IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

QUERY AND RETRIEVAL SYSTEM FOR TERRORISM DATABASE

From: Gaurav Dwivedi

Sricharan Sridhar

Vinayaka Srinivas

Abstract

Terrorism has always been a taboo subject and not much scaled and oriented information was always available over the web to the common people. The information which was available was always unorganized and not up to the mark. Through this paper, we are trying to create a database that retrieves the information of all the terrorist attacks such as the people involved (i.e. the leader and the terrorist group), the location and the date and time of each attack. The database created is in a form of search engine that uses retrieval algorithms like Positional Indexing, Inverted indexing, Signature files as well as clustering based algorithms to retrieve the specified query.

These retrieval algorithms are implemented over a dataset that has nearly 190,000 rows and 135 columns. After the preprocessing part, the main task was to make every single row of the database as a single document and then tokenize them so that we get relevant and specific information of all the data points present in the dataset. After breaking down the dataset, the proposed algorithms (mentioned above) are then implemented. During the process of implementation we broke down the dataset in 5000 documents so as for the smooth working of the python code according to the machine specifications.

After the coding phase the results and inferences were being drawn out. The main performance metric of our project are:

1. Precision
2. Recall
3. F1 score

Thus on the basis of these metrics the performance of the information retrieval system of this particular terrorism database was inferred.

Chapter 1

Introduction

Terrorism has been a persistent threat to global security for many years, and it has become an increasingly prevalent issue in today's world. The purpose of this research paper is to provide an introduction to the topic of terrorism and examine its different dimensions, causes, and effects on society.

The paper will start by providing a brief overview of the history of terrorism and how it has evolved over time. It will then examine the different types of terrorism, including religious terrorism, ethno-nationalist terrorism, and state terrorism, among others. The paper will also explore the root causes of terrorism, including political, economic, and social factors, and the ways in which these factors contribute to the rise of terrorist groups.

We introduce here a comprehensive global database, containing data on counterterrorist legislation in 219 countries and territories, and including over 1000 laws that were introduced between 1850 and 2009. The paper describes the dilemmas and difficulties in constructing a global terrorism database, including issues of standardized definitions and of data availability and reliability, and explains how the problem is being addressed while assembling the current database. It also describes the coding and searching procedures used during this process and the course of database creation and present descriptive statistics of the data, focusing on the historical development of global counterterrorist legislation and on the regional distribution of this legislation.

Moreover, the research paper will dive into the effects of terrorism on society, such as fear and panic, economic instability, and the erosion of civil liberties. The paper is derived from the database created on python which also used pyspark and spark libraries for a clearer analysis of data and a platform that uses a database containing the information of all the terrorist attacks since 1967 and so on. The dataset contained 190,000 rows and nearly 135 columns. The dataset was pre-processed and then broken down into 65000 rows and 63 columns. Each row in the dataset was converted into a single document and then the information retrieval techniques were applied to create the search engine and extract the goals of the project.

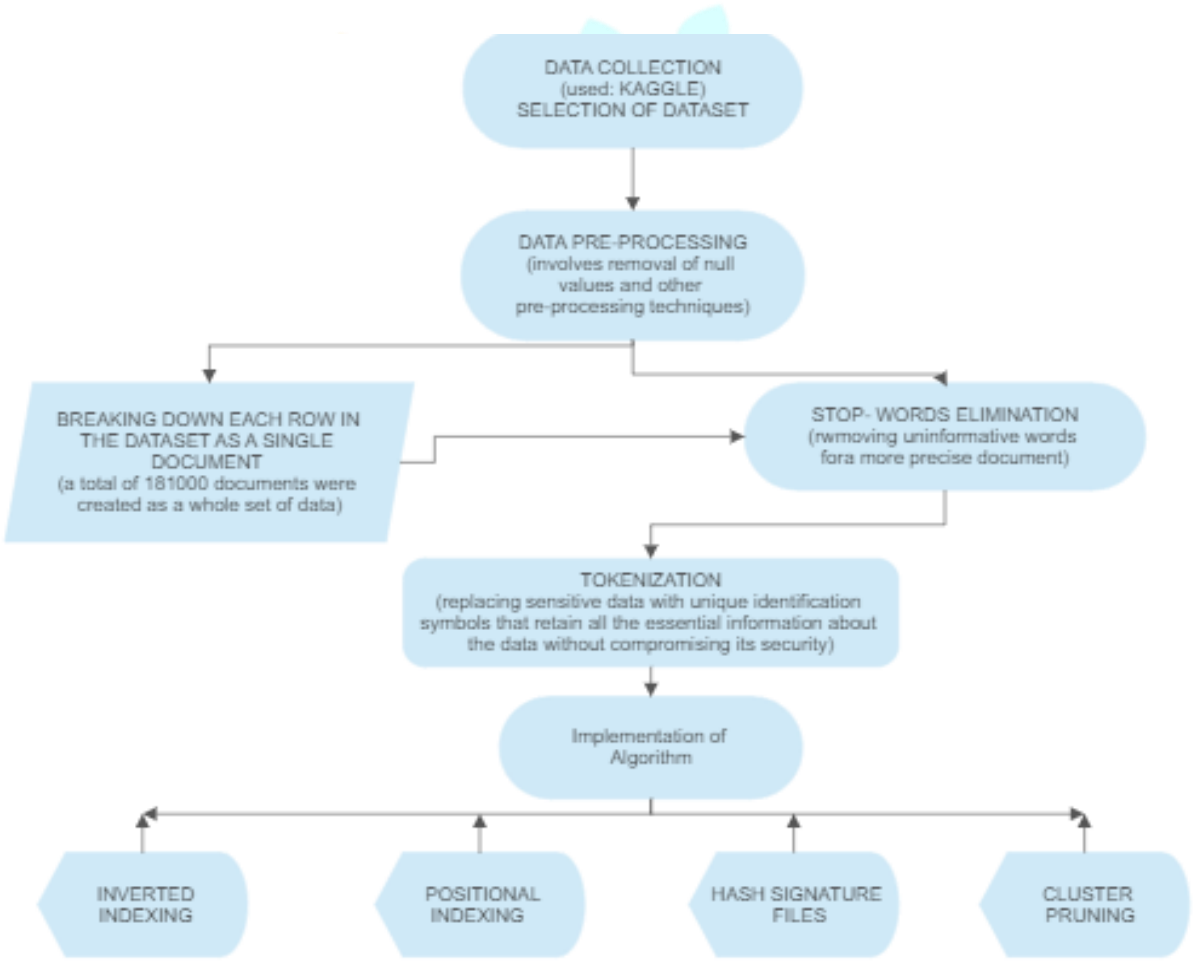
The coding part of the database creation involved the use of four Information Retrieval algorithms:

- Inverted Indexing
- Positional Indexing
- Hash-based Indexing
- Agglomerative Clustering

The performances of these algorithms are compared using the following performance metrics.

1. Precision
2. Recall
3. Accuracy score
4. F- score

The below flowchart will give a brief workflow of the overall working methodology of the project:



A data flow diagram that describes the steps and format of the research

Chapter 2

Methodology

Whenever dealing with large amounts of data, processing and analyzing it has always been a difficult task. Adding on to it the complexity of retrieving a specific data by sorting through the heap of data is truly time and computational power consuming. There have been previously many attempts at reducing the retrieval time of documents which have led to invention of various algorithms and techniques in the field of Information Retrieval and Organization. In this paper, we aimed to build a Terrorism Search Engine and to further compare the efficiency in computation and time of 4 different algorithms, namely: Inverted Indexing, Positional Indexing, Signature Files, and Cluster Pruning.

To build the database for the search engine, cleaning the dataset is imperative. Each of the given rows in the dataset (a csv file containing approximately 180k rows of data) was cleaned using various preprocessing techniques such as null value elimination, duplicate elimination along with other basic EDA procedures such as reducing the unnecessary columns and formatting the already present variables into necessary columns and data type.

2.1 Stop Word Elimination and Tokenization

With clean data in hand, the search engine building process begins with the elimination of stop words,which are uninformative components of the data, in the documents. Using the NLTK library available in Python. Once the stopwords are eliminated the focus shifts to tokenizing which is breaking the raw text into small chunks that breaks the raw text into words, sentences called tokens. These tokens help in understanding the context or developing the corpus of the search engine.

Once the database for the search engine was built choosing the right retrieving algorithm that is most efficient in terms of resource exhaustion has to be chosen.

2.2 Inverted Indexing

Inverted indexing is a technique used in information retrieval systems to efficiently store and retrieve information from large datasets. It is a process of reorganizing data to allow for faster searching and retrieval. In traditional indexing, data is indexed by its location or by the order in which it appears. In contrast, in inverted indexing, data is indexed by its content.

The basic idea behind inverted indexing is to create a lookup table that maps each unique term in a document to the document(s) that contain that term. This table is called an inverted index because it inverts the relationship between terms and documents. Instead of storing documents and looking up their contents to find relevant terms, inverted indexing allows for the direct lookup of terms and the corresponding documents that contain them.

To create an inverted index, a text corpus is first analyzed to identify and extract unique terms. These terms are then sorted and indexed by their frequency and location within the corpus. The resulting index can be used to quickly search for documents that contain specific terms or phrases.

One of the key advantages of inverted indexing is that it allows for faster search and retrieval of information from large datasets. This is because it eliminates the need to search through every document in a corpus to find relevant

information. Instead, the index can be used to quickly locate the documents that contain the desired terms, greatly reducing the search time.

Inverted indexing is used in a wide range of applications, including search engines, document management systems, and information retrieval systems. It is a powerful tool for efficiently storing and retrieving large amounts of text-based data, and has revolutionized the way we search for and access information.

2.3 Positional Indexing

Similar in nature to inverted indexing, positional indexing is traditional inverted index with modified entries of inverted list that include $f_{t,d}$ (frequency of term t in document d). With the additional knowledge the retrieval time is slightly decreased per retrieval which results in a significant difference in retrieval time when applied on a big dataset. Positional indexing is an advanced technique used in information retrieval systems that enhances the precision and relevance of search results. It is a type of indexing that takes into account not only the presence of keywords in a document but also their location and proximity to each other. This allows for more accurate and relevant search results.

In positional indexing, each occurrence of a keyword is assigned a unique identifier that includes the document ID, the position of the keyword within the document, and any other relevant metadata. This information is then stored in an inverted index, which allows for quick and efficient retrieval of documents based on specific keyword queries.

The use of positional indexing is particularly beneficial when searching for multi-word phrases or complex queries. By taking into account the relative position of each keyword in a phrase, positional indexing can accurately identify documents that contain the exact phrase or a close variation of it.

For example, if a user searches for "apple pie recipe," a positional index can identify documents that contain the exact phrase "apple pie recipe" as well as documents that contain variations such as "recipe for apple pie" or "how to make apple pie." This leads to more accurate and relevant search results and improves the overall user experience.

Another advantage of positional indexing is that it can be used to rank search results based on the proximity of keywords to each other. For example, a document that contains the phrase "apple pie recipe" in close proximity to each other (i.e., the words appear in consecutive positions within the document) may be ranked higher than a document that contains the same phrase with the words spread further apart.

Positional indexing is a powerful tool for information retrieval that improves the accuracy and relevance of search results. It is commonly used in search engines, document management systems, and other applications where efficient and precise search is essential. By taking into account the position and proximity of keywords, positional indexing enables more accurate and relevant search results, leading to a better user experience and improved productivity.

2.4 Hash-based indexing

Hash function signature files are a technique used in information retrieval systems to quickly and efficiently identify documents that contain specific keywords or phrases. This technique involves the creation of a signature file for each document in a corpus, which is then used to generate a hash value for each keyword or phrase.

To create a signature file, each document is first preprocessed to extract relevant keywords or phrases. These keywords or phrases are then hashed using a hash function, which generates a unique hash value for each term. These hash values are then stored in the signature file for the corresponding document.

When a user enters a keyword or phrase into the search engine, the hash function is applied to generate a hash value for the query. This hash value is then used to search the signature files for documents that contain the same hash value. This process is much faster than traditional keyword-based search methods because it eliminates the need to search through the entire document corpus.

Hash function signature files have several advantages over traditional indexing methods. Firstly, they are much faster and more efficient because they only need to search the signature files rather than the entire corpus of documents. This results in much faster search times and reduces the computational resources required for searching.

Secondly, hash function signature files are space-efficient. Because they only store the hash values of the keywords or phrases rather than the full text, they require less storage space than traditional indexing methods.

Finally, hash function signature files can also be used for approximate search. Because the hash function generates unique values for each keyword or phrase, similar terms can be identified by comparing their hash values. This allows for more accurate and flexible search results.

Hash function signature files are a powerful technique for information retrieval that offer significant advantages over traditional indexing methods. They are fast, space-efficient, and can be used for approximate search, making them an essential tool for any modern information retrieval system.

2.5 Agglomerative Clustering

Agglomerative clustering is a popular hierarchical clustering algorithm used to group similar objects together based on their proximity or similarity. This clustering algorithm starts by considering each object as a separate cluster and then combines them iteratively into larger clusters until all objects are in the same cluster. The resulting hierarchical structure of nested clusters can be represented by a dendrogram, a tree-like diagram that illustrates the merging process.

The agglomerative clustering algorithm works by computing the similarity between each pair of objects using a similarity metric, such as Euclidean distance or cosine similarity. It then combines the two most similar objects or clusters into a new, larger cluster. The similarity between the new cluster and the remaining clusters is then computed, and the process is repeated until all objects belong to the same cluster.

One of the advantages of agglomerative clustering is that it is flexible in terms of the similarity metric used, allowing for a wide range of applications in different fields. Additionally, the resulting hierarchical structure provides a useful visualization of the clustering process, allowing users to examine the relationships between clusters at different levels of granularity.

However, agglomerative clustering can be computationally expensive, particularly for large datasets, and can suffer from the "chaining effect" where clusters that are merged early in the process can have a strong influence on the final clustering, potentially leading to suboptimal results. To mitigate these issues, various modifications and extensions

have been proposed, such as using faster approximate algorithms, incorporating domain-specific knowledge or constraints, or combining agglomerative clustering with other clustering algorithms.

Overall, agglomerative clustering is a powerful and widely used clustering algorithm that provides a flexible and intuitive approach to grouping similar objects together.

Chapter 3

Literature Survey

3.1 Introduction:

A lack of knowledge about terrorism has made it a taboo subject and overall an inaccessible source for analysis given that there is no valid database that contains all the necessary information about previous attacks with the metadata in a single place. We intend to provide a large scale database with approximately 180,000 rows of information about various attacks all around the world in the last sixty years. We have further broken down our research into one that looks at the various algorithms used to index, query and retrieve the data. Thus we have scoured the internet and other physical textbooks to add to our knowledge to complete our mission.

3.2 Scope:

The literature review includes studies published between 2006 and 2022, focusing on the various methods used to build search engines, various retrieval and indexing algorithms, design a query based system to retrieve information, evaluating the performances of the algorithms and how to perform analysis on terrorism. The studies selected were based on their relevance to the research question and their methodological rigor.

3.3 Search Strategy:

A systematic search was conducted on several academic databases, including research gate, springer link, IOSR journal of engineering, indian national library of medicine, and science direct using a combination of keywords such as “efficient indexing algorithms," "terrorism analysis," "search engine performance," "inverted index,", “keyword based searching ” and "efficient retrieval algorithms." The search was limited to articles published in English and peer-reviewed journals.

3.4 Synthesis of Findings:

The literature review found that most search engines employ inverted indexing as a base for indexing their data irrelevant of the size, which shows that inverted indexing is an efficient algorithm that can be used to query any type of data. There were also multiple ideologies/methods to performing an index however, at a general level every article had the same intent or steps that they followed. There were many advantages and disadvantages mentioned about the various methods that could be used to develop a large scale querying system which could be used to our advantage depending on what type of system we intend to build.

3.5 Gaps in the Literature:

Despite the growing body of research on the topic, there are still several gaps and inconsistencies in the literature. For example, few studies have examined the efficiency of multiple algorithms on the same dataset whereas most primarily focused on using one specific algorithm for their analysis. Moreover, most studies have adhered to the standard of evaluation of the algorithms' performance using the similar metrics; however, some of the studies have relied on self-report measures, which may be subject to bias and social desirability effects.

3.6 Methodological Issues:

Some methodological issues were identified in the studies reviewed, such as small data sizes for the analysis and algorithms tested, cross-sectional designs of the model created for querying, and the use of non-validated measures for data analysis. These limitations may affect the generalizability and validity of the findings.

3.7 Implications:

The findings of the literature review have several implications for standard practices when it comes to analyzing a terrorism dataset or terrorism in general. It also had implications on the field of building a search engine or query system for large scale databases. For example, they suggest the use of inverted indexing algorithms for indexing purposes, as well as the need for more research on the improvement of more algorithms that can be used for state of the art performance. Moreover, they also suggested how the sensitivity of the subject was an obstacle to finding the whole truth of certain events given that governments would censor and withhold key information due to its sensitivity.

3.8 Conclusion:

Overall, the literature review highlights the use of various indexing algorithms to build search engines and query large scale databases. It also endeavors into the area of terrorism analysis and emphasizes the need for more rigorous and nuanced research on the sensitive topic. There are multiple ways to index, store, and retrieve information however the standard practices that have been followed until now are still the most efficient with minor tweaks to the methods.

Chapter 4

Results

After applying each of these models on the dataset, various performance metrics were used to evaluate the said models. Chief among them are precision, recall, confusion matrix along with F1 scores of each of the models.

These scores draw out the major inferences from the terrorist database which we scrolled out. The major focus was the F1-score since it mainly tells us the accuracy of the model as it is the harmonic mean of precision and recall, which gives equal weight to both metrics. It is a useful metric when there is an imbalance between precision and recall. The recall signifies a metric that quantifies the number of correct positive predictions made out of all positive predictions that could have been made

$$F1\ score = 2 * \frac{Precision\ * Recall}{Precision + Recall}$$

Formula for F1 Score

$$Precision = \frac{\#\ of\ True\ Positives}{\#\ of\ True\ Positives + \#\ of\ False\ Positives}$$

Formula for Precision

$$Recall = \frac{\#\ of\ True\ Positives}{\#\ of\ True\ Positives + \#\ of\ False\ Negatives}$$

Formula for Recall

We used the query: "United States bombing" to run the models and here are the different outputs for the different models.

4.1 Inverted Indexing:

The below confusion matrix denotes the performance of our implemented model on the database. The inverted indexing gave us the below printed values. The results which can be inferred are as follows.

Total Documents Retrieved	19457
True Positive	162234
False Positive	16232
False Negative	0
True Negative	3225
Precision	0.909
Recall	1.000
F1 Score	0.952
Total Occurance of Query	181691

```
Confusion matrix:
[[162234  16232]
 [     0   3225]]
Precision: 0.909
Recall: 1.000
F1-score: 0.952
```

It can be inferred from table1 that there were a total of 19457 documents that were retrieved of which there were 181691 instances of the query being found. Of the total 162,234 were true positives. In comparison with the false positive it is better however, the false positive is 10% of the true positive which can be reduced for a better performance in the future. Moreover, there were no false negatives in the retrieved document and 3225 True negatives were retrieved.

4.2 Positional Indexing:

The below confusion matrix denotes the performance of our implemented model on the database. The positional indexing gave us the below printed values. The results which can be inferred are as follows.

Total Documents Retrieved	266
True Positive	178463
False Positive	3
False Negative	3225
True Negative	0
Precision	1
Recall	0.982
F1 Score	0.991
Total Occurance of Query	181691

```
Confusion matrix:
[[178463      3]
 [  3225      0]]
Precision: 1.000
Recall: 0.982
F1-score: 0.991
```

As it can be inferred from the table there were a total of 266 documents that were retrieved of which there were 181691 instances of the query being found. Of the total 178463 were true positives. There are only 3 false positives and 0 true negatives. Moreover there were 3225 False Negatives which can be improved however since it is 1.8% of the True Positive it can be neglected and further improved in the future.

4.3 Signature Files using Hash Indexing:

The below confusion matrix denotes the performance of our implemented model on the database. The Hash indexing gave us the below printed values. The results which can be inferred are as follows.

Total Documents Retrieved	85540
True Positive	94262
False Positive	84204
False Negative	1889
True Negative	1336
Precision	0.528
Recall	0.980
F1 Score	0.686
Total Occurance of Query	181691

```
Confusion matrix:
[[94262 84204]
 [ 1889  1336]]
Precision: 0.528
Recall: 0.980
F1-score: 0.686
```

There is a huge imbalance in precision and recall which is where F1 score can be used to judge the output. The F1 score is a subpar .686 which indicates that the model has not performed well. The reason for this could be an improper suitability of the model for the dataset that we have.

There is also a widespread in between true positives and false positives which is never a measure of a well performing model. Moreover, the percentage similarity between false negative and true negative are also very similar which once again is not a good measure for a retrieval system.

4.4 Agglomerative Clustering:

The below confusion matrix denotes the performance of our implemented model on the database. The Agglomerative Clustering gave us the below printed values. The results which can be inferred are as follows.

Total Documents Retrieved	4155
True Positive	3919
False Positive	0
False Negative	1081
True Negative	0
Precision	0.780
Recall	1.000
F1 Score	0.880
Total Occurance of Query	181691

```
Confusion Matrix:
[[3919    0]
 [1081    0]]
Classification Report:
              precision    recall  f1-score

           0       0.78      1.00      0.88
```

The agglomerative clustering was a bit shoddy because to run the model we had to decrease the size of the dataset to 50,000 rows from its original length. Considering that it can be inferred that the .880 F1 score is very efficient. However, the result of this model can not be used to compare with the other models due to the change in the dataset, however, this can be used as a proof of concept to try running the model on a better system with the full dataset to get the proper output which can then be used to compare with the other models.

Chapter 5

References

Bramer, W. M., De Jonge, G. B., Rethlefsen, M. L., Mast, F., & Kleijnen, J. (2018). A Systematic Approach to Searching: an Efficient and Complete Method to Develop Literature Searches. *Journal of the Medical Library Association*, *106*(4). https://doi.org/10.5195/jmla.2018.283

Carterette, B. (2005). Comparing inverted files and signature files for searching a large lexicon. *Information Processing & Management*. Retrieved from https://www.academia.edu/2666174/Comparing_inverted_files_and_signature_files_for_searching_a_large_lexicon

Guohui, L., Song, L., Xudong, C., Hui, Y., & Heping, Z. (2014). Study on Correlation Factors that Influence Terrorist Attack Fatalities Using Global Terrorism Database. *Procedia Engineering*, *84*, 698–707. https://doi.org/10.1016/j.proeng.2014.10.475

Lafree, G., Dugan, L., Fogg, H., & Scott, J. (2006). *The author(s) shown below used Federal funds provided by the U.S. Department of Justice and prepared the following final report: Document Title: Building a Global Terrorism Database*. Retrieved from https://www.ojp.gov/pdffiles1/nij/grants/214260.pdf

López, J. A. H., & Cuadrado, J. S. (2021). An efficient and scalable search engine for models. *Software and Systems Modeling*. https://doi.org/10.1007/s10270-021-00960-4

Olajide, O., Elegbeleye, D., & Matthew. (2014). Performance Evaluation of Selected Search Engines. *International Organization of Scientific Research*, *ISSN*, 2. Retrieved from http://iosrjen.org/Papers/vol4_issue2%20(part-1)/A04210112.pdf

Stathoulopoulos, K. (2021, April 10). How to Build a Semantic Search Engine With Transformers and Faiss. Retrieved April 11, 2023, from Medium website: https://towardsdatascience.com/how-to-build-a-semantic-search-engine-with-transformers-and-faiss-dcbea307a0e8

Zobel, J., Moffat, A., & Ramamohanarao, K. (1998). Inverted files versus signature files for text indexing. *ACM Transactions on Database Systems*, *23*(4), 453–490. https://doi.org/10.1145/296854.277632

(n.d.-a). Retrieved from https://www.researchgate.net/publication/357362763_An_efficient_and_scalable_search_engine_for_models

(n.d.-b). Retrieved from https://arxiv.org/pdf/1302.2318.pdf

(n.d.-c). Retrieved from https://www.researchgate.net/publication/283541020_Proposed_Architecture_for_Terrorist_Web_Miner

(n.d.-d). Retrieved from https://link.springer.com/article/10.1007/s10270-021-00960-4

(n.d.-e). Retrieved from https://www.ojp.gov/pdffiles1/nij/grants/214260.pdf