NLP project

Satya Venkata Sai Teja P cs20b072[1,2*]
and Bhavadeep Bhukya cs20b015[2,3†]

*Corresponding author(s). E-mail(s): cs20b072@smail.iitm.ac.in;
Contributing authors: cs20b015@smail.iitm.ac.in;
†These authors contributed equally to this work.

Abstract

The idea of this project is to improve the basic vector space model built in the assignments . We used Spell check to correct query . Implemented LSA and two weighting schemes glasgow model and BM25 ranking algorithm. We used the following metrics : Precision, Recall, F1-score, MAP, ndcg to asses the performance of the model.

Keywords: LSA, Spellcheck, weighting, performance

1 Introduction

We aim to improve the model using following ideas:
1. **LSA**: We apply LSA to find relationships between terms, which accounts for the problems like polysemy and synonymy.
2. **Spellcheck**: we correct the spelling mistakes in the query given by the user.
3. **Using different weighting scheme** : Instead of using just generic TF-IDF we implemented Glasgow model and BM25 which aim to improve the performance.

2 LSA (Latent Semantic Analysis)

Latent Semantic Analysis (LSA) is a technique used in natural language processing and information retrieval to uncover the latent or underlying semantic structure in a collection of documents.
So we applied Latent Semantic Analysis (LSA) to the Cranfield dataset, with the idea that the technique can help uncover the latent semantic structure within the dataset

and provide insights into the relationships between scientific terms and documents. So we performed SVD on the term-document matrix and it got reduced into say K(k variable) dimensions or components, although the reduced dimensions are not easily interpretable, they represent the underlying semantic structure.
The value in the matrix represents the strength of the association of a term to a particular dimension. Hence terms with near values will have more text similarity, similar to documents.
For example the terms "lift" and "drag" will give high text similarity and correctly so because we are dealing with papers on aerodynamics mostly.

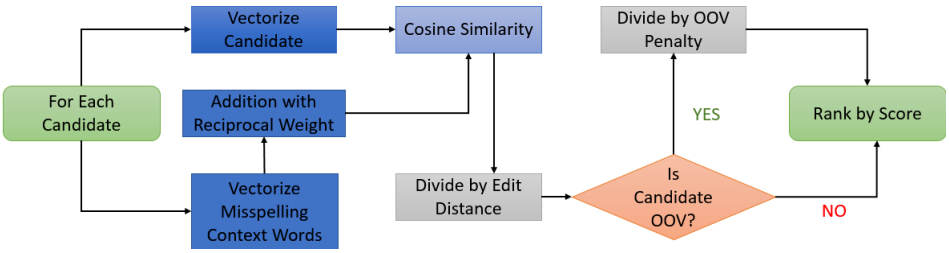
The math behind LSA is singular value decomposition of the term-document matrix, for a matrix A SVD is given by :

$$A = U\Sigma V^* \tag{1}$$

Where A is the term-document matrix of size n X m.
U represents left singular matrices of size n X r
Σ is a r X r diagonal matrix containing the singular values of the original matrix.
V^* is the transpose of V of size r X m which are the right singular vectors .
We take the top K vectors sorted by eigenvalues which are nothing but the diagonal entries of Σ we get the principal dimensions which will be used for the dimensionality reduction.
After this, we reduced the query vector to the same dimension and then calculated similarity using cosine similarity.

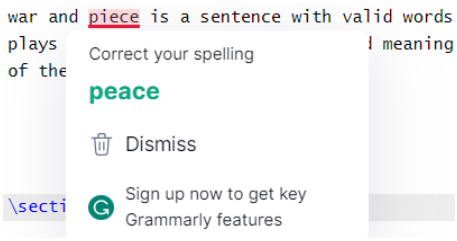
3 Spell check

contextual spell check :
Contextual spell check refers to the capability of a spell-checking system to consider the context of a word or phrase when determining whether it is spelled correctly or not. Unlike traditional spell checkers that simply compare words against a dictionary, contextual spell checkers take into account the surrounding words, grammar, and language patterns to provide more accurate suggestions for corrections.
In this aspect, we need not consider deep contexts, we can rather check before the word and next word i.e checking a window size of 1.



Example:

war and piece is a sentence with valid words. But **peace** plays a better role in giving a good meaning to the sentence instead of the **piece**.
Grammarly does this with good accuracy:



4 TF-IDF Model

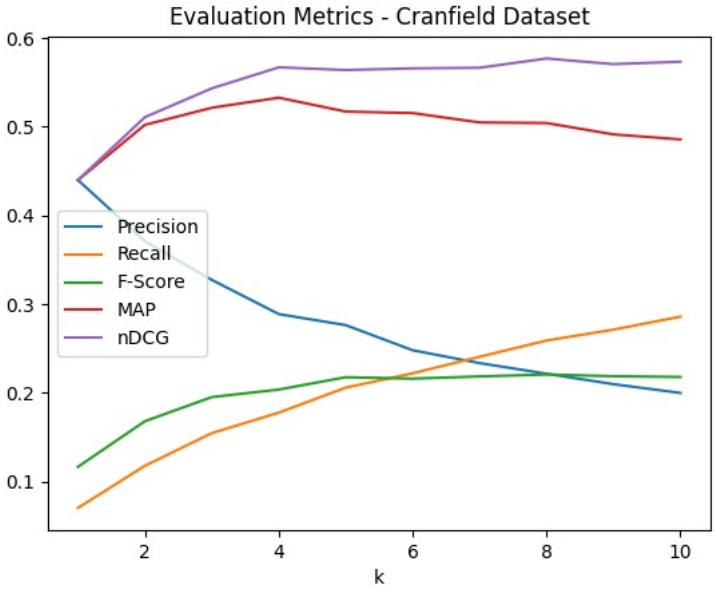
Glassgow model of term-document matrix

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

w_{ij} is the weight of term i in document j
$freq_{ij}$ is the frequency of term i in document j
$length_j$ is the number of unique terms in document j
N is the number of documents in collection
n_i is the number of documents term i appears in

Here log(N/n) gives inverse document frequency which gives some global information
This model is an extension to the regular tf-idf model and with some smoothing done to this.

Results with this model and spell check:



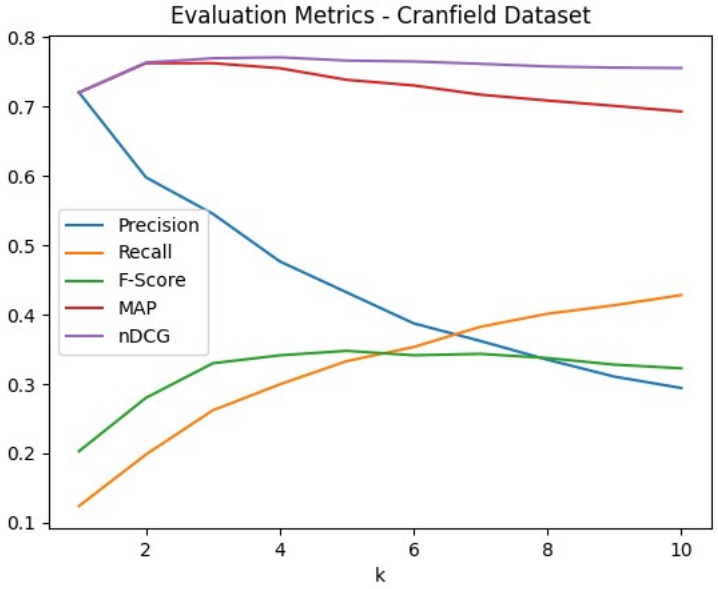
5 BM25

BM25, which stands for Best Match 25, is a bag-of-words retrieval function that ranks a set of documents based on the query terms appearing in each document, regardless of their proximity within the document.
The algorithm takes into account factors such as term frequency, inverse document frequency, and document length normalization to calculate the relevance score. Unlike the generic TF-IDF or Glasgow model it takes into consideration of document length. BM25 takes document length into account as part of its scoring algorithm to address the issue of document length bias. In information retrieval, longer documents may have a higher frequency of occurrence for a given query term simply because they contain more words. This can result in longer documents being ranked higher solely based on their length rather than their actual relevance to the query.
This allows shorter documents to compete more effectively against longer ones, ensuring that the ranking is based on the actual relevance of the document to the query rather than its length.

Method

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

$$\text{IDF}(q_i) = \ln\left(\frac{N-n(q_i)+0.5}{n(q_i)+0.5}+1\right)$$



6 Conclusions

Overall aspect for this project was focused on the performance part of the model , we can clearly see the improvements in every implementation, especially in the case of BM25.

The Model lacks in speed of search as it takes long time to fetch the documents for the queries. This problem can be solved using some techniques like : using small signature files or not too huge inverted files, computing subvectors (passage vectors) in long documents, not retrieving documents below a defined cosine threshold for fast evaluations.

Using spell check for documents : we intended to use spell check on all the documents during the preprocessing phase , upon doing it we found very little improvement but this is very rigorous task of preprocessing and generally scientific documents don't have many spelling errors , but this might work well with general documents

Below is the results after doing spell check on documents in dataset

```
Precision, Recall and F-score @ 1 : 0.4266666666666667, 0.07061909210414954, 0.11634518286502975
MAP, nDCG @ 1 : 0.4266666666666667, 0.4266666666666667
Precision, Recall and F-score @ 2 : 0.3377777777777778, 0.10528529503701912, 0.15161464459191987
MAP, nDCG @ 2 : 0.4888888888888889, 0.4992893291431742
Precision, Recall and F-score @ 3 : 0.31703703703703695, 0.14562070065221905, 0.18664145604539675
MAP, nDCG @ 3 : 0.5107407407407405, 0.5404546047577174
Precision, Recall and F-score @ 4 : 0.2811111111111111, 0.1709983032964886, 0.19776968001803347
MAP, nDCG @ 4 : 0.5077777777777777, 0.5414781475363014
Precision, Recall and F-score @ 5 : 0.2648888888888892, 0.19713985565643166, 0.2098062019376686
MAP, nDCG @ 5 : 0.5074753086419752, 0.5583176438056096
Precision, Recall and F-score @ 6 : 0.2466666666666665, 0.21787052803416293, 0.21411961726771603
MAP, nDCG @ 6 : 0.5012074074074077, 0.5629820444326218
Precision, Recall and F-score @ 7 : 0.2336507936507939, 0.24219557335920813, 0.21987548233670462
MAP, nDCG @ 7 : 0.4997456790123458, 0.573727962198569
Precision, Recall and F-score @ 8 : 0.22055555555555556, 0.26192835547704774, 0.22142525754674236
MAP, nDCG @ 8 : 0.4911931216931218, 0.5697507256058175
Precision, Recall and F-score @ 9 : 0.21283950617283967, 0.28001305122314707, 0.22417190139346366
MAP, nDCG @ 9 : 0.4857189090450997, 0.5763034888359452
Precision, Recall and F-score @ 10 : 0.20133333333333364, 0.29117212530381115, 0.22072435490997674
MAP, nDCG @ 10 : 0.4810311245485850, 0.5775920012599638
```

7 References:

http://en.wikipedia.org/wiki/Latent_semantic_indexing

https://en.wikipedia.org/wiki/Okapi_BM25

https://en.wikipedia.org/wiki/Tf%E2%80%93idf

https://www.kaggle.com/code/shivam1600/simple-information-retrieval-using-tf-idf-and-lsa

https://pypi.org/project/contextualSpellCheck/