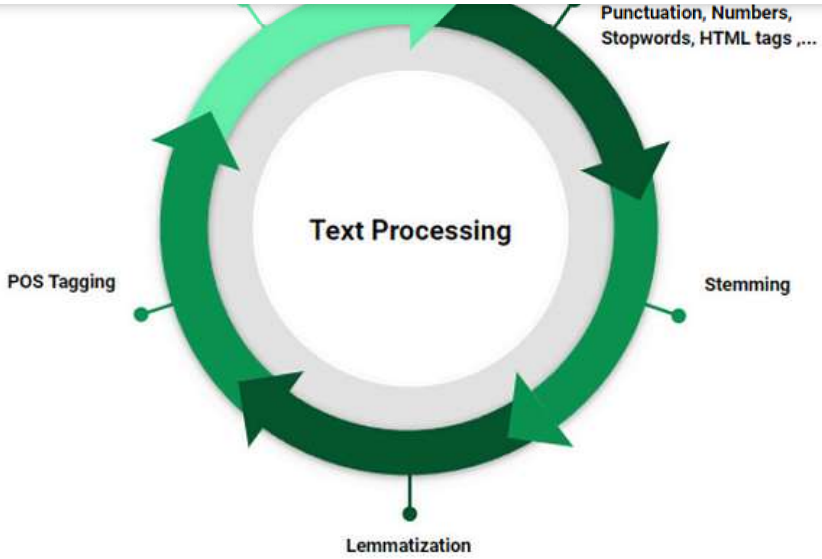
Aysel Aydin

Summary

The article outlines essential text preprocessing techniques for enhancing

1 — Text Preprocessing Techniques for NLP



In this article, we will cover the following topics:

- Why is text preprocessing important?
- Text Preprocessing Techniques

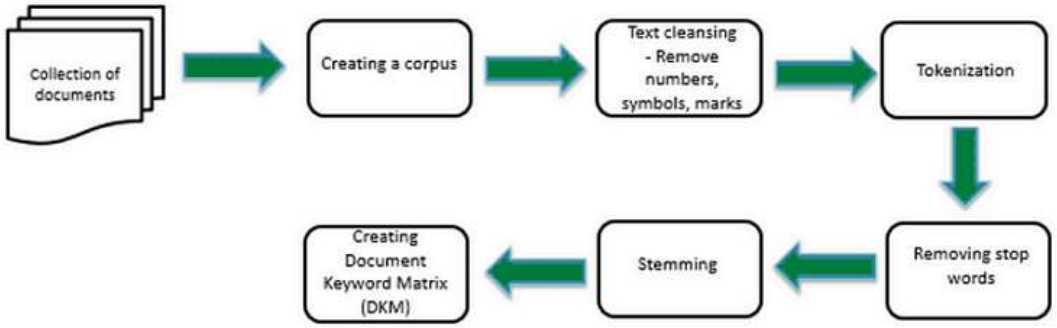
Why is text preprocessing important?

Data quality significantly influences the performance of a machine-learning model. Inadequate or low-quality data can lead to lower accuracy and effectiveness of the model.

In general, text data derived from natural language is unstructured and noisy. So text preprocessing is a critical step to transform messy, unstructured text

models, leading to better results and insights.

Text Preprocessing Techniques



Text preprocessing refers to a series of techniques used to clean, transform and prepare raw textual data into a format that is suitable for NLP or ML tasks. The goal of text preprocessing is to enhance the quality and usability of the text data for subsequent analysis or modeling.

Text preprocessing typically involves the following steps:

- Lowercasing
- Removing Punctuation & Special Characters
- Stop-Words Removal
- Removal of URLs

- Stemming & Lemmatization
- Tokenization
- Text Normalization

Some or all of these text preprocessing techniques are commonly used by NLP systems. The order in which these techniques are applied may vary depending on the needs of the project.

Let's explain the text preprocessing techniques in order.

Lowercasing Lowercasing is a text preprocessing step where all letters in the text are converted to lowercase. This step is implemented so that the algorithm does not treat the same words differently in different situations.

```
text = "Hello WorlD!"
lowercased_text = text.lower()

print(lowercased_text)
```

```
Output:
hello world!
```

periods, commas, exclamation marks, emojis etc.) from the text to simplify it and focus on the words themselves.

```
import re

text = "Hello, world! This is?* 💜an&/|~^+%'\" example- of text preprocessing

punctuation_pattern = r'[^\w\s]'

text_cleaned = re.sub(punctuation_pattern, '', text)

print(text_cleaned)
```

```
Output:
Hello world This is an example of text preprocessing
```

Stop-Words Removal Stopwords are words that don't contribute to the meaning of a sentence. So they can be removed without causing any change in the meaning of the sentence. The NLTK library has a set of stopwords and we can use these to remove stopwords from our text and return a list of word tokens. Removing these can help focus on the important words.

```
# remove english stopwords function
def remove_stopwords(text, language):
    stop_words = set(stopwords.words(language))
    word_tokens = text.split()
    filtered_text = [word for word in word_tokens if word not in stop_words]
    print(language)
    print(filtered_text)

en_text = "This is a sample sentence and we are going to remove the stopwords
remove_stopwords(en_text, "english")

tr_text = "bu cümledeki engellenen kelimeleri kaldıracağız"
remove_stopwords(tr_text, "turkish")
```

```
english
['This', 'sample', 'sentence', 'going', 'remove', 'stopwords']

turkish
['cümledeki', 'engellenen', 'kelimeleri', 'kaldıracağız']
```

> *If you examine the output closely, you'll notice that in the first sentence, the word 'this' was removed, but 'This' was not removed. So, it is necessary to convert the sentence to lowercase and remove punctuation marks before applying this step.*

```
def remove_urls(text):
    url_pattern = re.compile(r'https?://\S+|www\.\S+')
    return url_pattern.sub(r'', text)

text = "I hope it will be a useful article for you. Follow me: https://medium
remove_urls(text)
```

```
Output:
I hope it will be a useful article for you. Follow me:
```

Removal of HTML Tags Removal of HTML Tags is a text preprocessing step used to clean text data from HTML documents. When working with text data obtained from web pages or other HTML-formatted sources, the text may contain HTML tags, which are not desirable for text analysis or machine learning models. Therefore, it's important to remove HTML tags from the text data.

```
import re

text = """<html><div>
```

```
</div></html>"""

html_tags_pattern = r'<.*?>'

text_without_html_tags = re.sub(html_tags_pattern, '', text)

print(text_without_html_tags)
```

```
Output:
Aysel Aydin
Text Preprocessing for NLP
Medium account
```

I will cover the last 3 techniques in detail in the next article.

Conclusion

These are just a few techniques of natural language processing. Once the information is extracted from unstructured text using these methods, it can be directly consumed or used in clustering exercises and machine learning models to enhance their accuracy and performance.

I hope it will be a useful article for you. Happy coding ✊

NLP Text Preprocessing Data Text Naturallanguageprocessing

Recommended from ReadMedium

Aysel Aydin

10—Understanding Word2Vec 1: Word Embedding in NLP

In this article, we will talk about Word2vec one of the word embedding techniques. Before we start, I recommend you read the article I...

3 min read

Nivedita Bhadra

Topic modeling with Python : An NLP project

Explore your text data with Python

15 min read

K

Katabathina

NLP

What is Natural Language Processing? Well, you're doing it right now. You're listening to the words and the sentences that I'm forming, and...

Mdabdullahalhasib

A Complete Guide to Embedding For NLP & Generative AI/LLM

Understand the concept of vector embedding, why it is needed, and implementation with LangChain.

11 min read

Rana Munawar

NLP (Natural Language Processing)

2 min read

Dr. Walid Soula

Bag-of-Words

Explore the fundamentals of the Bag-of-Words model in natural language processing

4 min read

J

Jo Wang

Deep Learning Part 5 -How to prevent overfitting

Techniques used to prevent overfitting in deep learning models:

4 min read

Mastering NLP with GloVe Embeddings: Word Similarity, Sentiment Analysis, and More

Introduction

6 min read

Rahul Kumar

NLP Hands-On with Text Classification

This post is a part of the NLP Hands-on series and consists of the following tasks: 1. Text Classification 2. Token Classification 3...

3 min read

M

Mahesh

Data Science Codes

1. Data Cleaning

16 min read