies on blind matching of words if query and words of documents and never consider the semantic closeness and similarity of words. Documents with similar context but different term vocabulary won't be associated,resulting in a false negative match.
- The order in which the terms appear in the document is lost in the vector space representation that make it not suitable for modelling sequences of terms in documents.
- The vector space model is very computationally expensive involving a heavy number of intensive calculations. Also, the vector space model is not very flexible in the sense that any new additions to the term vocabulary require recalculation of all vectors thereby making it computationally expensive.
- The vector space model theoretically assumes that terms are statistically independent (orthogonality assumption) which is often not the case.So there are different varieties of model proposed to overcome these limitations.Some of them being Latent Semantic Analysis(LSA), Explicit Semantic Analysis (ESA), Query Expansion .... etc . So in this project we are going to explore some of them.

2 Problem definition

If we observe, a lot of limitations of the vector space model are revolving around the fact that this model fails to take into account the semantic relationship of the words (i.e., assuming orthogonality between words). Our hypothesis is that we may be able to improve the performance of our search engine by removing the previous orthogonal assumption of words and modeling relationships between words and also by considering the context of the words (i.e., the distributional hypothesis which states that words that are used and occur in the same contexts tend to purport similar meanings)

In this work, we aim to try to solve some of these limitations of the vector space model thereby boosting the performance of the search engine.

In this project ,our aim to explore different models that can address vector space model limitations (Especially LSA and ESA).

There are very different evaluation parameters but in our project we only considering mainly 5 evaluation matrix .They are Precision, Recall, F-score, MAP, and nDCG. Eventually our goal in this project is to improve the search engine on cranfield dataset.

3 Motivation

While working on building a search engine using a simple vector space model, we noticed many discrepancies in the output thereby pointing us to the various limitations of implementing our search engine using a just simple vector space model (as discussed in detail in section 1.1).

As part of the project, we were given the Carnfield dataset and were asked to create a toy search engine using the Vector Space Model, which, as previously noted, has its own set of limitations. As a result, we'd like to improve on those limitations and make our search engine more robust by experimenting with the various models and strategies we learned in class. Some of the strategies tried are

- Latent Semantic Analysis (LSA)
- Different Word Vectorizations (BOW, TFIDF, Word2Vec)
- Explicit Semantic Analysis (ESA)

3.1 Motivation for Latent Semantics Analysis

Ferdinand de Saussure proposed the Structural view of language (structural linguistics), which essentially put forth the idea that words don't refer to actual real-world objects rather they refer to some "abstract concepts". A concept can only be understood by its relationship with other concepts. So, in LSA, we try to find these concepts from the dataset itself without referring to any external source. LSA assumes that words that are close in meaning will occur in similar pieces of text (i.e., distributional hypothesis).

LSA improved one of the main problems of information retrieval techniques, that is, handling polysemous words, by assuming there is some underlying latent semantic structure in the data that is partially obscured by the randomness of word choice. It uses statistical techniques to estimate this latent structure and get rid of the obscuring "noise."

It gives information beyond lexical levels, revealing hidden (latent) semantic relationships between the elements of interest.

3.2 Motivation for Explicit Semantic Analysis

In this, we try to use the massive open source data present freely on the web. And what better source than Wikipedia. The aim here is essentially representing texts as a weighted mixture of a predetermined set of natural concepts, which

are defined by humans themselves and can be easily explained. To achieve this aim, we use Wikipedia articles as concepts. So that we get an advantage of using vast amounts of highly organized human knowledge encoded in Wikipedia. Furthermore, Wikipedia is always evolving with time.

3.3 Motivation for different Vectorization techniques

Text Vectorization is the process of converting text into numerical representation. Here, the motivation is that by using advanced text vectorization techniques, we can essentially capture some form of similarity between different words. We have explored three techniques i.e., Bag of Words, Term frequency - Inverse Document Frequency, Word2Vec.

Here, especially, Word2Vec is special as it provides embedded representation of words. Word2Vec starts with one representation of all words in the corpus and train a Neural Network (with 1 hidden layer) on a very large corpus of data.

4 Background and Related Work

Many studies and investigations have been conducted to improve the search methods utilised in IR systems in order to meet the specific user-defined query. Understanding the contents of documents is a critical component of developing an Information Retrieval (IR) system. As a result, "the better and more the system understands the contents of the documents, the more successful the retrieval outcomes will be." For this purpose, we went over some of the below research papers/articles:

- Foltz, Peter. (1996). Latent Semantic Analysis for Text-Based Research []
- Landauer, Thomas. et.al. An Introduction to Latent Semantic Analysis []
- Gabrilovich, Evgeniy Markovitch, Shaul. (2007). Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis. []

5 Proposed Methodology

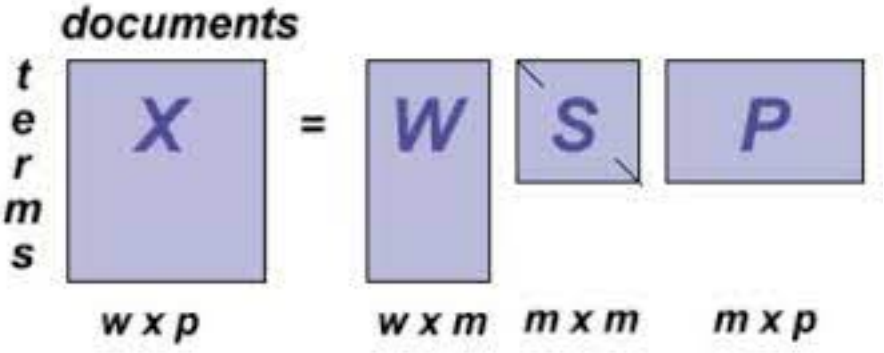
5.1 Latent Semantic Analysis

Latent semantic analysis (LSA) is a technique in natural language processing, in particular distributional semantics, of analyzing relationships between a set of documents and the terms they contain by producing a set of concepts related to the documents and terms. LSA assumes that words that are close in meaning will occur in similar pieces of text (the distributional hypothesis).

The first step is to represent the text as a matrix. And that matrix containing word counts(frequency) per document (rows represent unique words and

columns represent each document) is constructed from a large piece of text and a mathematical technique called singular value decomposition (SVD) is used to reduce the number of rows while preserving the similarity structure among columns.The cell entries are subjected to a preliminary transformation in which each cell frequency is weighted by a function that expresses both the word's importance in the particular document and the degree to which the word type carries information in the domain of discourse.

LSA applies SVD to the matrix.In SVD, a rectangular matrix is decomposed into the product of three other matrices. First component matrix describes the original row entities as vectors of derived orthogonal factor values, another describes the original column entities in the same way, and the third is a diagonal matrix containing scaling values such that when the three components are matrix-multiplied, the original matrix is reconstructed.



Documents are then compared by taking the cosine of the angle between the two vectors (or the dot product between the normalizations of the two vectors) formed by any two columns. Values close to 1 represent very similar documents while values close to 0 represent very dissimilar documents.Finally, the measure of similarity computed in the reduced dimensional space.

5.2 Explicit Semantic Analysis

Explicit semantic analysis (ESA) is a vectoral representation of text (individual words or entire documents) that uses a document corpus(usually Wikipedia) as a knowledge base. Specifically, in ESA, a word is represented as a column vector in the tf–idf matrix of the text corpus and a document (string of words) is represented as the centroid of the vectors representing its words. Explicit Semantic Analysis (ESA) represents the meaning of texts in a high-dimensional space of concepts derived from Wikipedia. We fetched documents from Wikipedia that are related to the title of the carnfield docs and use it to represent our document and query matrix.

To perform the basic variant of ESA, one starts with a collection of texts, say, all Wikipedia articles; let the number of documents in the collection be N. These are all turned into "bags of words", i.e., term frequency histograms,

stored in an inverted index. Using this inverted index, one can find for any word the set of Wikipedia articles containing this word "each word appearing in the Wikipedia corpus can be seen as triggering each of the concepts it points to in the inverted index". However, keeping into account the restrictions on computation power, we are using the cranfield documents titles instead of using the whole document text.

The output of the inverted index for a single word query is a list of indexed documents (Wikipedia articles), each given a score depending on how often the word in question occurred in them (weighted by the total number of words in the document). Mathematically, this list is an N-dimensional vector of word-document scores, where a document not containing the query word has score zero. To compute the relatedness of two words, one compares the vectors (say u and v) by computing the cosine similarity.

$$\mathsf{sim}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|} = \frac{\sum_{i=1}^{N} u_i v_i}{\sqrt{\sum_{i=1}^{N} u_i^2}\sqrt{\sum_{i=1}^{N} v_i^2}}$$

and this gives numeric estimate of the semantic relatedness of the words. The scheme is extended from single words to multi-word texts by simply summing the vectors of all words in the text.

5.3 Vectorizer

Word Embeddings or Word vectorization is a methodology in NLP to map words or phrases from vocabulary to a corresponding vector of real numbers which used to find word predictions, word similarities/semantics. The process of converting words into numbers are called Vectorization. In our project , we used three types of vectorizer: 1) TF-IDF 2) BOW (Bag of Word) 3) Word2Vec

5.3.1 TF-IDF

Term frequency-inverse document frequency (TF-IDF) gives a measure that takes the importance of a word into consideration depending on how frequently it occurs in a document and a corpus.

Term frequency denotes the frequency of a word in a document. For a specified word, it is defined as the ratio of the number of times a word appears in a document to the total number of words in the document. Or, it is also defined in the following manner:It is the percentage of the number of times a word (x) occurs in a particular document (y) divided by the total number of words in that document.

$$tf(term) = \frac{n}{N}$$

n = Number of times terms appears in a documents

N = Total number of items in the document

Inverse Document Frequency measures the importance of the word in the corpus. It measures how common a particular word is across all the documents in the corpus. It is the logarithmic ratio of no. of total documents to no. of a document with a particular word.

$$idf(terms) = log\frac{N}{n}$$

N = Total number of Documents

n = Number of Documents with term in it

This term of the equation helps in pulling out the rare words since the value of the product is maximum when both the terms are maximum. What does that mean? If a word appears multiple times across many documents, then the denominator df will increase, reducing the value of the second term.

$$TF - IDF(terms) = tf(terms) * idf(terms)$$

The product of TF x IDF of a word indicates how often the token is found in the document and how unique the token is to the whole entire corpus of documents.

5.3.2 BOW - Bag of Words

BOW is a text vectorization model commonly useful in document representation method in the field of information retrieval. In information retrieval, the BOW model assumes that for a document, it ignores its word order, grammar, syntax and other factors, and treats it as a collection of several words and uses a set of unordered words to express a text or a document. The appearance of each word in the document is independent of whether other words appear.

This vectorization technique converts the text content to numerical feature vectors. Bag of Words takes a document from a corpus and converts it into a numeric vector by mapping each document word to a feature vector for the machine learning model.

There is some limitations of BOW:

- This method doesn't preserve the word order
- It does not allow to draw of useful inferences for downstream NLP tasks

5.3.3 Word2Vec

Word2Vec — Word representations in Vector Space.

With Bag of Words and TF-IDF text vectorization techniques we did not get semantic meaning of words. But for most of the applications of NLP tasks like sentiment classification, information retrieval system, etc require semantic meaning of a word and semantic relationships of a word with other words. Word

embeddings captures semantic and syntactic relationships between words and also the context of words in a document. Word2vec technique used to implement word embeddings.

Word2vec model takes input as a large size of corpus and produces output to vector space. This vector space size may be in hundred of dimensionality. Each word vector will be placed on this vector space. In vector space words that share context commonly in a corpus are closer to each other. This works with similar words that tend to close with words that can have multiple degrees of similarity. Word vector having positions of corresponding words in a vector space. The Word2Vec method learns all those types of relationships of words while building a model.

In order to train with a larger dataset with complex models, the modern techniques use a neural network architecture to train complex data models and outperforms for huge datasets with billions of words and with millions of words vocabulary.

This technique helps to measure the quality of the resulting vector representations. Perhaps, one of the most famous examples of Word2Vec is the following expression:

$$\text{king} - \text{man} + \text{woman} = \text{queen}$$

Since every word is represented as an n-dimensional vector, one can imagine that all of the words are mapped to this n-dimensional space in such a manner that words having similar meanings exist in close proximity to one another in this hyperspace.

6 Experiments

We have entirely used the Cranfield dataset for our building and evaluation purposes. There are essentially three files in the dataset:

- **cran_docs** - which has the list of documents along with their ID.
- **cran_queries** - which has a list of queries along with their ID.
- **cran_qrels** - which has the relevance scores of the relevant documents for each query

The first step is to preprocess the data i.e., both documents as well as the queries. The steps done while preprocessing are:

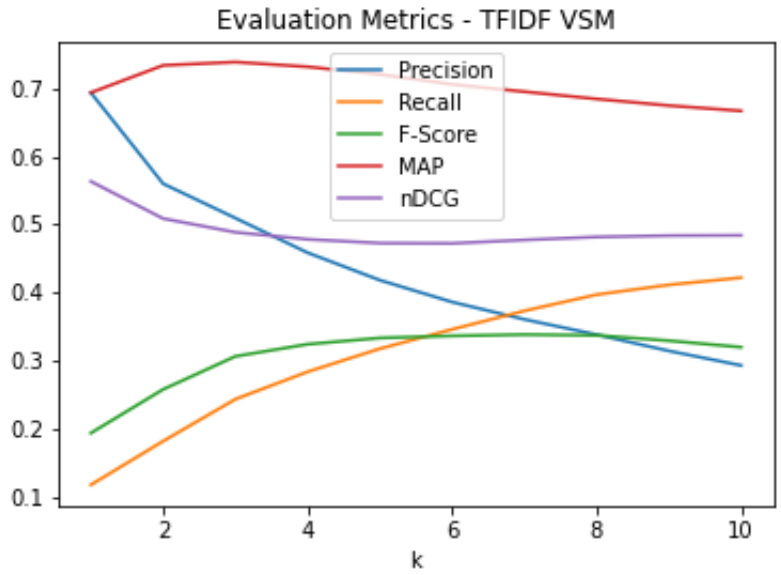
- **Sentence Segmentation** - essentially the process of breaking down a given document/text into individual processing units (called sentences) consisting of typically more than one word.
- **Tokenization** - Tokenization is the process of breaking up the existing sequence of the characters in a text by locating the word boundaries.

- **Inflection Reduction** - We used lemmatization here which is an intelligent and more sophisticated way of getting to the root form of the given word.
- **Stopword Removal** - Here we remove the useless words which often occur many times but dont add much value to our text.

6.1 Vector Space Model (VSM)

We built a simple vector space model by first vectorizing our documents using the TFIDF Vectorizer provided by sklearn library. After vectorizing, we built a simple document-term matrix and use this as index for computing the similarity values for the queries and documents. Then we evaluate our model using the metrics Precision, Recall, Fscore, Mean nDCG, Mean Average Precision at different rank values (till k = 10)

The result obtained is plotted as below:



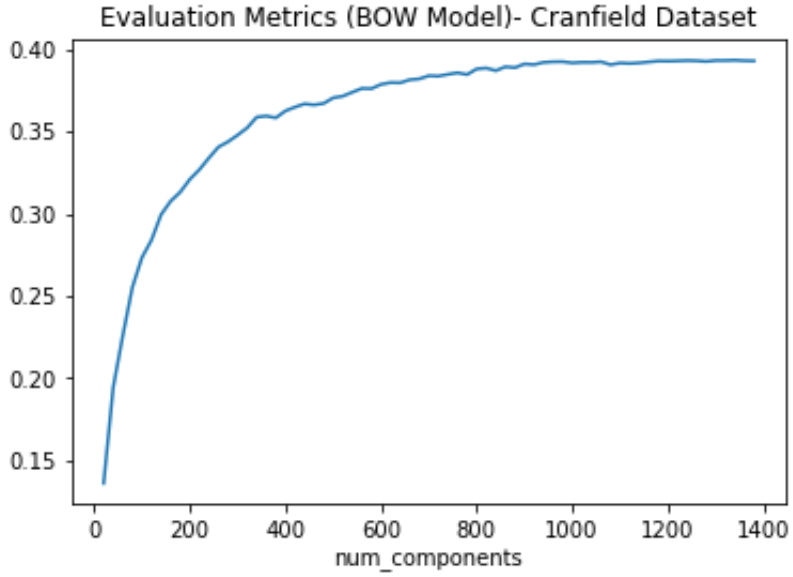
6.2 Latent semantic Analysis (LSA)

6.2.1 Using BOW Vectorizer

Here, we vectorized our documents using a simple Bag Of Words vectorizer provided by sklearn. Then we got a simple document-term matrix. Then we used TruncatedSvd class of sklearn library to perform SVD to get the reduced document term matrix with latent dimensions.

But here, we have a hyperparameter i.e., n_components i.e., number of latent dimensions that we want from LSA. To tune this hyperparameter, we used

kindof a Grid Search technique and ran all possible values and picked the best one. We started with n_components = 2 and did till 1400 (since number of documents). We used nDCG score as a metric to pick the best hyperparameter. The results are shown in the plot below:



As we can see from the above plot, the best value of n_components is 920.

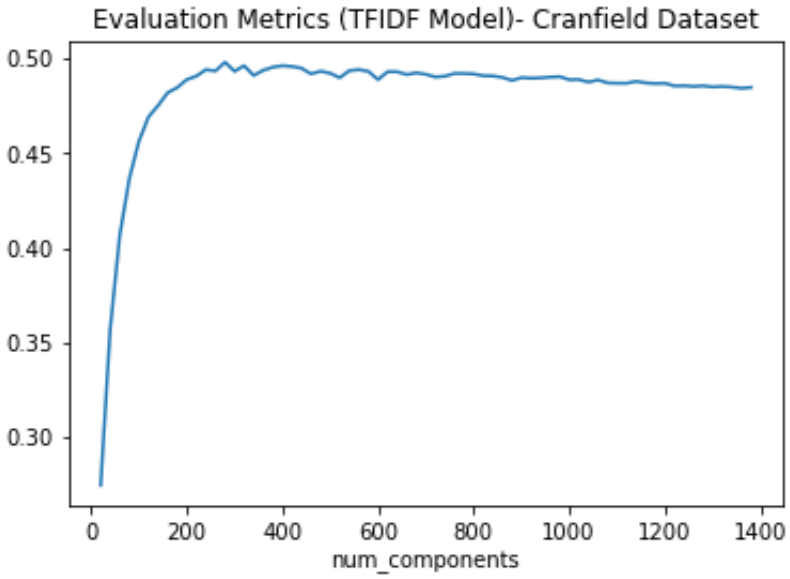
Therefore, using this, we built our index for computing the similarity values for the queries and documents. Then we evaluate our model using the metrics Precision, Recall, Fscore, Mean nDCG, Mean Average Precision at different rank values (till k = 10).

The result obtained is plotted as below:

6.2.2 Using TFIDF Vectorizer

Here, we vectorized our documents using a TFIDF vectorizer provided by sklearn. Then we got a simple document-term matrix which has tfidf weights. Then we used TruncatedSvd class of sklearn library to perform SVD to get the reduced document term matrix with latent dimensions.

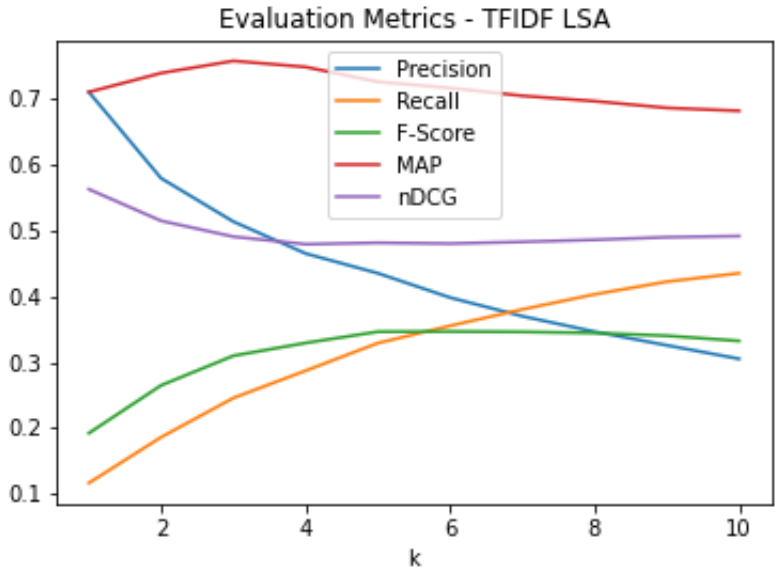
But here, again we have the hyperparameter i.e., n_components i.e., number of latent dimensions that we want from LSA. To tune this hyperparameter, we used kindof a Grid Search technique and ran all possible values and picked the best one. We started with n_components = 2 and did till 1400 (since number of documents). We used nDCG score as a metric to pick the best hyperparameter. The results are shown in the plot below:



As we can see from the above plot, the best value of n_components is 280.

Therefore, using this, we built our index for computing the similarity values for the queries and documents. Then we evaluate our model using the metrics Precision, Recall, Fscore, Mean nDCG, Mean Average Precision at different rank values (till k = 10).

The result obtained is plotted as below:



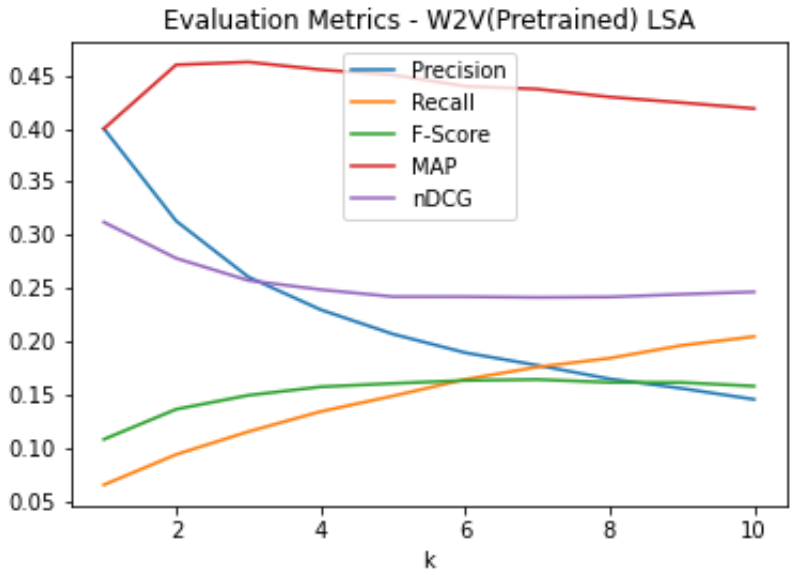
6.2.3 Using Pretrained Word2Vec Model

Here, we downloaded a pretrained word2vec model online and got the glove vectors pertaining to the train dataset used by the pretrained model. GloVe is actually an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

Then, for each sentence, we get a vector using the glove vectors and collecting all these vectors gives us the document-term matrix. Here we restricted ourselves to length 300 for vectors inorder to make the process computationally quicker.

Then, using this, we built our index for computing the similarity values for the queries and documents. Then we evaluate our model using the metrics Precision, Recall, Fscore, Mean nDCG, Mean Average Precision at different rank values (till k = 10).

The result obtained is plotted as below:



6.2.4 Using trained Word2Vec Model

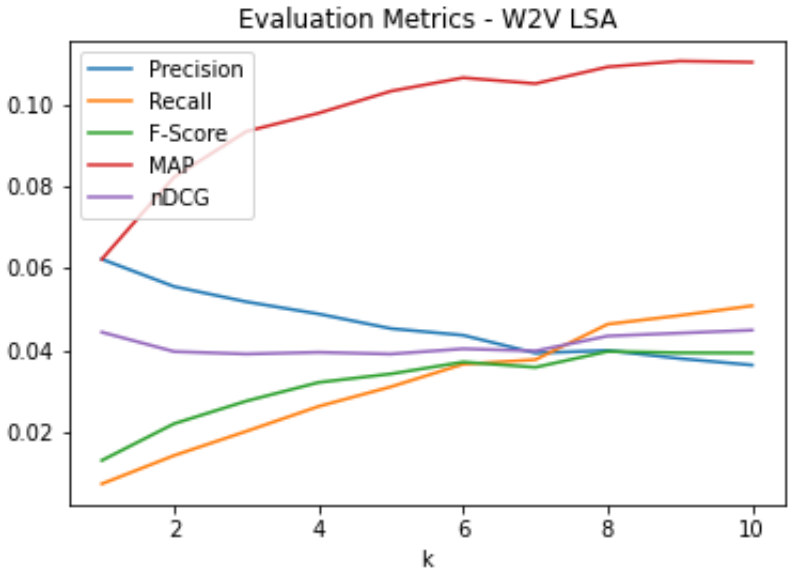
Here, instead of using a pretrained models, we used the Cranfield dataset to ourselves train the Word2Vec CBOW model. Using this trained model, we convert each of our document and query to a vector and obtain a simple document-term matrix.

Here we restricted ourselves to length 50 for vectors inorder to make the process computationally quicker.

Then, using this, we built our index for computing the similarity values for the queries and documents. Then we evaluate our model using the metrics

Precision, Recall, Fscore, Mean nDCG, Mean Average Precision at different rank values (till k = 10).

The result obtained is plotted as below:



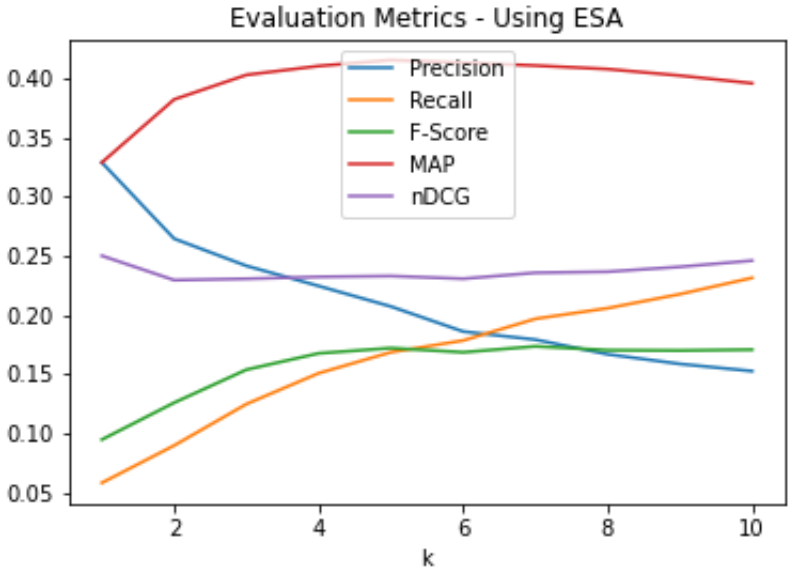
6.3 Explicit Semantic Analysis

We first install pymediawiki library to get the various wikipedia articles. Then we collect the document titles from Cranfield dataset and preprocess them using the same previous steps.

Then for each title, we collect the wikipedia articles that are similar to the title. We collect 5 such wikipedia articles for each document title. Then we convert these collected articles into TFIDF document-term matrix.

Then, using this, we built our index for computing the similarity values for the queries and documents. Then we evaluate our model using the metrics Precision, Recall, Fscore, Mean nDCG, Mean Average Precision at different rank values (till k = 10).

The result obtained is plotted as below:



7 Result

Model @ k =10	Precision	Recall	F-score	MAP	nDCG
VSM	0.2937	0.4223	0.3205	0.6665	0.4843
BOW-LSA	0.2128	0.3065	0.2324	0.5155	0.3417
TFIDF-LSA	0.3071	0.4386	0.3346	0.6828	0.4943
Pretrained W2V-LSA	0.3133	0.0940	0.136	0.46	0.2782
W2V-LSA	0.0324	0.0438	0.0346	0.0763	0.0357
ESA	0.1524	0.2313	0.1705	0.3958	0.2458

In the above table, green cells indicate the model which has the highest corresponding metric. As we can see from the above table, some of our models have worked as expected but remaining have given contradictory results.

Some observations based on the above conducted experiments are:

- Bag of Words model is not quite good even along with LSA as it doesnt consider anything while vectorizing except the presence or absence of a word.
- TFIDF vectorizing along with LSA works best across all metrics (except

precision but not by much) giving a much visible improvement as compared to the simple VSM model.

- Using Pretrained Word2Vec is not too good (just reasonable). Its quite obvious also as the pretrained model is trained on regular english sentences. Whereas our data i.e., Cranfield dataset contains a majority of scientific documents and queries. Maybe after finetuning our pretrained model, we may achieve greater performance. But nonetheless, this model gives the best precision but extremely bad recall. This means that for small amounts of retrieval, this model can be used.
- Training Word2Vec model from scratch is the worst and understandbly so. Since Word2Vec is a neural network model, it requires high amount of data to train and learn. Whereas our dataset has just 1400 documents which is too less. Therefore, this model performs the worst.
- ESA model also didnt perform as well as expected. So, its best to avoid this model for our dataset as it takes a lot of time without any significant increase in performance.

8 Conclusion

Hence, we can conclude that our initial hypothesis is correct that removing the orthogonal assumption of words and modeling relationships between words and also by considering the context of the words, we may be able to improve the accuracy of our system.

Modelling with LSA does better that VSM on the task of information retrieval for the evaluation measure nDCG on Cranfield Dataset.

9 Reference

- https://towardsdatascience.com/latent-semantic-analysis-sentiment-classification-with-python-5f657346f6a3
- https://www.datacamp.com/community/tutorials/discovering-hidden-topics-python
- https://medium.com/betacom/latent-semantic-indexing-in-python-85880414b4de
- https://www.kaggle.com/code/shivam1600/simple-information-retrieval-using-tf-idf-and-lsa/notebook
- https://machinelearninggeek.com/latent-semantic-indexing-using-scikit-learn/
- https://www.analyticsvidhya.com/blog/2021/06/part-5-step-by-step-guide-to-master-nlp-text-vectorization-approaches/
- https://towardsdatascience.com/word2vec-explained-49c52b4ccb71