Classification of Documents Using Graph-Features & KNN

CS-380 Graph Theory

Session: 2021-2025

Project Supervisor

Waqas Ali

Group Members

Shahzaib Rafi	2021-CS-2
Muhammad Subhan	2021-CS-35

Department of Computer Science

University of Engineering and Technology, Lahore

Pakistan

‘

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

In the current massive global world of the Internet, the number of documents being produced now is higher than ever. Daily, thousands of documents are added to this vast collection. Automated classification is an important research topic aimed at reducing time and costs. The vector model used for this purpose was slow and had low accuracy because it ignored much information regarding text structure, and this database isn't equipped to handle such. Therefore, it was necessary to use some other data structure that not only allows for efficient storage and retrieval of additional information (i.e., text structure, which helps to better categorize) but also provides techniques and concepts to reduce time and costs. That's where graphs come into play.

1.2 Objectives

The main objective of this project is to develop a robust system for document classification that utilizes graph-based features and the KNN algorithm. Specific objectives include:

- Collecting a large dataset of documents from the predefined categories.
- Representing each document as a directed graph.
- Identifying common subgraphs within the training set of documents.
- Implementing the KNN algorithm based on graph similarity measures.
- Classifying test documents based on their similarity to training documents in the feature space defined by common subgraphs.

2 Literature Review

2.1 Classification of Web Documents Using a Graph Model

This paper was the first to propose the use of graphs instead of vectors for document classification. It highlights that vectors are inefficient as they fail to capture important details such as text structure and sequence. Since vectors cannot efficiently represent such information, the paper advocates for the use of graphs. In this approach, each document is represented as a directed graph where edges connect consecutive words, with weights indicating the frequency of consecutive word pairs. The key feature for classification is the largest common subgraph between two graphs. The paper introduces a formula to measure the distance between two graphs:

$$dist_{MCS}(G_1, G_2) = 1 - \frac{|mcs(G_1, G_2)|}{\max(|G_1|, |G_2|)}$$

Using this distance measure, the KNN algorithm operates similarly to its vector-based counterpart, with the difference lying in the distance measure used. The paper conducted experiments on three different datasets, and the results indicate that accuracy increases with the number of nodes in the graphs.

2.2 A graph distance metric based on the maximal common subgraph

This research paper discusses graph distance measures and proposes a new one.

When comparing or matching graphs, a distance measure is needed to quantify how similar or different they are. Edit distance is a common approach, but it requires assigning costs to edit operations (like adding or removing nodes/edges), which can be challenging. This paper proposes a new graph distance measure based on the maximal common subgraph (MCS) of two graphs. The MCS is the largest subgraph that is present in both graphs. The paper proves that this new measure satisfies the properties of a metric, which makes it mathematically sound for measuring distance. The benefit of this new measure is that it doesn't rely on pre-defined edit costs, making it potentially more applicable in various scenarios.

2.3 References

- A. Schenker, M. Last, H. Bunke and A. Kandel, "Classification of Web documents using a graph model," Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings., Edinburgh, UK, 2003, pp. 240-244 vol.1, doi: 10.1109/ICDAR.2003.1227666.
- Horst Bunke, Kim Shearer, A graph distance metric based on the maximal common subgraph, Pattern Recognition Letters, Volume 19, Issues 3–4, 1998, Pages 255-259, SSN 0167-8655, https://doi.org/10.1016/S0167-8655(97)00179-7

3 Methodolgy

3.1 Data Collection and Preparation

3.1.1 Scraping

For each topic/category assigned to us, we looked for seperate blog websites to find the ones with the most relevance to the topic. Each website had a seperate html structure and implementation so each required a seperate script for it. The predefined topics assigned to our group along with the website used were:

- Food (newfoodmagazine.com)
- Sports (espn.in)
- Business and Finance (businessblogshub.com

So after understanding each website layout, we went for the the following **tools and libraries:**

- **Selenium:** Web automation library for controlling web browsers and automating web interactions.
- **Requests:** HTTP library for sending HTTP requests and interacting with web servers.
- **Beautiful Soup (bs4):** Python library for parsing HTML and XML documents, providing tools for web scraping and extracting data from web pages.

The need for multiple libraries was due to some website only being scrapped by selenium and some requiring to be scrolled again and again as they employ lazy loading.

Table 1: Scraping Statistics

Category	**Number of Documents**	**Avg Words**
Business and Finance	20	976
Sports	16	767
Food	20	854

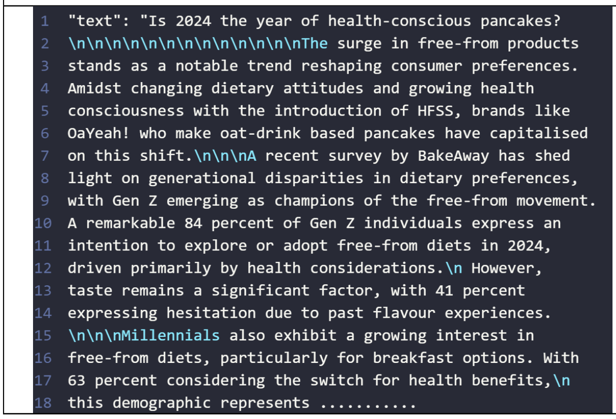
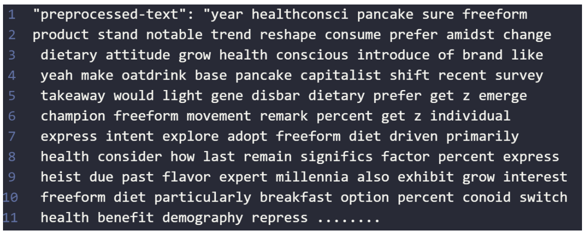
3.1.2 Preprocessing

It is the process of breaking down and removing unnecessary words from the text. We perform following preprocessing operations:

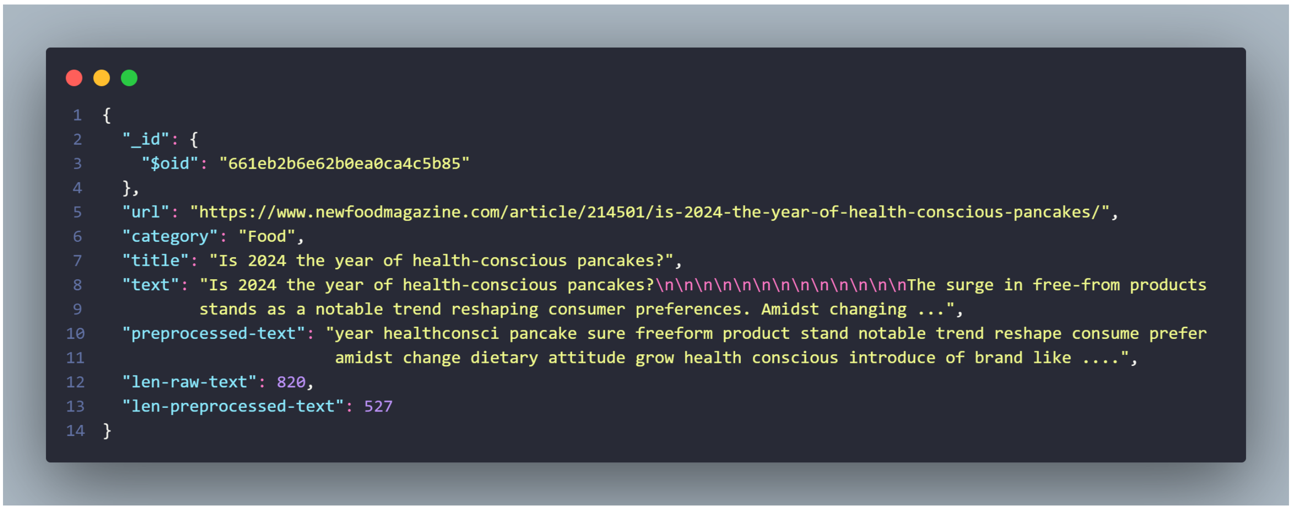
- **Stem Words:** Reducing words to their base or root form. Example: ”running” $->$ ”run”.
- **Lemmatize Words:** Similar to stemming but ensures resulting words are valid lemmas. Example: ”better” $->$ ”good”.
- **Correct Spellings:** Identifying and fixing misspelled words. Example: ”writting” $->$ ”writing”.
- **Remove Itemized Bullets and Numbering:** Eliminating itemized lists or numbering from text. Example: ”1. First item” $->$ ”First item”.
- **Remove URLs and Stopwords:** Removing web addresses and commonly occurring words (e.g., ”the”, ”is”). .

- **Remove HTML Tags:** Stripping HTML elements from text. Example: "$< p > Hello < /p >$" $- >$ "Hello".
- **Expand Contractions:** Expanding contracted forms of words. Example: "I'm" $- >$ "I am".
- **Remove Emails, Special Characters, Phone Numbers and Numbers:** Deleting email addresses, special characters, phone numbers, and numerical digits. Example: "john@example.com" $- >$ "" (removed).
- **Normalize Unicode:** Standardizing Unicode characters to their closest ASCII equivalents. Example: "ñ" $- >$ "n".
- **Remove Emoticons and Emojis:** Deleting emoticons and emojis from text. Example: ":)" $- >$ "" (removed).
- **Remove Punctuation:** Stripping punctuation marks from text. Example: "Hello!" $- >$ "Hello".
- **Remove Whitespace:** Eliminating extra spaces and line breaks. Example: " Hello " $- >$ "Hello".

Table 2: Comparison of text before and after preprocessing

Before Preprocessing	After Preprocessing
 "text": "Is 2024 the year of health-conscious pancakes? \n\n\n\n\n\n\n\n\n\n\n\nThe surge in free-from products stands as a notable trend reshaping consumer preferences. Amidst changing dietary attitudes and growing health consciousness with the introduction of HFSS, brands like OaYeah! who make oat-drink based pancakes have capitalised on this shift.\n\n\nA recent survey by BakeAway has shed light on generational disparities in dietary preferences, with Gen Z emerging as champions of the free-from movement. A remarkable 84 percent of Gen Z individuals express an intention to explore or adopt free-from diets in 2024, driven primarily by health considerations.\n However, taste remains a significant factor, with 41 percent expressing hesitation due to past flavour experiences. \n\n\nMillennials also exhibit a growing interest in free-from diets, particularly for breakfast options. With 63 percent considering the switch for health benefits,\n this demographic represents	 "preprocessed-text": "year healthconsci pancake sure freeform product stand notable trend reshape consume prefer amidst change dietary attitude grow health conscious introduce of brand like yeah make oatdrink base pancake capitalist shift recent survey takeaway would light gene disbar dietary prefer get z emerge champion freeform movement remark percent get z individual express intent explore adopt freeform diet driven primarily health consider how last remain significs factor percent express heist due past flavor expert millennia also exhibit grow interest freeform diet particularly breakfast option percent conoid switch health benefit demography repress ....

Following is an example document scrapped, preprocessed and stored in mongo database:



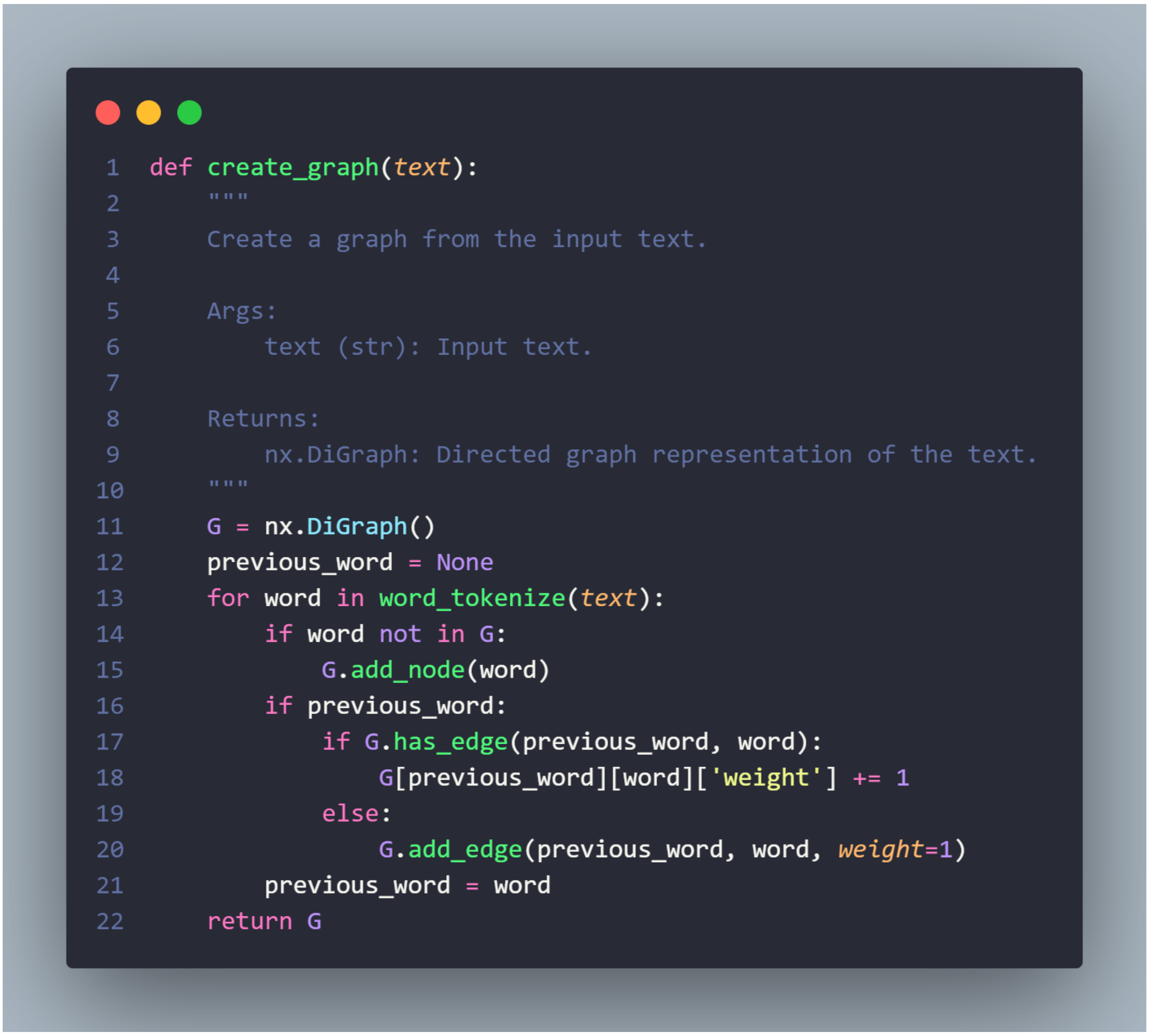
```
{
  "_id": {
    "$oid": "661eb2b6e62b0ea0ca4c5b85"
  },
  "url": "https://www.newfoodmagazine.com/article/214501/is-2024-the-year-of-health-conscious-pancakes/",
  "category": "Food",
  "title": "Is 2024 the year of health-conscious pancakes?",
  "text": "Is 2024 the year of health-conscious pancakes?\n\n\n\n\n\n\n\n\n\n\n\nThe surge in free-from products
          stands as a notable trend reshaping consumer preferences. Amidst changing ...",
  "preprocessed-text": "year healthconsci pancake sure freeform product stand notable trend reshape consume prefer
                       amidst change dietary attitude grow health conscious introduce of brand like ....",
  "len-raw-text": 820,
  "len-preprocessed-text": 527
}
```

Figure 1: Example document stored in a MongoDB

3.2 Graph Construction

After preprocessing, the next step is to create a directed graph for each document. For this purpose, we used the **networkx** library because of its ease of use and compatibility with other libraries for further processing and visualization.

The order of words was maintained in the graph. Each unique word from the preprocessed document becomes a node in the graph. An edge was created for every two consecutive words, with the tail pointing to the previous word and the head pointing to the following word, initially with a weight of 1, which would increase by 1 for every such relationship found. Following is the code snippet:



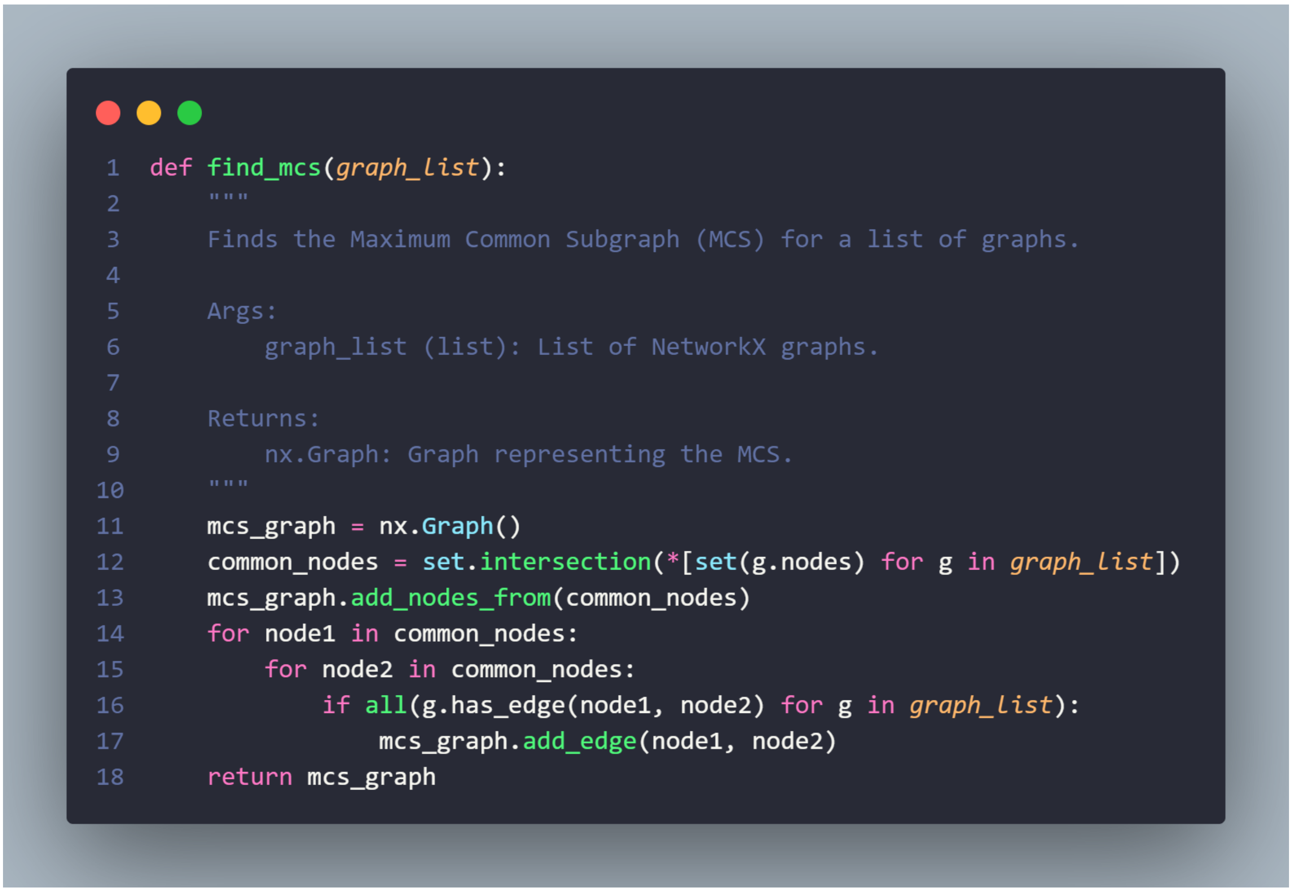
```python
def create_graph(text):
    """
    Create a graph from the input text.

    Args:
        text (str): Input text.

    Returns:
        nx.DiGraph: Directed graph representation of the text.
    """
    G = nx.DiGraph()
    previous_word = None
    for word in word_tokenize(text):
        if word not in G:
            G.add_node(word)
        if previous_word:
            if G.has_edge(previous_word, word):
                G[previous_word][word]['weight'] += 1
            else:
                G.add_edge(previous_word, word, weight=1)
        previous_word = word
    return G
```

Figure 2: Code snippet for constructing directed graphs

3.3 Feature Extraction via Common Subgraphs

The next step is to identify the feature(s) to correctly classify documents. That feature is the **maximal common subgraph (the largest common subgraph between two graphs)**.

The approach we took to find an MCS between two graphs is to first find all the common nodes. Then, for each node in the common nodes, we check against all the others for an edge to exist in both graphs. Only then do we add that edge in the MCS. Following is the code snippet for that:



```python
def find_mcs(graph_list):
    """
    Finds the Maximum Common Subgraph (MCS) for a list of graphs.

    Args:
        graph_list (list): List of NetworkX graphs.

    Returns:
        nx.Graph: Graph representing the MCS.
    """
    mcs_graph = nx.Graph()
    common_nodes = set.intersection(*[set(g.nodes) for g in graph_list])
    mcs_graph.add_nodes_from(common_nodes)
    for node1 in common_nodes:
        for node2 in common_nodes:
            if all(g.has_edge(node1, node2) for g in graph_list):
                mcs_graph.add_edge(node1, node2)
    return mcs_graph
```

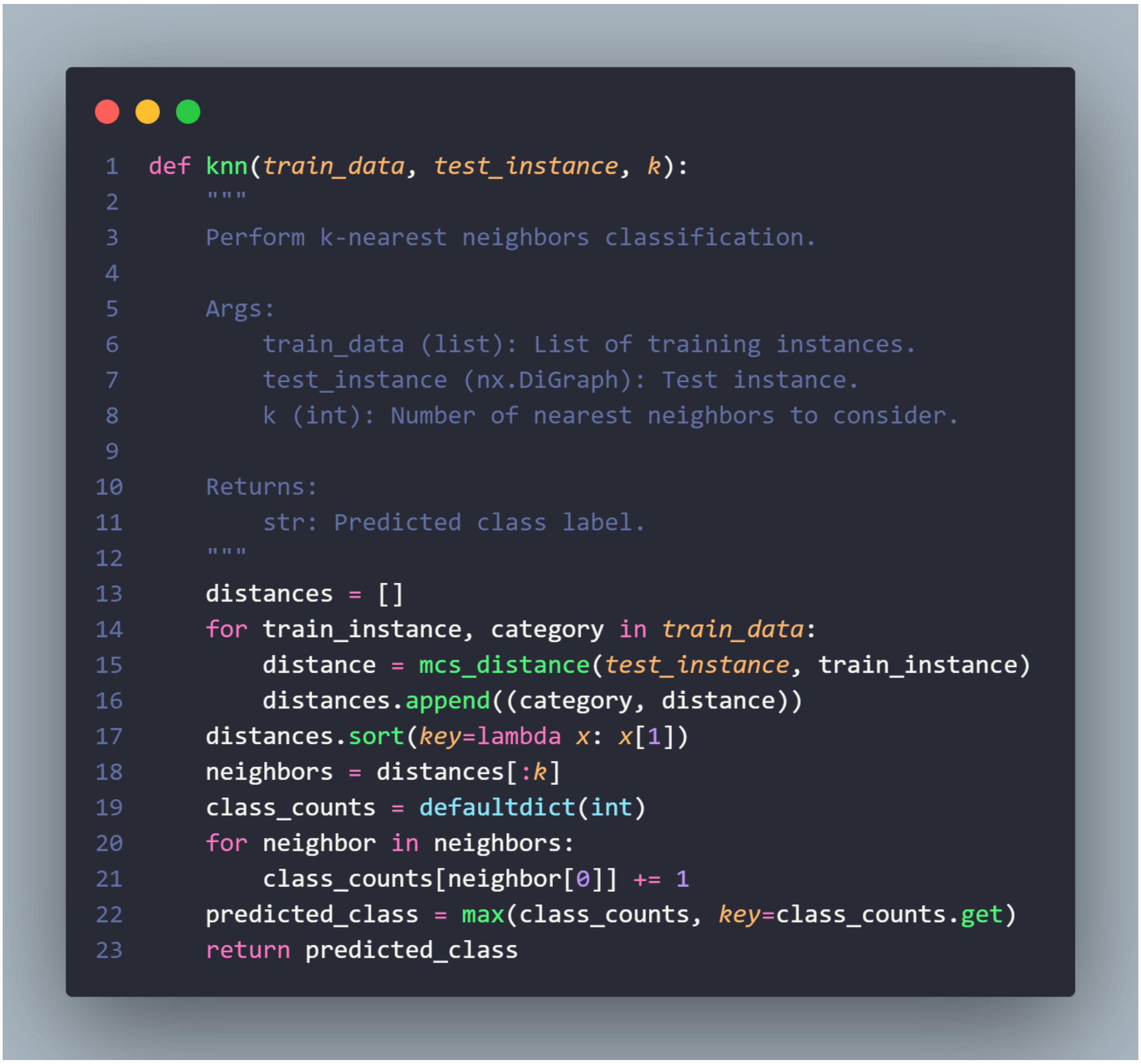
Figure 3: Code snippet for finding the maximal common subgraph

3.4 Classification with KNN

KNN basically classify by looking for which graph is closest to the input graph. This involves computing the similarity between graphs by evaluating their shared structure. If there's a tie, the class label with the highest count is chosen as the predicted class.The distance measure we used to compute the distance between two graphs is the same as proposed in the first paper we reviewd:

$$dist_{MCS}(G_1, G_2) = 1 - \frac{|mcs(G_1, G_2)|}{\max(|G_1|, |G_2|)}$$

Following is the code snippet for that:



```python
def knn(train_data, test_instance, k):
    """
    Perform k-nearest neighbors classification.

    Args:
        train_data (list): List of training instances.
        test_instance (nx.DiGraph): Test instance.
        k (int): Number of nearest neighbors to consider.

    Returns:
        str: Predicted class label.
    """
    distances = []
    for train_instance, category in train_data:
        distance = mcs_distance(test_instance, train_instance)
        distances.append((category, distance))
    distances.sort(key=lambda x: x[1])
    neighbors = distances[:k]
    class_counts = defaultdict(int)
    for neighbor in neighbors:
        class_counts[neighbor[0]] += 1
    predicted_class = max(class_counts, key=class_counts.get)
    return predicted_class
```

Figure 4: Code Snippet for Categorization using KNN

3.5 Evaluation

For evaluation, we considered following metrics:

Precision: Precision measures the proportion of true positive predictions out of all positive predictions made by the model. It is calculated as $\frac{\text{true positives}}{\text{true positives}+\text{false positives}}$.

Recall: Recall, also known as sensitivity or true positive rate, measures the proportion of true positive predictions out of all actual positives in the dataset. It is calculated as $\frac{\text{true positives}}{\text{true positives}+\text{false negatives}}$.

F1-Score: The F1-score is the harmonic mean of precision and recall. It provides a balance between precision and recall. It is calculated as $\frac{2\times\text{precision}\times\text{recall}}{\text{precision}+\text{recall}}$.

Confusion Matrix: A confusion matrix is a table that summarizes the performance of a classification model. It presents the counts of true positive (TP), false positive (FP), true negative (TN), and false negative (FN) predictions made by the model.

First we applied k-nearest neighbors (KNN) algorithm to classify each test instance by comparing it with the training instances.The test tain split was 20 80 percent. Then, we used

sklearn.metrics which collected the predicted classes and true labels for each test instance to evaluate the performance of the classification algorithm.

3.5.1 On 80 20 Split

Following is the result with the graphs on a 80 20 split:

Table 3: Classification Metrics using Graph on 80% 20% split

	Precision	Recall	F1-score	Support
Business & Finance	0.68	0.83	0.75	18
Food	0.81	0.94	0.87	18
Sports	1.00	0.61	0.76	18
Accuracy	-	-	0.80	54
Macro Avg	0.83	0.80	0.79	54
Weighted Avg	0.83	0.80	0.79	54

Afterwards, we ran the whole operation using vectors which yield following results:

Table 4: Classification Metrics using Vector on 80% 20% split

	Precision	Recall	F1-score	Support
Business & Finance	0.47	0.78	0.58	18
Food	0.75	1.00	0.86	18
Sports	0.00	0.00	0.00	18
Accuracy	-	-	0.59	54
Macro Avg	0.41	0.59	0.48	54
Weighted Avg	0.41	0.59	0.48	54

3.5.2 On 85 15 Split

Following is the result with the graphs on a 85 15 split:

Table 5: Classification Metrics using Graph on 85% 15% split

	Precision	Recall	F1-score	Support
Business & Finance	1.00	1.00	1.00	4
Food	1.00	1.00	1.00	4
Sports	1.00	1.00	1.00	4
Accuracy	-	-	1.00	12

Macro Avg	1.00	1.00	1.00	12
Weighted Avg	1.00	1.00	1.00	12

Afterwards, we ran the whole operation using vectors which yield following results:

Table 6: Classification Metrics using Vector on 85% 15% split

	Precision	Recall	F1-score	Support
Business & Finance	1.00	1.00	1.00	4
Food	1.00	1.00	1.00	4
Sports	1.00	1.00	1.00	4
Accuracy	-	-	1.00	12
Macro Avg	1.00	1.00	1.00	12
Weighted Avg	1.00	1.00	1.00	12

The confusion matrix of both implementation is same:

Table 7: Confusion Matrix

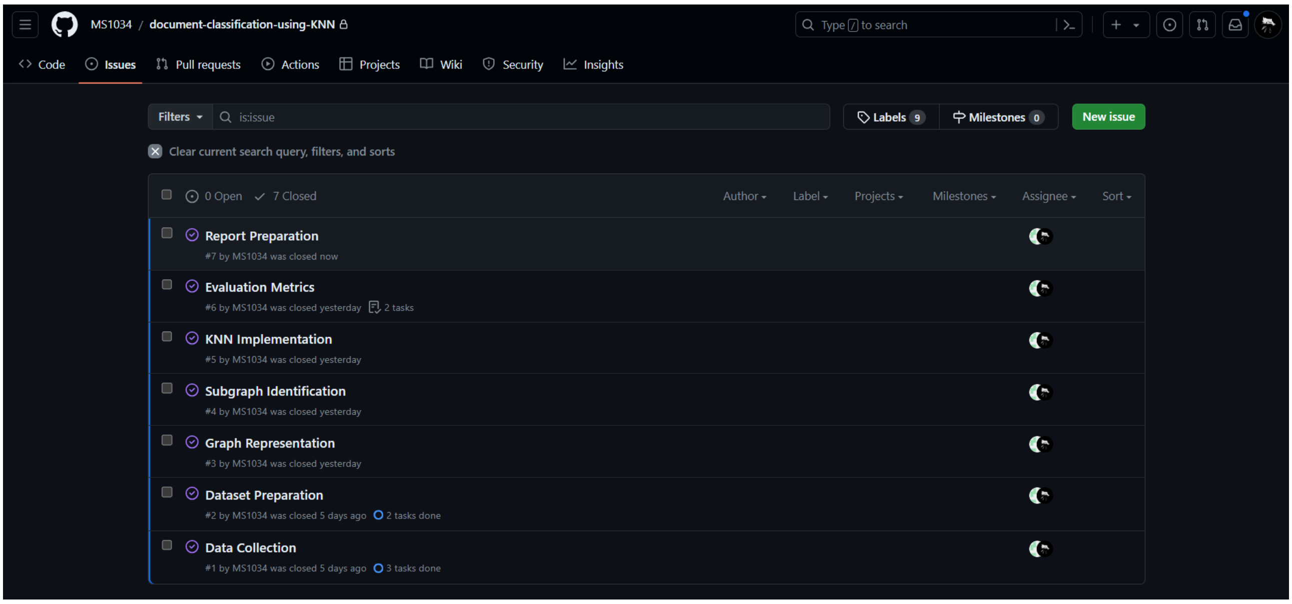
	Class 1	Class 2	Class 3
Class 1	4	0	0
Class 2	0	4	0
Class 3	0	0	4

So the graph implementation clearly yields better results and in the testing we did, we also found to be accurate than the vector implementation even at less data like (15% training data).

4 Project Management

The entire project was managed on GitHub. Visit here.

We created issues on GitHub to better cooperate and keep track of the project. Each issue primarily acted as a project milestone:



Report Preparation
#7 by MS1034 was closed now

Evaluation Metrics
#6 by MS1034 was closed yesterday 2 tasks

KNN Implementation
#5 by MS1034 was closed yesterday

Subgraph Identification
#4 by MS1034 was closed yesterday

Graph Representation
#3 by MS1034 was closed yesterday

Dataset Preparation
#2 by MS1034 was closed 5 days ago 2 tasks done

Data Collection
#1 by MS1034 was closed 5 days ago 3 tasks done

Figure 5: Project Management on Github

5 Challenges Faced:

From the start to the end of the project, we faced some problems, and we were able to overcome all of them:

- Some websites couldn't be accessed by the requests library, forcing us to explore other libraries.
- A website had slow loading, so we had to write a script to continuously scroll.
- Implementation of MCS as it was an NP problem, and our implementation needed to be in polynomial time and yield results.
- Selection of hyperparameters like K and choosing whether to use the number of edges or the number of nodes in the distance measure for the most optimized results.

6 Conclusion

In this project, we explored the use of graph-based methods for document classification, moving away from traditional vector-based approaches. By using the networkx library, we built directed graphs from preprocessed documents, keeping the word order intact and capturing sequential connections. Our method for feature extraction, focusing on maximal common subgraphs (MCS), proved effective in recognizing significant patterns for classification purposes.

Throughout our project journey, we encountered several methodological challenges, such as complexities in web scraping and optimizations in algorithms. Overcoming these hurdles required careful deliberation and continuous refinement of our techniques. Despite the inherent difficulty in efficiently implementing MCS, our persistent efforts led to the creation of a robust classification model.