# DON'T CRAWL IN WEB

A project report stage II submitted in partial fulfillment of the requirements for the degree of Bachelor of Engineering in Computer Engineering

by

**Sameer Mishra (40)**

**Harshil Patel (49)**

**Pratik Patil (51)**

Under the guidance of

**Prof.(Mrs.) Shilpali Bansu**


Department of Computer Engineering

A. C. Patil College of Engineering, Kharghar, Navi Mumbai

University of Mumbai

2016-2017

Jawahar Education Society's

A. C. Patil College of Engineering, Kharghar

# CERTIFICATE

This is to certify that the project entitled

## DON'T CRAWL IN WEB

by

**Sameer Mishra (40)**

**Harshil Patel (49)**

**Pratik Patil (51)**

is successfully completed for the degree of Bachelor of Engineering as prescribed by University of Mumbai.

| | |
|---|---|
| Guide | HOD |
| Examiner 1 | Examiner 2 |

Principal

# DECLARATION

I declare that this written submission represents my ideas and does not involve plagiarism. I have adequately cited and referenced the original sources wherever others' ideas or words have been included. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action against me by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Date: _____

Sameer Mishra (40)

Harshil Patel (49)

Pratik Patil (51)

# Abstract

At present, a vast amount of text resources can easily be accessed on the Internet. Although such high frequency of web documents provides us with rapid and immediate access to information, similar and duplicated data results in wasted time and confusion on the user part who re-reads any previous material. Text or content similarity detection can be employed in paraphrasing identification, plagiarism, text summarizing, sentiment analysis, text clustering, text entailment, tracking text news flow on web, etc. For instance, accurate text similarity detection leads to better performance in paraphrasing identification and can improve text clustering. Numerous studies have been conducted to detect similar documents as well as plagiarism, but most of them have not taken the types of content and domain- specific knowledge as well as style and structural of writing into account. For instance, content type varies when moving from a well-documented content to web. From another perspective, in these studies the methods of text similarity measurement which exert a major influence on the accuracy of evaluation have been used for a certainty and express in crisp way. For instance the word Process has different importance in image processing content versus e-learning. As a result, while comparing two texts in specialized or academic domain we will face ambiguity in word or text pairs comparisons since previous methods dont consider structural features such as authors style of writing. Such limitations in similarity measurement reduce the effectiveness of previous methods in surface and semantic level of text. To overcome these limitations, we have proposed a new method that can deal with the ambiguity in the similarity measurement and also consider structural features of text. This method deploys fuzzy linguistic variables to express knowledge about text similarity in surface level of text. The output of this system for all documents is combined with specific factor and the extent of similarity between the contents is determined.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Problem statement

The search engines are the chief gateways for access of information in the web. Search engines in response to user query produces a list of documents ranked according to closest to the user's request by employing the process of web crawling that populates an indexed repository of web pages. The search engine uses data filtering algorithm which can prevent or detect near duplicate documents to save user's time and effort. Duplicate and Near Duplicate web pages accelerate the space for indexes and cost of serving results. A Search engine with a good ranking will provide most of the relevant results early in the list. Therefore a plot against recall will generally slow down to the right. For a web search engine, removing duplicates from its indexed results is desirable since searches get faster and users are not annoyed by several identical responses to a query. Detecting Duplicate and Near Duplicate web pages also helps us to delete duplicate links during crawling, ranking, clustering and archiving caching which can lead to significant savings in network and storage systems. Image below contains text from two separate documents and highlighted content show the similar sub-content material. It is clearly visible, the second document was able to duplicate and re-arrange items from the first document. A search engine would find equal number of repetitions in both the documents and hence would publish these results as two separate relevant results in the list. The goal of both the documents remain same i.e provide the user with news. But, the user is discouraged when he/she sees the same content being repeated on the results after having read the first document.

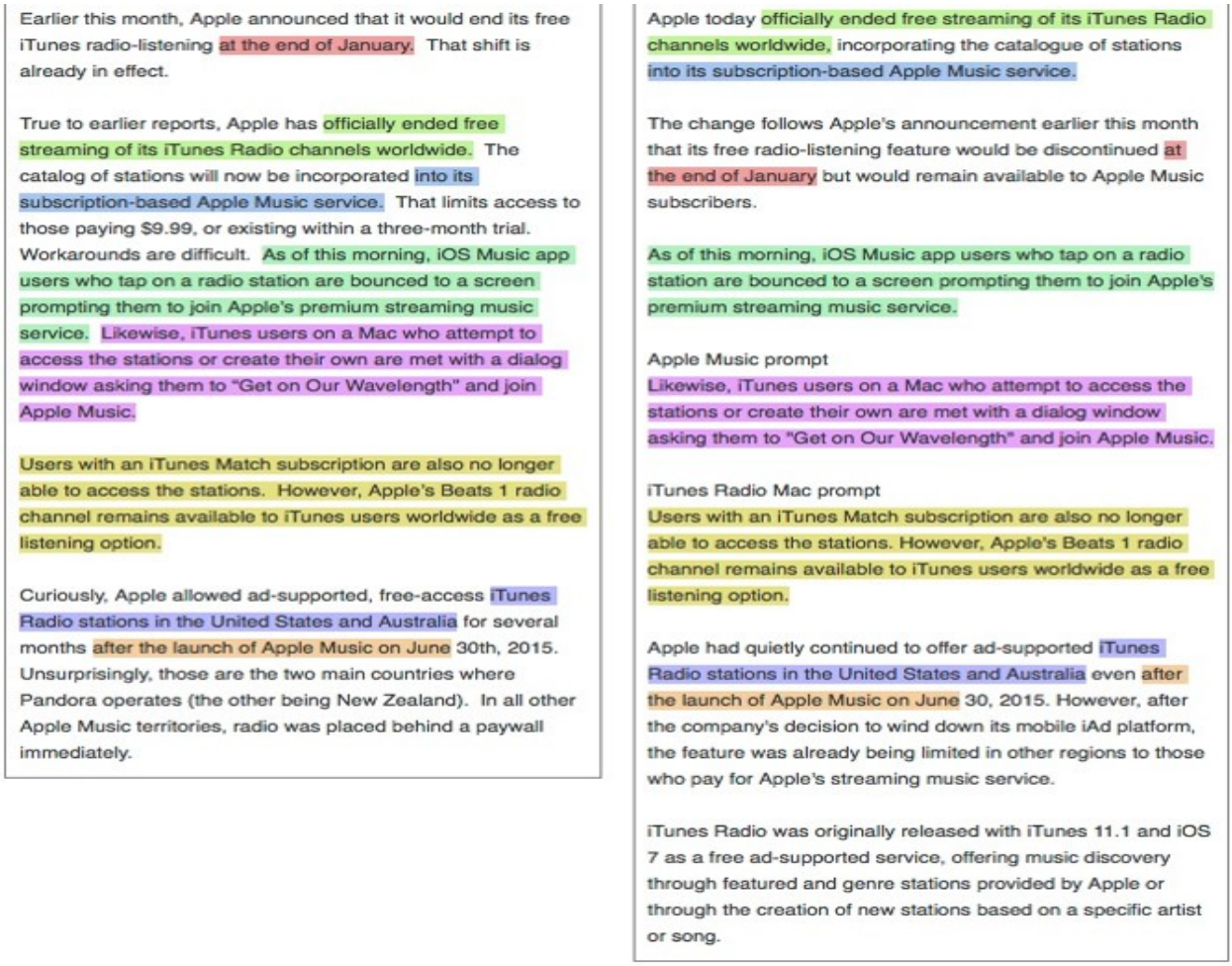


Figure 1.1: Search Result Comparision

## 1.2 Scope of Project

- In order to curb plagiarism in the web domain,of collaborating the search results provided by any leading search engine with improved graphical representation of the search results.
- It would be very helpful for the user to get to the correct document,hyperlinks,images etc without going through all the pages of the search result thereby saving his time and efforts in reading whole documents etc.
- The obtained results are efficiently optimized and well collaborated hence giving the utmost accurate result of the users desire.

## 1.3 Problem Formulation

Text similarity detection can be stated as finding two similar parts of two different documents. Since users change the word order of sentences, style of writing, transpose different sections of a text or rewrite a text by changing a word to its equivalent semantic meaning (substituting hyponyms, synonyms or paraphrasing), detecting these types of text similarity is too difficult, especially when we are concerned with specific content domain texts and in low resource scenarios like less spoken about domain. However in specific content domain texts usually it is difficult to change the style of origin author. This will help us significantly to detect similar document.

- Given two web page content from a search result in specific content domain, how can we assess the degree of similarity between two texts with high accuracy and precision ?

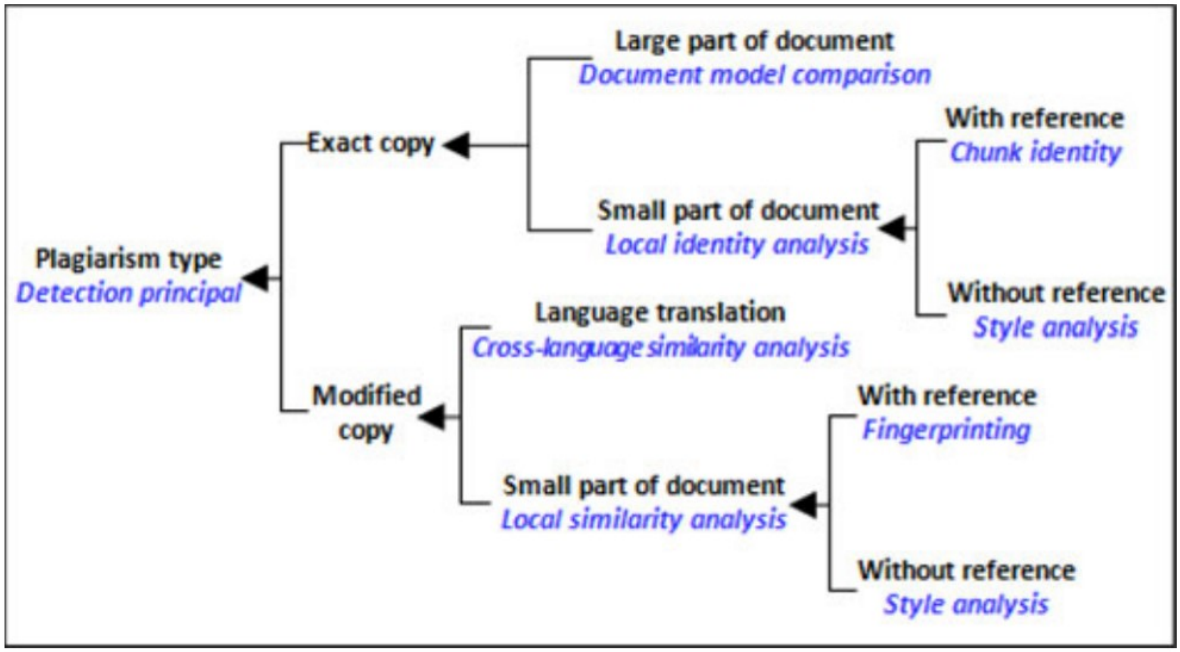


Figure 1.2: Problem formulation

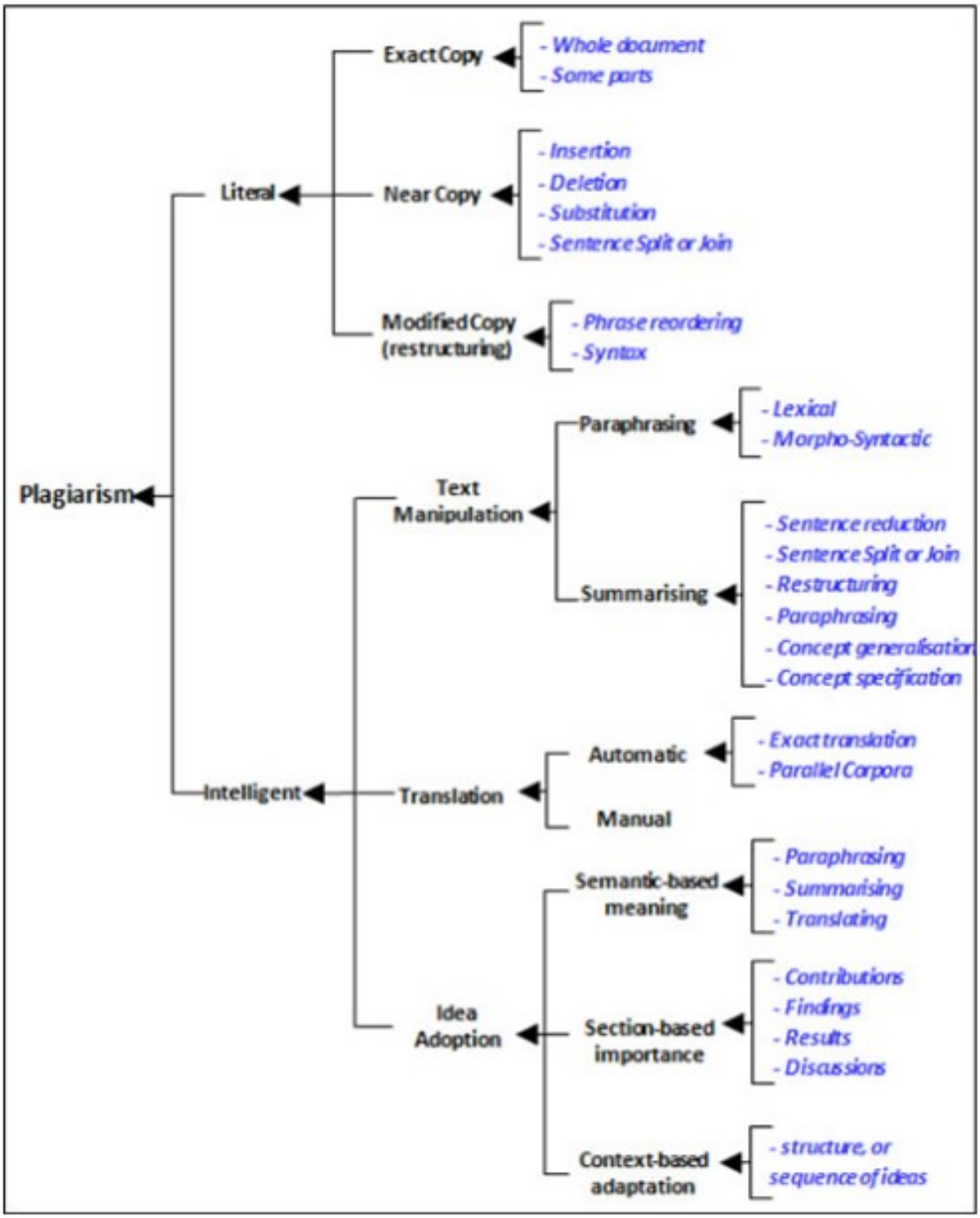


Figure 1.3: Problem Formulation

# Chapter 2

# Literature Survey

In the previous text similarity methods[from survey] -they could classify methods into two categories: Literal and Intelligent. With respect to literal methods, individual use simple operations such as copy and paste, which is the most common form of plagiarism and in intelligent methods they use more sophisticated methods such as paraphrasing, obfuscation, changing the structure to hide cheating (redrafting), translation from one language to another and adopting other peoples ideas. Techniques were classified into six groups: character-based, vector- based, grammar-based, semantics-based, fuzzy-based, and structure-based. After our literature review we concluded and propose semantic- and fuzzy- based methods as a suitable fit, due to their better detection and estimation of text similarity and plagiarism detection. By harnessing the power of fuzzy logic and mathematical cryptographic techniques, we propose a solution that fits well when performed at scale.

## 2.1 Survey Tables

| YEAR | TOPIC | OBSERVATIONS | Advantages | Dis-Advantages |
|---|---|---|---|---|
| 2006 | Detecting Nepotistic Links by Language Model Disagreement | Filters out hyperlinks with no relevance to the target page without the need of white and blacklists or human interaction | Searches and removes spams, ads and other invalid things by link analysis. | Doesnt consider documents or anything else other than the hyperlink keywords while crawling |
| 2006 | A Web-based Kernel Function for Measuring the Similarity of Short Text Snippets | Uses cosine coefficient to determine degree of similarity and compares text | Effective and highly efficient on less quantity of data. | Cannot crawl in todays world as the quantum of data is more.Searches only for keywords and hence decides degree of similarity |
| 2007 | Using Spam Farm to Boost PageRank | Optimal spam farm claimed by Gyongyi and Garcia-Molina through dropping the assumption that leakage is constant. | Reduces the 6-8% spams in the search engine results by a greater margin. | Considers only spammed and hijacked websites. |
| 2007 | Measuring Similarity to Detect Qualied Links | Link-based ranking algorithms, merit based decision making. | Using a classier, noisy links are detected and dropped. Then, link analysis algorithms are performed on the reduced link graph. | Link farming and making classifiers is kind of hectic and doubles the performance of Kleinbergs HITS and boosts Bharat and Henzingers imp algorithm by close to 9% in terms of precision |
| 2007 | Do Not Crawl in the DUST: Different URLs with Similar Text | Examines the URLs and the Text while crawling. | Considers every aspect to reduce the similarity in the search results and provides most accurate data. | The efficiency of the algorithm can be enhanced with modern techniques. |
| 2009 | Removing web spam links from search engine results | Considers page ranks for operations. | Advances Gyngyi and Garcia-Molina methodologies. | Does not consider text or documentations, just the page rankings. Attackers can manipulate with page ranks |

Table 2.1: Survey 2006-2009

| YEAR | TOPIC | OBSERVATIONS | Advantages | Dis-Advantages |
|---|---|---|---|---|
| 2010 | Learning URL Patterns for Webpage De-duplication | Map Reduce Framework | Irrelevant for project. | Focus remains on urls, Cares less for content. |
| 2010 | A Pattern Tree-based Approach to Learning URL Normalization Rules | URL normalization in both crawling and index compression. | Uses both content-based de-duping and URL-based de-duping. | Follows a graph based approach, useful for only duplicates arriving same host, rather than distributed. |
| 2013 | Web Spam Detection Using MapReduce Approach to Collective Classification | MapReduce programming model. | Helpful in spam detection. | Useful is initial stages of scraping out irrelavant content. Limited scope. |
| 2013 | An Unsupervised Model to detect Web Spam based on Qualified Link Analysis and Language Models | Self Organizing Map (SOM) and the Adaptive Resonance Theory (ART) | Complex Algorithms for vast computations and analysis | Follows optimization based on quality of links rather than content or history of publications. |
| 2013 | A Survey on Near Duplicate Web Pages Crawling | Map Reduce programming model. | Projects few useful proven methods. | Methods like shingling, SPEX, Simhash method do not prove useful for deep comparisons |

Table 2.2: Survey 2010-2013

| YEAR | TOPIC | OBSERVATIONS | Advantages | Dis-Advantages |
|---|---|---|---|---|
| 2014 | Intelligent Mechanism for Redundant Result Removal from Search Results | Extensive literature survey- Redunant Document Detection and Removal Framework+ Shingle Generation Agent - fetched urls for a specific query.+ Shingle Comparison Agent - [Similarity and Containtment Measures] + Query Processor Agent - removes redundant results from SGA. | Url fetching from query and perforrming document content analysis. Based on a similarity index, decide if the search result is relevant/redundant. | - No defined way/method to define category/objective of document/article/webpage.- Doesn't highlight main content- How to distinguish among main content in the document.[Ads/extra headers] ? |
| 2014 | Link-based web spam detection using weight properties | Focuses on discovery of host serving web pages for increased visibility and no real content. Relies on website history and spams. Host Specific data collection - then classifies data. | | Useful for spam detection alone. Doesn't consider content. Only host relationship is analysed. |
| 2015 | A Survey on Design and Implementation of Clever Crawler based on DUST Removal | Fetching urls using any crawler for some query. Form Clusters for similar urls. Uses Jaccard similarity coefficient. Declare similar if count of similar words pass a threshold. | Requires counting of words followed by comparison. | Focus is majorly on url alone. |
| 2016 | Detection of Distinct url and removing DUST using muliple alignments of sequences | URL Normalization. Sequence Alignment. Pairwise Sequence Alignment. Multiple Sequence Alignment breaks URL, makes tokens and then finds similar url's. | Good for url cleaning up. Finds similar url's returning same page/document | Focus on URL alone. Focus on single host/domain. Doesn't care about content.No methodology for comparison on different domain/host. |

Table 2.3: Survey 2014-2016

## 2.2 Previous work

Increasing number of websites generate content every hour/every day. More options to choose from with varying quality i.e increasing reading time per document. Search Engines like google still rely on page-ranking algorithms which dont judge a link by its content quality. Engines index the pages based on their content and the publisher specifications of category and timestamp.

## 2.3 Drawback of Existing System

Although, more options helps in getting accurate results, this often leads to more and more repeated results with varying degree of similarity. Content in the first results is often seen repeated in the third, fifth result. Requires more of user time to go through the results

# Chapter 3

# Proposed System

## 3.1 Architecture

The goal for the project was to look for a solution that could work at scale and not be biased when comparing documents from different domain. Any word belonging to a particular domain has its own unique meaning. But, this becomes an over analysis, since our solution is being built on an existing qualified engine. The search engines returns a list of web document target link. Post this, Individual Documents from the results is processed and converted to a vector representation. Using this representation, documents can be compared and assigned a similarity index using fuzzy linguistic variables for the end user.

Initially on a search keyword, the search engine is queried with the keyword. The list returned contains a short description and a link on the web. We process these links. The links could be either a web page, document or an image, these are filtered to return pure textual content. We process every result and convert it into a mathematical representation.

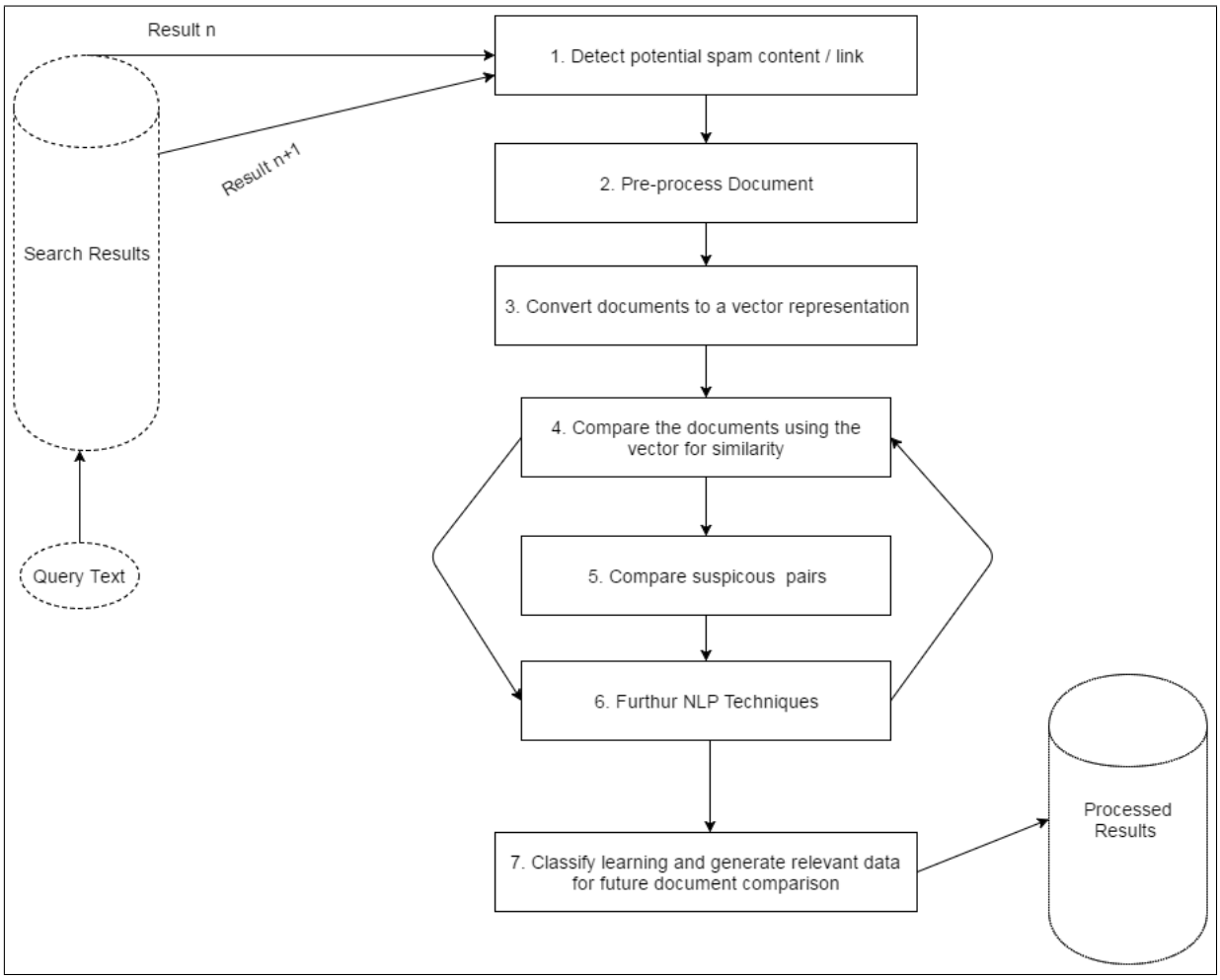


Figure 3.1: Conceptual Framework

Pre-processing involves multiple stages:

1. Spam removal : filter out items which redirect to another link, meaning less relevant content.

2. Document specific filtering : removing style sheets, hidden content, links in the form of text.

3. Apply language filtering: remove unicode errors, remove stop words, normalize the text.

4. Tokenize: Shingling is performed. Sentences are converted to shingles and indexed as per their position.

5. Words are then assigned index based on their position in the document. The index value contains a 4 digit value in the following way. [Document, Paragraph, Sentence, Word].

6. Then, every word in converted to a (16*6=96) bit digest. Every digest represents a 6-bit hexadecimal color code. The method proposed here is based on the Md scheme that most of the hashing function like SHA/MD5 are based on. An existing algorithm could not be chosen since our data size and message digest size requirements are must smaller.

7. Using the indexing and hash values, the document is converted to a mathematical format.

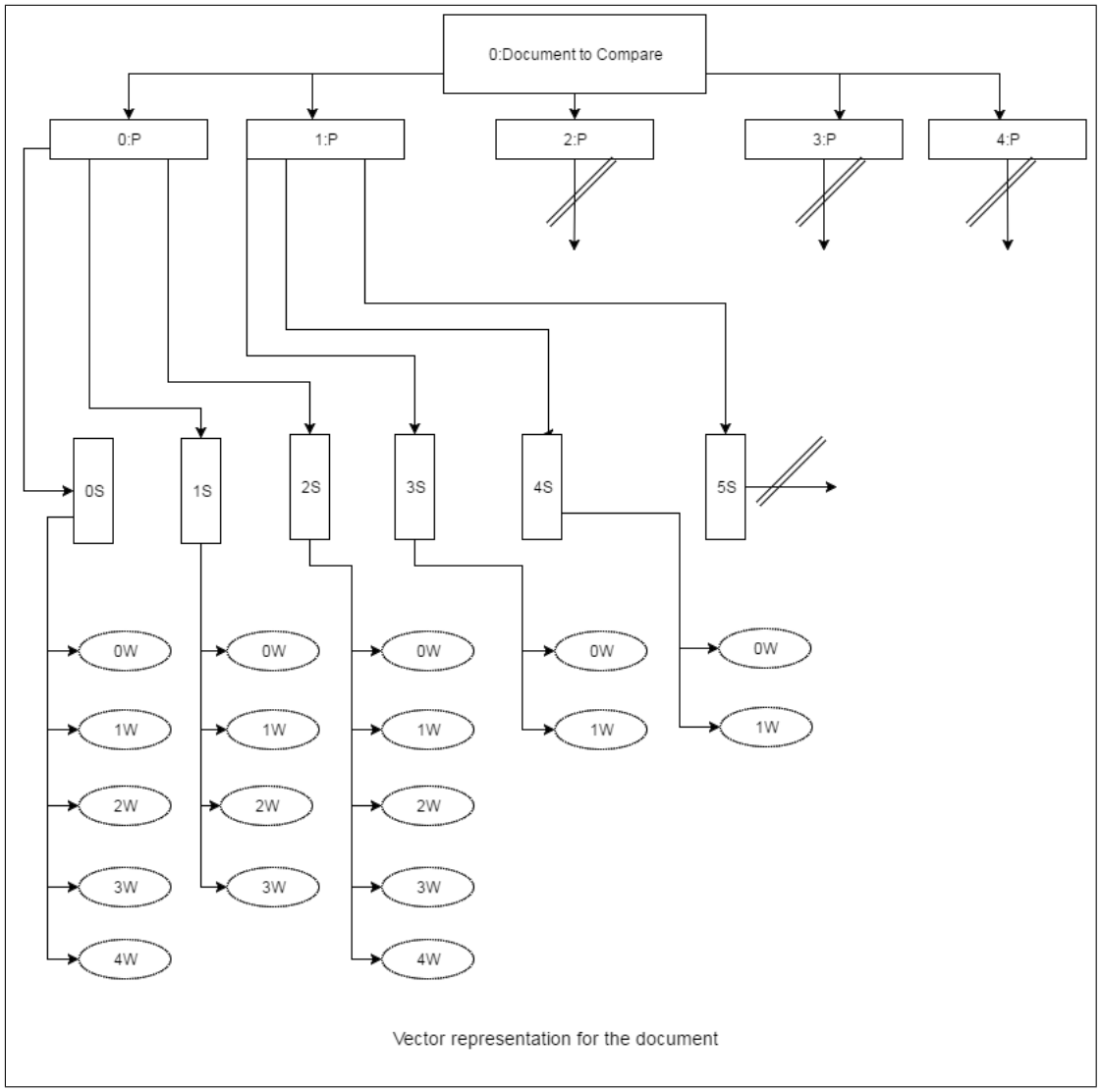


Figure 3.2: Indexing Document

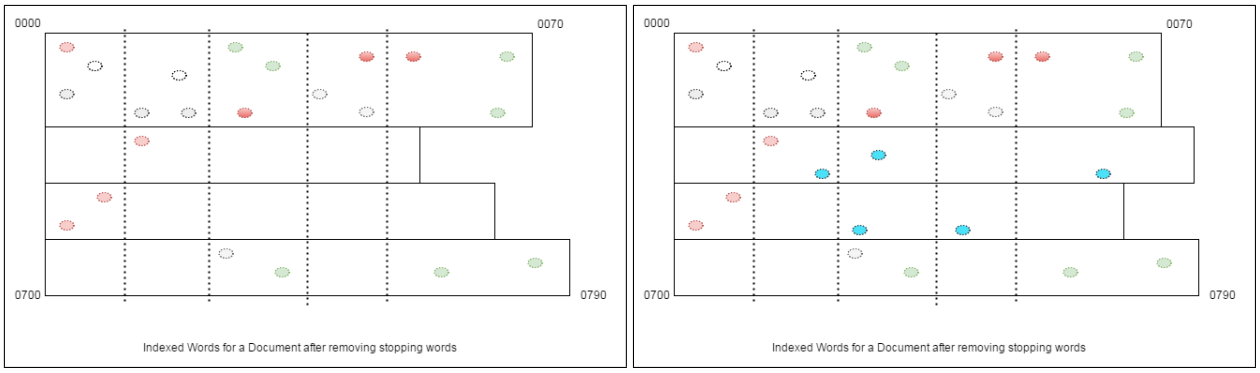


Figure 3.3: Document Representation

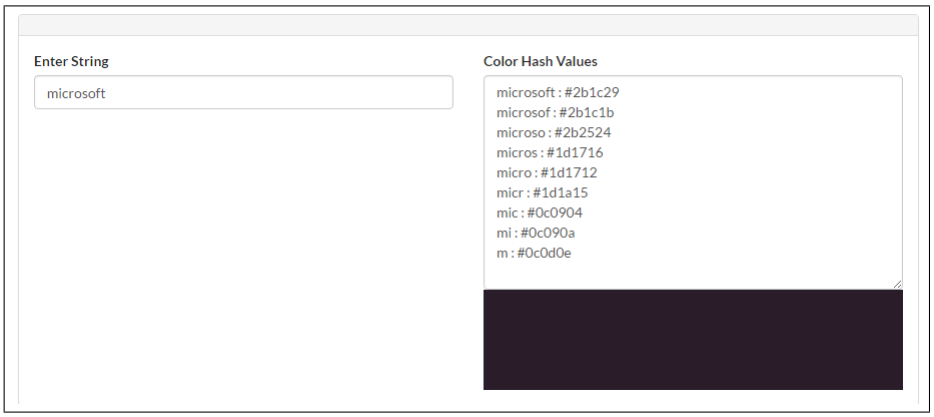


Figure 3.4: Hashing words

This representation is then used to compare two documents. This method is useful in document comparison at various hierarchical levels. For instance, comparing sub-content like paragraph, sentences becomes useful and hence we can assign similarity index.

When comparing sub-sections further nlp/mining techniques can be applied for local adjustments in order to make up for re-arrangements in duplicated material.

For document pair with similarity greater than 0.5, furthur nlp techniques are appplied to use domain specific knowledge, which helps in increasing the score by replacing different words with same meaning.

# Chapter 4

# Design

## 4.1 Use Case Diagram

An Use Case diagram is a representation of user interaction with a system that show the relation between the user and different use cases in which the user is involved. A use case diagram can identify the different type of user of a system.

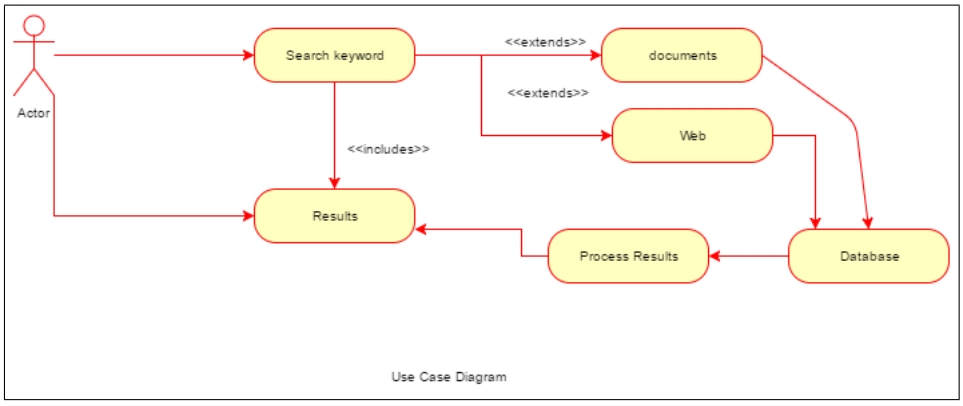


Figure 4.1: Use Case Diagram

## 4.2 Activity Diagram

An Activity Diagram is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. Activity diagrams deals with all type of flow control by using different elements like fork,join etc.

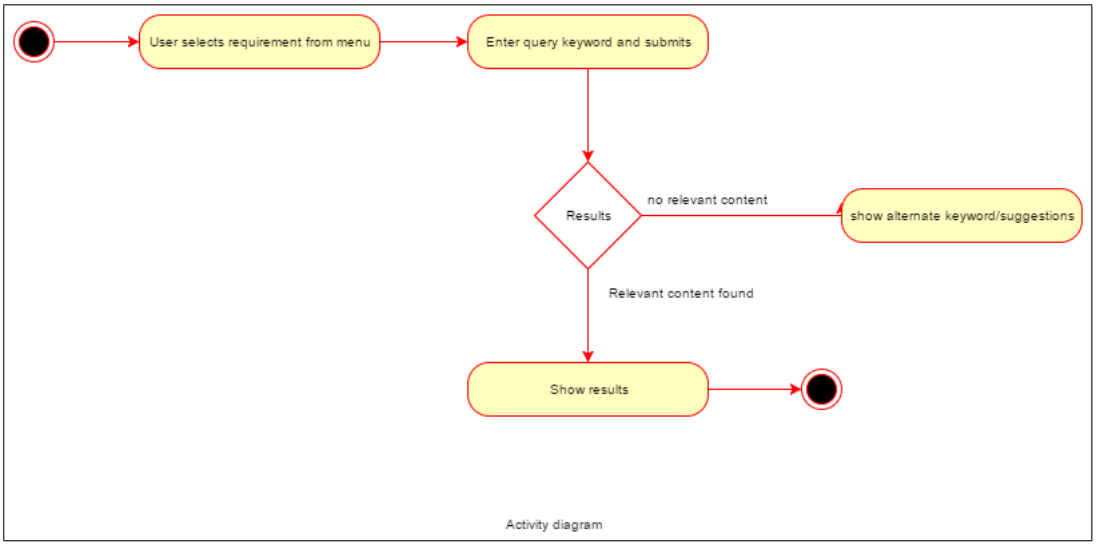


Figure 4.2: Activity Diagram

## 4.3 Collaboration Diagram

A Collaboration Diagram emphasizes on the structural organization of the objects that send and receive messages.

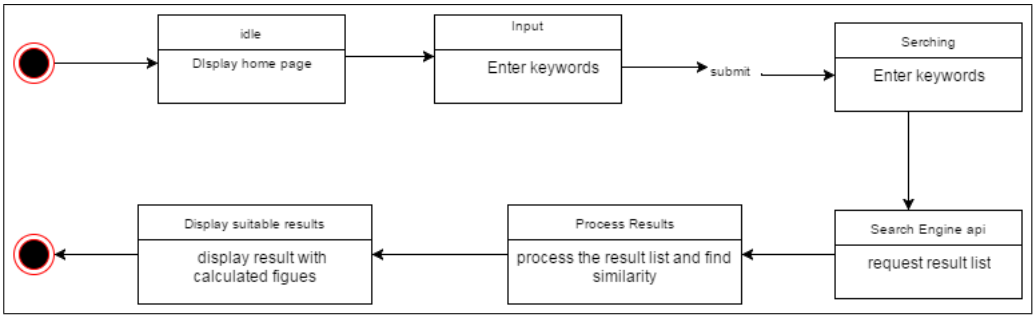


Figure 4.3: Collaboration Diagram

## 4.4 Sequence Diagram

A Sequence Diagram is an interaction diagram that allows how processes operate with another and what is their order. A Sequence Diagram shows object interaction arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of scenario.

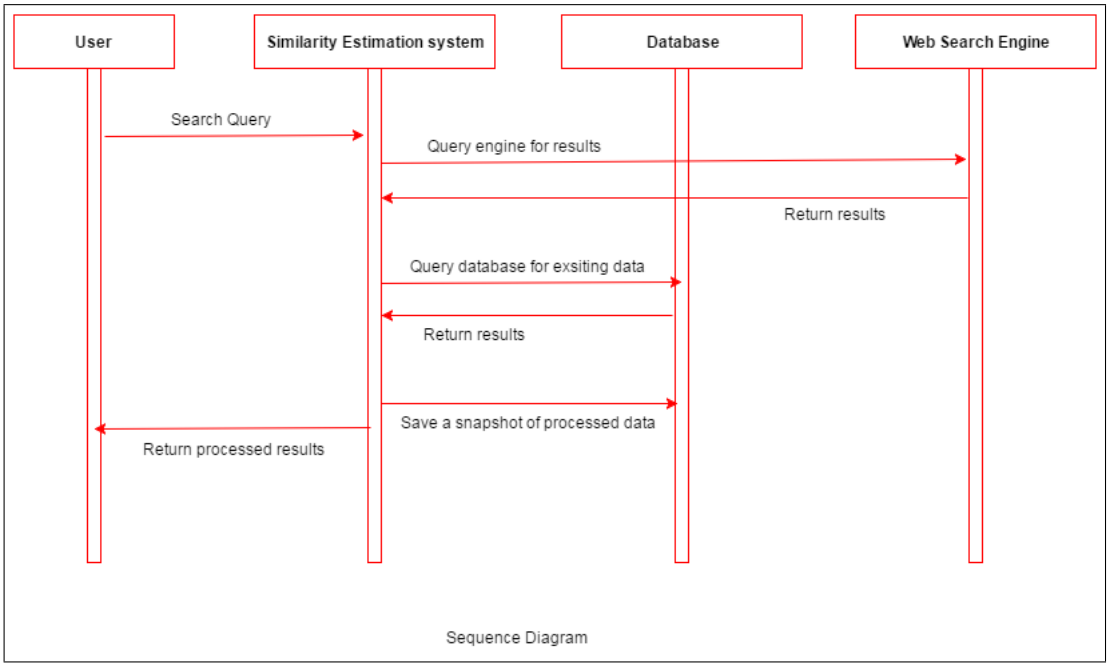


Figure 4.4: Sequence Diagram

# Chapter 5

# Description

## 5.1 Theory

The project will be hosted on a cloud platform, therefore it will remain accessible on all devices with a web browser. The hosted environment will be a (Platform as a service) Pass platform, hence deployment will remain independent of the device or operating system used on the user side.

## 5.2 Inputs to the system

1. User will insert his query over the internet and specify the type of search to perform.

## 5.3 Outputs from the system

1. Refined Search results.

2. Detailed observation displayed beside the search results for accurate results.

## 5.4 Design Approach

The tool compares a bunch of search results and compares them individually in a sequential manner. This keeps the ranking set up by the search engine in the same order and builds its analysis over its list.

The search results is available at an api endpoint. The format required for this endpoint as

follows. URL Regex: tool/search Parameters: query : , quantity : , force : . The url response to any form of GET or POST requests sent to the server. The params are required as mentioned, else a fallback default is applied.

| Parameters | Purpose |
|---|---|
| Query | The query string to search on engine. |
| Quantity | The number of results required to be analysed. The first n results are analyses on n requested items. |
| Force | Lets you force parse all links and refresh the parsed/cached content. |

Following the concept stages required for the proper implementation: The framework for external plagiarism detection involves five stages. It is an expansion of the the common three-stage approach described in standard nlp processing:

**Stage 1**: Pre-processing: This stage prepares the input text collection, that is the prior search results. Text pre processing and shallow NLP techniques are applied to the texts, such as Stemming, Punctuation repair, Stop word filtering.

**Stage 2**: Similarity comparison: This stage performs pair-wise comparisons between each suspicious text against all source texts. One or more similarity metrics are applied to give each suspicious-source text pair a similarity score.

**Stage 3**: Filtering: The similarity scores generated in Stage 2 are used to judge the likelihood of a suspicious-source pair being listed as a candidate pair. The likelihood is determined by setting a threshold on the similarity scores. This can be done either by using a machine learning algorithm to learn the threshold, or by manually defining such a threshold. If a pair has reached a certain threshold, the pair is listed as a candidate pair; otherwise the pair is discarded as not plagiarised.

**Stage 4**: Further processing: As deep linguistic features are computationally expensive, this stage is only applied to candidate pairs. Candidate pairs from Stage 3 are further processed; then Stage 2 is repeated for the pairs of Stage 4 to generate a similarity score.

**Stage 5**: Classification - The final stage is to use the similarity scores from the previous stage to assign each text pair a classification as Plagiarised or Clean. In some cases the class Plagiarised can be further defined at various levels, such as Near Copy, Heavy Revision, or Light Revision.

The classification is either done by setting thresholds, or by using similarity scores generated from various modules as features in a machine learning classifier. Classifications are verified by applying standard evaluation metrics which include precision, recall, f-score and accuracy. The processing flow chart shows the general framework proposed in this study. A text collection pass through various stages of processing, and then similarity metrics are applied to compute the similarity between texts for each suspicious-source pair. The similarity scores resulting from shallow techniques are used as features in text classification or in the filtering stage before applying deep NLP techniques. This five-stage framework has been applied in our small-scale project. The setup although not fit for a production ready deployment, it still performs considerably well with all the restrictions provided.

### 5.4.1 Text Pre-processing Theory

Here, we describe the pre-processing and prior shallow nlp techniques applied to text prior to analysis.

### 5.4.2 Pre-processing

These techniques are available from the Python module of the Natural Language Processing Toolkit 12 (NLTK), which aids text analysis and development. The techniques used are as follows (example texts excerpted from the PAN-PC-10 corpus): Sentence segmentation This technique splits the text in the document into sentences, which allows sentence-by-sentence processing in the subsequent stages.

For example:

1. Raw text: Apple is expected to start ”trial assembly” of iPhones in India next month. Taiwanese original design manufacturer Wistron with which Apple has a contract will begin assembling the iconic phones at its unit in a Bangalore suburb

2. Sentence segmentation: (Sentence 1) (Apple is expected to start ”trial assembly” of iPhones in India next month.) (Sentence 2) (Taiwanese original design manufacturer Wistron with which Apple has a contract will begin assembling the iconic phones at its unit in a Bangalore suburb.)

3. LowerCasing: This technique substitutes every uppercase letter with lowercase to generalise

the matching.

Eg: Raw Text: Apple is expected to start ”trial assembly” of iPhones in India next month. Processed Text: apple is expected to start ”trial assembly” of iphones in india next month.

4. Tokenisation: This technique determines token boundaries, such as words and punctuation symbols in sentences. Eg: Raw Text: apple is expected to start ”trial assembly” of iphones in india next month Processed Text: [[apple] [is] [expected] [to] [start] [”][trial] [assembly][”] [of] [iphones] [in] [india] [next] [month]]

5. Stopword Removal: This technique removes function words, which include articles, pronouns, prepositions, complementizers, and determiners, such as the, of, a, and.

6. Punctuation Removal: This technique removes punctuation symbols to generalise matching between tokens.

### 5.4.3 Shallow NLP Techniques

**Lemmatization**: This technique transforms words into their dictionary base forms, which generalises the texts for similarity analysis. For example, produce and produced are normalised to produce. **Stemming**: This technique transforms words into their stems, which generalises the texts for similarity analysis. For example, both computer and computers are normalised to comput, and product, produce, and produced to produc.

### 5.4.4 Similarity metrics

In this framework, text pre-processing and shallow NLP techniques are applied before the filtering stage. Deep NLP techniques are applied when the texts have been filtered and further investigation is needed for deeper analysis on candidate texts if required. This section describes the similarity metrics that are applied after the corpus has been processed. Different similarity metrics are computed depending on the type and level of processing performed. The application of similarity metrics is essential to feature generation, as each feature consists of similarity scores generated by comparing processed text pairs, and the level of similarity for each suspicious-source text pair is determined by the similarity score.

The overlap coefficient is a variant of the Jaccard coefficient.

Let S(A,n) and S(B,n) be the unique n-grams contained in the suspicious text A and the source

$$Sim_{Overlap}(A, B) = \frac{|S(A, n) \cap S(B, n)|}{min(|S(A, n)|, |S(B, n)|)}$$

text B respectively. The intersection of both sets is divided by the smaller set of S(A,n) or S(B,n). This is very useful when the size is suspicious and source text varies.

## 5.5 Software Dependencies

### 5.5.1 Python

Python is a widely used high-level programming language for general-purpose programming. An interpreted language, Python has a design philosophy which emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly braces or keywords), and a syntax which allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale. Python works best for scientific computing, where code is easier and manageable. The general philosophy is that once, the concept is stable in python, it could port relevant and very frequent memory hungry code to lower level languages like c++.

### 5.5.2 Numpy

NumPy is the fundamental package for scientific computing with Python. It contains among other things: ? a powerful N-dimensional array object ? sophisticated (broadcasting) functions ? tools for integrating C/C++ and Fortran code ? useful linear algebra, Fourier transform, and random number capabilities Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

### 5.5.3 NLTK

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries.

# Chapter 6

# Snapshots

The project was implemented as enclosed in an mvc framework. The analysis tool was integrated alongside a mvc request-response cycle, so that it could extended to other projects as well in the future. The project has a few heavy dependency on a few pre-existing and well maintained libraries as listed above. The corpus used for shallow and deep nlp techniques are available from stanford.

### 6.0.1 GUI Description

The interface was build on the Angularjs platform. Angularjs provides convenience since it reduces the lines of code required for templating and configuration. Since the the interface is a single page application., it is not required to refreshed as other simple web platforms.
The two way code binding, enables data to be persistent and is easily updated as user actions are executed. The screenshot below is a simple view of a three page setup which easy transition between them.

### 6.0.2 GUI Snapshot


Figure 6.1: Main Search Box

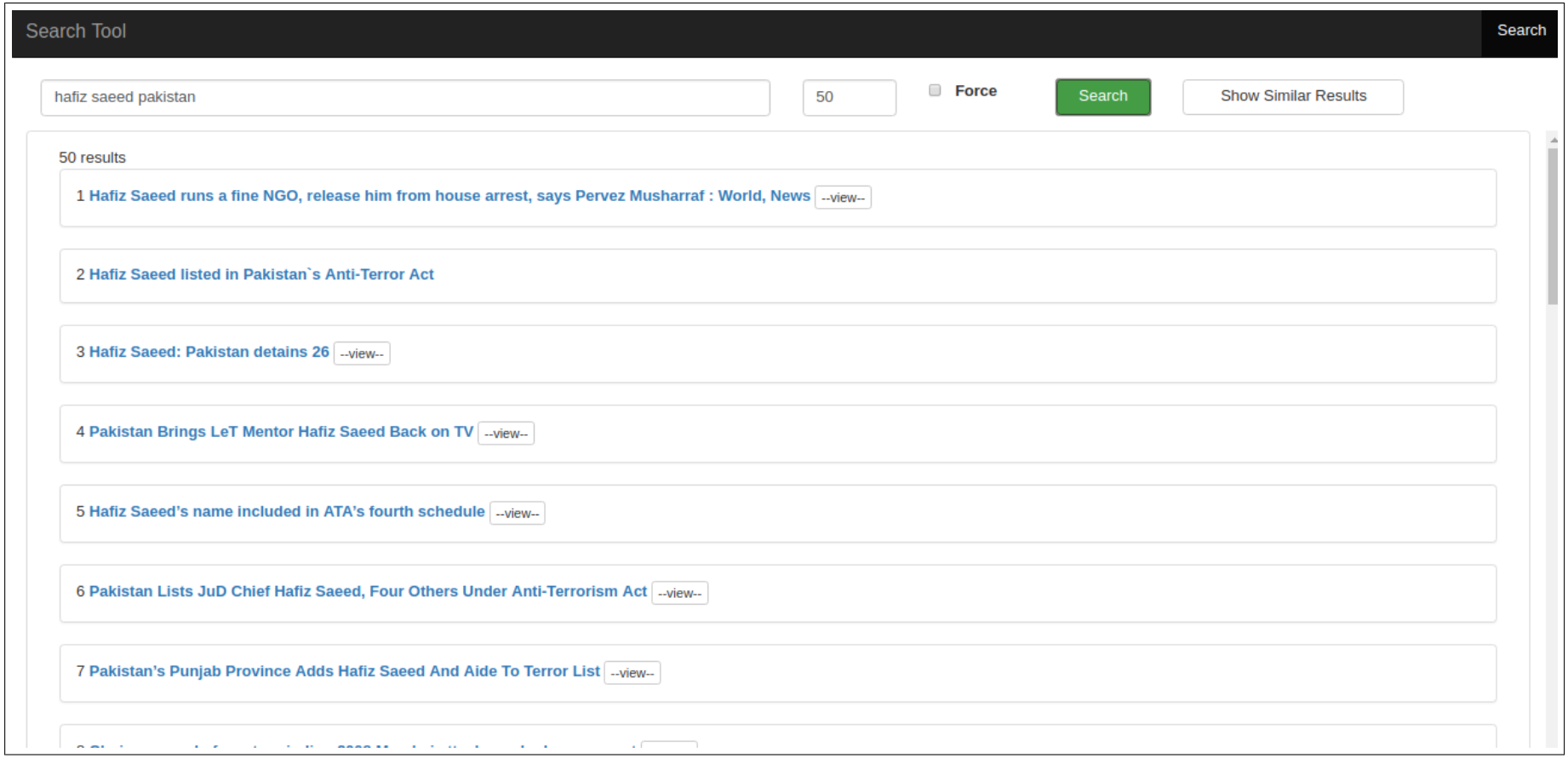


Figure 6.2: After Entering Query
23

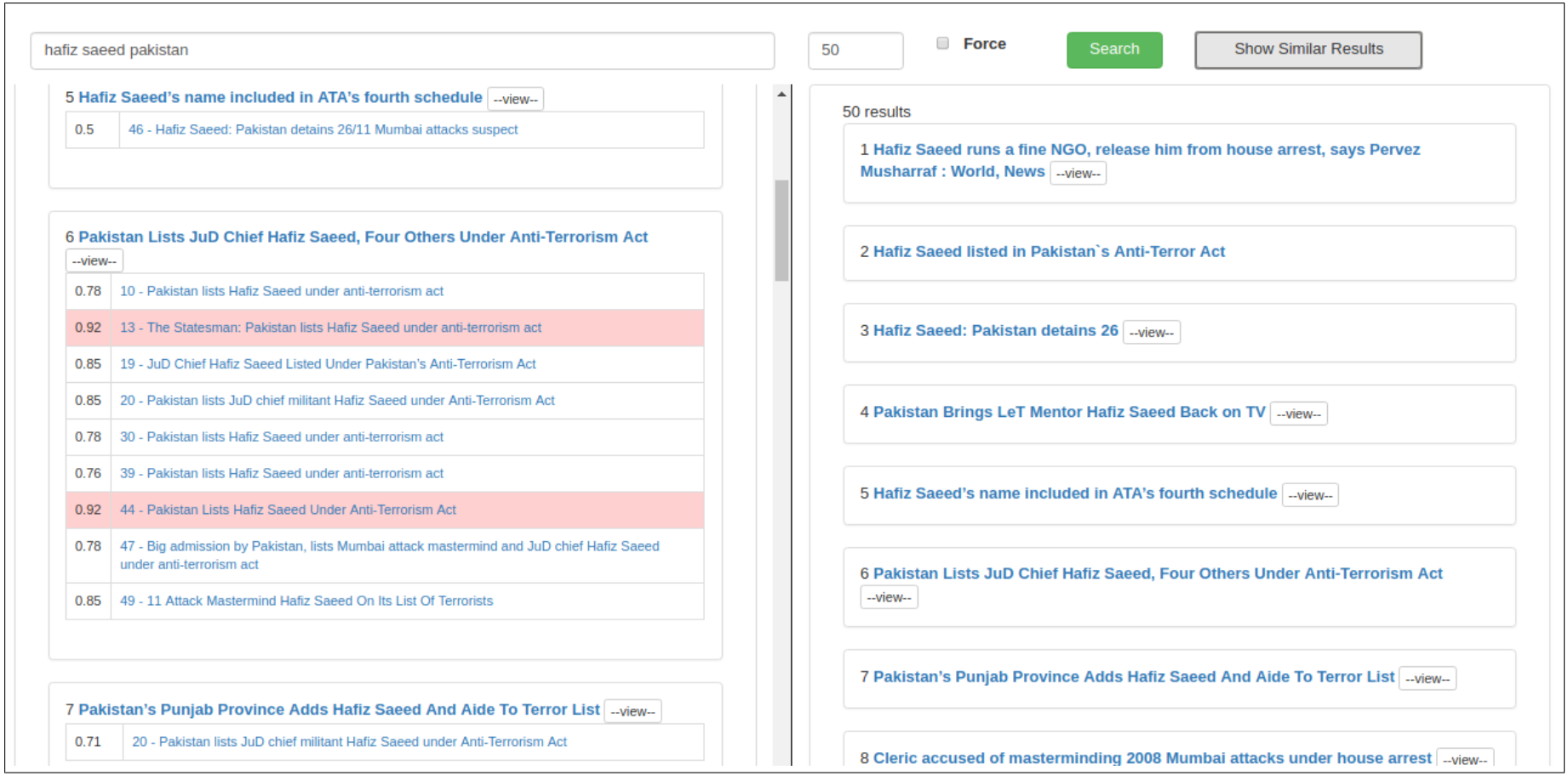


Figure 6.3: Comparisions view

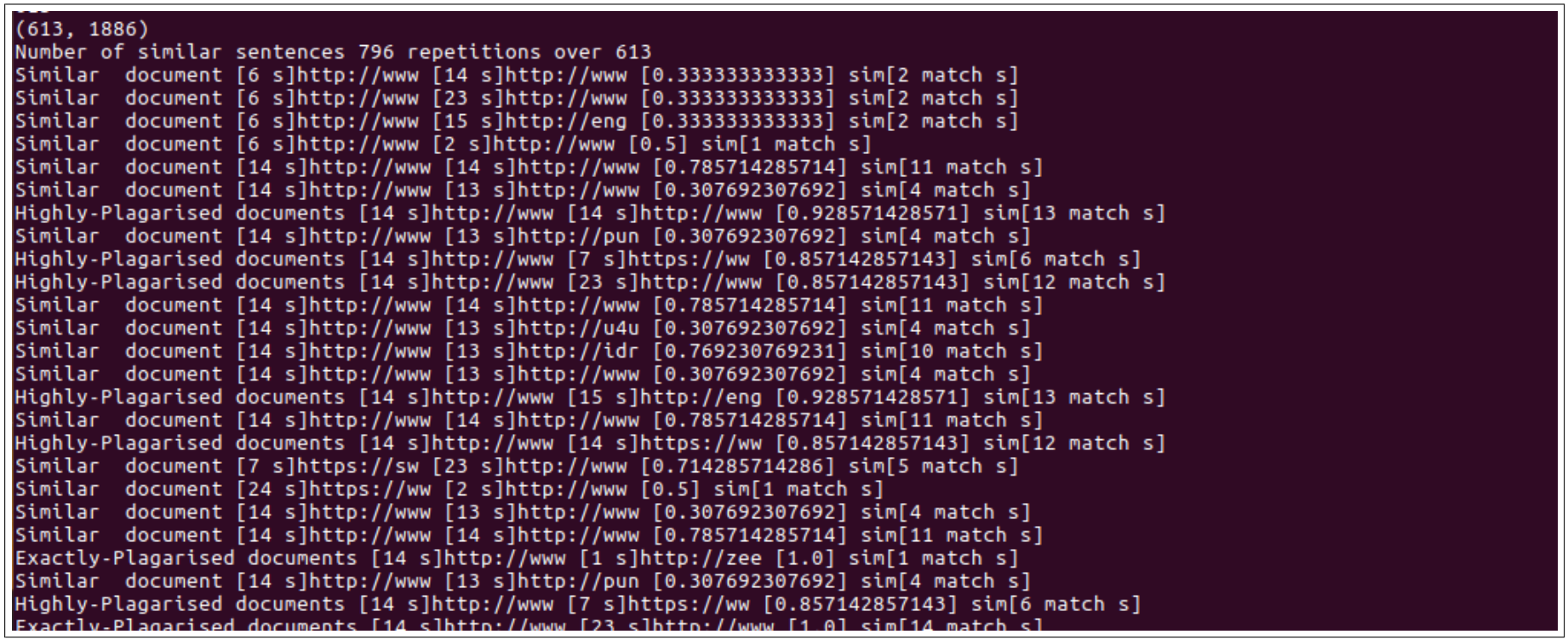


Figure 6.4: Results and observations

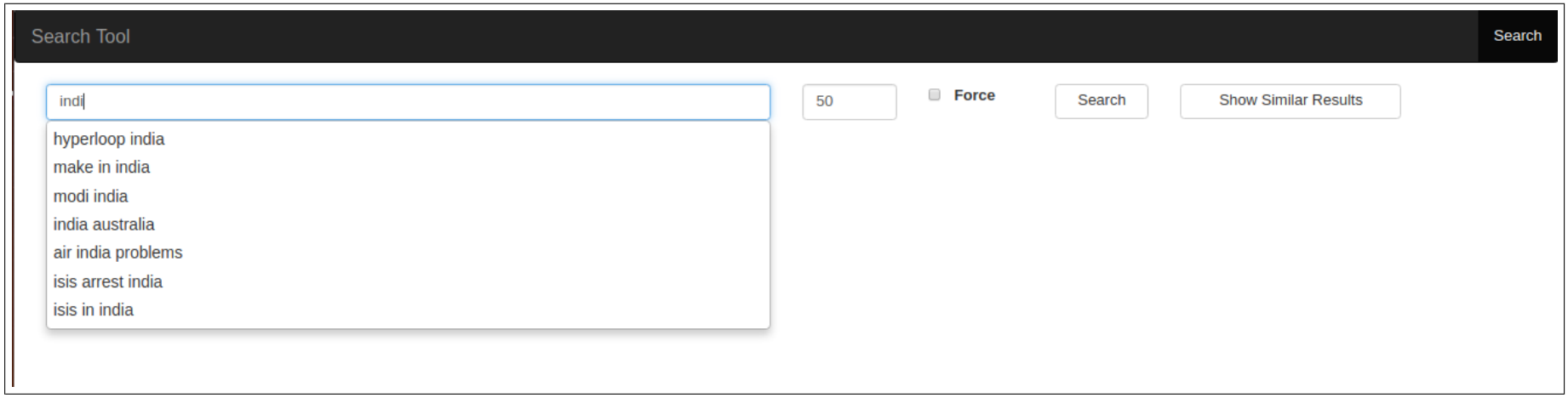


Figure 6.5: auto complete

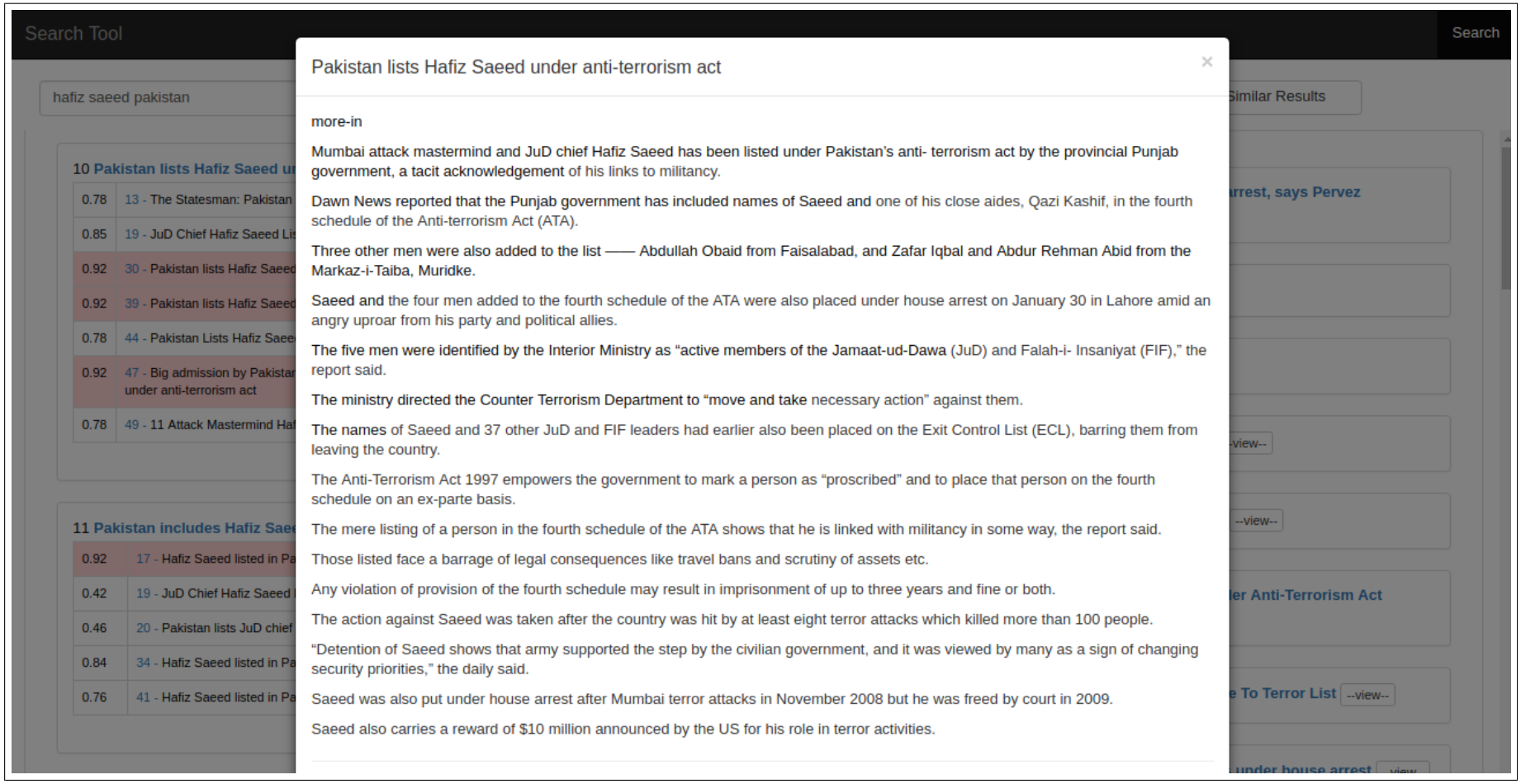


Figure 6.6: Resultant document

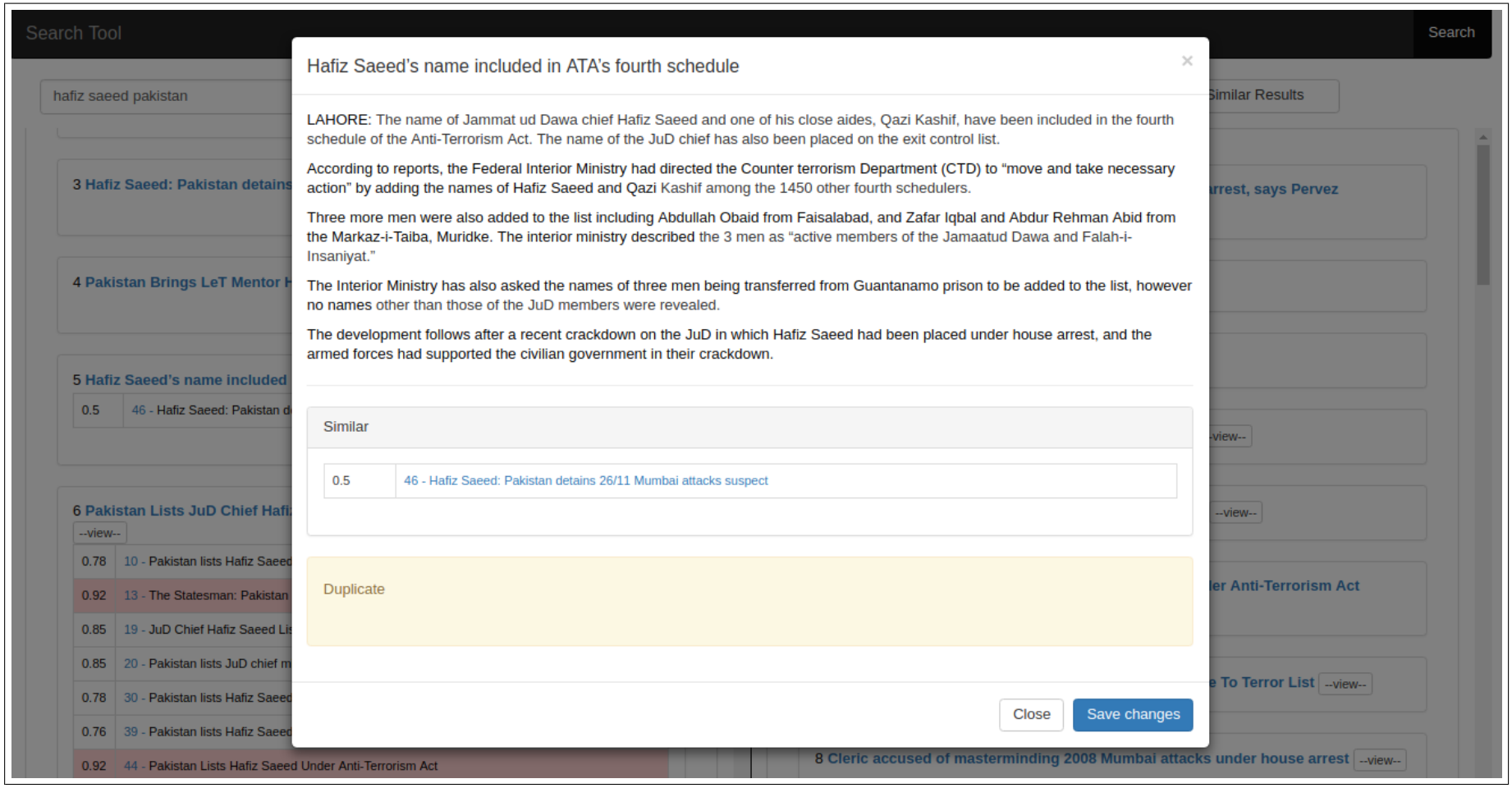


Figure 6.7: similar result
28

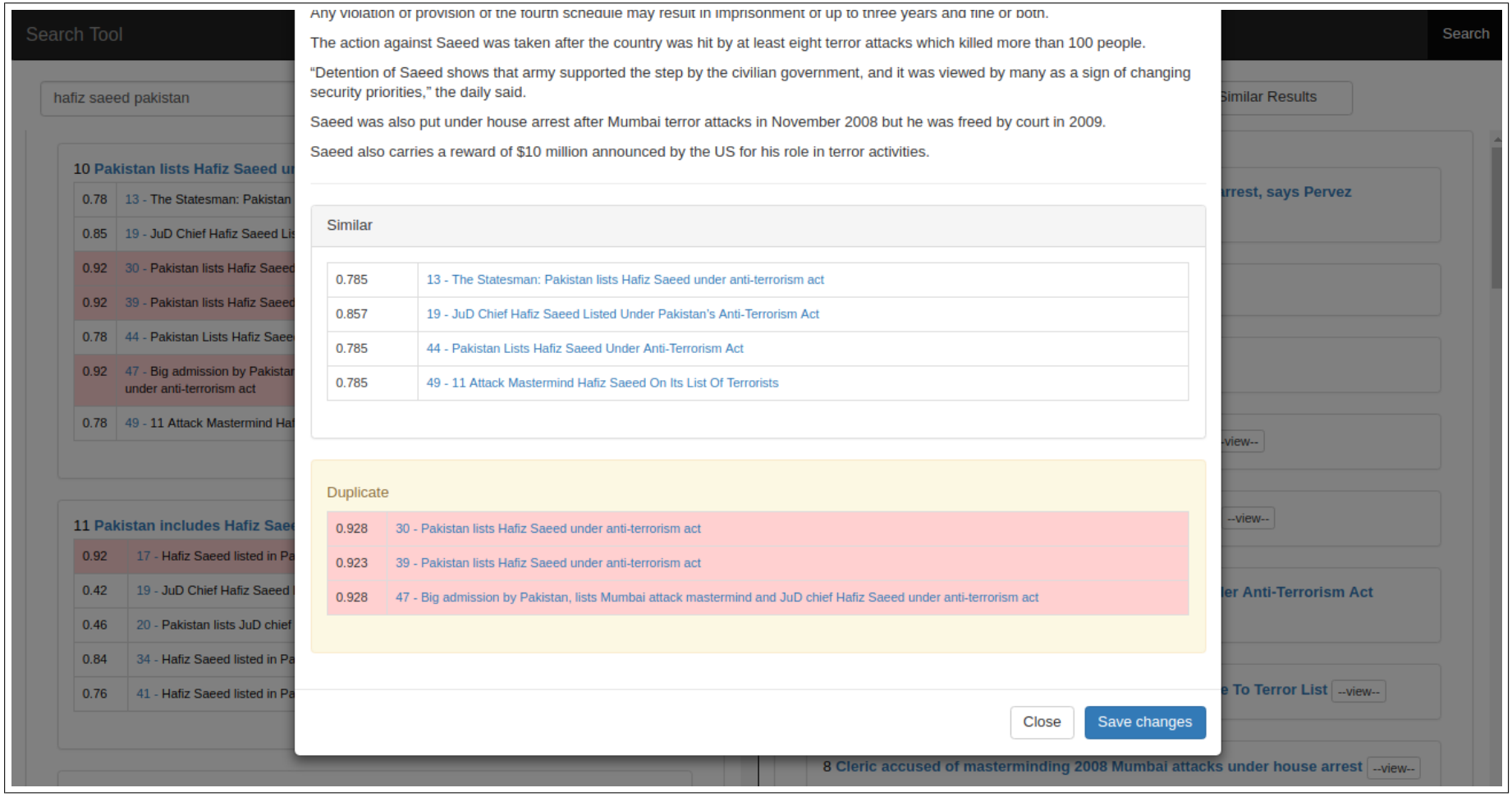


Figure 6.8: Very, highly similar result

# Chapter 7

# Summary

The Results are processed with the five-stage framework, which include the pre-processing stage, the similarity comparison stage, the filtering stage, the further processing stage, and the classification stage. The final stage is to use the similarity scores generated from the similarity comparison stage to give each document pair a binary classification of Plagiarised or Clean, or a multiclass classification for each document pair in three levels: Near Similar, Heavy Revision, or Light Revision. However, the two revisions were shown as similar in the GUI. The classification is either done by setting thresholds, or by using similarity scores depending upon the task. Document classifications are evaluated by applying evaluation metrics.
The text pre-processing techniques include:

1. Sentence segmentation
2. Tokenisation
3. Lowercasing
4. Stopword removal
5. Punctuation removal

The shallow NLP techniques include:

1. Part-of-Speech Tagging
2. Stemming
3. Lemmatization

4. Chunking

After applying these techniques, the output texts were further processed using Jaccard coefficient.

A Similarity in model consists of similarity scores generated using a combination of processing techniques and one of these metrics for each instance of a suspicious source document pair. The features therefore are a representation of the outcome of similarity scores that correspond to a specific set of processed documents.

A Duplicate in model consists of set of sources with very heavy revision passing the hard threshold set for measurement.

# Chapter 8

# Hardware and Software Requirements

## 8.1 Hardware Configuration

1. Memory: 2,4,8 GB
2. Hard Drive Capacity: 100 GB and above
3. Chip Type: Intel Core i7,i5,i3 or AMD A7,A10,A5
4. Processor Speed: 2.60 GHz and above
5. Graphics Processing Type: Integrated/On-Board Graphics

## 8.2 Software Configuration

1. Operating System: Windows XP,7,8,10,VISTA,Ubuntu,FEDORA.
2. C-python IDE
3. Chrome
4. Microsoft WINDOWS NT 4.

# Chapter 9

# Conclusion

In this Project, we will try to simplify the search results into a simpler and much more refined context document for the ease of the user so that he can jump into conclusions of his search results within no time and saving the trouble of going through many documents at he same time. The Project focused on :

1. Quality of document.
2. Quick response to larger data searches through millions of pages over the web.
3. Reliability of the document with utmost accuracy.
4. Availability of the crawler over any platform.

With more upcoming technologies to enhance the search engines, this is our approach to provide the end user a much more better result so that their time in going through all the document is saved and thereby providing much more time for the actual work.
Soon much more technologies would emerge for the betterment of search engines and our minor contribution towards it may make a difference as the method we propose is scalable for real-time requirements.
The proposed framework integrated linguistic and statistical traits. Instead of following a traditional brute-force pair-wise comparison approach, the experiment focused on fitting individual texts into their respective class patterns. The results showed that the identification of plagiarism direction can be easily performed using statistical and linguistic features. These features showed promising results even when they were tested on manually rewritten texts that are challenging for human beings to identify.

On the challenge of filtering candidate documents, one of the main issues of the plagiarism detection approach is that the mechanical means cannot prove the absence of plagiarism. Instead, the approach can only provide indications as to what parts of the text might have been copied from a potential source.

# Chapter 10

# Future Scope

The project could be furthur extended to compare documents in real-time. This would provide a useful way to retrieve same documents floating around the web in different formats. The benefits here would be huge considering the fact reading documents like file .pdf, .word from search results are not only time consuming but often lead to malware distribution upon downloading from un-known hosts. This work could taken care of before hand by an advanced search engine like ours to ease the process.

# Bibliography


[1] Jiang-Ming Yang Yan Ke Xiaodong Fan Lei Zhang Tao Lei, Rui Cai. A pattern tree-based approach to learning url normalization rules. 2010.

[2] Amit Agarwal Hema Swetha Koppula, Krishna P. Leela. Learning url patterns for webpage de-duplication. 2010.

[3] Timothy D. Heilman Mehran Sahami. A web-based kernel function for measuring the similarity of short text snippets. 2006.

[4] Naomie Salim Chow Kok Kent. Features based text similarity detection. 2010.

[5] Christian Platzer Manuel Egele, Clemens Kolbitsch. Removing web spam links from search engine results. 2010.

[6] Lan Nie Xiaoguang Qi and Brian D. Davison. Measuring similarity to detect qualified links. 2007.

[7] Kroly Csalogn Mt Uher Andrs A. Benczr, Istvn Br. Detecting nepotistic links by language model disagreement. 2006.

[8] Xin Zhao Ye Du, Yaoyun Shi. Using spam farm to boost pagerank. 2007.

[9] Uri Schonfeld Ziv Bar-Yossef, Idit Keidar. Do not crawl in the dust:ydifferent urls with similar text. 2007.

[10] J. Daniel Samson K. Mukesh Kumar I. Monish Niveth Tadepalli Sarada Kiranmayee, S. Sai Vimal Prasath. Removing duplicate urls using duster. 2016.

[11] B.Lakshmipathi Shrijina Sreenivasan. An unsupervised model to detect web spam based on qualified link analysis and language models. 2013.

[12] Rajeev Soni Aarti Singh. Intelligent mechanism for redundant result removal from search results. 2014.

[13] Wojciech Indyk, Tomasz Kajdanowicz, Przemyslaw Kazienko, and Slawomir Plamowski. Web spam detection using mapreduce approach to collective classification. 2013.

[14] P. L. Ramteke Kanchan S. Khedkar. A survey on design and implementation of clever crawler based on dust removal. 2015.

[15] Prof. Pankaj Agarkar Jayashri Waman. Eliminate duplicate urls using multiple alignment of sequences. 2015.

[16] Prof. Pankaj Agarkar Jayashri Waman. A survey on near duplicate web pages for web crawling. 2013.

[17] Ashutosh Singh Kwang Leng, Ravi Kumar. Link based web spam detection using weight properties. 2014.

# Acknowledgments

This work was influenced by countless individuals whom we were fortunate enough to meet during our phase-II of the project duration, while space does not permit us to acknowledge them all, we would be remiss if we did not acknowledge the following individuals whose guidance, support and wisdom so greatly influenced this work.

In particular, we would like to deeply thank our Head of Computer Engineering Department **Dr. M. M. Deshpande** for his moral support and guidance and for giving us innovative ideas to complete phase-II of our project in time, he guided us at every step of the project.

We also wish to thank our Project Head **Prof. (Mrs.) Shilpali Bansu** for her moral support and guidance to complete our Phase-II of Project. Further we also wish to extend our thanks to all the staff members as well as other colleagues for attending our Seminars and for their insightful comments and constructive suggestions to improve the quality of this Project work.

Date: _____