LEMMATIZATION AND STEMMING

IN PYTHON

NLTK spaCy

@LEARN.MACHINELEARNING

WHY???

- In grammar, inflection is the modification of a word to express different grammatical categories such as tense, case, voice, aspect, person, number, gender, and mood.
- An inflection expresses one or more grammatical categories with a prefix, suffix or infix, or another internal modification such as a vowel change

- The degree of inflection may be higher or lower in a language.
- Both tries to do the same thing(root forms of words) in a different approch.

WHY???

- The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. For instance:
  - am, are, is ==> be
  - car, cars, car's, cars' ==> car
- The result of this mapping of text will be something like:
  - the boy's cars are different colors ==> the boy car be differ color

STEMMING??

- Stemming refers to reducing a word to its root form. While performing natural language processing tasks, you will encounter various scenarios where you find different words with the same root.
- For instance, compute, computer, computing, computed, etc. You may want to reduce the words to their root form for the sake of uniformity. This is where stemming comes in to play.
- Root is the part of the word to which you add affixes such as (-ed,-ize, -s,-de,mis).

⊳⊳⊳

STEMMING USING NLTK

- There are two types of stemmers in NLTK: Porter Stemmer and Snowball stemmers.
- Both of them have been implemented using different algorithms.

```
>>> #Porter Stemmer
>>> import nltk
>>> from nltk.stem.porter import *
>>> stemmer = PorterStemmer()
>>> #Example words
>>> tokens = ['compute', 'computer', 'computed',
'computing']
>>> for token in tokens:
>>>       print(token + ' --> ' + stemmer.stem(token))
>>> #Result
compute --> comput, computer --> comput
computed --> comput, computing --> comput
```

NLTK

STEMMING USING NLTK

```
>>> #Snowball Stemmer
>>> from nltk.stem.snowball import SnowballStemmer
>>> stemmer = SnowballStemmer(language='english')
>>> tokens = ['compute', 'computer', 'computed',
'computing']
>>> for token in tokens:
>>>     print(token + ' --> ' + stemmer.stem(token))
>>> #Result
compute --> comput
computer --> comput
computed --> comput
computing --> comput
```

LEMMATIZATION (SPACY)

- Lemmatization, unlike Stemming, reduces the inflected words properly ensuring that the root word belongs to the language.
- In Lemmatization root word is called Lemma.
- A lemma (plural lemmas or lemmata) is the canonical form, dictionary form, or citation form of a set of words.

```
>>> import spacy
>>> sp = spacy.load('en_core_web_sm')
>>> tex = sp(u'compute computer computed computing')
>>> for word in tex:
>>>   print(word.text,  word.lemma_)
>>> #Result
compute compute, computer computer
computed compute, computing computing
```

LEMMATIZATION

- You can see that unlike stemming where the root we got was "comput", the roots that we got here are actual words in the dictionary.
- Lemmatization converts words in the second or third forms to their first form variants.

STEMMING OR LEMMATIZATION?

- Stemming and Lemmatization both generate the root form of the inflected words. The difference is that stem might not be an actual word whereas, lemma is an actual language word.
- Stemming follows an algorithm with steps to perform on the words which makes it faster. Whereas, in lemmatization, you used WordNet corpus and a corpus for stop words as well to produce lemma which makes it slower than stemming. You also had to define a parts-of-speech to obtain the correct lemma.

APPLICATIONS

- Tokenization, Stemming and Lemmatization are some of the most fundamental natural language processing tasks
- text categorization
- text clustering
- concept/entity extraction
- production of granular taxonomies
- sentiment analysis
- document summarization
- entity relation modeling