Text Processing Using Spacy

Stop Words

- Stop words are words that are filtered out before or after the natural language data(text) are processed.
- **stop words** typically refers to the most common words in a language.
- There is no universal list of **stop words** that is used by all NLP tools in common.

- **what are stop words?**
  - **Stopwords** are the **words** in any language which does not add much meaning to a sentence.
  - They can safely be ignored without sacrificing the meaning of the sentence.
  - For some search engines, these are some of the most common, short function words, such as the, is, at, which, and on.

- **when to remove stop words?**
  - If we have a task of **text classification** or **sentiment analysis** then we should remove stop words as they do not provide any information to our model i.e. **keeping out unwanted words out of our corpus**.
  - But, if we have the task of **language translation** then **stopwords** are useful, as they have to be translated along with other words.
  - There is no hard and fast rule on when to remove stop words
    1. Remove **stopwords** if task to be performed is one of **Language Classification**, **Spam Filtering**, **Caption Generation**, **Auto-Tag Generation**, **Sentiment analysis**, or something that is related to **text classification.**
    2. Better not to remove **stopwords** if task to be performed is one of **Machine Translation**, **Question Answering problems**, **Text summarization**, **Language Modeling**.

- **Pros of Removing stop words**
  - **Stopwords** are often removed from the text before training deep learning and machine learning models since stop words occur in abundance, hence providing little to no unique information that can be used for classification or clustering.
  - On removing **stopwords**, dataset size decreases, and the time to train the model also decreases without a huge impact on the accuracy of the model.
  - **Stopword** removal can potentially help in improving performance, as there are fewer and only significant tokens left. Thus, the classification accuracy could be improved.

- **Cons of Removing Stop Words**
  - Improper selection and removal of **stop words** can change the meaning of our text. So we have to be careful in choosing our stop words.
  - **Example:** ***This movie is not good***
    - If we **remove (not)** in pre-processing step the sentence (this movie is good) indicates that it is positive which is wrongly interpreted.

- **Removing Stop words using SpaCy Library**
  - Comparing to **NLTK**, spacy got bigger set of stop words (326) than that of NLTK (179)
  - **installation: (spacy, English Language Model)**
    - *pip install -U spacy*
    - _python -m spacy download en_core_web*sm*
  - **Demo shown in below Cell:**

In [1]:

```
#installation
!pip install -U spacy
!python -m spacy download en_core_web_sm
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: spacy in /home/anish/.local/lib/python3.8/site-packages (2.3.4)
Requirement already satisfied: srsly<1.1.0,>=1.0.2 in /home/anish/.local/lib/python3.8/site-packages
(from spacy) (1.0.4)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /home/anish/.local/lib/python3.8/site-packag
es (from spacy) (3.0.4)
Requirement already satisfied: numpy>=1.15.0 in /usr/lib/python3/dist-packages (from spacy) (1.17.4)
Requirement already satisfied: catalogue<1.1.0,>=0.0.7 in /home/anish/.local/lib/python3.8/site-pack
ages (from spacy) (1.0.0)
Requirement already satisfied: wasabi<1.1.0,>=0.4.0 in /home/anish/.local/lib/python3.8/site-package
s (from spacy) (0.8.0)
Requirement already satisfied: setuptools in /usr/lib/python3/dist-packages (from spacy) (45.2.0)
Requirement already satisfied: thinc<7.5.0,>=7.4.1 in /home/anish/.local/lib/python3.8/site-packages
(from spacy) (7.4.3)
Requirement already satisfied: plac<1.2.0,>=0.9.6 in /home/anish/.local/lib/python3.8/site-packages
(from spacy) (1.1.3)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /home/anish/.local/lib/python3.8/site-packages
(from spacy) (4.48.2)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/lib/python3/dist-packages (from spacy
) (2.22.0)
Requirement already satisfied: blis<0.8.0,>=0.4.0 in /home/anish/.local/lib/python3.8/site-packages
(from spacy) (0.7.3)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /home/anish/.local/lib/python3.8/site-packages
(from spacy) (2.0.4)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /home/anish/.local/lib/python3.8/site-pa
ckages (from spacy) (1.0.4)
Requirement already satisfied: numpy>=1.15.0 in /usr/lib/python3/dist-packages (from spacy) (1.17.4)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /home/anish/.local/lib/python3.8/site-packages
(from spacy) (2.0.4)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /home/anish/.local/lib/python3.8/site-pa
ckages (from spacy) (1.0.4)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /home/anish/.local/lib/python3.8/site-packages
(from spacy) (4.48.2)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /home/anish/.local/lib/python3.8/site-packag
es (from spacy) (3.0.4)
Requirement already satisfied: numpy>=1.15.0 in /usr/lib/python3/dist-packages (from spacy) (1.17.4)
Requirement already satisfied: catalogue<1.1.0,>=0.0.7 in /home/anish/.local/lib/python3.8/site-pack
ages (from spacy) (1.0.0)
Requirement already satisfied: wasabi<1.1.0,>=0.4.0 in /home/anish/.local/lib/python3.8/site-package
s (from spacy) (0.8.0)
Requirement already satisfied: plac<1.2.0,>=0.9.6 in /home/anish/.local/lib/python3.8/site-packages
(from spacy) (1.1.3)
Requirement already satisfied: blis<0.8.0,>=0.4.0 in /home/anish/.local/lib/python3.8/site-packages
(from spacy) (0.7.3)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /home/anish/.local/lib/python3.8/site-packages
(from spacy) (2.0.4)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /home/anish/.local/lib/python3.8/site-pa
ckages (from spacy) (1.0.4)
Requirement already satisfied: srsly<1.1.0,>=1.0.2 in /home/anish/.local/lib/python3.8/site-packages
(from spacy) (1.0.4)
WARNING: You are using pip version 20.3; however, version 20.3.1 is available.
You should consider upgrading via the '/usr/bin/python3 -m pip install --upgrade pip' command.
/usr/bin/python: No module named spacy
```

In [6]:

```
import spacy
from nltk.tokenize import word_tokenize

def remove_stopwords(text):
    """
    Removes stopwords passed from the text passed as an arguments

    Arguments:
    text: raw text from where stopwords need to removed

    Returns:
    tokens_without_sw: list of tokens of raw text without stopwords
    """

    # loading english language model of spacy
    nlp = spacy.load("en_core_web_sm")

    # getting list of default stop words in spaCy english model
    stopwords =nlp.Defaults.stop_words

    # tokenize text
    text_tokens = word_tokenize(text)

    # remove stop words:
    tokens_without_sw = [word for word in text_tokens if word not in stopwords]

    # return list of tokens with no stop words
    return tokens_without_sw

# define sample text
sample_text = "Oh man, this is pretty cool. We will do more such things."

# remove stopwords calling defined functions
filtered_sentence = remove_stopwords(sample_text)
print("filtered sentence without stop words:\n", filtered_sentence)
```

```
filtered sentence without stop words:
 ['Oh', 'man', ',', 'pretty', 'cool', '.', 'We', 'things', '.']
```

Tokenization

- Tokenization refers to dividing the text into a sequence of words or sentences.
- **Word Tokenization**
  - Word Tokenization simply means splitting sentence/text in words.
  - Using attribute **token.text** to tokenize the **doc**
- **Sentence Tokenization**
  - Sentence Tokenization is the process of splitting up strings into sentences.
  - A sentence usually ends with a full stop (.), here focus is to study the structure of sentence in the analysis
  - use **sents** attribute from spacy to identify the sentences.

In [19]:

```python
# Word Tokenization

# import the spacy library
import spacy

# load the english model and initialize an object called 'nlp'
nlp = spacy.load("en_core_web_sm")

def tokenize_word(text):
    """
    Tokenize the text passed as an arguments into a list of words(tokens)

    Arguments:
    text: raw text

    Returns:
    words: list containing tokens in text
    """
    # passing the text to nlp and initialize an object called 'doc'
    doc = nlp(text)

    # Tokenize the doc using token.text attribute
    words = [token.text for token in doc]

    # return list of tokens
    return words

# define sample text
sample_text =  "Oh man, this is pretty cool. We will do more such things."

# tokenize  words
words = tokenize_word(sample_text)

# print tokens
print("**Word tokens**\n", words)
```

```
**Word tokens**
 ['Oh', 'man', ',', 'this', 'is', 'pretty', 'cool', '.', 'We', 'will', 'do', 'more', 'such', 'things
', '.']
```

In [20]:

```python
# Sentence Tokenization

# import the spacy library
import spacy

# load the english model and initialize an object called 'nlp'
nlp = spacy.load("en_core_web_sm")

def tokenize_sentence(text):
    """
    Tokenize the text passed as an arguments into a list of sentence

    Arguments:
    text: raw text

    Returns:
    sentences: list of sentences
    """
    # passing the text to nlp and initialize an object called 'doc'
    doc = nlp(text)

    # tokenize the sentence using sents attributes
    sentences = list(doc.sents)

    # return tokenize sentence
    return sentences

# define sample text
sample_text =  "Oh man, this is pretty cool. We will do more such things."

# tokenize sentence
sentences = tokenize_sentence(sample_text)

# print sentences
print("**Sentence tokens**\n", sentences)
```

```
**Sentence tokens**
 [Oh man, this is pretty cool., We will do more such things.]
```

Punctuation

- **punctuation** are special marks that are placed in a text to show the division between phrases and sentences.
- There are 14 punctuation marks that are commonly used in English grammar.
- They are, **period, question mark, exclamation point, comma, semicolon, colon, dash, hyphen, parentheses, brackets, braces, apostrophe, quotation marks, and ellipsis**.
- We can **remove punctuation** from text using **is_punct** attribute.

In [27]:

```python
# remove punctuations

# import the spacy library
import spacy

# load the english model and initialize an object called 'nlp'
nlp = spacy.load("en_core_web_sm")

def remove_punctuation(text):
    """
    removes punctuation symbols present in the raw text passed as an arguments

    Arguments:
    text: raw text

    Returns:
    not_punctuation: list of tokens without punctuation
    """
    # passing the text to nlp and initialize an object called 'doc'
    doc = nlp(text)

    not_punctuation = []
    # remove the puctuation
    for token in doc:
        if token.is_punct == False:
            not_punctuation.append(token)

    return not_punctuation

# define sample text
sample_text =  "Oh man, this is pretty cool. We will do more such things."

# remove punctuation
not_punctuation = remove_punctuation(sample_text)

print("**list of tokens without punctutaions:**\n", not_punctuation)
```

```
**list of tokens without punctutaions:**
 [Oh, man, this, is, pretty, cool, We, will, do, more, such, things]
```

As we can see all the punctuation symbol is removed.

Lower Casing

- Converting word to lower case (NLP->nlp).
- **Q.Why Lower Casing**
    - Words like **Book** and **book** mean the same,
    - When not converted to the lower case those two are represented as two different words in the vector space model (resulting in more dimension).
    - Higher the dimension, more computation resources are required.

In [42]:

```python
def lower_casing(text):
    """
    Accepts text as arguments and return text in lowercase

    Arguments:
    text: raw text

    Returns:
    text_to_lower: text converted to lower case
    """
    text_to_lower = text.lower()

    return text_to_lower

sample_text = "Books are on the table."

# lower casing
print(lower_casing(sample_text))
```

```
books are on the table.
```

Lemmatization

- Lemmatization is the process of converting a word to its base form.
- For example, lemmatization would correctly identify the base form of **caring** to **care**
- Lemmatization can be carried out using the attribute **token.lemma_**

In [29]:

```
# lemmatization

# import the spacy library
import spacy

# load the english model and initialize an object called 'nlp'
nlp = spacy.load("en_core_web_sm")

def lemmatization(text):
    """
    obtain the lemma of the each token in the text, append to the list, and returns the list

    Arguments:
    text: raw text

    Returns:
    token_lemma_list: list containing token with its lemma
    """

    # passing the text to nlp and initialize an object called 'doc'
    doc = nlp(text)

    token_lemma_list = []
    # Lemmatization
    for token in doc:
        token_lemma_list.append((token.text, token.lemma_))

    return token_lemma_list

# define sample text
sample_text = "The Republican president is being challenged by Democratic Party nominee Joe Biden"

# Lemmatization
token_lemma_list = lemmatization(sample_text)

#printing
for token_lemma in token_lemma_list:
    print(token_lemma[0], '-->', token_lemma[1])
```

```
The --> the
Republican --> republican
president --> president
is --> be
being --> be
challenged --> challenge
by --> by
Democratic --> Democratic
Party --> Party
nominee --> nominee
Joe --> Joe
Biden --> Biden
```

The word **is** converted into **be**, **being** -> **be**, **challenged** -> **challenge.**

POS-Tagging

- Parts-of-speech tagging is the process of tagging words in textual input with their appropriate parts of speech.
- This is one of the core feature loaded into the pipeline.

In [46]:

```python
# import the spacy library
import spacy

# load the english model and initialize an object called 'nlp'
nlp = spacy.load("en_core_web_sm")

def pos_tagging(text):
    # passing the text to nlp and initialize an object called 'doc'
    doc = nlp(text)

    pos_list = []
    for token in doc:
        pos_list.append((token.text, token.pos_, token.tag_))
    return pos_list

# define sample text
sample_text = 'Antibiotics do not help, as they do not work against viruses.'

# pos_tagging
pos_list = pos_tagging(sample_text)

# display
for pos in pos_list:
    print(pos[0], pos[1], pos[2])
```

```
Antibiotics NOUN NNS
do AUX VBP
not PART RB
help VERB VB
, PUNCT ,
as SCONJ IN
they PRON PRP
do AUX VBP
not PART RB
work VERB VB
against ADP IN
viruses NOUN NNS
. PUNCT .
```

As you can see, the words are tagged with appropriate parts of speech.
One important note, some words can be both noun or verb depending on context.

Named entity recognition

- It is the process of detecting the named entities such as the person name, the location name, the company name, the quantities and the monetary value.
- We can find the named entity using spaCy **ents** attribute class.
- **Entity attributes details**

In [47]:

```
# Named Entity Recognition

# import the spacy library
import spacy

# import displacy
from spacy import displacy


# load the english model and initialize an object called 'nlp'
nlp = spacy.load("en_core_web_sm")

def named_entity_recognition(text):
    """
    returns entity_text and entity labels as a tuple

    Arguments:
    text: raw text

    Returns:
    entity_text_label: entity text and labels as a tuple
    """
    # passing the text to nlp and initialize an object called 'doc'
    doc = nlp(text)

    #named entity recogniton using doc.ents
    entity_text_label = []

    for entity in doc.ents:
        entity_text_label.append((entity.text, entity.label_))

    return entity_text_label


# define sample text
sample_text = "The Republican president is being challenged by Democratic Party nominee Joe Biden, who \
                is best known as Barack Obama’s vice-president but has been in US politics since the 1970s"

# Named Entity Recognition
entity_text_label = named_entity_recognition(sample_text)

# display entity text and label
for text_label in entity_text_label:
    print(text_label[0], '->', text_label[1])

# Visualizing the named entity description
print("\n***VISUALIZING NAMED ENTITY RECOGNIZER***")
displacy.render(nlp(sample_text), style = "ent",jupyter = True)
```

```
Republican -> NORP
Democratic Party -> ORG
Joe Biden -> PERSON
Barack -> GPE
US -> GPE
the 1970s -> DATE

***VISUALIZING NAMED ENTITY RECOGNIZER***
```

The Republican **NORP** president is being challenged by Democratic Party **ORG** nominee Joe Biden **PERSON** , who is best known as Barack **GPE** Obama’s vice-president but has been in US **GPE** politics since the 1970s **DATE**