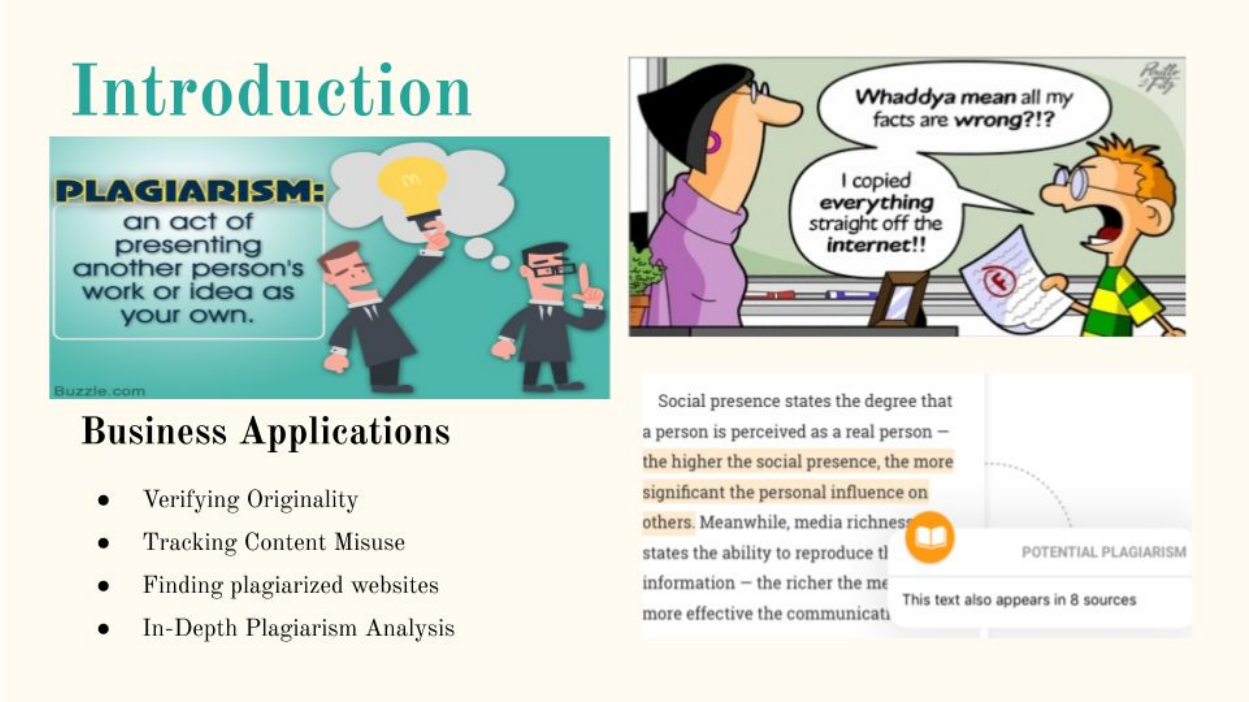
DS 5230 Unsupervised Machine Learning

Project Report

Plagiarism Detection

Authors

Aishwarya Vantipuli, Spatika Krishnan, Nandini Jampala

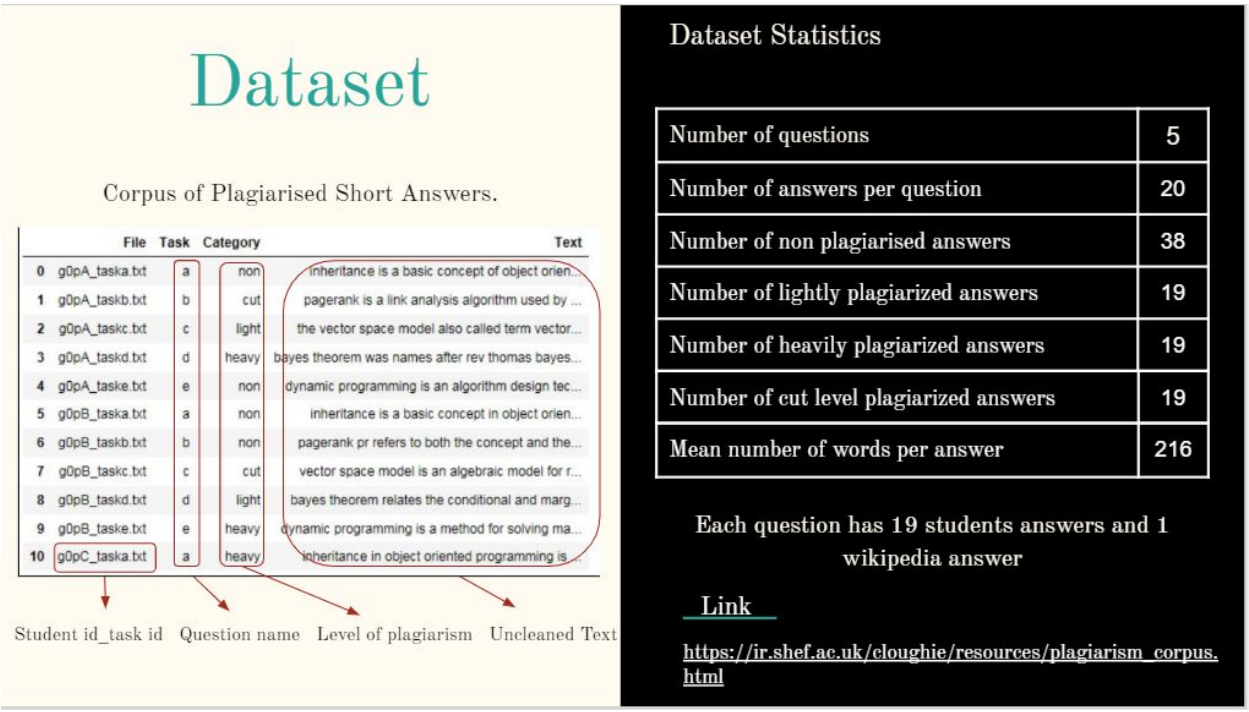


Introduction:

Plagiarism is the “wrongful appropriation” and “stealing and publication” of another author’s “language, thoughts, ideas, or expressions” and representing them as one’s own original work. Checking assignments for plagiarism is essential in academics as assignments are used to evaluate students in their courses.. If a course consists of a large number of students, it is impractical to check each assignment by a human inspector. Therefore it is essential to have automated tools in order to assist detection of plagiarism in assignments.

Business Applications:

- Students may plagiarize by copying ideas from friends,web or private tutors. Most courses in universities evaluate students based on the marks of their assignments.
- A quality plagiarism detector has a strong impact on law suit prosecution.
- To create websites quickly with less effort and to follow the quality content of a reputed website people commit plagiarism.
- Some level plagiarism checkers also provide the accuracy of grammar and paragraphing.



Dataset:

The dataset is taken from a freely available Clough-Stevenson corpus . The corpus consists of answers to five short questions on a variety of topics in the Computer Science field. The five short questions are:

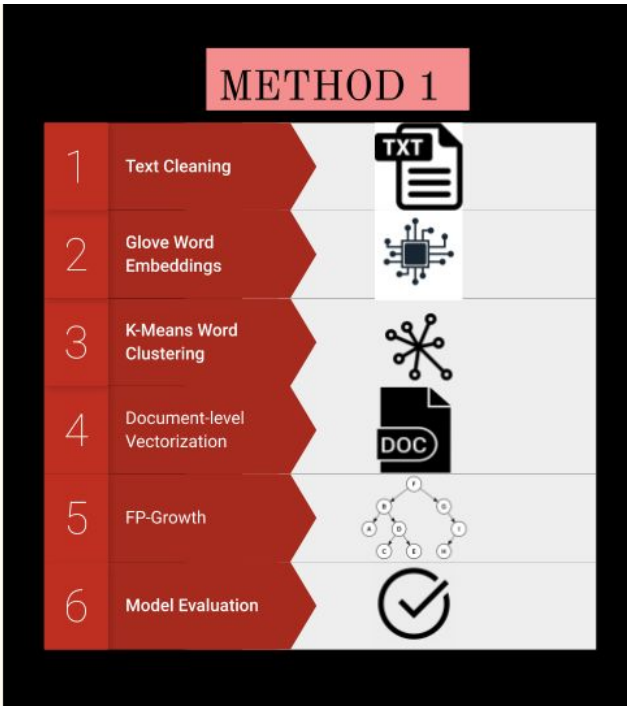
1. What is inheritance in object oriented programming?
2. Explain the PageRank algorithm that is used by the Google search engine.
3. Explain the Vector Space Model that is used for Information Retrieval.
4. Explain Bayes Theorem from probability theory.
5. What is dynamic programming?

These questions are represented as tasks(a-e) in the dataset where each task maps to each of the five questions respectively.

Each question has 19 students answers and 1 corresponding wikipedia answer.

We are given the extent of plagiarism for every answer in a separate file_information file. We use this file to check accuracy and precision of the results achieved by our algorithm.

There are four different levels of plagiarisms. They are cut, light, non and heavy. Cut refers to answers that are cut, copy, pasted from Wikipedia answers. Heavy refers to answers that are heavily copied from Wikipedia answers, light refers to answers which are slightly copied from Wikipedia answers with extreme changes to the structure of answers. Non refers to non-plagiarised work.

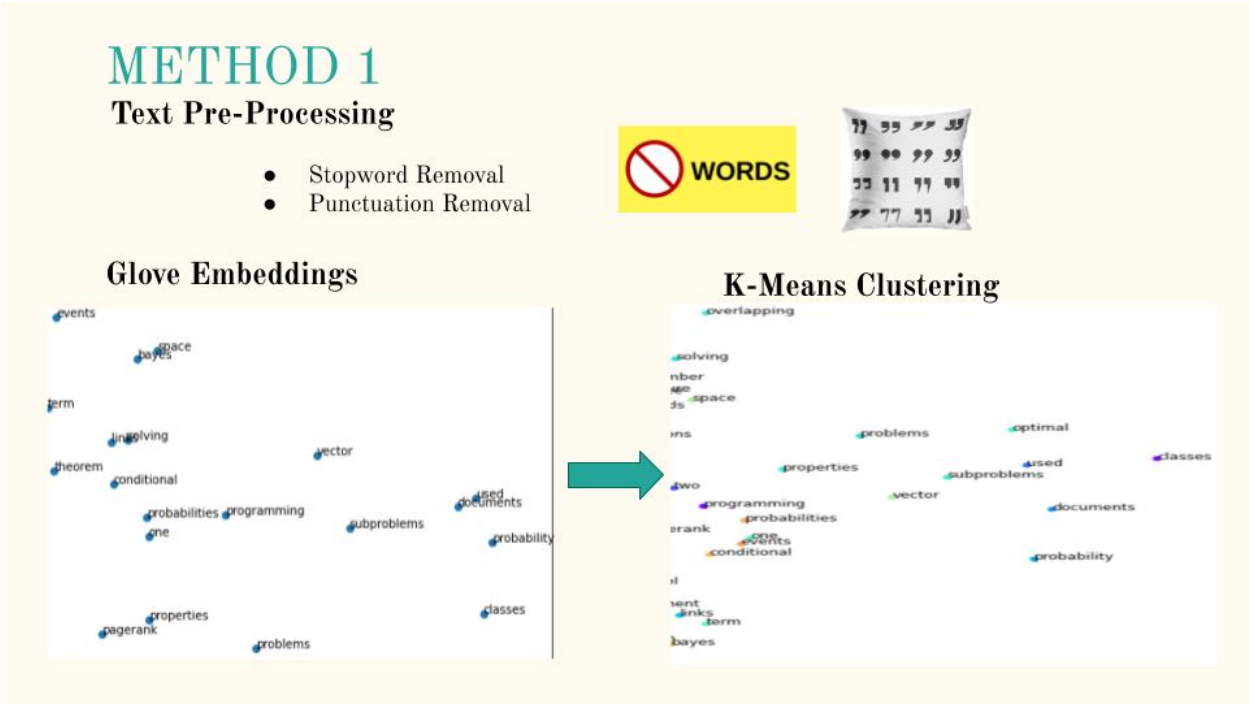


Project Goal:

The goal of this project is to find the extent of plagiarism in each student's submission for every question using unsupervised machine learning methods.
For this, we have Proposed two methods.

Method 1 Overview:

1. **Text Cleaning:** Initial step for building text models. Accuracy of machine learning models on textual data depends heavily on text pre-processing.
2. **Glove Embeddings:** GloVe is a log-bilinear model that is trained on the non-zero entries of a global word-word co-occurrence matrix, which tabulates how frequently words co-occur with one another in a given corpus. Output of this step is vector representation for each unique word in the corpus.
3. **K-Means Clustering:** Cluster these word vectors into K clusters based on their contextual meaning. The output of this step is K word clusters**.**
4. **Document Level Clustering:** From the obtained clusters and its respective labels from the K-means clustering, we now use these clusters to convert our data in each document to vectors.
5. **FP-Growth Algorithm:** In this step, we mine frequent patterns in documents based on sentence similarities which outputs a list of items which were plagiarised.
6. **Model Evaluation:** We evaluated the results manually calculating accuracy, precision, F1- score .



Text Pre-Processing:

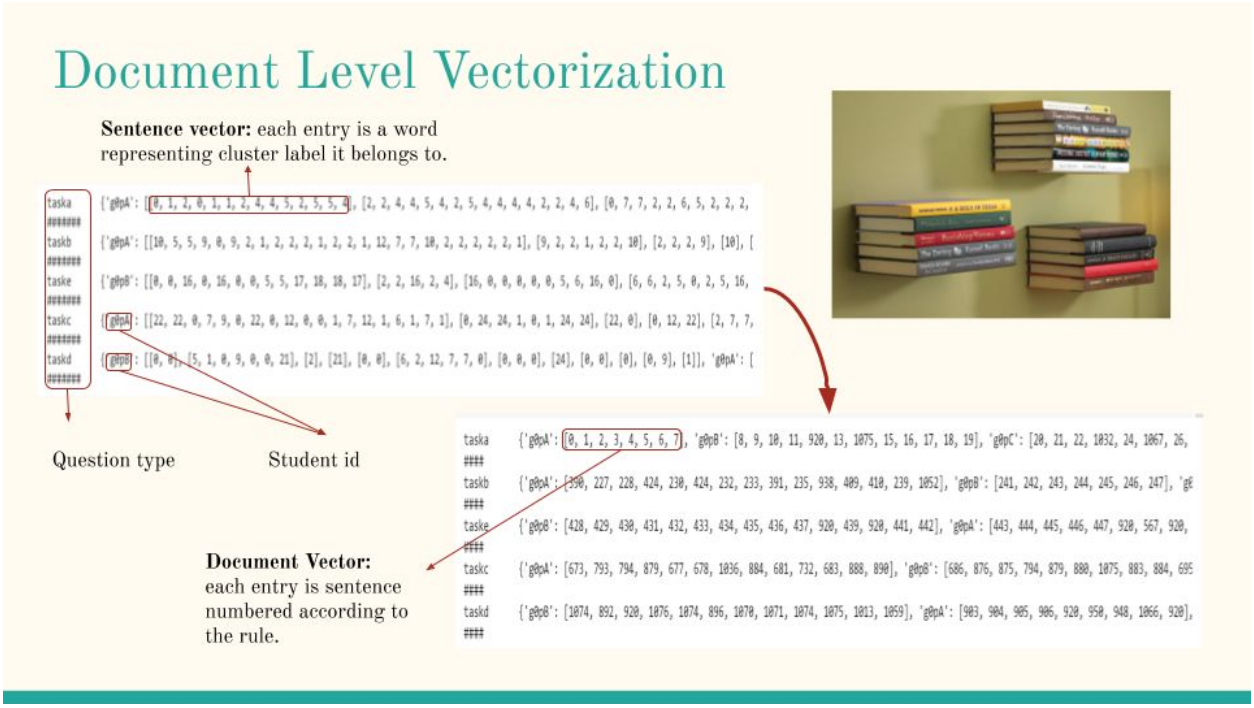
For pre-processing we have to make sure that all the words are lowercase with all the punctuations removed. We use the NLTK sentence tokenizer to get a list of sentences so that we know where each sentence begins and ends. There are various latin words in our answers as well so we use ”latin1” encoding for reading the text. Implemented EDA on text data to get insights like top most used words in each question, Bigram representation in word clouds etc.,

Glove Embeddings:

For creating vector word embeddings, we used a method called GloVe. GloVe is an unsupervised learning algorithm for obtaining vector representations for words. List of words in sentences of the documents is fed to the Algorithm. After performing GloVe on our preprocessed data we get vectors of each unique word in our corpus. We can use this representation for clustering and further processing.

K-means clustering:

K-means clustering is a type of unsupervised learning, which is used when we have unlabelled data. The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable K. We determined K value using Grid Search, a hyperparameter selection method. The output of this step is K (=25) word clusters.



Vector Representation for document-level embeddings:

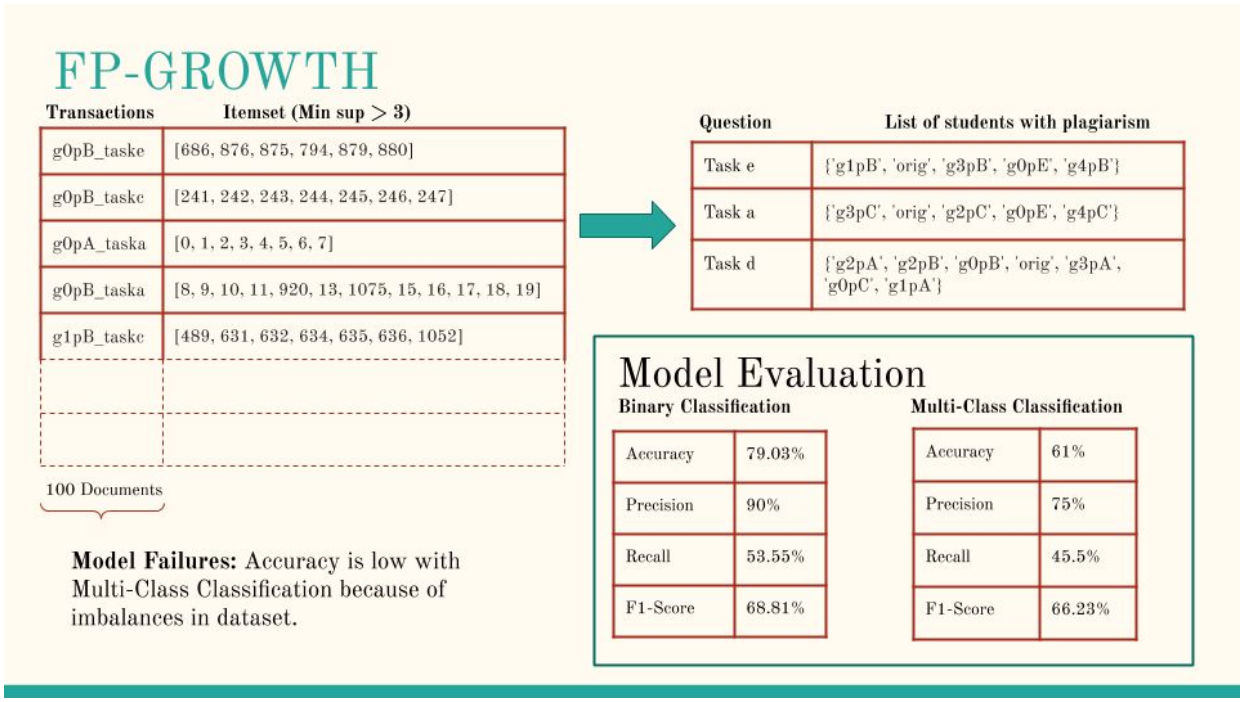
Sentence Vector: Each entry is a word representing the cluster label it belongs to.

Document Vector: Each entry represents sentence number according to the following rule.

From the obtained word clusters and its respective labels from the K-means clustering, we now use these clusters to convert data in each document to vectors. For this, we do the following:

- Determine the cluster label for each of the words in sentences and substitute the word with its respective cluster label.
- From the above step,we merge all the words and form sentence vectors.
- From these sentence vectors, we form an answer(document) vector by assigning a unique number if two sentence vectors are different. If two sentence vectors are the same, we assign each of them the same number.

Thus we get a vector for each answer(document).This vector can be viewed as an item-set and each answer can be viewed as a transaction. Thus we get a list of transactions which contain item-sets.We can feed this into the FP-Growth Algorithm to get frequent item sets,i.e., plagiarised answers.



FP-Growth Algorithm:

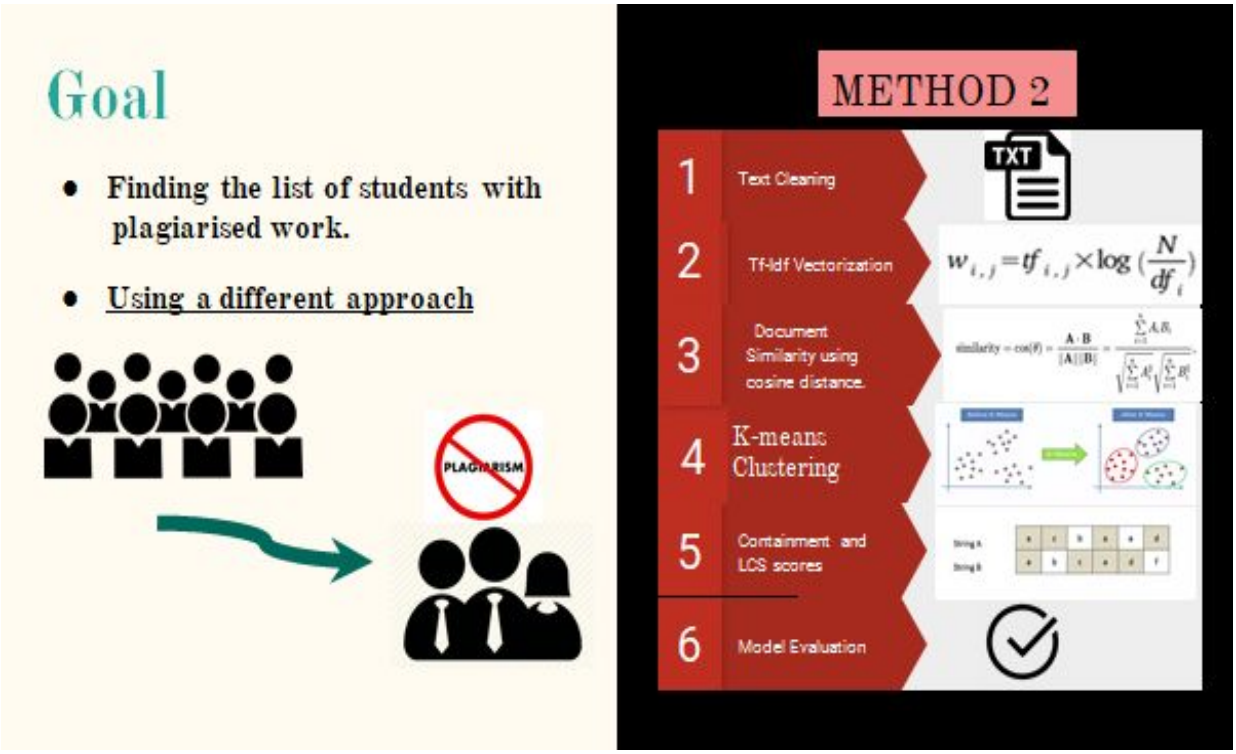
The FP-growth algorithm is currently one of the fastest approaches to frequent itemset mining. We can find which documents are plagiarised by just looking at the transactions which in our case are student answers. Here each student answer is one transaction and their vector representation of sentences are itemsets.

Model Evaluation:

Based on Support counts output is classified into 4 classes. For heavy, minsup is 8, for cut its 6, for light its 4 , for non its 3.
Multi-Class Classification: Evaluated final results with actual labels given in dataset i.e., light, heavy, cut, non. Accuracy is as low as 60% but high precision is to be noted. This is usually due to class imbalances.
Binary Classification: To Tackle Class Imbalance problems**,** Binary Classification is applied. Documents with light,cut,heavy labels are categorized as Plagiarized and with non as Non-Plagiarized. Accuracy is as high as 80% and precision is 90%

Method Failures:

This approach of using FP-Growth for Detecting Plagiarism got pretty decent results. Although, not absolutely great, the results were better than the traditional methods used for this task. The main drawback is insufficient data and class imbalances. One of the possible ways to tackle this problem is collecting more data as we can't regenerate data in these types of applications and experimenting with different support counts for class thresholds.



Pre-Processing Techniques

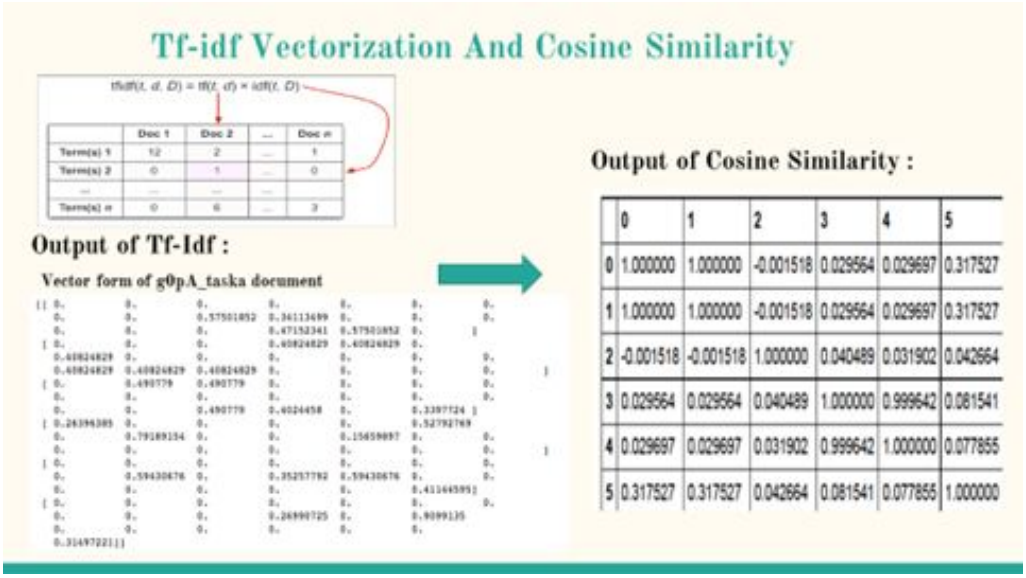
We have experimented with other unsupervised machine learning algorithms, to get better results for multi-class classification. So, we have implemented a whole new approach to achieve our objective.

We started our method-2 by cleaning and preprocessing our documents as the optimal quality and performance of the clustering methods depend largely on the pre-processing techniques. For each document in the corpus we followed the 3 main steps:

1) Normalization: This means to have each word in the uniform format. We achieved this by converting all text to the lower case, removing punctuation and converting numbers to their word equivalents.

2) Stemming: It is a process of reducing words to its root form. We have used Porter stemming algorithm for this. This is used to catch the similar words even if they are in its non-root forms.

3) Tokenization and removal of stop-words: This is the process of splitting the given text into smaller pieces called tokens. So this represents every document as a list of tokens without having the stop-words.



	0	1	2	3	4	5
0	1.000000	1.000000	-0.001518	0.029564	0.029697	0.317527
1	1.000000	1.000000	-0.001518	0.029564	0.029697	0.317527
2	-0.001518	-0.001518	1.000000	0.040489	0.031902	0.042664
3	0.029564	0.029564	0.040489	1.000000	0.999642	0.081541
4	0.029697	0.029697	0.031902	0.999642	1.000000	0.077855
5	0.317527	0.317527	0.042664	0.081541	0.077855	1.000000

After preprocessing to apply data mining techniques we have converted these words to a machine-readable format**.**So, to achieve a relationship between the important words that are not just placed adjacently in a document but also placed throughout the corpus we used Tf-idf vectorization.

Tf-idf Vectorization:

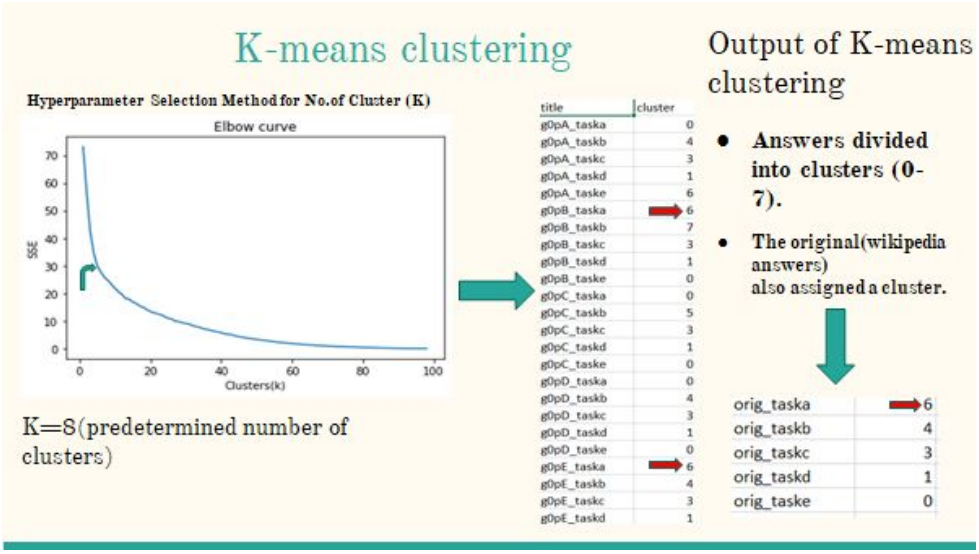
Term frequency (Tf): Count of how frequently that expression (term, word) occurs in the document.

Inverse document frequency (idf): It is used to calculate the weight of rare words across all documents in the corpus. This value is obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient.

Words that occur frequently within a document but not frequently within the corpus receive a higher weighting as they are assumed to contain more meaning in the document. Now each document in the corpus is represented in a vector form that contains the product of Tf and idf values for each word in the document.

Cosine Similarity:

This is measured against the tf-idf matrix and can be used to generate a measure of similarity between each student's answers and the other answers in the corpus. High values indicate a high level of similar words between two documents.



K-means clustering

The K-means clustering was iterated multiple times with different values of k(number of clusters) to compare the SSE(Sum of Squared Errors) and find the optimal value of k.

K-Means clustering was performed in two ways:

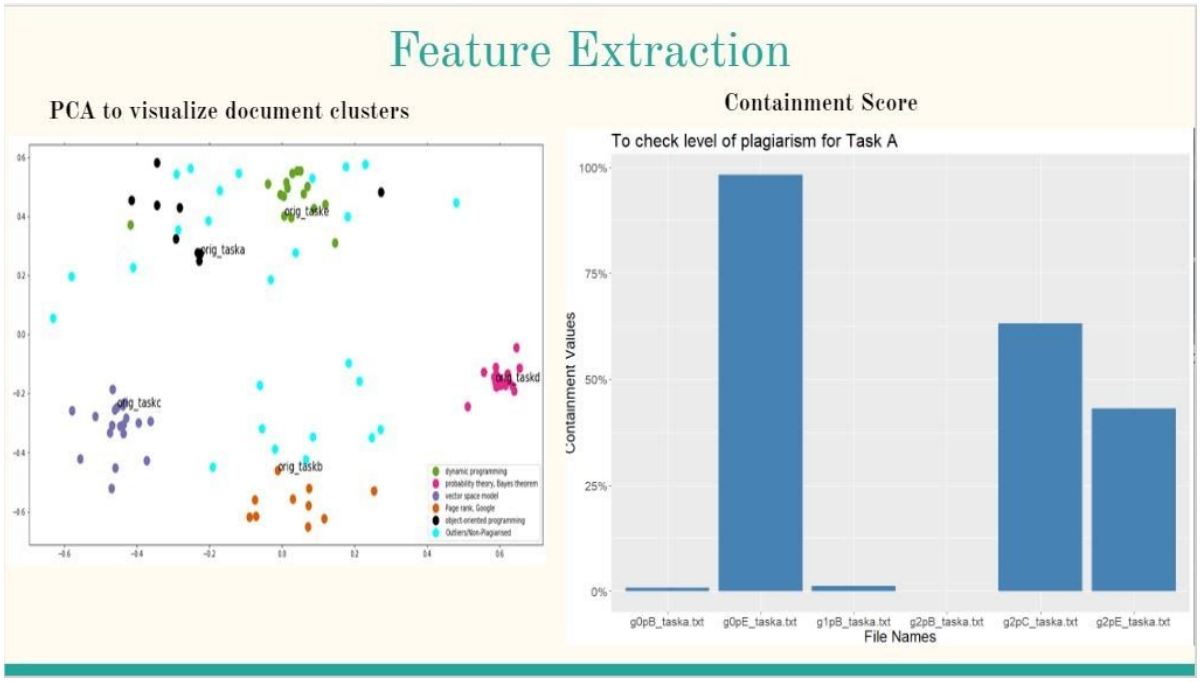
a) Only Using the Tf-idf matrix.
b) Reducing the dimensionality using PCA i.e by calculating Eigenvectors and Eigenvalues from the Cosine Similarity matrix and using k Eigenvectors corresponding to the topmost k Eigenvalues to cluster into k-dimensional document vectors.

The Elbow Method is an effective way to find the k parameter. If the line chart looks like an arm, then the "elbow" on the arm is the value of k that is the best.(In our case,k=8)

Failure of method using Tf-idf:

We have implemented method ‘b’ because we observed that method ‘a’ gave a rather smooth SSE curve with all high values of SSE whereas Method ‘b’ gave significantly low SSEs for any comparable value of k because of the reduced dimensions. For method ‘b’ we see that the SSE curve is shown above with an elbow near k=8.

Now our student answers and our reference(wikipedia) answers are assigned a cluster label from(0-7). So if the same cluster label is assigned to the reference and to the student answers that would imply there is some level of plagiarised work.



As shown in the above cluster figure we can see that there are student answers belonging to the same cluster as the reference answer , and there are some outliers not belonging to any of the clusters from the reference answers, thus the outliers are falling under non-plagiarised work.

Now to classify, upto which extent the students have plagiarised the work we calculate the Containment and the LCS scores in each cluster between the student answer document and the reference(wikipedia) document that is falling under the same cluster.

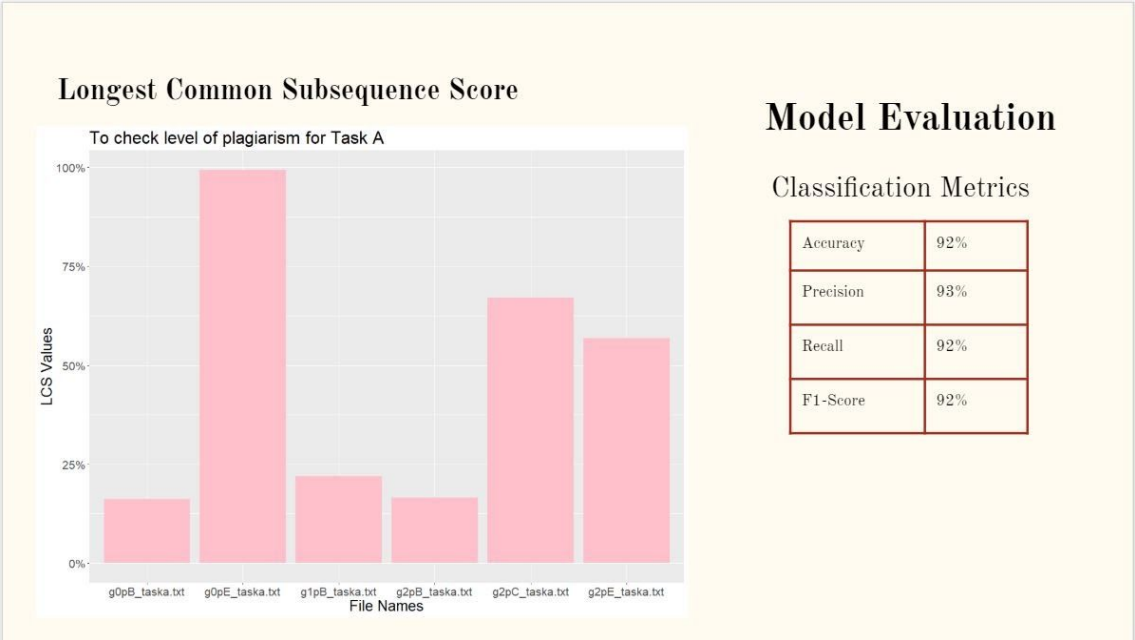
Containment Score:

Containment is a good way to find overlap in word usage between two documents. it may help identify cases of cut-and-paste as well as paraphrased levels of plagiarism.

We are calculating the containment values between an answer and source text belonging to the same cluster according to the following equation.

$$\frac{\sum count(\text{ngram}_A) \cap count(\text{ngram}_S)}{\sum count(\text{ngram}_A)}$$

High containment scores refers to high levels of plagiarised work. Since plagiarism is a complex task with varying levels of plagiarism, it's useful to include other measures of similarity too like LCS.



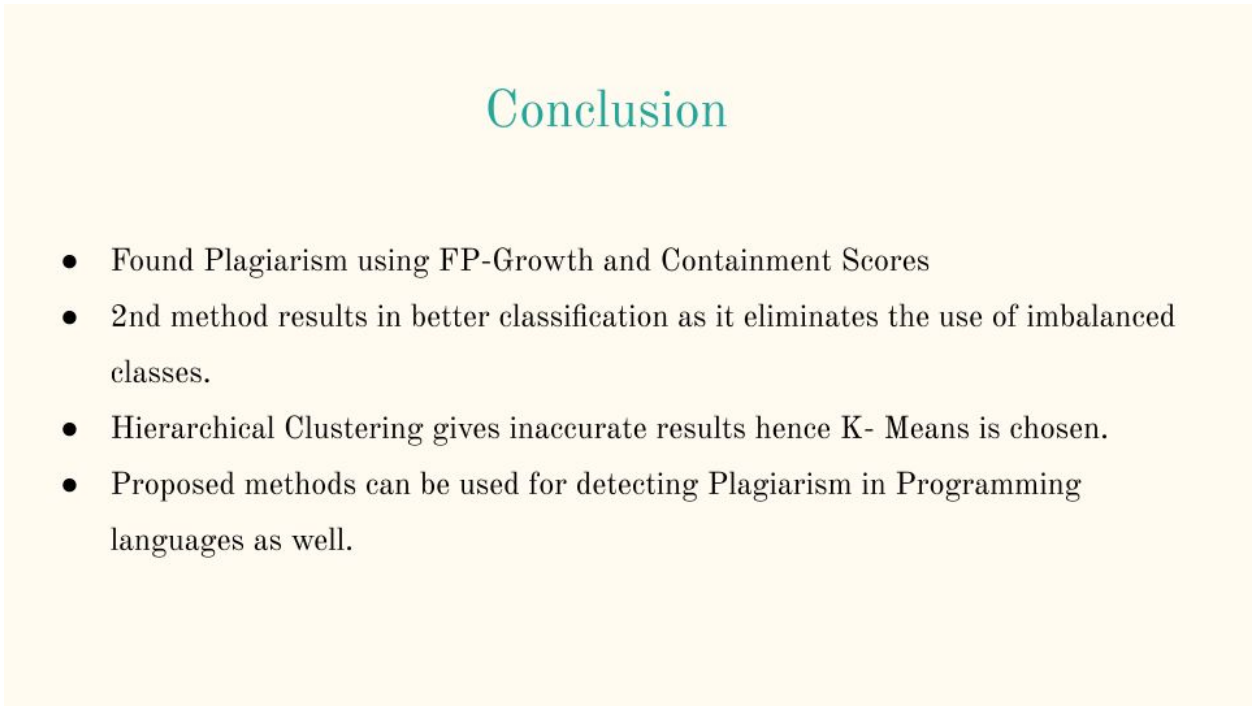
Longest Common Subsequence:

The longest common subsequence is the longest string of words that are the same between the Wikipedia Source Text and the Student Answer Text. This value is then normalized by dividing by the total number of words in the Student Answer Text. High LCS score refers to high levels of plagiarised work.

We can classify the student answers into different levels of plagiarism by looking at their LCS and containment scores by comparing them with the wikipedia answers. For eg : If the containment and the LCS scores are in the range of 80-100 % then we can say that it falls under the “cut” classification. If it is in the range of 50-80% then we can say that it falls under the “heavy” classification. If it is in the range of 15-50% then we can say that it falls under the “light” classification.

Model Evaluation:

Multi-Class Classification: Evaluated final results with actual labels given in dataset i.e., light, heavy, cut, non. We saw a high rise in accuracy as it is 92% and also a high rise in precision is to be noted.



Conclusion:

In our first method we have tried to find the extent of plagiarism using FP-growth and in the second method we have used Containment and LCS scores to the clusters. We have developed a system based on principles of vector representations. We also compared the results between Hierarchical Clustering and K-Means. K-means give better results than hierarchical clustering as the latter is not suitable for large datasets and Initial seeds have a strong impact on the final results thus giving us inaccurate cluster labels.

We have seen a rise in the accuracy for multi-class classification in the second method because we have clustered the abundant class and when we apply containment and LCS scores between documents in each cluster, we do not face the problem of imbalanced classes as it itself finds the similarity score and classifies according to the score.

As an extension to this work, we can include vector representations using other techniques such as word2vec or fast text. Another future work is to change the domain from documents to programming. In programming, we would need stricter noise removal as well as the support count of FP-growth algorithms will increase.