CS6370 Natural Language Processing Project

Chakirala Sai Prasanna, CS20B019
Nayani Chetana, CS20B055

May 10, 2023

Abstract

To improve the search engine built on Cranfield Dataset by addressing the current limitations of vector space model for unigrams. We experiment with various techniques like bigrams, LSA, Glasgow weighting model, BM25, spell-check, and LDA for this purpose.

Introduction

The problem statement is to build an Information Retrieval System on the Cranfield dataset. The dataset contains 1400 documents and 225 queries. These queries are used to retrieve the relevant documents. Each query is also given a set of relevant documents and relevancy scores, which can be used in nDCG calculation. We intend to address several shortcomings of the vector space model, like polysemy, synonymy, order of terms and more, by experimenting with other language modelling methods. We evaluate these methods on evaluation measures of MAP (Mean Average Precision), F-Score, nDCG and Recall.

Baseline

The following figure (Fig 1) is a plot evaluated for the vector space model using the tf-idf weighting scheme for unigrams and using stemming, starting with MAP 0.608 and reaching the highest value of 0.669 @ k = 3, with nDCG averaging at 0.42 and F-Score at 0.3. The aim of this project is to improve using this as a baseline model.

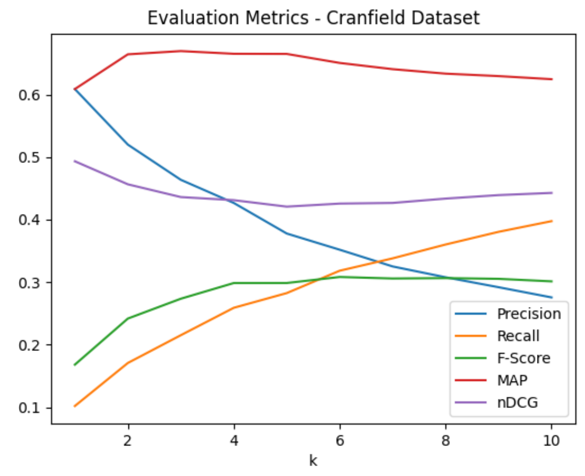


Fig 1.

Methodology

Below are the methods we experimented with in order to improve our information retrieval system.

Bigrams

In a vector space model built using unigrams, each word is considered a dimension. Documents and queries are therefore represented as vectors in these dimensions, and cosine similarity is used to measure similarity. But using unigrams does not capture the order of terms or the context. To address this, we use bigrams, wherein we consider two adjacent words as a dimension. Using bigrams, we can potentially capture more complex relationships between words, such as idiomatic expressions and phrasal verbs.

Latent Semantic Analysis

Vector space model uses each word as a dimension, and all words are taken as orthogonal to each other, but this is rarely true. Words share similarities with other words, i.e., synonymy. This needs us to capture the semantic relations between words. Latent Semantic Analysis is one technique which can help us achieve this. It is an efficient way to find underlying concepts in a set of documents. We perform SVD (Singular Value Decomposition) of the term-document matrix while preserving relations between terms and documents. The singular value decomposition (SVD) of A is given by:

$$A = U\Sigma V^T$$

where A is an $m \times n$ matrix, U is an $m \times m$ orthogonal matrix, Σ is an $m \times n$ diagonal matrix, and V^T is an $n \times n$ orthogonal matrix. The diagonal entries of Σ are the singular values of A, and the columns of U and V^T are the left and right singular vectors of A, respectively. By selecting k singular values, we can construct a matrix in a lower dimension. Where $[:, 1 : k]$ denotes the first k columns of a matrix.

$$X' = U[:, 1 : k]\Sigma[1 : k, 1 : k]V[:, 1 : k]^T$$

Glawgow Weighting Model

We also experimented with a weighting model other than standard tf-idf. This is the Glasgow model presented in class.

$$w_{ij} = \frac{\log(freq_{ij} + 1)}{\log(length_j)} \bullet \log\left(\frac{N}{n_i}\right)$$

w_{ij} = tf•idf weight of term i in document j

$freq_{ij}$ = frequency of term i in document j

$length_j$ = number of unique terms in document j

N = number of documents in collection

n_i = number of documents term i occurs in

Okapi BM25

BM25 uses a probabilistic approach to ranking, taking into account the frequency of query terms in the document, the length of the document, and the average length of documents in the collection being searched.

$$score(d, Q) = \sum_{i=1}^{|Q|} IDF(q_i) \cdot \frac{f(q_i, d) \cdot (k_1 + 1)}{f(q_i, d) + k_1 \cdot (1 - b + b \cdot \frac{|d|}{avgdl})}$$

Where:
- d is the document being scored
- Q is the set of query terms
- $|Q|$ is the number of query terms
- $f(q_i, d)$ is the frequency of term q_i in document d
- $|d|$ is the length of document d in words
- $avgdl$ is the average document length in the collection being searched
- $IDF(q_i)$ is the inverse document frequency of term q_i, defined as
$IDF(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}$
where N is the total number of documents in the collection and $n(q_i)$ is the number of documents containing term q_i
- k_1 and b are tuning parameters that control the impact of term frequency and document length, respectively.

Spellcheck

Doing spell-check on the Cranfield data set would be a time-intensive task, more over Cranfield dataset doesn't have any spelling errors; hence there is no necessity to perform spell-check. But there is a possibility of a typing error in case of a custom query, in this case, we used TextBlob spell checker.

Latent Dirichlet Allocation

In LDA, each document is represented as a distribution over a fixed number of latent topics, and each topic is represented as a distribution over the words in the vocabulary. The goal of LDA is to learn the topic distributions and the word distributions that best explain the observed document collection.

Experiments and Results

Unigrams

Firstly, we sought to improve the results using naive unigrams and vector space model. We tried the following ways. Starting with adding the title to the document body, gave an improvement of 3% in MAP start value over not having the title in the body. Giving more weightage to the title may be a good idea, as the title captures the essence of the document. To decide what weightage to give to the title with respect to the body, we did parameter tuning to get the best values by giving the title 3 times more weightage than the body. This finally gave the following result - starting MAP at 0.6755, reaching the highest value of 0.7148 @ k = 3, with nDCG averaging at 0.44 and F-score at 0.32. Adding authors to the document body gave a very slight improvement in the results. We also checked this model with lemmatization to make sure that stemming gave better results and this was verified. Hence the best we could do with naive unigrams was starting MAP at 0.6755, reaching the highest value of 0.7159 @ k = 3, with nDCG averaging at 0.45 and F-score at 0.32.

Bigrams

We extended our code from assignment-2 to work for Bigrams. But this resulted in a steep degradation in the results and an increase in time taken as the number of dimensions is huge in case bigrams. Before implementing hybrid, we decided to switch to libraries like sklearn so as get better results. On experimenting with library functions for unigrams, we noticed considerable improvements in our results. We also observed a small vertical shift in nDCG and MAP curves. Evaluated VSM for unigrams with the help of libraries using stemming, starting with MAP 0.7111 and reaching the highest value of 0.7533 @ k = 2, nDCG averaging at 0.485 and F-Score at 0.337. On changing this code for bigrams, we noticed a decrease in performance, as seen in Fig 2. We concluded that this might be due to the fact that two words occurring together in a query might not have occurred in documents together as often in the Cranfield data set. Hence to capture the order of terms and at the same time capture more information, we built a hybrid model. After experimenting more, we found that not having the title in the body improved the results for the hybrid model, evaluation for the hybrid model with the help of libraries using stemming, started with MAP 0.7155 and reached the highest value of **0.7714** @ k = 3, nDCG averaging at 0.49 and F-Score at 0.337. Before we proceeded further, we also tried out lemmatisation for our hybrid model (see Fig 3). This gave slight improvement over the stemming with respect to MAP.

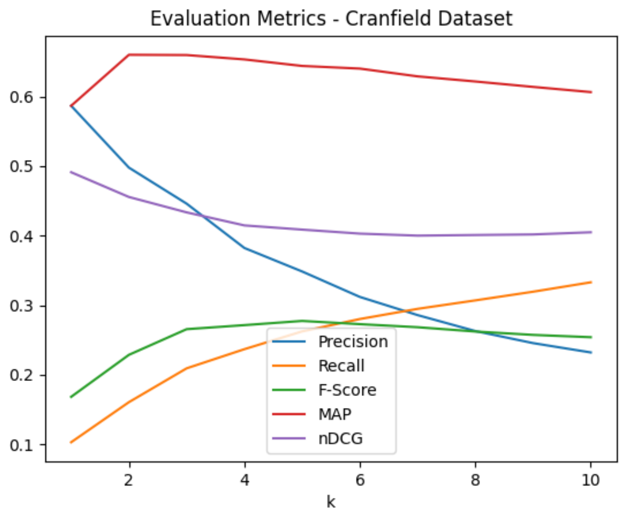


Fig 2. Evaluated for bigrams with the help of libraries using stemming, starting with MAP 0.604 and reaching the highest value of 0.6603 @ k = 3, nDCG averaging at 0.39 and F-Score at 0.25.

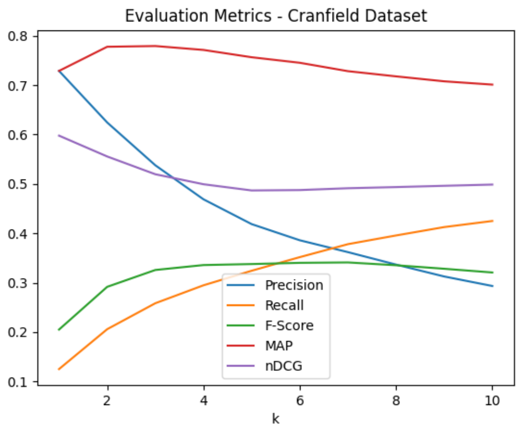
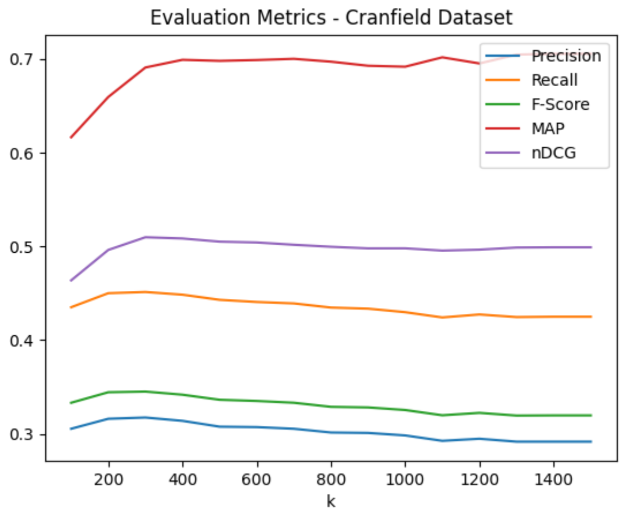


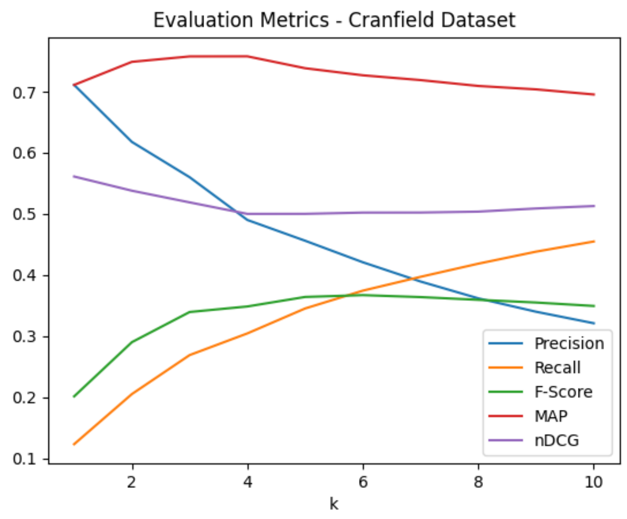
Fig 3.Evaluated for the hybrid model with the help of libraries using lemmatization, starting with MAP 0.728 and reaching the highest value of **0.77926** @ k = 3, nDCG averaging at 0.49 and F-Score at 0.337.

LSA

An important task at hand while using LSA is to find the value of c, i.e. the number of dimensions we would want to use to get maximum performance. To determine this, we used different values of c ranging from 100 to 1500 in intervals of 100, while calculating the measures @ k=10, for both stemming and lemmatization. We found a local maximum at 300 for nDCG, after plotting graphs, we found that all values were stagnating after c = 1400 as expected, as the number of documents is 1400, with the highest value at c = 1400. After more experimenting and rigorous analysis, we found an interesting observation, using c = 1350 starting value of MAP reached **0.733** the highest value of MAP, achieving a value of **0.7837** @k = 3 using lemmatisation also by adding title during LSA, we achieved MAP of **0.7903** @ k = 3. As we know, more concepts lead to more precision, while a lesser number of concepts gave a better nDCG. We chose c = 250 for better nDCG performance. While values using stemming stagnated at around 75%, those with lemmatization were 1% higher on average. In LSA, unigram, bigram and hybrid models were tested and found that the hybrid model worked the best. The curves in LSA are smoother than the curves that we got for bigrams, thereby providing slightly better performance overall.



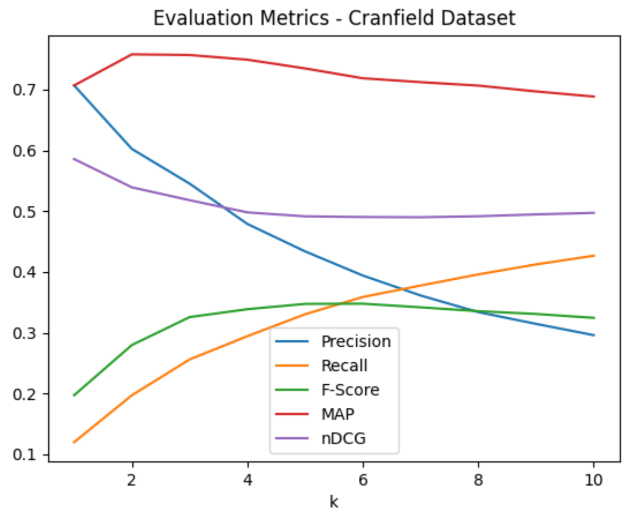
Experimenting with c-values to find the best number of concepts for LSA @ k = 10.



Evaluated using LSA with the help of libraries, starting with MAP **0.7022** and reaching the highest value of **0.7611** @k = 2 and reaching **nDCG = 0.513** at k = 10 and F score of 0.35 using lemmatization.

Okapi BM25

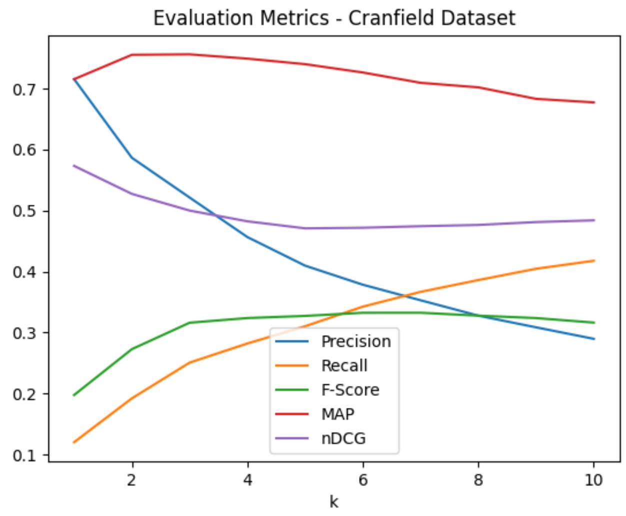
We directly used a Python library present in GitHub. This gave good enough results and was comparable to other models.



Evaluated for BM25 model, started at MAP 0.7066 and reached highest value of 0.7577 @ k = 2 and had an nDCG of 0.497 @ k = 10

Glasgow Model

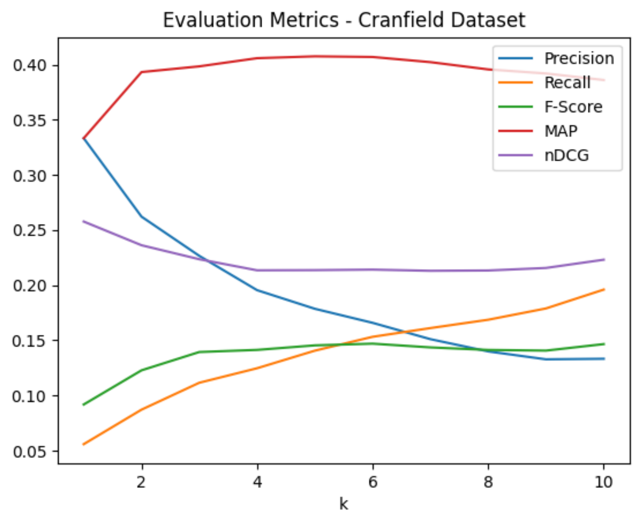
We used the Glasgow weighting model on LSA with c = 250 and used lemmatization, but this gave results inferior to VSM. After some more testing using c = 1350, we got results almost similar to the previous models. Here are the results we got.



Evaluated using LSA and Glasgow model, started with MAP 0.706 and reaching the highest value of 0.758 @k = 3 and reaching nDCG = 0.49 at k = 10 and F score of 0.32.

LDA

We also experimented with LDA, which did not give a satisfactory result. It performed worse within the hybrid model. While LDA can be a useful tool for topic modelling and content analysis, it was not the best approach for search engine applications, which require more specialized techniques for ranking documents based on their relevance to a query.



Spellcheck

We used Textblob spell checker. This is not a very efficient spellchecker, but it does the job for small errors, in terms of time complexity for handling custom queries.

Conclusions

- LSA performs better than VSM using 250 latent concepts in the retrieval of documents on evaluation measure nDCG over the Cranfield Dataset.
- Using hybrid model performs better than unigrams and only bigrams in vector space model.
- While stemming works better for VSM using unigrams, lemmatization works better for LSA.
- Using more concepts leads to higher MAP values in LSA.
- LDA is may not be a suitable tool for building a search engine.

References

We worked heavily with sklearn library, here are some of the resources used-
https://machinelearninggeek.com/latent-semantic-indexing-using-scikit-learn/
https://www.geeksforgeeks.org/python-textblob-correct-method/
https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
https://pypi.org/project/rank-bm25/
https://stackoverflow.com/questions/18424228/cosine-similarity-between-2-number-lists