

# Classification of Documents Using Graph-Features KNN

CS-380 Graph Theory



Submitted By:

**Syed Abdul Rehman**  
**Hammad Ejaz**

**2021-CS-62**  
**2021-CS-102**

Submitted To:

**Mr. Waqas Ali**

**Department of Computer Science**  
University of Engineering and Technology, Lahore

## **Acknowledgement**

We are grateful to Allah Almighty that he provided me with the strength and power and we want to say a big thank you to our project supervisor, Waqas Ali, for guiding and supporting us throughout the completion of our term project. His help was important in making sure we finished the project on time. we also want to thank our family and friends for encouraging us and giving us helpful feedback. They were always there for us and helped us stay on track to finish the project successfully. Lastly, we are very grateful to our family for always loving and supporting us during our academic journey. Their support helped us to achieve this milestone.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Background . . . . .	1
1.2	Objectives . . . . .	1
<b>2</b>	<b>Methodolgy</b>	<b>1</b>
2.1	Data Collection and Preparation . . . . .	1
2.1.1	Scraping . . . . .	1
2.1.2	Preprocessing . . . . .	2
2.1.3	Graph Construction . . . . .	3
2.2	Feature Extraction . . . . .	4
2.3	Classification . . . . .	5
2.4	Evaluation . . . . .	5
<b>3</b>	<b>Project Management</b>	<b>6</b>
<b>4</b>	<b>Challenges Faced:</b>	<b>7</b>
<b>5</b>	<b>Conclusion:</b>	<b>7</b>

## List of Figures

1	Before Preprocessing . . . . .	2
2	After Preprocessing . . . . .	3
3	Create graph code . . . . .	3
4	Example graph . . . . .	4
5	MCS algorithm . . . . .	4
6	kNN algorithm . . . . .	5
7	Project Management on Github . . . . .	7

# 1 Introduction

## 1.1 Background

In today's huge online world, there's a ton of new stuff being written every day. Trying to organize all of it is a big challenge. Researchers are working on ways to do this automatically to save time and money. The old way of doing it, called the vector model, was slow and not very accurate because it didn't understand how text is structured. So, they needed a better way to store and organize all this info, and that's where graphs come in. Graphs are like a cool new tool that can help sort through all the text quickly and accurately, making things easier and cheaper for everyone involved.

## 1.2 Objectives

The primary aim of this project is to create a sturdy document classification system, leveraging graph-based characteristics alongside the KNN algorithm. Specific goals encompass:

- Develop a document classification system.
- Represent each document as a directed graph.
- Identify common subgraphs within the documents.
- Utilize the K-Nearest Neighbors (KNN) algorithm.
- Base the classification on graph similarity measures.
- Aim to classify documents into predefined topics efficiently and accurately.

# 2 Methodolgy

## 2.1 Data Collection and Preparation

### 2.1.1 Scraping

For each assigned topic, namely Food, Travel, and Sport, we sourced data from Wikipedia. Each topic had its own dedicated Wikipedia page, each with a distinct HTML structure, necessitating separate scripts for data retrieval. The tools and libraries utilized for this task were:

- **Requests:** Employed for sending HTTP requests and interacting with web servers to fetch Wikipedia page content.
- **Beautiful Soup (bs4):** Utilized for parsing HTML documents and extracting relevant data from Wikipedia pages.
- **Regular Expressions (re):** Applied for pattern matching and text processing to refine the extracted data.

This approach was necessary due to variations in Wikipedia page layouts and content presentation. Below are the details of the data retrieved:

Topic	Number of Articles	Average Word Count
Food	16	800
Travel	15	750
Sport	15	700

These statistics showcase the quantity and average word count of articles retrieved for each topic.



```

Travel - Wikipedia
Jump to content,Main page,Contents,Current events,Random article,About
Wikipedia,Contact us,Donate,Help,Learn to edit,Community portal,Recent
changes,Upload file,Create account,Log in,learn
more,Contributions,Talk,Afrikaans,Asturianu,Azərbaycanca,Català,etina,Cymraeg,Da
nsk,Deutsch,Deutsch,Eesti,Español,Esperanto,Euskara,Français,Frysk,Gaeilge,Galeg
o,Hausa,Hawaiian,Hrvatski,Bahasa Indonesia,Italiano,Jawa,Kiswahili,Kreyl
ayisyen,Latina,Latvian,Lietuvi,Magyar,Nederlands,Norsk bokmål,Norsk
nynorsk,Ozbekcha,Plattdütsch,Polski,Portuguese,Română,Runa Simi,Scots,Shqip,Simple
English,Slovenian,Soomaaliga,srpski,Srpskohrvatski,Svenska,Tagalog,Tačlit,Trke
,Türkmençe,Tiếng Việt,emaitka,Edit links,Article,Talk,Read,View source,View
history,Read,View source,View history,What links here,Related changes,Upload
file,Special pages,Permanent link,Page information,Cite this page,Get
shortened URL,Download QR code,Wikidata item,Download as PDF,Printable
version,Wikimedia Commons,Wikiquote,Wikivoyage,Travel
disambiguation,Travelling disambiguation,Hong Kong,most visited
city,Euromonitor,CrossHarbour Tunnel,MTR,a series,Homestays,Hospitality
exchange services,BeWelcome,Couchsurfing,Dachgeber,Hospitality Club,Pasporta
Servo,Servas International,Trustroots,Warm
Showers,HelpX,Workaway,WWOOF,9flats,Airbnb,Booking.com,GuestReady,misterbb,Vrb
o,Home exchange,Friendship Force International,HomeExchange.com,Intervac
International,ThirdHome,v,t,e,Train,Nilgiri Mountain
Railway,Mettupalayam,Ootacamund,Tamil
Nadu,India,locations,foot,bicycle,automobile,train,boat,bus,airplane,ship,lugg
age,1,tourism,Old French,2,citation needed,Mount Everest,Amazon
rainforest,extreme tourism,adventure travel,bus,cruise ship,bullock
cart,3,recreation,4,5,tourism,4,vacationing,4,research,4,volunteer
travel,charity,migration,pilgrimages,4,mission trips,business
travel,4,trade,4,commuting,4,fleeing war,humanpowered
transport,walking,bicycling,vehicles,public
transport,automobiles,trains,ferries,boats,cruise
ships,airplanes,Pleasure,6,Relaxation,Discovery,exploration,4,Adventure,Interc

```

Figure 2: After Preprocessing

### 2.1.3 Graph Construction

After preprocessing of the data, the next task was to construct graphs from the data to use in the classification algorithm. We settled on **networkx** python package for graph construction. We followed the format give in Paper 1. The data was divided into 3 section: **Title**, **Link** and **Text**. In each section, if **b** follows **a**, then an edge will be created with the section being the label of the edge **a-text-b**. Punctuation were taken into account and for simplification, multiple edges nodes were not allowed. For debugging, we used **gravis** python library for graph visualisation. The following shows the graph creation function and an example graph.

```

def create_graph(title, links, text):
    graph = nx.DiGraph()

    graph.add_nodes_from(get_nodes(title, links, text))

    # Title Section
    process_section_tokens(graph, remove_stopwords(get_tokens(title, False)), "title")

    # Links Section
    for link in links:
        process_section_tokens(graph, remove_stopwords(get_tokens(link, False)), "link")

    # Text Section
    process_section_tokens(graph, remove_stopwords(get_tokens(text, False)), "text")

    # return gv.vis(graph, edge_label_data_source = "section")
    return graph

```

Figure 3: Create graph code

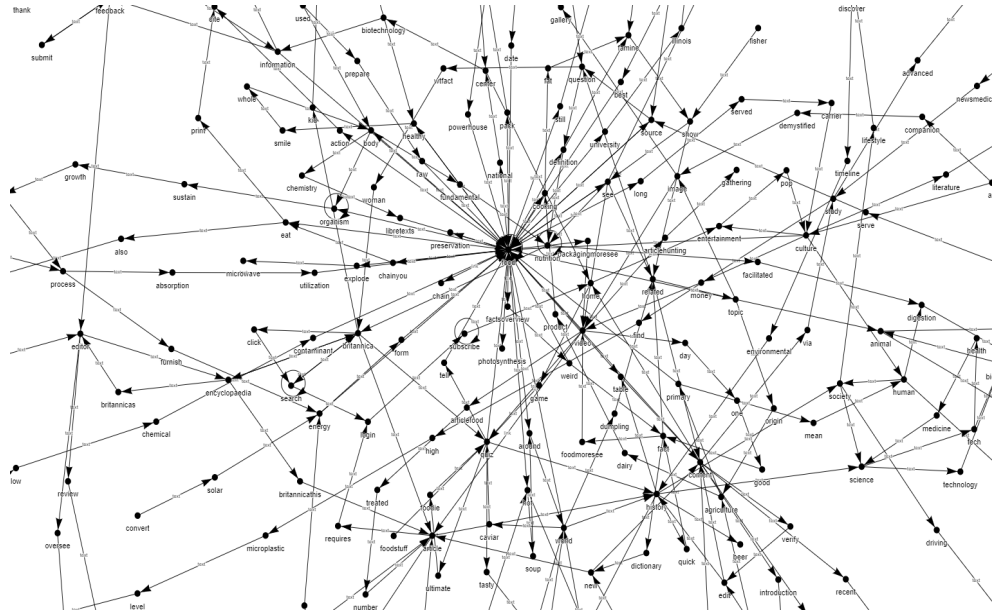


Figure 4: Example graph

## 2.2 Feature Extraction

Next we had to perform feature extraction to use in the classification algorithm. Again, using the approach of Paper 1. We used maximum common graph (MCS) as the feature for the classification. Extraction of MCS riddled us for a while since it is an NP-complete problem. After exploring VF2, GSPAN, ISAMGS, we settled for a simpler algorithm which first extracts all common parts between two graphs and gives the connected component with the largest number of nodes, since the MCS needs to be a connected subgraph of the original graphs. Following is the algorithm we use:

```
def mcs(g1, g2):
    matching_graph = nx.Graph()

    for n1,n2 in g2.edges():
        if g1.has_edge(n1, n2):
            matching_graph.add_edge(n1, n2)

    components = list(nx.connected_components(matching_graph))

    if len(components) == 0:
        return nx.DiGraph()

    largest_component = max(components, key = len)
    return nx.induced_subgraph(g1, largest_component)
```

Figure 5: MCS algorithm

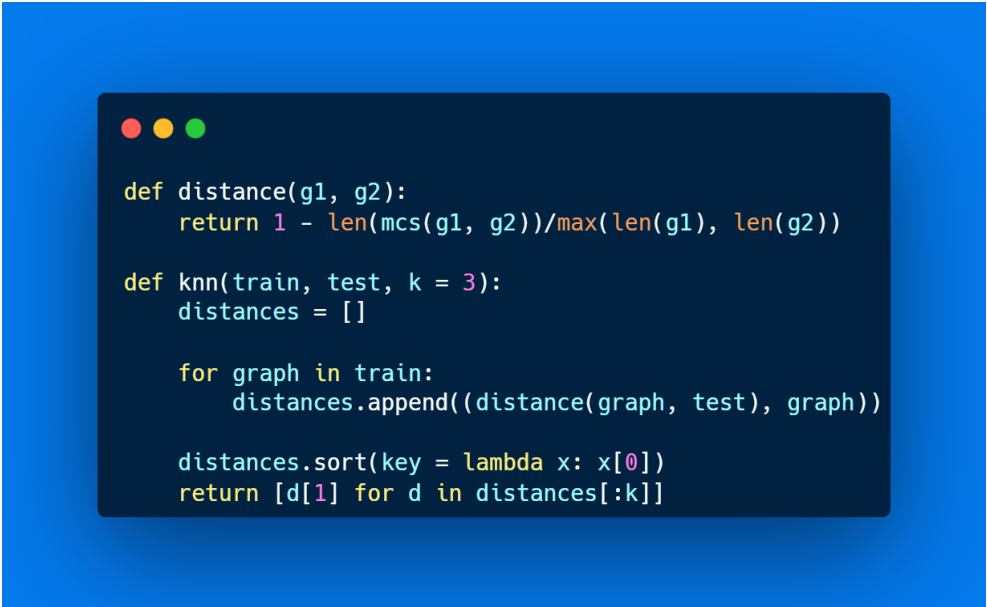


## 2.3 Classification

For classification, we used a kNN based model. Paper 1 suggested a formulae for the calculation of distance between two graphs, based on MCS feature, which is used in kNN for classification of said documents. A test sample is tested against all training graphs and the categories of **k** most similar graphs are selected and the most common category of them is predicted to be the category of the test document. The formulae is given below:

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

Implemented code:



```
def distance(g1, g2):  
    return 1 - len(mcs(g1, g2))/max(len(g1), len(g2))  
  
def knn(train, test, k = 3):  
    distances = []  
  
    for graph in train:  
        distances.append((distance(graph, test), graph))  
  
    distances.sort(key = lambda x: x[0])  
    return [d[1] for d in distances[:k]]
```

Figure 6: kNN algorithm

## 2.4 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 train 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. Following is the result with the graphs

For the actual gains of graph method, we also used the vector classification model and the results are below:

	Precision	Recall	F-1 score	Support
Food	1.00	0.75	0.86	4
Sport	1.00	1.00	1.00	2
Travel	0.75	1.00	0.86	3
Accuracy	-	-	0.89	9
Macro Avg.	0.92	0.92	0.90	9
Weighted Avg.	0.92	0.89	0.89	9

Table 1: Table of metrics of Graph classification

	Food	Sport	Travel
Food	3	0	1
Sport	0	2	0
Travel	0	0	3

Table 2: Confusion matrix of Graph classification

	Precision	Recall	F-1 score	Support
Food	0.80	1.00	0.89	4
Sport	1.00	1.00	1.00	1
Travel	1.00	0.75	0.86	4
Accuracy	-	-	0.89	9
Macro Avg.	0.93	0.92	0.90	9
Weighted Avg.	0.91	0.89	0.89	9

Table 3: Table of metrics of Vector classification

	Food	Sport	Travel
Food	4	0	0
Sport	0	1	0
Travel	1	0	3

Table 4: Confusion matrix of Vector classification

### 3 Project Management

The entire project was managed on GitHub. Here is the repository link: <https://github.com/Hammad-Ejaz/Document-Classification>

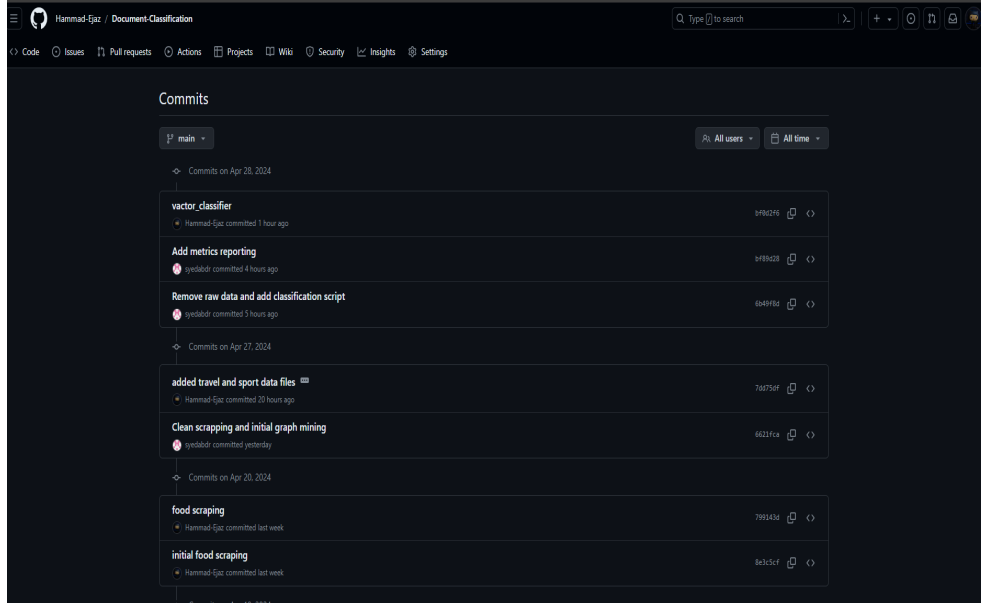


Figure 7: Project Management on Github

## 4 Challenges Faced:

Throughout the project lifecycle, we encountered several challenges, each of which we successfully addressed:

- Certain websites were inaccessible via the `requests` library, prompting us to investigate alternative libraries for data retrieval.
- Implementing the Maximum Common Subgraph (MCS) algorithm posed a challenge due to its NP complexity. Our solution required efficient polynomial-time implementation to deliver feasible results.
- Determining optimal hyperparameters, such as the value of  $K$  and selecting between the number of edges or nodes for distance measurement, presented challenges in achieving the most optimized outcomes.

## 5 Conclusion:

In this project, we delved into the application of graph-based techniques for document classification, departing from conventional vector-based methods. Leveraging the `networkx` library, we constructed directed graphs from preprocessed documents while preserving the original word order and capturing sequential relationships. Our approach to feature extraction, centered on maximal common subgraphs (MCS), demonstrated effectiveness in identifying meaningful patterns crucial for classification tasks.

Throughout our project endeavor, we encountered various methodological challenges, including complexities in web scraping and algorithmic optimizations. Overcoming these obstacles demanded meticulous problem-solving and continual refinement of our strategies. Despite the inherent intricacies involved in implementing MCS efficiently, our persistent efforts culminated in the development of a robust classification model.