Graph-Based Classification using KNN

Muhammad Shahzaib Ijaz
2021-CS-75

Introduction

In this presentation, we will explore the **text classification** using the *K-Nearest Neighbors (KNN) algorithm*. We will delve into the key concepts and strategies for KNN in text classification tasks.

Objectives

- **Collect large dataset** from predefined categories.
- Represent documents as **directed graphs**.
- Identify **common subgraphs** in training set.
- Implement **KNN algorithm** using graph similarity.
- Classify test documents based on **common subgraphs**.

Methodology

- Data Collection
- Text Preprocessing
- Graph Creation
- Feature Extraction via common subgraphs
- Classify with KNN

Data Collection

- Utilized Selenium, powerful web scraping tool, to automate the data collection process.
- Identified relevant websites containing documents related to each of the three assigned topics.
- Extracted text content from each webpages and stored it in a text files.

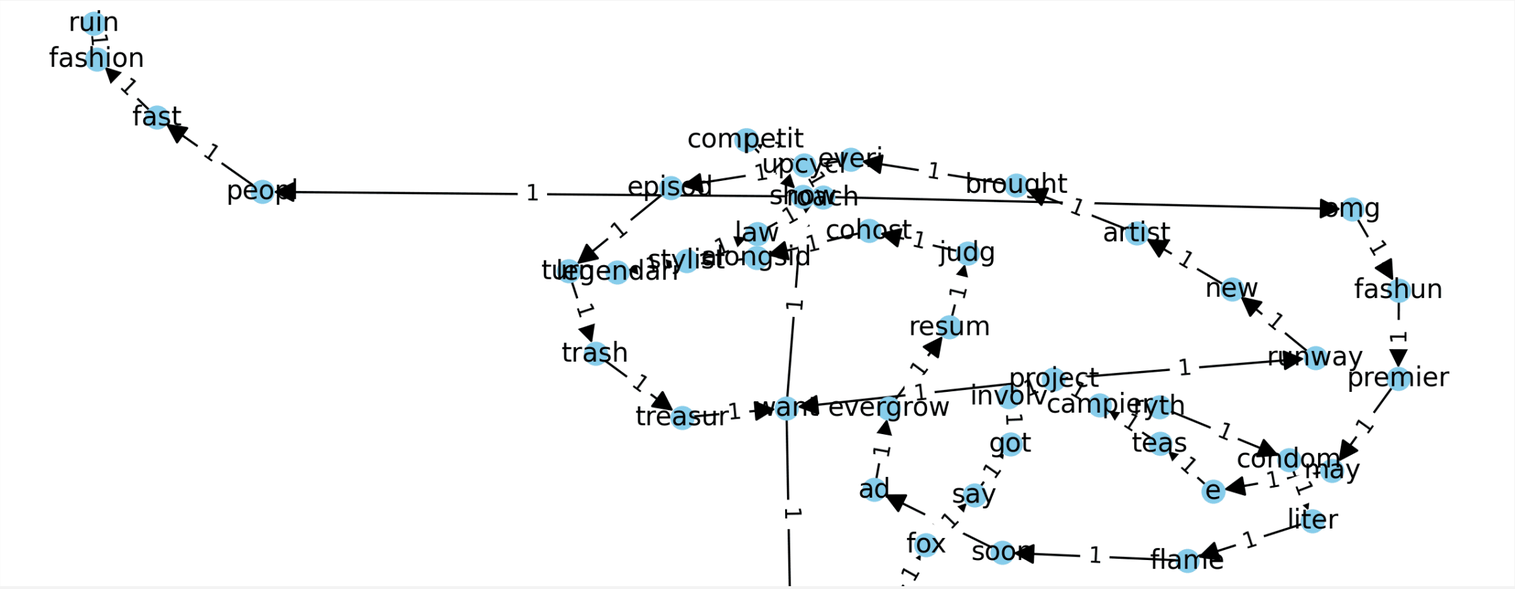
```python
preprocessText(text):
ps = PorterStemmer()
preprocessedText = []
# Convert text to lowercase
text = text.lower()
# Tokenize text into words
words = word_tokenize(text)
# define stop words
stopWords = set(stopwords.words('english'))
# Remove stopwords
for word in words:
    if word.isalpha() and word not in stopWo
        stemmedWord = ps.stem(word)
        preprocessedText.append(stemmedWord)

# words = [word for word in words if word no
text =  " ".join(preprocessedText)
return text
```

Preprocessing

- Converts the text to lowercase for consistency.
- Tokenizes the text into words using NLTK's word tokenizer.
- Retrieves English stopwords and removes them along with non-alphabetic characters.
- Reconstructs the preprocessed text by joining the stemmed words.
- Stems each word using the Porter Stemmerto reduce inflectional forms.

GRAPH CONSTRUCTION

- Each unique word became a node.
- Edges linked consecutive words.
- The weight of each edge reflected the frequency of consecutive word occurrences



FEATURE SELECTION TECHNIQUES

Feature selection methods such as **TF-IDF**, **word embeddings**, and **n-grams** play a vital role in capturing the most discriminative and relevant features for text classification tasks.

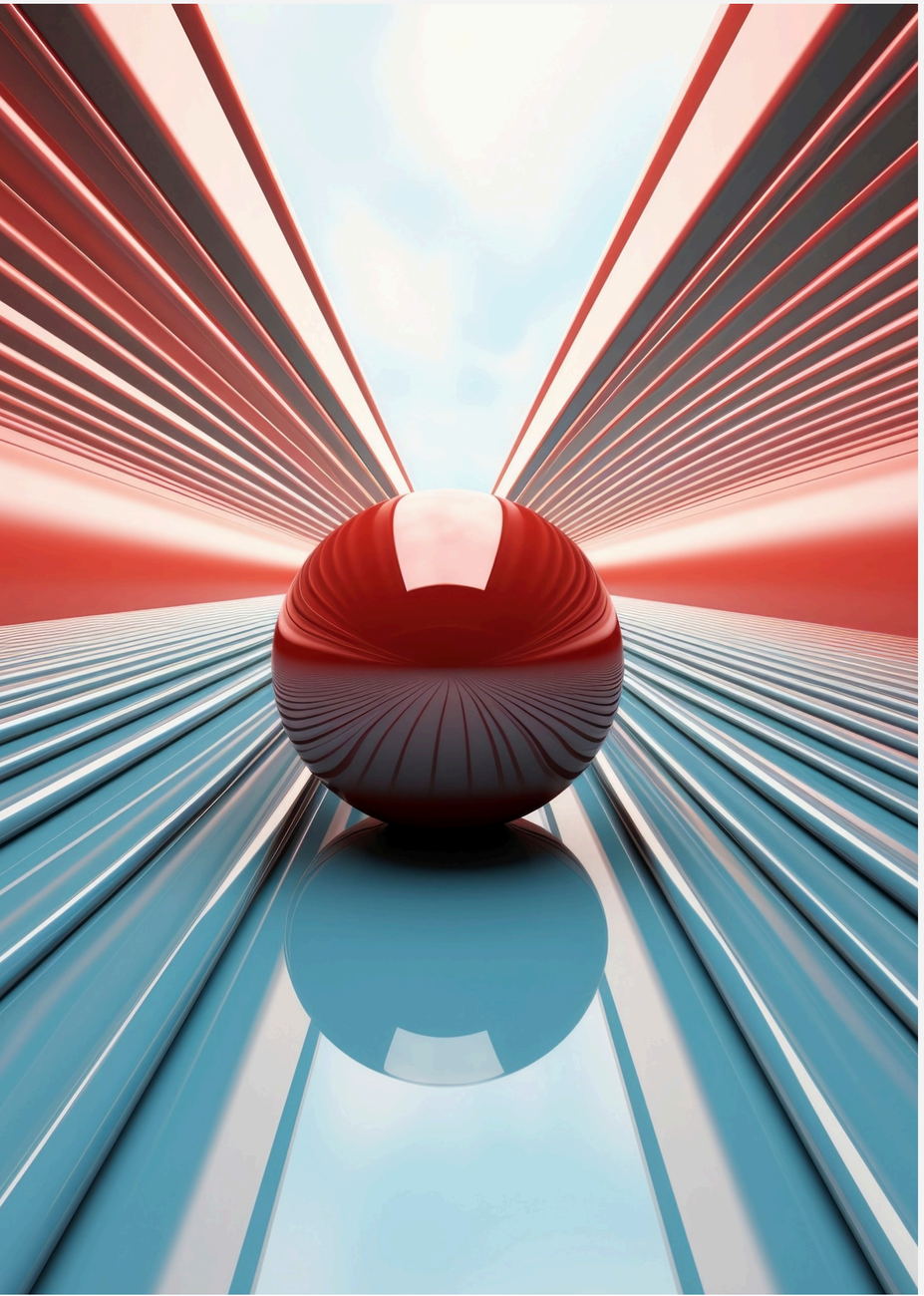
K-Nearest Neighbors (KNN) Algorithm

KNN is a **non-parametric** and **instance-based** algorithm used for classification and regression tasks. It classifies data points based on the majority class of their **nearest neighbors** in the feature space.

K-Nearest Neighbors (KNN) Algorithm

- Classify by looking for which graph is closest to the input graph.
- **Tie Breaker**: selects the class with the highest count among tied neighbors.
- The distance measure used:

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



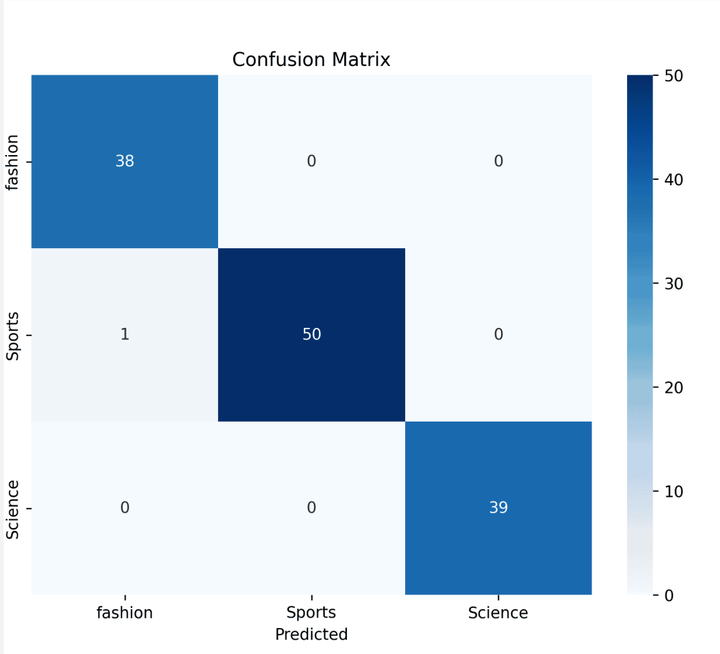
Evaluation Metrics for KNN

Metrics such as **precision**, **recall**, and **F1 score** are commonly used to evaluate the performance of KNN in text classification. Understanding these metrics is essential for assessing the algorithm's effectiveness.

EVALUATION

- **Precision**: Measures true positive predictions relative to all positive predictions.
- **Recall**: Measures true positive predictions relative to all actual positives.
- **F1-Score**: Harmonic mean of precision and recall, providing a balanced measure.
- **Confusion Matrix**: A table summarizing classification model performance.

```
Predicted Class : science, Actual_Category : science
Classification Report:
              precision    recall  f1-score   support

     fashion       1.00      0.97      0.99        38
     science       0.98      1.00      0.99        51
      sports       1.00      1.00      1.00        39

    accuracy                           0.99       128
   macro avg       0.99      0.99      0.99       128
weighted avg       0.99      0.99      0.99       128

              precision    recall  f1-score   support

     fashion       1.00      0.97      0.99        38
     science       1.00      1.00      1.00        51
      sports       0.97      1.00      0.99        39

    accuracy                           0.99       128
   macro avg       0.99      0.99      0.99       128
weighted avg       0.99      0.99      0.99       128

Accuracy is: 99.21875
```



Major Challenges faced

- **Websites inaccessible** via requests library, requiring alternative libraries.
- **Slow-loading website** necessitating script for continuous scrolling.
- Implementing **MCS** as **NP** problem, needing polynomial time and results.
- **Hyperparameter selection** for optimized results: K and distance measure.

Thanks!

ANY QUESTIONS?

muhammadshahzaibijaz@gmail.com

+92 348 6304008