A Project Report

On

**Locality Sensitive Hashing**

BY

Under the supervision of

**Dr. Aruna Malapati**

**SUBMITTED IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS OF**

**CS F469: Information Retrieval**



**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI (RAJASTHAN)**

**HYDERABAD CAMPUS**

**(March 2023**)

**Group Details**

|  |  |
| --- | --- |
| **Name** | **ID no.** |
| Aaditya Mahesh Rathi | 2020A7PS2191H |
| Sankalp Kulkarni | 2020A7PS1097H |
| Akshat Oke | 2020A7PS0284H |
| Rishi Poddar | 2020A7PS1195H |

**ACKNOWLEDGEMENTS**

We are grateful to BITS Pilani for giving us this insightful opportunity of working on such an interesting course assignment. We would also like to thank the CSIS department for allowing us to work on a problem in the domain of Information Retrieval.

Finally, we would like to express my profound gratitude to Dr. Aruna Malapati for allowing us to work under her supervision and clearing our doubts throughout the duration of the course. This project would have been impossible without her help.

# Table of Contents

Contents

[Table of Contents 4](#_Toc129039069)

[1. Pipeline 5](#_Toc129039070)

[1.1 Data Pre-processing 5](#_Toc129039071)

[1.2 Document Expansion 5](#_Toc129039072)

[1.3 Tokenization and Stemming 5](#_Toc129039073)

[1.4 Index Construction 6](#_Toc129039074)

[1.5 Ranking 6](#_Toc129039075)

[2. References 8](#_Toc129039076)

[References 8](#_Toc129039077)

# Pipeline

## K- Shingles

First we experimented by forming character-wise shingles(k=9), but we were getting a lot of false positives .We decided to have word-wise trigram(k=3) shingles. Based on literature survey and experimentation we decided to take the value of k as 3.

## Pre-processing

During normalization special characters were removed and replaced with whitespaces. This choice was done because dataset had many tokens separated by special characters. The text was then converted into lowercase.

## Hashing

We used the min-hashing technique. Hash functions of the form (a\*x + b) % c were generated to get true permutations. This was ensured as the pairs (a, c) and (b, c) were taken as co-primes. These were chosen randomly so the results varied over multiple runs of the program.

## Index Construction

Our IR system supports free text queries and phrase queries. Based on our analysis of dataset, we decided that wild-card queries weren’t relevant. Also, computational cost for it wasn’t worth the benefit it would provide. We built two types of indexes. Inverted Index for normal free text queries and Bigram Index for phrase queries.  Based on our dataset,  we anticipated that most phrase queries would consist of two or three words, so we decided to use the bigram index. Since size of our dataset was so small, it could easily fit into the main memory, so we avoided using data structures like linked lists or B-trees, which would have slowed down our system with unnecessary memory references. Using such data structures is useful when the data doesn't fit entirely in the main memory and we need to fetch multiple blocks of the disk. We used Python lists to store the posting lists. Python arrays are just wrappers around C arrays and are more memory efficient, but Python lists are highly optimized and provide greater flexibility. They are recommended to be used in almost all cases other than where compatibility with C code is necessary. We used a dictionary to store terms and their corresponding posting lists.

## Ranking

We used BM25 algorithm for ranking the retrieved documents. TF-IDF penalizes document frequency and favors term frequency. It doesn’t consider the issue of term saturation. If a document contains a word 50 times and another one contains it 100 times, is it twice as relevant as the others? In most cases, both documents would be equally important. If document is shorter and contains the term then it is more likely to be relevant than a longer document that contains the term, as it is likely to mention multiple things. Below is the formula for the BM25 ranking function.

Text

Description automatically generated

b and k are hyperparameters here. b controls effect of length of document in its score. If b is larger, documents with longer length than average document length are penalized more (scored less). Literature indicates that value of b = 0.75 works well in most IR tasks. k1 controls term saturation. Higher k1 indicates that saturation value for term frequency is higher. Increase in score when term frequency grows beyond certain value will be minimal. Literature shows that k1 = 1.2 works well in most IR tasks. We used b = 1 and k1=0.5 for standard inverted index. For bigram index we used b=1 and k1=0.2. Since the document length in our case was less, the term frequency was also less, so we decided to use these values.

## 6 Time Analysis

|  |  |  |
| --- | --- | --- |
| Example Query | Number of documents retrieved | Time (in ms) |
| Earthquake explosions fire | 179 | 2.7818 |
| “contamination of property” | 213 | 3.96 |
| Radioactive contamination policy | 1276 | 10.6982 |
| Bodiliy injuries damages pay | 1665 | 12.0556 |
| Unidentified Automobiles accidents -death | 488 | 7.5362 |

The time taken is linear with respect to number of retrieved documents for every query term to find the documents and ranking the documents takes O(n logn) for the sorting algorithm.

\*Time varies depending on the load on the processor at that moment.

# References

# References

|  |  |
| --- | --- |
| [1] | W. Y. J. L. K. C. Rodrigo Nogueira, "Document Expansion by Query Prediction," 2019. |
| [2] | C. L. B. P. S. C. Thibault Formal, "SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval," 2021. |
| [3] | J. C. Zhuyun Dai, "Context-Aware Sentence/Passage Term Importance Estimation For First Stage Retrieval," 2019. |