CS6370: Course Project Report - Team 31*

Saarthak Sandip Marathe ME17B162[1] and Pragalbh Vashishtha MM19B012[1]

Indian Institute of Technology Madras, Chennai, Tamil Nadu, India - 600036

Abstract. This paper tackles the problem of Information Retrieval in the Toy Search engine problem. The goal of this project is to make information retrieval efficient through use of various Natural Language Processing methods. We use a baseline Vector Space Model (VSM) method to create the search engine. Multiple pre-processing techniques were tried and implemented to find the most optimum for our case. We tried addressing few of the limitations of VSM by trying out multiple improvement methods like LSA, Query expansion. For evaluation of these implementations, we used different metrics which are discussed in the report. Some future work into the improvement and reduction of retrieval time is also discussed.

Keywords: Information Retrieval · Search Engine · Query Expansion · Vector Space Model · LSA · Clustering

1 Introduction

The major dataset used here is the Cranfield dataset which is available publicly. To start, we shall list a few definitions which are the keystones for the project

2 Definitions

2.1 Vector Space Model

Vector space model refers to the representation of documents and queries as an array of numbers, called vectors. The model uses term frequency- inverse document frequency, **(tf-idf)**, as the measures in the vectors.
During retrieval, the cosine similarity, i.e, the cosine of the angle made between the document and the query is used for ranking the documents.

2.2 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. Higher order structure implicit in association of the terms with documents are utilized in this approach. We expect to tackle polysemy and synonymy using LSA.

2.3 TF-IDF

TF-IDF is a statistical measure that assesses the relevance of a word to a document in a set of documents. This is accomplished by multiplying two metrics: the number of times a word appears in a document and the word's inverse document frequency over a collection of documents. It has a variety of applications, the most important of which is automatic text analysis, as well as scoring words in machine learning algorithms for NLP.

2.4 Query Expansion

Over here, we expand on the abbreviations and other similar short forms used in the query to give a better query to work with.
For example: 'Company depends on their *PM* for improvement in the product' will become 'Company depends on their *Product Manager* for improvement in the product'

* Supported by Prof Sutanu and Indian Institute of Technology Madras. We would like to thank the TAs as well for their constant guidance.

2.5 Query auto-completion Method

We plan to use an LSTM based text-generation to predict the next k-words based on the context. The algorithm predicts the words based on the training data

3 Pre-processing

3.1 Sentence Segmentation

Here we divide the sentences into meaningful units which could be used further for other pre-processing tasks and model using

3.2 Normalization

All the uppercase words/letters are normalized and converted to lowercase. This will be useful for the further pre-processing that we do after this. For example: *Googles* become *googles.* The word essentially remains the same but if you do not convert them to lowercase, then the models (especially Vector Space Model) treats both words differently which in turn increase our dimensionality of vector.

3.3 Tokenization

Tokenization is the process of breaking down a large chunk of text into smaller tokens. Tokens can be words, characters, or subwords in this case. Tokenization can thus be divided into three categories: word, character, and subword (n-gram characters) tokenization. Here we used "Treebank tokenizer" from nltk package.

3.4 Stopword Removal

Different articles, prepositions, conjunctions and other similar words are removed. Here we remove all the punctuation marks as well. (', ", ;, :, /, ?, [,], |, etc)

3.5 Handling the numbers

We plan on removing the numbers from the given dataset given the less significance of the same and also so that we only deal with words in the following methods after it.

Another method to deal with the numbers could be to use the *word2num* model which would convert all word-written numbers to numerical values. For example: *He had seven cars* would become *He had 7 cars*

3.6 Inflection Removal

Both stemming and lemmatization are useful in this setting. Stemming is faster and easier to perform compared to lemmatization, hence initially stemming was used by search engines. However it can return mistaken words as caring gets cut to “car”, thus‘over-stemming’ can result in mistaken results.Stemming has a high recall and so does lemmatization. Lemmatization however uses morphological analysis and also achieves a higher precision. It is computationally intensive,but, with the increased computational power, it is not an issue to modern search engines.Hence lemmatization is ‘better’ than stemming for search engine applications for better results.

Overall the findings suggest that language modeling techniques improve document retrieval, with **lemmatization techniques producing the best result**.

4 Evaluation

The performance is evaluated by checking the following measures for the quality of the retrieved documents:-

- Precision@K:- Precision refers to the fraction of retrieved documents that are relevant to the query. Precision@K is evaluated at a given cut-off rank, considering only the topmost results returned by the system.
- Recall:- Recall is the fraction of the relevant documents that are successfully retrieved. Similar to precision, recall@k is evaluated at a given cut-off rank.
- MAP:- Mean Average Precision (MAP), which provides a single-figure measure of quality across recall levels. Among evaluation measures, MAP has been shown to have especially good discrimination and stability. For a single information need, Average Precision is the average of the precision value obtained for the set of top k documents existing after each relevant document is retrieved, and this value is then averaged over information needs. That is, if the set of relevant documents for an information need $q_j \in Q$ is $\{d_1, \dots d_{m_j}\}$ and R_{jk} is the set of ranked retrieval results from the top result until you get to document d_k, then

$$\text{MAP}(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} \text{Precision}(R_{jk})$$

- nDCG:-normalized discounted cumulative gain (NDCG). NDCG is designed for situations of non-binary notions of relevance (cf. Section 8.5.1). Like precision at k, it is evaluated over some number k of top search results. For a set of queries Q, let $R(j, d)$ be the relevance score assessors gave to document d for query j. Then,

$$\text{NDCG}(Q, k) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} Z_{kj} \sum_{m=1}^{k} \frac{2^{R(j,m)} - 1}{\log_2(1 + m)},$$

  where Z_{kj} is a normalization factor calculated to make it so that a perfect ranking's NDCG at k for query j is 1. For queries for which $k' < k$ documents are retrieved, the last summation is done up to k'.
- F-Score:- It is the harmonic mean of precision and recall, hence tackles the issues faced when using only one by combinig both into a single number

4.1 Ranking

Prepossessing is applied on the query as well to get the query vector. The similarity between documents and queries obtained by cosine similarity. Cosine similarity close to 1 signifies high similarity while close to 0 signifies low similarity

4.2 Evaluation Metrics

The retrieval of documents are measured by different metrics including but not limited to, Mean-Precision@k, Mean-Recall@k, MAP@k, nDCG@k, Mean F-Score@k

5 Vector Space Model (VSM)

5.1 Pre-processing

The pre-processing methods used here are:

- Segmentation
- Normalization
- Tokenization
- Stopword Removal
- Handling the numbers - using the first method (removing them all)
- Lemmatization

5.2 Problems faced initially

Previously, the model used to run TF-IDF functions over each sentences. This increased the overall run time and repeated vectorization. Same words which showed up repeatedly and frequently are given different vectors so they are interpreted wrongly. This creates issues while implementing TF-IDF method and thus interpretation going ahead. The final results were sub-par and needed to be improved.

5.3 Improvements in VSM

Changes in the TF-IDF model was done in batches so that we include the whole document and queries as a whole. Going forward, the TF-IDF functionalities worked well and thus the final results were good.

5.4 Results

– We run the VSM search engine over the Cranfield data and calculate the evaluation metrics as shown below:

k	**Precision**	**Recall**	**F-score**	**MAP**	**nDCG**
1	0.68	0.116	0.191	0.68	0.511
2	0.562	0.181	0.257	0.724	0.406
3	0.502	0.235	0.298	0.734	0.406
4	0.448	0.272	0.316	0.728	0.380
5	0.406	0.306	0.321	0.720	0.381
6	0.371	0.328	0.321	0.711	0.384
7	0.345	0.353	0.322	0.700	0.389
8	0.322	0.374	0.319	0.680	0.396
9	0.299	0.389	0.312	0.676	0.400
10	0.282	0.405	0.308	0.663	0.404

Table 1: Table of evaluation metrics - VSM

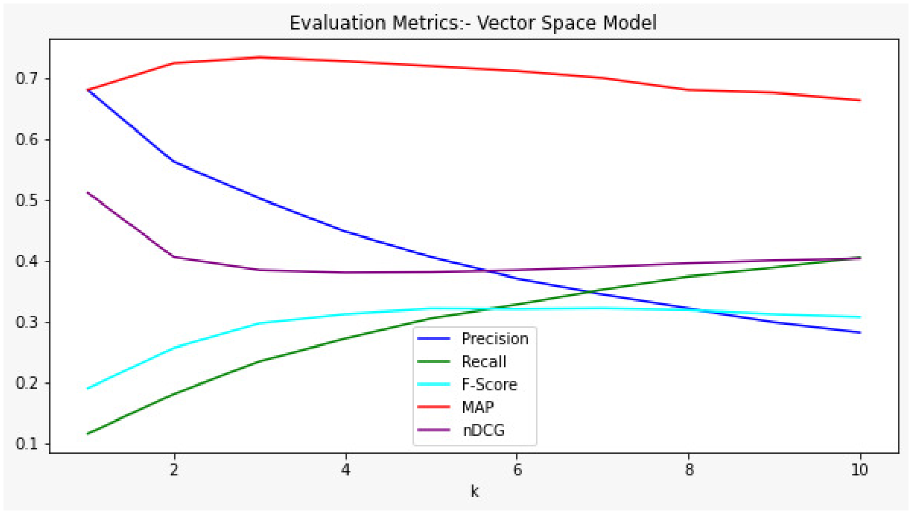


Fig. 1: Plots for the evaluation metrics - VSM

We see observe the following from the image shown:

– Precision value decreases with k values rapidly with a non-linear graph

- Recall values increases with k values. It starts from about 0.12 and increases upto 0.42 as expected
- F-score increases upto k=4 but then it remains almost constant (with a small dip) for the rest of the period
- MAP follows a similar graph as that of F-score where it reaches an asymptote after a point but it is offsetted by 0.15 values.
- nDCG decreases value decreases non-linearly till k=4 and then remains almost constant for the rest of the k values. nDCG curve shows that the IR system could perform better (because ideal nDCG equal to 1) it shows more or less constant variation as k is increased

5.5 Drawbacks and Limitations

- Long documents are poorly represented because they have poor similarity values, i.e., a small scalar product and a large dimensionality
- The sparsity of document vectors leads to poor similarity values between documents and queries.
- Semantic sensitivity; documents with similar context but different term vocabulary won't be associated. This is due to only first order similarity being used.
- The order in which the terms appear in the document is lost in the vector space representation.
- Theoretically assumed terms are statistically independent. This is due to orthogonality of dimensions in vector spaces
- Difficulty in processing long documents and causes problems in calculating cosine similarity with high dimensionality

We use the following methods to overcome the drawbacks

6 Latent Semantic Analysis

This tries to solve the 1 and 2 drawbacks faced. LSA assumes there exists a hidden semantic structure in the data that is partially obscured by the randomness of word choice with respect to retrieval. [7] **Singular value decomposition** is a statistical technique used to identify the latent structure and remove the arbitrary noise in the data. The reason why we implemented this:

- Original term-document matrix is presumed too large for the computing resources; in this case, the approximated low rank matrix is interpreted as an approximation
- Noisy original term-document matrix: for example, anecdotal instances of terms are to be eliminated. Denoisification needs to be done

6.1 Tuning the hyperparameters of LSA

We test the hyperparameters of LSA - number of latent variables over the evaluation metrics. We choose the hyperparameter which gives the best results. The figure showing the same is given below:

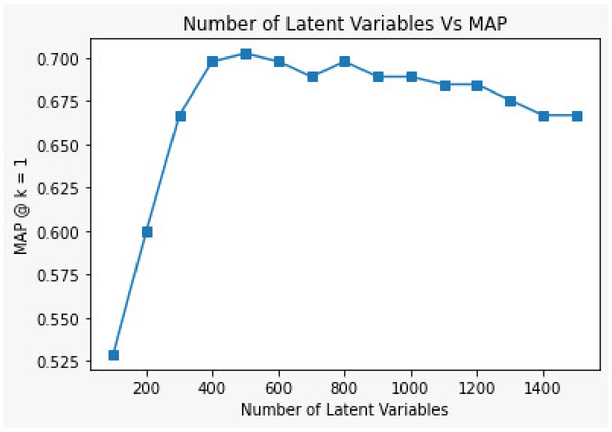


Fig. 2: Hypertuning LSA parameters

From this, we can see that the best evaluations happen at k=500.

6.2 Results of LSA

– We run the LSA search engine over the Cranfield data and calculate the evaluation metrics as shown below:

k	Precision	Recall	F-score	MAP	nDCG
1	0.697	0.116	0.191	0.698	0.523
2	0.573	0.185	0.264	0.74	0.417
3	0.510	0.242	0.306	0.742	0.398
4	0.463	0.289	0.329	0.728	0.394
5	0.415	0.315	0.331	0.719	0.393
6	0.380	0.341	0.331	0.702	0.398
7	0.361	0.373	0.339	0.686	0.408
8	0.332	0.388	0.330	0.680	0.408
9	0.311	0.401	0.324	0.671	0.413
10	0.292	0.417	0.318	0.663	0.417

Table 2: Table of evaluation metrics - LSA

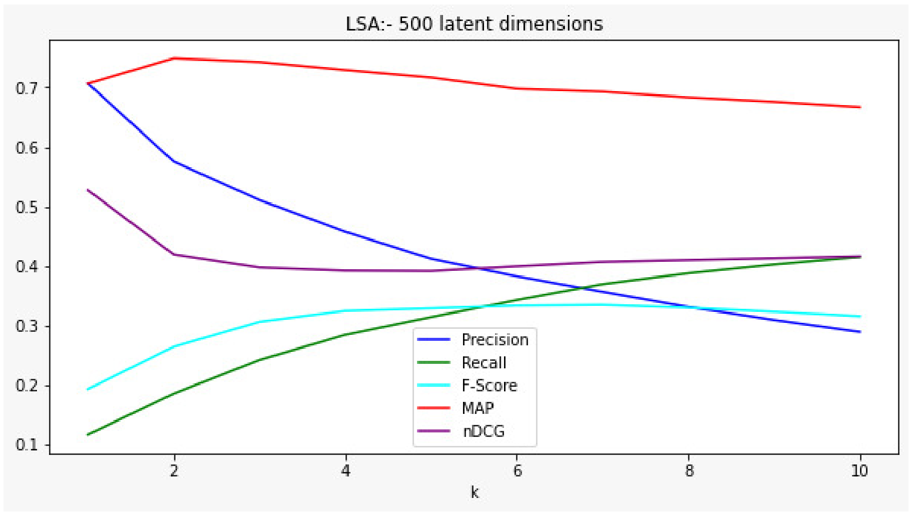


Fig. 3: Plots for the evaluation metrics - LSA

Observations for this graph are similar to those of VSM implementations.

6.3 Drawbacks of LSA

– The final metrics that we see in the results are not an improvement of the previous VSM
– It somewhat solves synonymy (words with similar meaning), but not polysemy (words with different meaning)
– Complexity of this model is high which is why it takes a lot of time

7 Query Expansion (QE) and Similar words

This method tries to solve the issue of the abbreviations used in the dataset and the queries. [5] Expanding the short forms and identifying similar words would help us in solving the problem faced in the VSM limitation of semantic sensitivity. We used word2vec model [2] to get similar words. We train the model on the corpus provided and thus find the similar words within the given dataset by the help of cosine similarity. Once the similarity is found, the word is appended in the phrase. Also, the abbreviated words transformations like *md* to *managing director* or ceo to *chief* *executive officer* are also done here.

After implementing this and comparing it with the baseline VSM, we didn't find significant change and improvement.

7.1 Results of QE

– We run the QE model search engine over the Cranfield data and calculate the evaluation metrics as shown below:

k	**Precision**	**Recall**	**F-score**	**MAP**	textbfnDCG
1	0.547	0.0926	0.152	0.547	0.391
2	0.482	0.157	0.223	0.611	0.334
3	0.416	0.199	0.250	0.616	0.319
4	0.372	0.234	0.265	0.613	0.318
5	0.340	0.258	0.270	0.609	0.318
6	0.304	0.274	0.266	0.604	0.318
7	0.281	0.291	0.264	0.600	0.321
8	0.263	0.306	0.261	0.594	0.325
9	0.246	0.320	0.257	0.589	0.330
10	0.234	0.335	0.254	0.576	0.335

Table 3: Table of evaluation metrics - QE

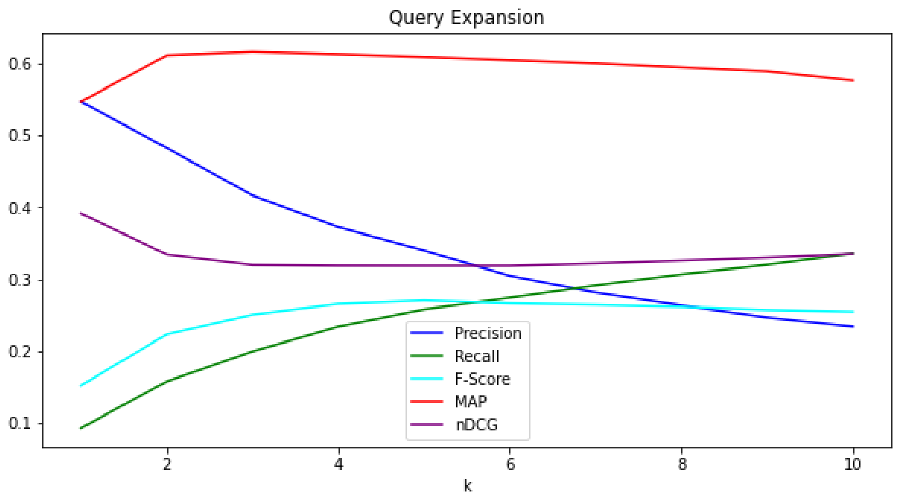


Fig. 4: Plots for the evaluation metrics - QE Model

– Observations for this model are worse than the VSM baseline model as the overall highest precision value itself is much lower than the other two models.
– The trends of this graph and the metrics are similar to the VSM model

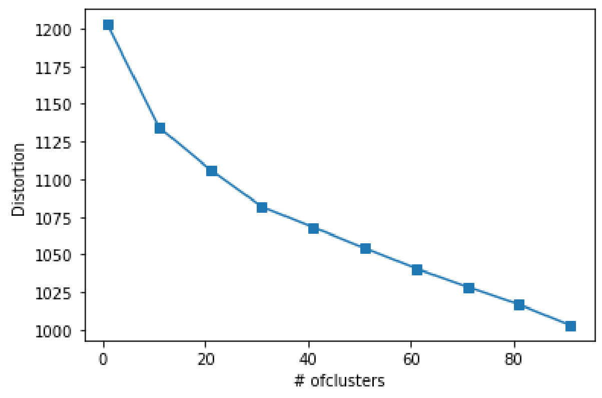
7.2 Drawbacks of QE

– The final metrics that we see in the results are not an improvement of the previous QE model
– It works well with few queries but worse for others. It works good when the consecutive words are highly correlated and expansion helps in that

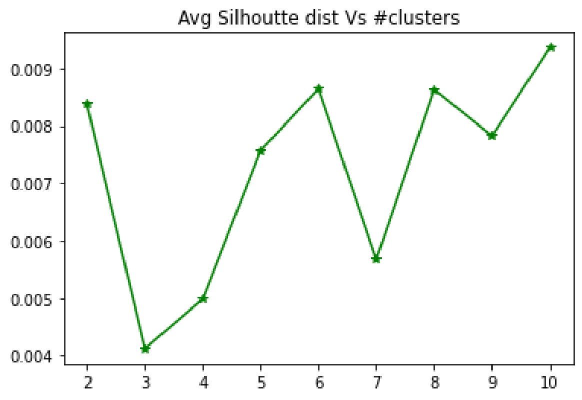
8 Clustering methods

8.1 K-Means Clustering

– The reason this is implemented is to reduce the latency time required for processing the various documents. [1]
– We group the TF-IDF matrix into 'k' clusters and based on the elbow values we see which is the optimum numbers



(a) K-means elbow



(b) K-means silhouette

Fig. 5: Graphs for K-Means Clustering

– As it is apparent from the graph, k= 6/8 is optimal number of clusters is 6/8.
– K-means clustering is optimized by the silhouette distance, which is empirical for the quality of the clustering
– For higher values, the trade-off is too much, but for 6 and 8 it is quite good. however the retrieval time of 8 is more than for 6 so 6 is chosen

8.1.1 Retrieval time comparison

– We compare the documentation retrieval time when clustering is used and when it is not used.
– We see that the time required when clustering is used is much lesser when compared to the time required to retrieve when clustering is not used
– We see a 50-70% decrease in the retrieval time

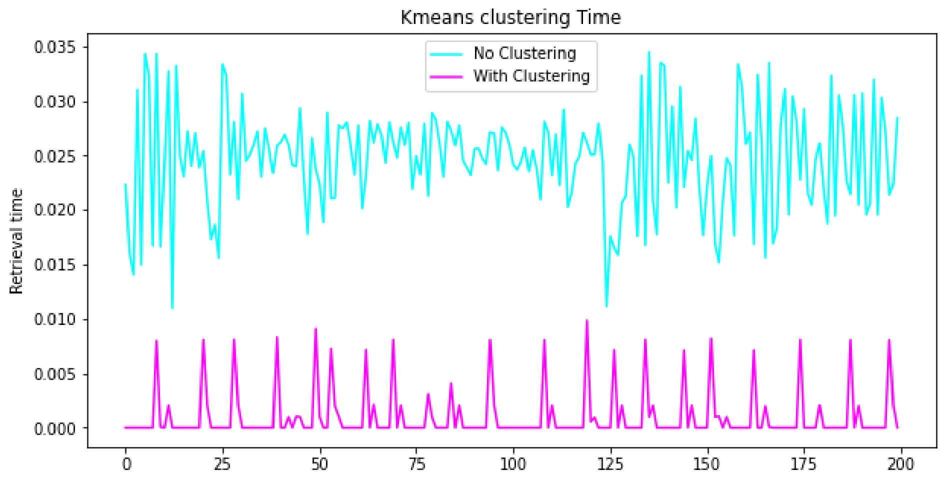


Fig. 6: Documentation retrieval time: K-Means and no clustering

8.2 LDA Method

Latent Dirichlet Allocation (LDA) [3] is similar to that of LSA as it focuses on similar topic clustering. Main difference between LSA and LDA is that LDA assumes that the distribution of topics in a document and the distribution of words in topics are Dirichlet distributions. [6]

We have two hyperparameters (alpha and beta) to control for topic and document similarities. A low alpha value will result in fewer topics being assigned to each document, whereas a high alpha value will have the reverse effect. A low beta value will use fewer words to model a topic, whereas a high one will use more words, resulting in themes that are more comparable.

The algorithm outputs the clustering for each of the document based on the topic. It first maps the documents in a space. Finds the centroid to the clusters [4] made and then tries to improve the clustering in each iteration with Dirichlet distributions approach. We saw that a cluster of 8 gave the best results.

8.2.1 Retrieval time comparison

– The results for this comparison are similar to the one we did for K-Means clustering
– But the time reductions are not as much as that of K-Means clustering

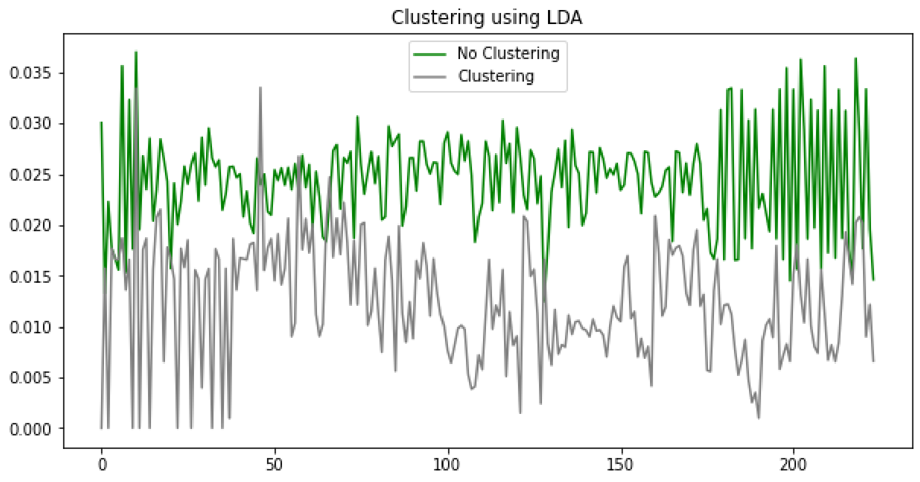


Fig. 7: Documentation retrieval time: LDA and no clustering

9 Inference and Summary

– During this project, we built a Vector Space Model which was used as a base for comparing it with the performance of the other methods we used to improve the system.
– During the implementation, we used extensive pre-processing techniques like segmentation, tokenization, normalization, inflection removal, handling numbers, stopword removal to make sure the further use of the data gives better results.
– We see that the LSA, QE modelling which were tried did overcome few of the drawbacks faced by the VSM implementation
– But, the results of LSA, QE were similar to that of VSM. We don't see drastic changes in the results
– The clustering methods were tried to reduce the time required to reduce the retrieval time. The K-means method worked better than the LDA modelling used
– From all the analysis, we understand the best implementation is highly dependent on the set of queries that we take into consideration. Each model and method works best in different scenarios as we saw through this report.

Bibliography

[1] "Wikipedia". *K-Means Clustering.* URL: `https://en.wikipedia.org/wiki/K-means_clustering`.

[2] "Wikipedia". *Word2Vec.* URL: `https://en.wikipedia.org/wiki/Word2vec`.

[3] David M Blei, Andrew Y Ng, and Michael I Jordan. "Latent dirichlet allocation". In: *the Journal of machine Learning research* 3 (2003), pp. 993–1022.

[4] Eui-Hong Sam Han and George Karypis. "Centroid-based document classification: Analysis and experimental results". In: *European conference on principles of data mining and knowledge discovery.* Springer. 2000, pp. 424–431.

[5] Saar Kuzi, Anna Shtok, and Oren Kurland. "Query expansion using word embeddings". In: *Proceedings of the 25th ACM international on conference on information and knowledge management.* 2016, pp. 1929–1932.

[6] "Edward Ma". *2 latent methods for dimension reduction and topic modeling.* URL: `https://towardsdatascience.com/2-latent-methods-for-dimension-reduction-and-topic-modeling-20ff6d7d547`.

[7] "Kushal Vala". *Latent Semantic Analysis: Distributional Semantics in NLP.* URL: `https://towardsdatascience.com/latent-semantic-analysis-distributional-semantics-in-nlp-ea84bf686b50`.