Complete Tutorial On Natural Language Processing using spaCy

BEGINNER MACHINE LEARNING NLP PYTHON

This article was published as a part of the Data Science Blogathon

In this tutorial, we will learn:

- Introduction to Natural Language Processing
- Phases of Natural Language Processing
- Introduction to spaCy
- Installation and local setup of spaCy
- Statistical models available in spaCy
- Reading and processing text
- Spans
- Tokenization
- Sentence Detection
- Stopwords removal
- Word Frequency graph
- Parts of speech tagging
- Named Entity recognition
- Graphical representation using displacy
- Conclusion

Introduction to Natural Language Processing

Natural Language Processing**(NLP)** is defined as the branch of Artificial Intelligence that provides computers with the capability of understanding text and spoken words in the same way a human being can. It incorporates **machine learning models, statistics, and deep learning models** into **computational linguistics** i.e. rule-based modeling of human language to allow computers to understand text, spoken words and understands human language, intent, and sentiment.

NLP is used to convert text from one language to another, provide a summary to a large amount of text, respond to customer queries in **chatbots**, **digital assistants**. It also found application in **voice-operated GPS systems** and other consumer conveniences. NLP is becoming increasingly popular in companies for providing business solutions to enhance **customer experiences, streamline operations and increase profit**.

spaCy is an open-source library used for natural language processing in python. It is extremely popular for processing a large amount of unstructured data generated at a vast scale in the industry and generate useful and meaningful insights from the data.

The task of NLP is complex as the **natural language is ambiguous and uncertain**. There are different types of ambiguities present in natural language:

1. **Lexical Ambiguity:** It is defined as the ambiguity associated with the meaning of a single word. A single word can have different meanings. Also, a single word can be a noun, adjective, or verb. For example, The word “bank” can have different meanings. It can be a financial bank or a riverbank. Similarly, the word “clean” can be a noun, adverb, adjective, or verb.

2. **Syntactic Ambiguity:** It is defined as the ambiguity associated with the way the words are parsed. For example, The sentence “Visiting relatives can be boring.” This sentence can have two different meanings. One is that visiting a relative’s house can be boring. The second is that visiting relatives at your place can be boring.

3. **Semantic Ambiguity:** It is defined as ambiguity when the meaning of the words themselves can be ambiguous. For example, The sentence “Mary knows a little french.” In this sentence the word “little french” is ambiguous. As we don’t know whether it is about the language french or a person.

Phases of Natural Language Processing

There are roughly five phases of Natural language processing:

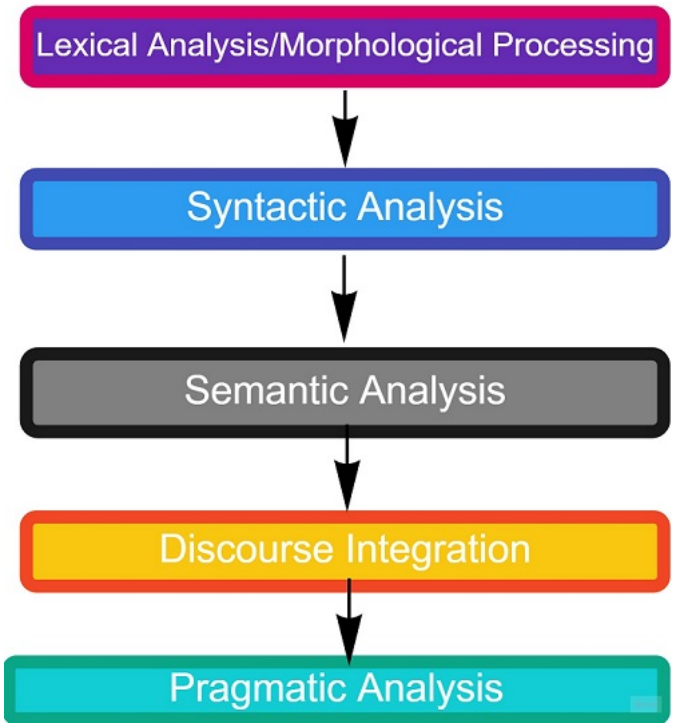
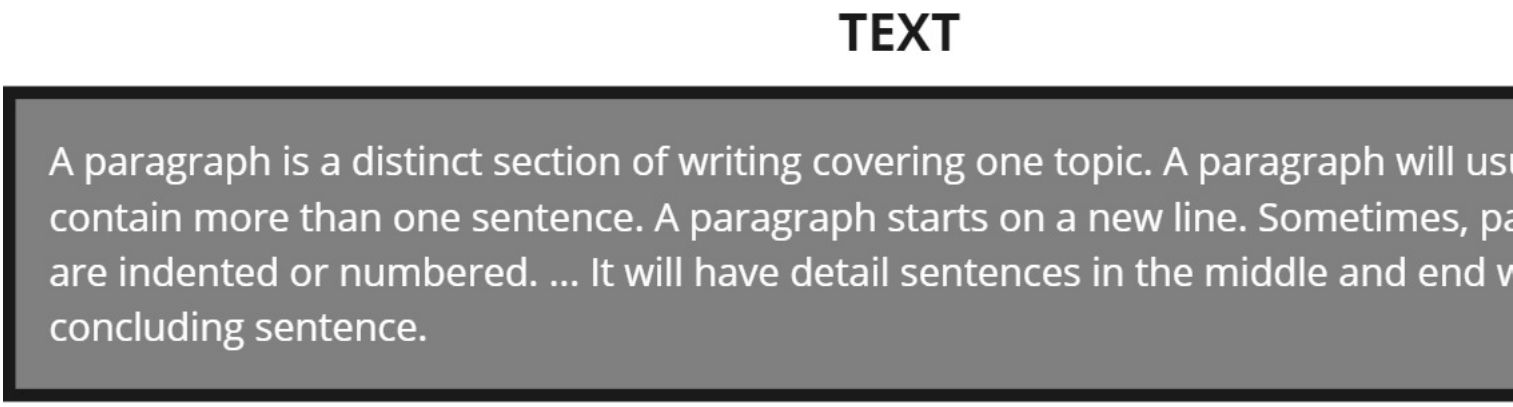


Image 1

1. **Lexical Analysis:**

The first phase is lexical analysis/morphological processing. In this phase, the sentences, paragraphs are broken into tokens. These tokens are the smallest unit of text. It scans the entire source text and divides it into meaningful lexemes. For example, The sentence “He goes to college.” is divided into [‘He’ , ‘goes’ , ‘to’ , ‘college’, ‘.’] . There are five tokens in the sentence. A paragraph may also be divided into sentences.



2. **Syntactic Analysis/Parsing:**

The second phase is Syntactic analysis. In this phase, the sentence is checked whether it is well-formed or not. The word arrangement is studied and a syntactic relationship is found between them. It is checked for word arrangements and grammar. For example, the sentence “Delhi goes to him” is rejected by the syntactic parser.

3. **Semantic Analysis:**

The third phase is Semantic Analysis. In this phase, the sentence is checked for the literal meaning of each word and their arrangement together. For example, The sentence “I ate hot ice cream” will get rejected by the semantic analyzer because it doesn’t make sense.

4. **Discourse Integration:**

The fourth phase is discourse integration. In this phase, the impact of the sentences before a particular sentence and the effect of the current sentence on the upcoming sentences is determined.

5. **Pragmatic Analysis:**

The last phase of natural language processing is Pragmatic analysis. Sometimes the discourse integration phase and pragmatic analysis phase are combined. The actual effect of the text is discovered by applying the set of rules that characterize cooperative dialogues.

Introduction to Natural Language Processing With spaCy

spaCy is a free, open-source python library used for natural language processing. It is written in Cython. It provides all the features required for natural language processing. It provides production-ready code. It is very popular and widely used. It contains processing pipelines and language-specific rules for tokenization. It excels at large-scale information extraction tasks and is written in memory-managed cython. In the last five years, spaCy has become an industry standard with a huge ecosystem.

Salient Features:

- Supports 64+ languages.
- 63 trained pipelines for 19 languages.
- Production-ready training system.
- Multi-task learning
- Built-in visualizers.
- Robust and fast.
- Easily extensible with custom components.
- Multi-language support. It has models for systems with more than one language together.

Installation and Local setup of spaCy

spaCy can be installed using python package manager pip. You can first create a virtual environment.

```
python3 -m venv env $ source ./env/bin/activate $ pip install spacy
```

Once spaCy is installed we are good to go.

Statistical models available in spaCy

spaCy's models and trained pipelines can be installed easily as python packages. They are available in a variety of languages. They act as components of your application. Their data can be stored anywhere in your system or provided as a dependency in the requirement.txt file of the docker image. The pipelines can be installed manually, using pip from a downloaded URL. It also asks whether we want a default trained pipeline for accuracy or efficiency.

There are four trained pipelines available for the English language. We will install "en_core_web_sm" pipeline.

```
python -m spacy download en_core_web_sm import spacy nlp = spacy.load("en_core_web_sm")
```

The pipeline has been downloaded.

Reading and Processing text with spaCy

To perform analysis on a text, it must be first read and then processed to gain insights.

The text needed to be processed with the **nlp** object loaded above. The nlp object is called the natural language instance loaded by en_core_web_sm. It processes the text and converts it into a **Doc** object.

The **Doc** object is a type of container that contains all the linguistic information like tokens, annotations, and relationships about the text.

Reading a string using spaCy

Let us first read a string, process it by nlp object of the spacy library, and then convert it into Doc object.

```
text = ('''Natural Language Processing is defined as the branch of Artificial Intelligence that provides computers the capability of understanding text and spoken words in the same way a human being can.''')
text_doc_object = nlp(text) # Extract tokens for the given doc print ([token.text for token in text_doc_object])
```

As we can see, the nlp object first processes the text then converts it into a Doc object. We have shown the token attribute of the doc object. Other attributes will be covered later in this tutorial.

Reading a text file

Let us now read the text file and perform the above steps.

```
text = open("C:/Users/Dell/Desktop/example.txt",encoding='UTF-8').read() text_doc_object = nlp(text) # Extract tokens for the given doc print ([token.text for token in text_doc_object])
```

Spans

Span is defined as a part/ substring of a text. They are treated token-wise.

So text_doc[1:5] is a span with token indices starting from 1 to 4(5 is not inclusive). You can simply make a span out of doc object or create it manually as shown below:

```
from spacy.tokens import Span # Span for token index 1 to 5 with label GPE (geopolitical) span = Span(text_doc_object, 1, 5, label="GPE") span.text
```

Tokenization

A **token** is defined as the smallest meaningful part of the text. The process of dividing a text into a list of meaningful tokens is called **tokenization**. Tokens are of great importance as they provide the basis for analysis. In many applications like social media analytics where hashtags play an important part, tokenization provides major benefits.

As we have already seen above, we can find the tokens using the Doc object of spaCy. There are various attributes of the token class. You can find the list of all the attributes here. Let us explore some of the attributes.

```
for token in text_doc_object: print (token, token.idx, token.text_with_ws,token.is_digit,token.is_ascii,
token.like_url,token.like_email, token.is_alpha, token.is_punct, token.is_space, token.shape_, token.is_stop)
```

In the above code, we have used the following token attributes:

token/token.text – The text of the token

idx- It is the character offset of the token with respect to the parent text document.

text_with_ws- The text of the token if trailing whitespace if any.

is_digit- Whether the token is a digit or not.

is_ascii- Whether the token is an ASCII value or not.

like_url- Whether the syntax of the token is like a URL or not.

like_email– Whether the syntax of the token is like an email id or not.

is_alpha– Whether the token is alphanumeric or not.

is_punct- Whether the token is a punctuation mark or not.

shape– The structure of the token

is_space– Whether the token is a space or not.

Sentence Detection

A sentence is defined as a linguistic expression/ grammatical unit of words conveying a meaning together. Sentence detection is an important phase of natural language processing.

spaCy provides sents property to detect sentences from the text. By default, it uses full stop as a separator for sentences but you can customize it to use any other separator by adding a pipeline before parsing the text.

```
[sent.text for sent in text_doc_object.sents]
```

Stopwords Removal

Stopwords are defined as the words that occur frequently in language for phrase-making but do not have any significance in analysis. The words like is, are, am, are examples of stopwords. They should be removed from the text.

Let us first look at the stopwords in spaCy.

```
stopwords = spacy.lang.en.stop_words.STOP_WORDS len(stopwords) for stopword in list(stopwords)[1:15]: print(stopword)
```

Now we will remove the stopwords from the text.

```
[token for token in text_doc_object if not token.is_stop and not token.is_punct]
```

Word Frequency Graph

Now we would like to see how are words distributed in the text.

```
from collections import Counter words = [token.text for token in text_doc_object if not token.is_punct and not token.is_stop and not token.is_space] word_freq = Counter(words) common_words = word_freq.most_common(30) first=[] second=[] for i in common_words: first.append(i[0]) second.append(i[1]) import matplotlib.pyplot as plt plt.bar(first, second, color='orange') plt.xticks(range(len(first)), first, rotation='vertical') plt.show()
```

Parts of Speech Tagging

There are eight parts of speech:

- Noun
- Verb
- Adverb

- Adjective
- Preposition
- Conjunction
- Interjection
- Pronoun

spaCy provides a way to tag each part of a text to specific parts of speech tags. This process of assigning POS tags to tokens is called Parts of Speech Tagging. It gives deep insights into the tokens and their usage in the text.

Let us find out the POS tags of the above text:

```
for token in text_doc_object: print (token, token.tag_, token.pos_, spacy.explain(token.tag_))
```

In the above code, tag, pos, and explain attributes of the token class of spaCy library are used. the tag gives the fine-grained part of the token, pos gives the coarse-grained part of the tokens, and explain gives the detailed explanation of the same.

Named Entity Recognition with spaCy

Named Entity Recognition is the process of finding named entities from the text and assigning them to categories such as organization's names, timelines, dates, etc.

The property ents help to find the named entities from the text.

```
for ent in text_doc_object.ents: print(ent.text, ent.start_char, ent.end_char,ent.label_, spacy.explain(ent.label_))
```

Graphical representation

Let us now graphically analyze the relationship between the tokens of the text using **displacy**. displacy renders the graph at **http://127.0.0.1:5000** as a server. We can clearly visualize the semantic relationship between named entities.

```
from spacy import displacy text = ("Natural Language Processing provides computers the capability of understanding text") text_doc_object = nlp(text) displacy.serve(text_doc_object, style='dep')
```

Conclusion

In this tutorial, we learned how spaCy can be used to perform natural language processing. It provides production quality code and is considered the most commonly used library for natural language processing. Natural language processing finds great application as text serves as an integral part of any application like chatbots, digital assistants.

And finally, it does not go without saying,

```
Thanks for reading!
```

Reference :
Image 1: https://www.google.com/search?q=phases+of+nlp&rlz=1C1ONGR_enIN945IN945&sxsrf=AOaemvJU5vJ3GeGXUnZUV8EfgQHrydw-vg:1631866504218&source=lnms&tbm=isch&sa=X&sqi=2&ved=2ahUKEwjn2be6yIXzAhW-lZUCHTZWBdcQ_AUoAXoECAEQAw&biw=1536&bih=696&dpr=1.25#imgrc=HepC3kLxoaq9bM

Article Url - https://www.analyticsvidhya.com/blog/2021/09/complete-tutorial-on-natural-language-processing-using-spacy/

Siddharth Sachdeva
Computer science enthusiast