**NLP(Natural Language Processing)**

NLP is a subfield of Artificial Intelligence and is concerned with interactions between computers and human languages.

Use cases of NLP are:

* Automatic summarization
* Named entity recognition
* Question answering systems

**SPACY**

spacy is a free, open-source library for NLP in Python. It’s written in Cython.

USING SPACY

In [ ]:

import spacy

nlp=spacy.load('en')

THE STEPS OF NLP ARE:

IMG\_20200828\_181111.jpg

1. **SENTENCE DETECTION(SEGMENTATION):**

Sentence Detection is the process of locating the start and end of sentences in a given text.

In spaCy, the sents property is used to extract sentences.

In [ ]:

about\_text = ('Gus Proto is a Python developer currently'

' working for a London-based Fintech'

' company. He is interested in learning'

' Natural Language Processing.')

about\_doc = nlp(about\_text)

sentences = list(about\_doc.sents)

len(sentences)

Out[ ]:

2

In [ ]:

for sentence in sentences:

print (sentence)

Gus Proto is a Python developer currently working for a London-based Fintech company.

He is interested in learning Natural Language Processing.

2. TOKENIZATION:

Tokenization is the next step after sentence detection.It allows us to identify the basic units in given text.These basic units are called tokens. Tokenization is useful because it breaks a text into meaningful units.These units are used for further analysis , like part of speech tagging.

In [ ]:

[token.text for token in about\_doc]

Out[ ]:

['Gus',

'Proto',

'is',

'a',

'Python',

'developer',

'currently',

'working',

'for',

'a',

'London',

'-',

'based',

'Fintech',

'company',

'.',

'He',

'is',

'interested',

'in',

'learning',

'Natural',

'Language',

'Processing',

'.']

1. **Parts of Speech Tagging:**

Parts of speech or POS is a grammatical role that explains how a particular word is used in a sentence.

Parts of speech tagging is the process of assigning a POS tag to each token depending on its usage in the sentence. POS tags are useful for assigning a syntactic category like noun or verb to each word.

Here, two attributes of the Token class are accessed:

tag\_ lists the fine-grained part of speech.

pos\_ lists the coarse-grained part of speech.

spacy.explain gives descriptive details about a particular POS tag. spaCy provides a complete tag list along with an explanation for each tag.

In [ ]:

for token in about\_doc:

print (token, token.pos\_, token.tag\_, spacy.explain(token.tag\_))

Gus PROPN NNP noun, proper singular

Proto PROPN NNP noun, proper singular

is AUX VBZ verb, 3rd person singular present

a DET DT determiner

Python PROPN NNP noun, proper singular

developer NOUN NN noun, singular or mass

currently ADV RB adverb

working VERB VBG verb, gerund or present participle

for ADP IN conjunction, subordinating or preposition

a DET DT determiner

London PROPN NNP noun, proper singular

- PUNCT HYPH punctuation mark, hyphen

based VERB VBN verb, past participle

Fintech PROPN NNP noun, proper singular

company NOUN NN noun, singular or mass

. PUNCT . punctuation mark, sentence closer

He PRON PRP pronoun, personal

is AUX VBZ verb, 3rd person singular present

interested ADJ JJ adjective

in ADP IN conjunction, subordinating or preposition

learning VERB VBG verb, gerund or present participle

Natural PROPN NNP noun, proper singular

Language PROPN NNP noun, proper singular

Processing PROPN NNP noun, proper singular

. PUNCT . punctuation mark, sentence closer

**4. Lemmatization:**

Lemmatization is the process of reducing inflected forms of a word while still ensuring that the reduced form belongs to the language. This reduced form or root word is called a lemma.

For example, organizes, organized and organizing are all forms of organize. Here, organize is the lemma. The inflection of a word allows you to express different grammatical categories like tense (organized vs organize), number (trains vs train), and so on.

If you do not lemmatize the text, then organize and organizing will be counted as different tokens, even though they both have a similar meaning. Lemmatization helps you avoid duplicate words that have similar meanings.

spaCy has the attribute lemma\_ on the Token class. This attribute has the lemmatized form of a token:

In [ ]:

for token in about\_doc:

... print (token, token.lemma\_)

Gus Gus

Proto Proto

is be

a a

Python Python

developer developer

currently currently

working work

for for

a a

London London

- -

based base

Fintech Fintech

company company

. .

He -PRON-

is be

interested interested

in in

learning learn

Natural Natural

Language Language

Processing Processing

. .

**5. STOP WORDS:**

Stop words are the commmon words in a language such as the,are, but, and they.Most sentences need to contain stop words in ordr to be full senetence that make sense. Generally,stop words are removed beacuse they aren't significant and distort the word frequency analysis. Spacy has list of stop words for the English language.

In [ ]:

import spacy

from spacy.lang.en.stop\_words import STOP\_WORDS

spacy\_stopwords = spacy.lang.en.stop\_words.STOP\_WORDS

len(spacy\_stopwords)

for stop\_word in list(spacy\_stopwords)[:10]:

print(stop\_word)

as

fifty

they

ever

via

third

n‘t

on

sometimes

beside

You can remove stop words from the input text;

In [ ]:

for token in about\_doc:

if not token.is\_stop:

print(token)

Gus

Proto

Python

developer

currently

working

London

-

based

Fintech

company

.

interested

learning

Natural

Language

Processing

.

Stop words like is,a,for,the, and in are not printed in the output above. You can also create a list of tokens not cintaining stop words:

In [ ]:

about\_no\_stopword\_doc = [token for token in about\_doc if not token.is\_stop]

print (about\_no\_stopword\_doc)

[Gus, Proto, Python, developer, currently, working, London, -, based, Fintech, company, ., interested, learning, Natural, Language, Processing, .]

about\_no\_stopword\_doc can be joined with spaces to form a sentence with no stop words.

**6. Dependency Parsing:**

Dependency parsing is the process of extracting the dependency parse of a sentence to represent its grammatical strcuture. It defines the dependency realtionship between headwords and their dependents.The head of a sentence has no dependency and is called the root of the sentence.The verb is usually the head of the sentence.All other words are linked to the headword.

The dependence can be mapped in a directed graph representation:

Words are the nodes.

The grammatical realtionships are the edges.

Dependency parsing helps you know what role a word palys in the text and how different words relate to each other.It's also used in Shallow parsing and named entity recognition.

In [ ]:

piano\_text = 'Gus is learning piano'

piano\_doc = nlp(piano\_text)

for token in piano\_doc:

print (token.text, token.tag\_, token.head.text, token.dep\_)

Gus NNP learning nsubj

is VBZ learning aux

learning VBG learning ROOT

piano NN learning dobj

nsubj is the subject of the word.Its headword is a verb.

aux is an auxiliary word . Its headword is a verb.

dobj is the direct object object of the verb.Its headword is a verb.

**7. Noun Pharse:**

A noun pharse is a pharse that has a noun as its head.It could that has a nou as its head.it could also include other kinds of words, such as adjectives,ordinals,determiners.noun pharse are useful for explainiing the context of the sentence.They help you infer whta is being talked about in the sentence.

Spacy has the property noun\_chunks on Doc object.You can use it to extract noun phrases:

In [ ]:

conference\_text = ('There is a developer conference'

' happening on 21 July 2019 in London.')

conference\_doc = nlp(conference\_text)

# Extract Noun Phrases

for chunk in conference\_doc.noun\_chunks:

print (chunk)

File "<ipython-input-1-b3e36bf20030>", line 3

conference\_doc = nlp(conference\_text)

By looking at noun pharse, you can get information about your text. For example,a developer conference indicates that the text mentions a conference , while the data 21 july lets you know that conference is scheduled for 21 july.You can figure out whether the conference is in past or the future .London tells you that the conference is in London.

**8. Named Entity Recognition:**

Named Entity Recognition(NER) is the proces sof locating named entities in unstrcuted text and then classifying them into pre-defined categories,such as person names,organizations,locations,monetary values,percentage,time expressions and so on.

You can use NER to know more about the meaning of your text. For example,you could use it to populate tags for a set of documents in order to improve the keyword search. You could also use it to categorize customer support tickets into relevant categories.

Spcay has the property ents on Doc objects. You can use it to extract named entities:

In [ ]:

piano\_class\_text = ('Great Piano Academy is situated'

' in Mayfair or the City of London and has'

' world-class piano instructors.')

piano\_class\_doc = nlp(piano\_class\_text)

for ent in piano\_class\_doc.ents:

print(ent.text, ent.start\_char, ent.end\_char,

ent.label\_, spacy.explain(ent.label\_))

Great Piano Academy 0 19 ORG Companies, agencies, institutions, etc.

Mayfair 35 42 GPE Countries, cities, states

the City of London 46 64 GPE Countries, cities, states

In the above example ,ent is a Span object with variours attribute:

text gives the Unicode text representation of the entity.

start\_char denotes the character offset for the start of the entity.\

end\_char denotes the character offset for the end of the entity.

label\_gives teh label of teh entity.