Applied Text Analytics & Natural Language Processing

with Dr. Mahdi Roozbahani
& Wafa Louhichi

Text Preprocessing Techniques

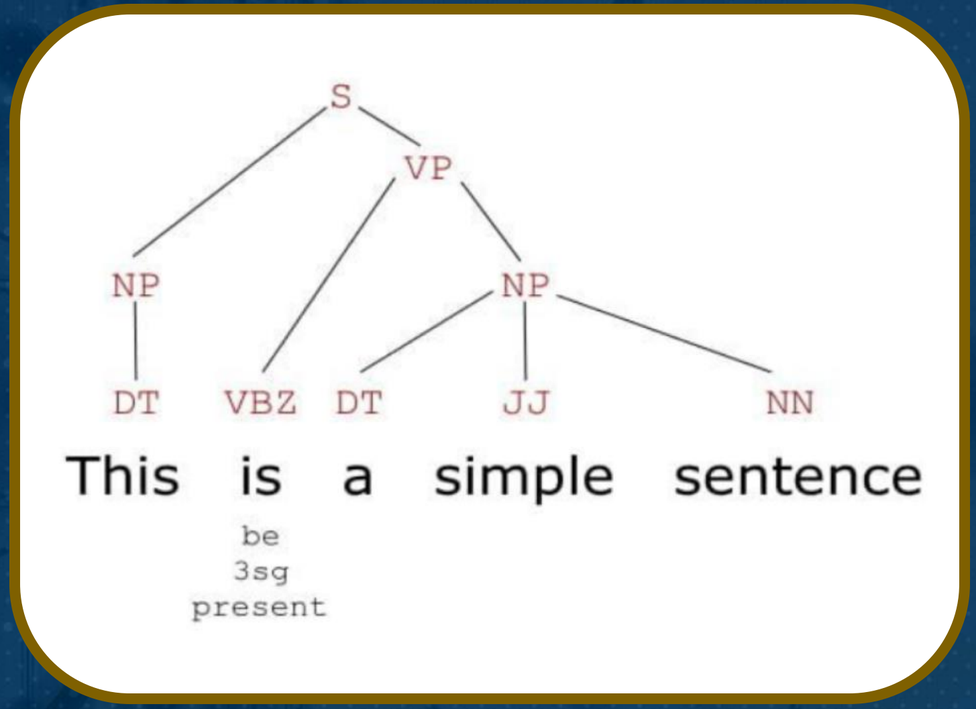
Learning Objectives

In this lesson, we will start diving into NLP!

- You will be introduced to some NLP terminology
- Practice basic preprocessing techniques on text data

Some NLP Terminology

- Corpus: collection of text
    - Example: Yelp Reviews, Wikipedia...
- Syntax: the grammatical structure of the text
- Syntactic parsing: process of analyzing natural language with grammatical rules
- Semantics: the meaning of text



Some NLP Terminology

- Tokenization: splitting longer pieces of texts into smaller pieces: tokens. Example of splitting a sentence into words: This is a simple sentence -> "This" "is" "a" "simple" "sentence"
- Vocabulary: list of unique words

Some NLP Terminology

- Stop words: commonly used words such as "the", "a", "an, "is", "are" … that don't contribute a lot to the overall meaning
- N-Grams: a consecutive sequence of n words (2-5 is common) in a text. 1-gram (unigram), 2-gram(bigram), 3-gram(trigram). Example of bigrams: This is a simple sentence -> "This is", "is a", "a simple", "simple sentence"

Preprocessing Text Data

- Text data is a form of unstructured data. Text preprocessing is the first step to prepare and clean the text data to perform an NLP task
- Useful libraries for text preprocessing: re (Regular expression operations) and NLTK (Natural Language Toolkit)
- Common steps in text preprocessing:
    - Noise removal
    - Tokenization
    - Text normalization

Noise Removal

Noise removal refers to removing unwanted text formatting information

- Punctuation and accent marks
- Special characters
- Numeric digits or replacing with the word
- Leading, ending, and vertical whitespace
- HTML formatting

```
1 import re
2 text = "    # This is a''simple sentence !!!! 1+ \n"
3
4 # remove punctuation and special characters
5 re.sub('^\s+|\W+|[0-9]|\s+$',' ', text).strip()
6
```

```
'This is a simple sentence'
```

Tokenization

- Example of word tokenization using nltk

```
1 from nltk.tokenize import word_tokenize
2 text="This is a simple sentence."
3 word_tokenize(text)

['This', 'is', 'a', 'simple', 'sentence', '.']
```

Text Normalization

- Text normalization aims at removing variations in the text and bringing it to a standard form.
    - Convert all letters to lower or upper case
    - Removing stop words, sparse terms, and particular words

```
from nltk.corpus import stopwords

text="This is a Simple senTence"

# convert to lower case
text = text.lower()
# remove stop words
stop_words = list(stopwords.words('english'))
text = ' '.join([i for i in text.split() if not i in stop_words])
text
```

```
'simple sentence'
```

Text Normalization

- Stemming: reducing words to their word stem, base or root form
- Lemmatization: similar to stemming, reduces inflectional forms to a common base form : the *lemma*. Lemmatization does not simply chop off inflections, instead, it uses lexical knowledge bases to get the correct base forms of words

```
1 from nltk.stem import PorterStemmer, WordNetLemmatizer
2
3 def stemming(mytext):
4     stemmer= PorterStemmer()
5     return ' '.join([stemmer.stem(word) for word in mytext.split()])
6 def lemmatization(mytext):
7     lemmatizer=WordNetLemmatizer()
8     return ' '.join([lemmatizer.lemmatize(word) for word in mytext.split()])
9 text = 'There are several types of stemming algorithms.'
10 print('Stemming: ', stemming(text))
11 print('Lemmatization: ', lemmatization(text))

Stemming:  there are sever type of stem algorithms.
Lemmatization:  There are several type of stemming algorithms.
```

Summary

- Introduction to NLP terminology
- Different text preprocessing steps