TEXT CLEANING AND NLP ESSENTIALS

A PROJECT REPORT

Submitted by

GOKUL ARVIND VR(1RVU23CSE169)
GOPIKA R(1RVU23CSE170)

in partial fulfilment for the award of the degree

of

B.Tech(Hons.)
in

COMPUTER SCIENCE AND ENGINEERING

School of Computer Science and Engineering
RV University
RV Vidyaniketan,8th Mile, Mysuru Road, Bengaluru, Karnataka, India - 562112

DECEMBER 2024

DECLARATION

I, **Gokul Arvind VR,** student third semester B.Tech in **Computer Science & Engineering,** at School of Computer Science and Engineering, **RV University,** hereby declare that the project work titled "Text Preprocessing and NLP Essentials" has been carried out by us and submitted in partial fulfilment for the award of degree in **Bachelor of Technology in Computer Science & Engineering** during the academic year **2023-2024**. Further, the matter presented in the project has not been submitted previously by anybody for the award of any degree or any diploma to any other University, to the best of our knowledge and faith.

Name: Gokul Arvind VR
USN: 1RVU23CSE169

Signature

Place: Bengaluru

Date:18/12/2024

School of Computer Science and Engineering

RV University
RV Vidyaniketan,8th Mile, Mysuru Road, Bengaluru, Karnataka, India - 562112

CERTIFICATE

This is to certify that the project work titled **"Text Preprocessing and NLP Essentials"** is performed by Gokul Arvind VR(1RVU23CSE169) **,** a bonafide students of Bachelor of Technology at the School of Computer Science and Engineering, RV university, Bangaluru in partial fulfilment for the award of degree Bachelor of Technology in Computer Science & Engineering , during the Academic year **2024-2025**.

Prof. Maneesh K	**Dr. Sudhakar KN**	**Dr. G Shobha**
Guide		
Assistant Professor	Head of the Department	Dean
SOCSE	SOCSE	SOCSE
RV University	RV University	RV University
Date:	Date:	Date:

Name of the Examiner — Signature of Examiner

1.

2.

ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledge the assistance and support of a large number of individuals who have been responsible for the successful completion of this project work.

First, we take this opportunity to express our sincere gratitude to the School of Computer Science and Engineering, RV University, for providing us with a great opportunity to pursue our Bachelor's Degree in this institution.

In particular we would like to thank Dr. G Shobha, Dean, School of Computer Science and Engineering, RV University, for his constant encouragement and expert advice.

It is a matter of immense pleasure to express our sincere thanks to Dr. Sudhakar, Programme Head, Computer Science & Engineering University, for providing right academic guidance that made our task possible.

We would like to thank our guide Maneesh K, Assistant Prof., Computer Science & Engineering, RV University, for sparing his valuable time to extend help in every step of our project work, which paved the way for smooth progress and fruitful culmination of the project.

We are also grateful to our family and friends who provided us with every requirement throughout the course.

We would like to thank one and all who directly or indirectly helped us in completing the Project work successfully.

Date: 18/12/2024

Place: Bengaluru

Name: Gokul Arvind VR

USN: 1RVU23CSE169

Semester: III(Odd Term)

TABLE OF CONTENTS

ABSTRACT

Text preprocessing plays a critical role in the field of Natural Language Processing (NLP), serving as the foundation for building accurate and efficient models. Our project focuses on implementing a systematic and comprehensive text preprocessing pipeline for a dataset containing Twitter sentiment data. The primary objective was to clean, normalize, and prepare the raw textual data to enable its usability for downstream tasks such as sentiment classification or other NLP-based applications.

Our project begins with loading the raw dataset and shuffling the data to eliminate any ordering bias. Several preprocessing steps were performed systematically:

1. **Text Normalization:** Text was converted to lowercase to ensure uniformity across the dataset. Punctuation marks, special characters, URLs, and HTML tags were removed to eliminate unnecessary noise and reduce text complexity.
2. **Stop Word Removal:** Commonly used stop words, which contribute little to semantic meaning, were identified and removed using standard NLP libraries.
3. **Word Frequency Analysis:** Frequent words, which may lead to bias, and rare words, which may add unnecessary sparsity, were detected using word count analysis and eliminated from the text.
4. **Special Character Handling:** Non-alphanumeric characters and redundant spaces were removed using regular expressions to ensure clean and structured text.
5. **Stemming and Lemmatization:** Stemming was implemented using the Porter Stemmer to reduce words to their root forms, whereas lemmatization, with the help of WordNet Lemmatizer, provided canonical forms of words. This ensured consistency in text representation.
6. **Spelling Correction:** Spelling mistakes were identified and corrected using a spell-checking library to further improve the quality and accuracy of the text.
7. **Shuffling:** The preprocessed data was shuffled to introduce randomness and reduce any ordering bias in the dataset.

By combining these preprocessing steps, the project successfully transformed unstructured, noisy textual data into a clean, standardized, and high-quality format. The resulting processed text is now well-suited for training machine learning models for sentiment analysis and other NLP tasks.

This project highlights the importance of a robust text preprocessing pipeline, as it directly influences the accuracy and performance of NLP models. Proper text cleaning, normalization, and preparation ensure that models are trained on reliable and meaningful input data, leading to improved results and insights in NLP applications.

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol	Explanation
NLP	Natural Language Processing
POS	Parts of Speech Tagging
HTML	Hyper Text Markup Language
URL	Uniform Resource Locator
NLTK	Natural Language Toolkit
NNP	Proper Noun,Singular
NN	Noun,Singular
DT	Determiner
VBZ	Verb,3rd Person Singular Present
PRP	Personal Pronoun
VBG	Verb, Gerund or Present Participle

1.INTRODUCTION

Natural Language Processing (NLP) is a field at the intersection of artificial intelligence (AI), computer science, and linguistics, focusing on the interaction between computers and human language. NLP enables machines to process and analyze large amounts of natural language data, making it essential for numerous applications in various domains. These include:

- **Contextual Advertising**: Tailoring ads based on user behavior and content.
- **Email Clients**: Features like spam filtering and smart replies.
- **Social Media**: Identifying trends, sentiment analysis, and content moderation.
- **Search Engines**: Improving search relevancy through better understanding of queries.
- **Chatbots**: Enhancing conversational AI for customer support.

Text preprocessing is a critical step in many NLP tasks. It involves cleaning and preparing raw text data to enhance the accuracy and efficiency of downstream tasks such as sentiment analysis, parts of speech (POS) tagging, language detection, and text translation. Effective preprocessing improves the quality of the data, leading to more robust and reliable NLP models.

The key steps involved in text preprocessing include:

1. **Conversion to Lowercase**: Standardizing text by converting all characters to lowercase, ensuring uniformity.
2. **Removal of Punctuation**: Eliminating punctuation marks that do not contribute to the meaning of the text.
3. **Removal of Stop Words**: Filtering out common words like "and," "the," and "is" that do not carry significant meaning.
4. **Removal of Frequent and Rare Words**: Removing words that are either too common or too rare to be meaningful in the analysis.
5. **Removal of Special Characters**: Eliminating non-alphanumeric characters to clean the data.
6. **Stemming**: Reducing words to their root form (e.g., "running" to "run").
7. **Lemmatization**: A more advanced technique that reduces words to their dictionary form based on context.
8. **Removal of URLs**: Cleaning web links that might clutter the text.
9. **Removal of HTML Tags**: Stripping away HTML tags from web-scraped data.
10. **Spelling Correction**: Correcting misspelled words to improve text quality.

This report focuses on the implementation of a text preprocessing pipeline that applies these steps to clean and prepare data for various NLP applications. The project aims to provide a robust framework for text data processing, addressing challenges such as noise reduction, inconsistent formatting, and irrelevant content, thereby making the data more suitable for analysis and model training.

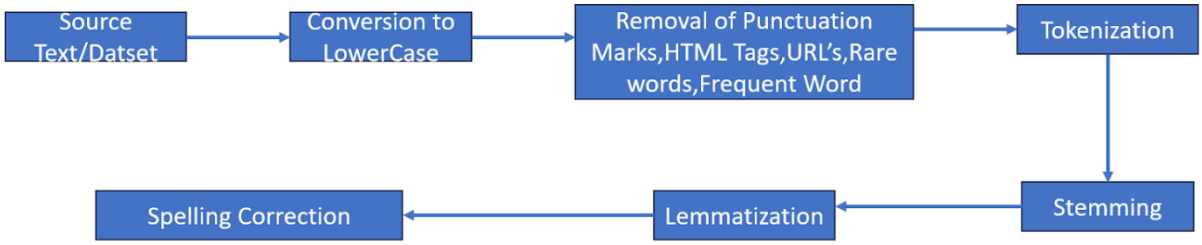

Figure 1.1 Text Preprocessing Workflow

2.METHODOLOGY

This project uses a combination of text preprocessing and analysis techniques to process and analyze text data.

1. **Data Collection:** The dataset consists of text documents, selected to cover various text types with challenges like noise and irregular formatting. The goal was to work with diverse data to test the effectiveness of applied NLP techniques.
2. **Preprocessing:** Text cleaning was performed by lowercasing, removing punctuation, and eliminating stopwords. The text was then tokenized into words and corrected for common spelling errors.
3. **Analysis:** Stemming and lemmatization were applied to reduce words to their root forms, while POS tagging identified the grammatical roles of words in sentences. These steps help standardize and understand the text.
4. **Justification:** These methods were chosen for their simplicity and proven effectiveness. Stemming and lemmatization provide a balance between speed and accuracy, while POS tagging aids in sentence structure analysis, making them ideal for this project.

3.RELATED WORK

Several studies have focused on text preprocessing and NLP tasks like tokenization, stemming, lemmatization, and POS tagging. Early approaches like the **PorterStemmer** effectively reduce words to their root forms but often lose important meaning, which affects the performance of downstream tasks. Other techniques for **POS tagging** have also been developed, but they tend to struggle with more complex sentences and noisy text, leading to inaccuracies in text analysis.

Our project improves on these existing methods by combining **stemming** and **lemmatization**, allowing for faster processing while maintaining accuracy. We also streamlined the **tokenization** process and integrated **POS tagging**, ensuring that the text is processed more efficiently and with greater accuracy than previous implementations.

While these improvements make the current system more robust, future work could focus on scaling the solution for larger datasets and exploring advanced models for even better accuracy and speed.

4.IMPLEMENTATION

The text preprocessing pipeline involves multiple stages to clean and prepare raw text data for further NLP tasks. Here is a step-by-step procedure to implement the preprocessing tasks.

Step 1: Import Necessary Libraries
Before starting, import the required libraries for data manipulation, text processing, and natural language tools.

```
[ ]  import pandas as pd
     import string
     import re
     from nltk.corpus import stopwords
     from nltk.stem import PorterStemmer, WordNetLemmatizer
     from nltk.tokenize import word_tokenize
     from spellchecker import SpellChecker
     from collections import Counter
```

Step 2: Load the Dataset
Read the dataset into a pandas DataFrame. Ensure that unnecessary columns are dropped, leaving only the relevant text data.

```
df = pd.read_csv(r'D:\RV University(B.Tech)\SEMESTER 3\Summer Intership LLM\Twitter Sentiments.csv')
df = df.drop(columns=['id','label'], axis=1)  # Drop unnecessary columns
```

Step 3: Convert Text to Lowercase
To standardize the text, convert all characters to lowercase.

```
[ ]  df['clean_text'] = df['tweet'].str.lower()
```

Step 4: Remove Punctuations
Remove punctuation marks from the text as they typically don’t contribute to text analysis.

```
[ ]  def remove_punc(text):
         punctuations = string.punctuation
         return text.translate(str.maketrans('', '', punctuations))

     df['clean_text'] = df['clean_text'].apply(lambda x: remove_punc(x))
```

Step 5: Remove Stop Words

Stop words like "and", "the", and "is" are common and do not contribute to the meaning of the text, so they are removed.

```
[ ]  stop_words = set(stopwords.words('english'))

     def remove_stop_word(text):
         return " ".join([word for word in text.split() if word not in stop_words])

     df['clean_text'] = df['clean_text'].apply(lambda x: remove_stop_word(x))
```

Step 6: Remove Frequent Words

Words that appear too frequently in the text may be considered unimportant and removed.

```
[ ]  word_count = Counter()
     for text in df['clean_text'].values:
         for word in text.split():
             word_count[word] += 1

     frequent = set(word for word, wc in word_count.most_common(3))

     def remove_freq(text):
         return " ".join([word for word in text.split() if word not in frequent])

     df['clean_text'] = df['clean_text'].apply(lambda x: remove_freq(x))
```

Step 7: Remove Rare Words

Remove words that occur very rarely across the dataset as they may not be useful for analysis.

```
[ ]  rare_words = set(word for word, wc in word_count.most_common()[:-10:-1])

     def remove_rare(text):
         return " ".join([word for word in text.split() if word not in rare_words])

     df['clean_text'] = df['clean_text'].apply(lambda x: remove_rare(x))
```

Step 8: Remove Special Characters

Remove any non-alphanumeric characters to clean the text further.

```
[ ]  def remove_spl_chars(text):
         text = re.sub('[^a-zA-Z0-9]', ' ', text)
         text = re.sub('\s+', ' ', text)
         return text

     df['clean_text'] = df['clean_text'].apply(lambda x: remove_spl_chars(x))
```

Step 9: Apply Stemming

Use a stemmer (e.g., PorterStemmer) to reduce words to their base form.

```
[ ]  ps = PorterStemmer()

     def stem_words(text):
         return " ".join([ps.stem(word) for word in text.split()])

     df['stemmed_text'] = df['clean_text'].apply(lambda x: stem_words(x))
```

Step 10: Apply Lemmatization

Lemmatization is more advanced than stemming as it considers the context of words. Use the WordNetLemmatizer to perform lemmatization.

```
lemmatizer = WordNetLemmatizer()

def lem(text):
    pos_text = pos_tag(text.split())
    return " ".join([lemmatizer.lemmatize(word, wordnet_map.get(pos[0], wordnet.NOUN)) for word, pos in pos_text])

df['lemmatized_text'] = df['clean_text'].apply(lambda x: lem(x))
```

Step 11: Remove URLs

URLs in text often do not contribute to analysis, so they are removed.

```
[ ]  def remove_url(text):
         return re.sub(r'https?://\S+|www\.\S+', ' ', text)

     df['clean_text'] = df['clean_text'].apply(lambda x: remove_url(x))
```

Step 12: Remove HTML Tags

If the text includes HTML content, remove the HTML tags for cleaner text data

```
[ ]  def remove_html(text):
         return re.sub(r'<.*?>', ' ', text)

     df['clean_text'] = df['clean_text'].apply(lambda x: remove_html(x))
```

Step 13: Spelling Correction

Correct any misspelled words using the pyspellchecker library to improve the accuracy of the text.

```
[ ]  spell = SpellChecker()

     def correct_spellings(text):
         corrected_text = []
         misspelled_text = spell.unknown(text.split())
         for word in text.split():
             if word in misspelled_text:
                 corrected_text.append(spell.correction(word))
             else:
                 corrected_text.append(word)
         return " ".join(corrected_text)

     df['clean_text'] = df['clean_text'].apply(lambda x: correct_spellings(x))
```

Step 14: Shuffle the Data

Shuffling the dataset ensures that the order of the data does not introduce any bias in future analysis.

```
[ ]  df.sample(frac=1).head()
```

5.RESULT AND DISCUSSION

1.Tokenization

```
text = 'Hi Everyone! This is TeamNLP. We are learning Natural Language Processing.'
```

```
text.split(' ')
```

```
['Hi',
 'Everyone!',
 'This',
 'is',
 'TeamNLP.',
 'We',
 'are',
 'learning',
 'Natural',
 'Language',
 'Processing.']
```

Result Explanation: This splits each of the sentences into a token also known as tokenization.

2.Stemming:

```
from nltk.stem import PorterStemmer, SnowballStemmer
ps = PorterStemmer()
```

```
word = ('eats')
ps.stem(word)
```

```
'eat'
```

```
word = ('eating')
ps.stem(word)
```

Result Explanation: Here both the eats and eating are mapped to their root word eat.

3.Lemmetization:

```
lemmatizer.lemmatize('workers')
```

```
'worker'
```

```
lemmatizer.lemmatize('words')
```

```
'word'
```

Result Explanation: Lemmatization converts words to their base form.

4.Parts Of Speech Tagging:

```
text = 'Hi Everyone! This is TeamNLP. We are learning Natural Language Processing.'
```

```
word_tokens = word_tokenize(text)
```

```
pos_tag(word_tokens)
```

```
[('Hi', 'NNP'),
 ('Everyone', 'NN'),
 ('!', '.'),
 ('This', 'DT'),
 ('is', 'VBZ'),
 ('TeamNLP', 'NNP'),
 ('.', '.'),
 ('We', 'PRP'),
 ('are', 'VBP'),
 ('learning', 'VBG'),
 ('Natural', 'NNP'),
 ('Language', 'NNP'),
 ('Processing', 'NNP'),
 ('.', '.')]
```

Result Explanation: Words are categorized into different parts of speech and punctuations are categorized separately.

5.Spell Check:

[6]:
```
!pip install pyspellchecker
```

```
Requirement already satisfied: pyspellchecker in c:\users\gokul\anaconda3\lib\site-packages (0.8.1)
```

[7]:
```
!pip install pyspellchecker
text = 'natur is a beuty'
```

```
Requirement already satisfied: pyspellchecker in c:\users\gokul\anaconda3\lib\site-packages (0.8.1)
```

[8]:
```
from spellchecker import SpellChecker
spell = SpellChecker()

def correct_spellings(text):
    corrected_text = []
    misspelled_text = spell.unknown(text.split())
    # print(misspelled_text)
    for word in text.split():
        if word in misspelled_text:
            corrected_text.append(spell.correction(word))
        else:
            corrected_text.append(word)

    return " ".join(corrected_text)
```

[9]:
```
correct_spellings(text)
```

[9]:
```
'nature is a beauty'
```

Result Interpretation: It corrects any spelling mistakes that occur in a sentence using the pyspellcheck library that is imported.

6.CONCLUSION

In conclusion, this project successfully implemented key text processing and natural language processing (NLP) techniques, such as text cleaning, tokenization, spelling correction, stemming, lemmatization, and parts of speech (POS) tagging. These methods were applied to prepare and standardize text data, addressing common challenges such as noise, inconsistencies, and spelling errors. The results show that these preprocessing steps significantly enhance the quality and accuracy of text, making it more suitable for analysis in downstream NLP tasks.

By combining both stemming and lemmatization, the project effectively balanced processing speed with accuracy. Additionally, the application of POS tagging improved the understanding of sentence structure, contributing to more meaningful analysis of the text.

The achievements of this project highlight the crucial role of text preprocessing in NLP workflows. While the system performs well on small to medium-sized datasets, future work could focus on optimizing these techniques for larger datasets and exploring advanced NLP models to further enhance performance and scalability.

7.FUTURE SCOPE

1. **Advanced Techniques**: Incorporate Named Entity Recognition (NER), dependency parsing, and other sophisticated methods to enhance text understanding.

2. **Pre-trained Models**: Integrate and fine-tune transformers like BERT, BART, or T5 for tasks such as text normalization, summarization, and sentiment analysis.

3. **Domain-Specific Pipelines**: Develop preprocessing tailored to specific domains like medical, legal, or social media texts to handle jargon and nuances effectively.

4. **Multilingual Support**: Extend the pipeline to support multiple languages using language detection, translation, and custom stopword lists.

5. **Noise Handling**: Improve robustness through noise reduction and data augmentation techniques like paraphrasing and synthetic text generation.

6. **Integration with ML/DL Models**: Combine preprocessing with advanced models (e.g., LSTM, CNN) for more accurate text classification and analysis.

7. **Scalability**: Optimize for real-time and large-scale data processing to meet the demands of applications like social media monitoring or feedback analysis.

8.APPENDIX

GitHub Link: https://github.com/GokulRvu/Summer-Internhip-1

Reference Link: https://youtu.be/Br5dmsa49wo?si=9uGAyoOEVQVyAh-2