# Design Document


## Made By:-

Tanay Gupta : 2018AAPS0343H

Mudit Chaturvedi : 2018A7PS0248H

Sristi Sharma : 2018A7PS0299H


## Creating Database –

Outline of database –

- The database is divided into topics (Machine Learning, Information Retrieval, Computer Graphics, Computer Science and Business) to make the plagiarism checker scalable.
- While adding a document to the database the topic of the document must be mentioned.
- While checking for plagiarism the checker won't need to go through every document in the database, we made a function that can detect the document's topic(s) and return the relevant documents for it to search.
- The information of every topic is stored in a dictionary for fast retrieval. Every topic has following data under it –
    1. The common words that occur in that topic. Sets are used to store these words to avoid repeats and to compute intersection with the words of the document to be checked.
    2. The list (array) of documents about that topic.
- The database also contains a term to frequency dictionary (term_freq_db) for every document. This dictionary stores the frequency of every word for every document, along with this a term to frequency dictionary for the entire corpus (doc_freq_db) as a whole is also used. This dictionary stores the number of documents a word has occurred in, for all the words in the corpus.
  Dictionary data structure was used to retrieve data for every document and every word quickly.
- Finally the database contains a dictionary which stores the name of documents corresponding to their Id number. This will help in returning the name of the document from which plagiarism was done.

Preprocessing document –

- The document is tokenized, this removes the punctuations from the document and the words are stored in an array.
- The stop words are removed using nltk's inbuilt English stop words library.
- Array of words is returned after this process.

- The time complexity of this process is O(n) where n is the number of words in the document, as we need to check every word to see whether it is a stop word or not, checking this takes O(1) time for every word as stop words are stored in sets.

Computing term frequencies –

- Each preprocessed document is traversed for each word and accordingly the values are updated in the dictionaries.
- For every word encountered in a document ‘d’, term_freq_db [d][word] stores the frequency of the word’s occurrence in document ‘d’. (While creating the database containing ‘m’ documents and each document containing ‘n’ words this step takes O(mn) time).
- While doc_freq_db[word] stores the total number of documents, the word has occurred in. (While creating the database containing ‘m’ documents and each document containing ‘n’ words this step takes O(mn) time).
- In order to assign weights and relative importance to the occurrence of each term, tf-idf (term frequency – inverse document frequency) is calculated for each term in a particular document and is stored in a dictionary named tf_idf_db. This dictionary will later be used to calculate similarity between the test document and the documents of the corpus.
- tf-idf approach is used to assign importance and weights to each term in each document, based on their discriminability and relevance to the optimal results.
- For ‘m’ documents in the corpus with ‘n’ words , this calculation takes O(mn) time.

## Checking Plagiarism –

Categorizing Documents-

- To find the most probable topic(s) of the document we compute the intersection of the words of the document with the words that appear in that category (already stored in set). We set a threshold value, if the ratio of number of words in intersection of the document with the words of category and number of words in the document exceed this value then the document belongs to that category.
- If we have ‘m’ categories and our document has ‘n’ words in it then this step takes O(mn) time.

Calculating Similarity Scores-

- The frequency of occurrence of each term, in the test document is calculated and using this term frequency and the previously computed document frequency for the corpus, tf-idf values are calculated for the test document.
- Based on the categories that the test document falls under, its similarity score is calculated against the documents in those categories.
- The similarity score between test document and a document ‘d’ is calculated based on cosine similarity, in order to prevent the document length from biasing our results.

- Treating the test document and each document in the categories to be matched against, as vectors, the cosine similarity is calculated and top 10 documents with the highest score from amongst the categories are returned.
- For ‘k’ categories having ‘m’ documents and ‘n’ words in them, the similarity calculation takes O(kmn) time. While getting the top 10 documents based on these scores takes , O(n) time.
- These 10 documents are most similar to our test document and if the score is above an allowed level of plagiarism/similarity, then the test document can be said to have been plagiarized from that particular document.

## Analyzing the Results –

- The checker will efficiently recognize identical stemmed words between the test document and corpus documents, a 100% similarity score with a document ‘d’ will indicate that the test document is the same as ‘d’.
- Under normal circumstances we can consider that if the test document has more than ‘s’ similarity to a corpus document then a large subset of the test document has been plagiarized from the corpus one. This value ‘s’ can be considered as the limit for plagiarism.
- The results do not account for synonyms used and regards synonyms as different terms. This restriction holds for multiple languages as well because the stemmer used doesn’t form equivalence classes or mappings between synonyms etc.
- Hence a way to get away with plagiarism, undetected from the checker will be to make sufficient changes in the document in terms of language, synonyms etc.
- The categories used in plagiarism detection, streamline our results and give more relevant answers because mostly whenever someone checks for plagiarism in a document then it belongs to a particular subject/course.
- While the category-based approach used, works well for documents closely related to the predefined 5 categories (documents in each of the categories), it has a restriction that if sufficiently large number of words of the test document do not belong to any of the categories (i.e. the intersection with all categories individually is less than the threshold value , 0.5 in trials) then the document will be recognized as not plagiarized from the corpus.
- A way to improve on the results would be to add more documents and categories to the corpus for better analysis.
- Overall, the Checker approach combining categories and similarity scores (cosine) is beneficial as it will give more relevant and authentic results than simple similarity scores-based approach, for larger dataset where we have millions of documents across millions of categories.