TIC

ACTIVIDAD #3

Tipo actividad: reading activity: "Natural Language Processing (NLP), Multiple choice activity

Socialization key vocabulary reading activity #4: "Natural Language Processing (NLP) Data Preprocessing Techniques"

1. Text Preprocessing: Text preprocessing is a crucial step in natural language processing (NLP) that involves cleaning, transforming, and organizing raw textual data to make it suitable for machine learning or NLP tasks. The primary goal is to enhance the quality and usability of text data for subsequent analysis or modeling.

2. Lowercasing: Lowercasing is a text preprocessing technique where all letters in the text are converted to lowercase. This step is implemented to ensure that the algorithm treats the same words consistently in different situations, preventing variations in casing from affecting the analysis.

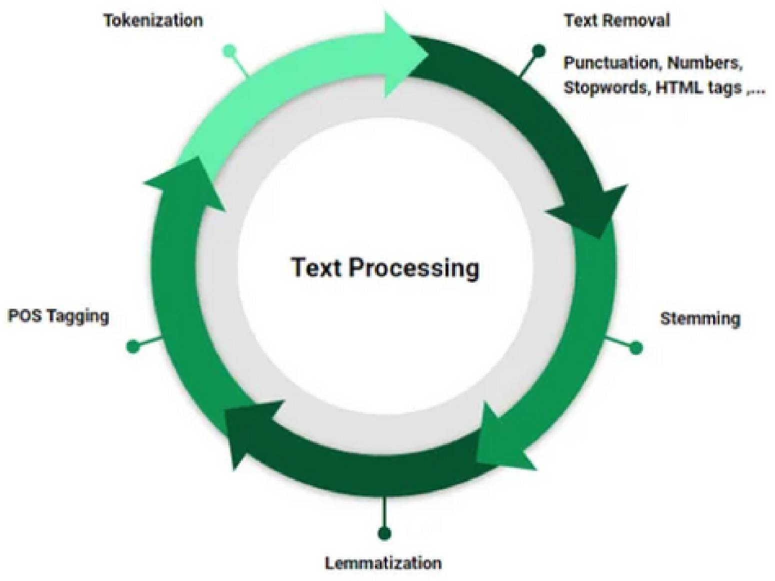
3. Stop-Words Removal: Stop-words removal is a text preprocessing step aimed at eliminating words that do not contribute to the meaning of a sentence. These words, known as stopwords, are removed to focus on important words, enhancing the efficiency of natural language processing tasks.

4. Tokenization: Tokenization is a text preprocessing technique that involves breaking down a text into smaller units, called tokens. These tokens can be words, punctuation marks, or numbers. Tokenization is a fundamental step in preparing text data for analysis or machine learning models.

5. HTML Tags Removal: HTML tags removal is a text preprocessing step used to clean text data extracted from HTML documents. When dealing with text obtained from web pages or other HTML-formatted sources, removing HTML tags is essential to ensure the accuracy and relevance of the text for analysis or modeling.

9) Reading comprehension activity #4: "Natural Language Processing (NLP) Data Preprocessing Techniques"

Text Preprocessing Techniques for NLP



In this article, we will cover the following topics:

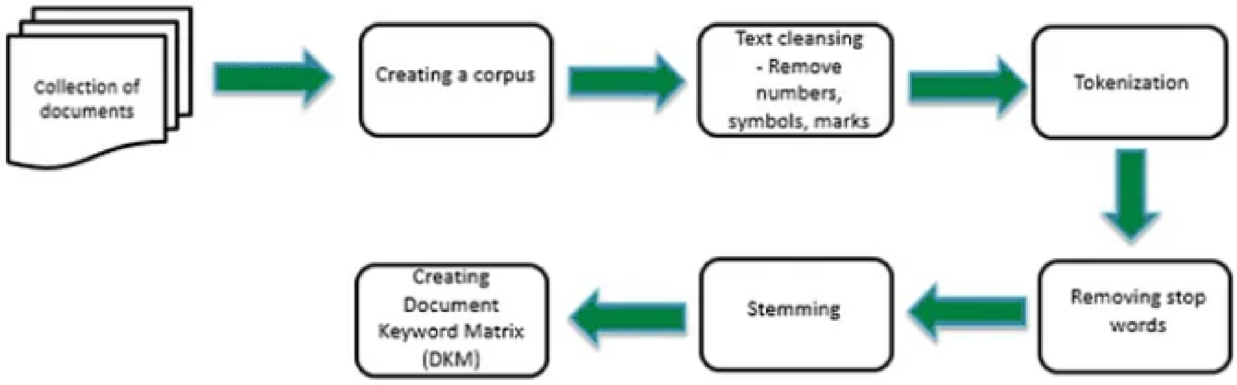
- Why is text preprocessing important?
- Text Preprocessing Techniques

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 data into a form that can be effectively used to train machine learning models, leading to better results and insights.

Text Preprocessing Techniques


Collection of documents
Creating a corpus
Text cleansing - Remove numbers, symbols, marks
Tokenization
Removing stop words
Stemming
Creating Document Keyword Matrix (DKM)

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
- Removal of HTML Tags
- 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!
```

Removing Punctuation & Special Characters

Punctuation removal is a text preprocessing step where you remove all punctuation marks (such as 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.

```
from nltk.corpus import stopwords

# 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 from this"
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.

Removal of URLs

This preprocessing step is to remove any URLs present in the data.

```
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.com/@ayselaydin"
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>
<h1>Aysel Aydin</h1>
<p>Text Preprocessing for NLP</p>
<a href="https://medium.com/@ayselaydin">Medium account</a>
</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
```

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.

10) Multiple choice questions about the previous reading

1. Why is text preprocessing important in natural language processing (NLP)?
   - A. Enhances data quality.
   - B. Introduces complexity.
   - C. Ensures consistent treatment.
   - D. Increases dataset size.

2. What is the purpose of lowercasing in text preprocessing?
   - A. Ensure consistent treatment.
   - B. Introduce casing variation.
   - C. Confuse the algorithm.
   - D. Remove stopwords.

3. Which technique is used to eliminate words that do not contribute to the meaning of a sentence?
   - A. Tokenization
   - B. Lemmatization
   - C. Stop-Words Removal
   - D. Stemming

4. What is the primary goal of HTML tags removal in text preprocessing?
   - A. Add HTML tags for clarity.
   - B. Enhance text structure.
   - C. Preserve HTML formatting.
   - D. Clean text data from HTML documents.

5. What does tokenization involve in text preprocessing?
   - A. Break down text into smaller tokens.
   - B. Convert all letters to lowercase.
   - C. Remove punctuation marks.
   - D. Remove URLs from the text.