Preprocessing Steps for Natural Language Processing

1. Text Cleaning

In this step, we will perform fundamental actions to clean the text. These actions involve transforming all the text to lowercase, eliminating characters that do not qualify as words or whitespace, as well as removing any numerical digits present.

I. Converting to lowercase

Python is a case sensitive programming language. Therefore, to avoid any issues and ensure consistency in the processing of the text, we convert all the text to lowercase.

II. Removing URLs

When building a model, URLs are typically not relevant and can be removed from the text data.
For removing URLs we can use ‘regex’ library.

III. Removing remove non-word and non-whitespace characters

It is essential to remove any characters that are not considered as words or whitespace from the text dataset.

```
df = df.replace(to_replace=r'[^\w\s]', value='', regex=True)
```

IV. Removing digits

It is important to remove all numerical digits from the text dataset. This is because, in most cases, numerical values do not provide any significant meaning to the text analysis process.

```
df = df.replace(to_replace=r'\d', value='', regex=True)
```

2. Tokenization

Tokenization is the process of breaking down large blocks of text such as paragraphs and sentences into smaller.

3. Stopword Removal

Stopwords refer to the most commonly occurring words in any natural language.

For the purpose of analyzing text data and building NLP models, these stopwords might not add much value to the meaning of the document. Therefore, removing stopwords can help us to focus on the most important information in the text and improve the accuracy of our analysis.

One of the advantages of removing stopwords is that it can reduce the size of the dataset, which in turn reduces the training time required for natural language processing models.

4. Stemming/Lemmatization

What's the difference between Stemming and Lemmatization?

Stemming is a simple and practical approach that involves cutting off the ends of words with the intention of obtaining the correct root form.	Lemmatization aims to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the '*lemma*'.
Eg: automate, automatic, automation → automat	Eg: car, cars, cars', car's → car am, is, are → be

There are various algorithms that can be used for stemming,

· Porter Stemmer algorithm
· Snowball Stemmer algorithm
· Lovins Stemmer algorithm

sses → ss	Eg: caresses → caress
ies → i	Eg: ponies → poni
ational → ate	Eg: international → internate
tional → tion	Eg: conditional → condition

Types of Tokenization

Tokenization methods vary based on the granularity of the text breakdown and the specific requirements of the task at hand. These methods can range from dissecting text into individual words to breaking them down into characters or even smaller units

1. **Word tokenization.** This method breaks text down into individual words. It's the most common approach and is particularly effective for languages with clear word boundaries like English.

2. **Character tokenization.** Here, the text is segmented into individual characters. This method is beneficial for languages that lack clear word boundaries or for tasks that require a granular analysis, such as spelling correction.

3. **Subword tokenization**. Striking a balance between word and character tokenization, this method breaks text into units that might be larger than a single character but smaller than a full word. For instance, "Chatbots" could be tokenized into "Chat" and "bots". This approach is especially useful for languages that form meaning by combining smaller units or when dealing with out-of-vocabulary words in NLP tasks.

Vectorization

In Multi-Layer Perceptrons (MLPs) refers to organizing computations using matrix and vector operations instead of iterative loops. It leverages libraries like NumPy or TensorFlow to perform operations efficiently by taking advantage of optimized low-level implementations and parallel computing capabilities. Vectorization improves training speed and scalability, particularly for large datasets and complex models.

TF-IDF (Term Frequency-Inverse Document Frequency):

- A statistical method to evaluate the importance of a word in a document relative to a collection (corpus) of documents.
- **Purpose:** Represents text as numerical feature vectors for traditional machine learning models. It emphasizes words that are unique to specific documents.

Word2Vec:

- A neural network-based model that learns dense, low-dimensional representations (embeddings) of words by capturing semantic and syntactic relationships.
- **Purpose:** Useful for deep learning tasks, including NLP, by providing meaningful word representations.

s1: "earth is the third planet from the sun"
s2: "jupiter is the largest planet"

Words	TF(s1)	TF(s2)	IDF ($\log(\mathrm{TD}/\mathrm{AD})$)	TF-IDF(s1)	TF-IDF(s2)
earth	$1/8$	0	$\log(2/1) = 0.3$	-	-
is	$1/8$	$1/5$	$\log(2/2) = 0$	-	-
the	$2/8$	$1/5$	$\log(2/2) = 0$	-	-
third	$1/8$	0	$\log(2/1) = 0.3$	-	-
planet	$1/8$	$1/5$	$\log(2/2) = 0$	-	-
from	$1/8$	0	$\log(2/1) = 0.3$	-	-
sun	$1/8$	0	$\log(2/1) = 0.3$	-	-
jupiter	0	$1/5$	$\log(2/1) = 0.3$	-	-
largest	0	$1/5$	$\log(2/1) = 0.3$	-	-

Index:

Words Order: earth, is, the, third, planet, from, sun, jupiter, largest