**Spacy**

**When developed**

2015 by explosion AI .stable release 2020

**What ?**

Spacy is free open source library for NLP task that is designed to process large amount of text.

It uses CNN to build models for POS tagging,NER,dependency parser.

It is built using Thinc: A machine learning library optimized for CPU usage and deep learning with text input.

**AI Model**

POS tagging

NER

dependency parsing

text categorization

**RUle based model**

lemmatization,tokenization,segmentation

**why Spacy**

* It is more efficient than other nlp library.

Package Tokenizer Tagging Parsing

spaCy 0.2ms 1ms 19ms

CoreNLP 2ms 10ms 49ms

NLTK 4ms 443ms –

* It is more accurate than other nlp tools. The pos accurcay is 92 % trained on ontonote5plus.

Refer the below link for facts and figures of spacy.

<https://spacy.io/usage/facts-figures>

* It provides us the provision to train customised model for pos tagger,NER and dependency parser using spacy standard format.
* Other NLP libraries provides a set of algorithms to be used for processing text while spacy provides single optimized algorithm.

**Lemmatization algorithm**

1. tokenize the text.

2. assign POS tag to the tokens.

3. convert the tokens to lower case.

4. lemma\_forms=[]. check if the word is in the index.

e.g: longer -adjective

INDEX = { exception rules- contains symantically associated words.

"adj": ADJECTIVES, e.g:bad-> worst

"adv": ADVERBS,

"noun": NOUNS,

"verb": VERBS

}

RULES = {

"adj": ADJECTIVE\_RULES,e.g: ["er", ""],["est", ""]

"noun": NOUN\_RULES,

"verb": VERB\_RULES,

"punct": PUNCT\_RULES

}

Look for the exceptions, get them if the lemma from the exception list if the word is in it.

Apply the rules

Save the ones that are in the index lists

5. oov list-out of vocabulary.there are no lemma form,the word is returned as it is and append to oov list.

**Spacy tokenization**

* Tokenizer exception: Special-case rule to split a string into several tokens or prevent a token from being split when punctuation rules are applied. e.g:don't do,n't
* Prefix: Character(s) at the beginning, e.g. $, (, “, ¿.
* Suffix: Character(s) at the end, e.g. km, ), ”, !.
* Infix: Character(s) in between, e.g. -, --, /, ….

priorities-token match,special cases,prefix,suffix,,url,infix

Algorithms of tokenizer used by spacy

1. split the text based on the white space.

"'Let's go to U.K.!'"

'Let's go to U.K.!

2. search for the match token. (match token is the one which does not contain prefix,suffix and infix)

'Let's go to U.K.!

3. search for the special case for tokens found in step 2.(exclusive special case ,prefix,suffix should not be present)

4. search for the prefix,if no prefix,then find suffix.If both found,split at prefix and suffix.

' Let's go to U.K!'

' Let's go to U.K! '

' Let's go to U.K ! '

5.After step4 ,search for special case and split into tokens.

' Let 's go to U.K. ! '

**How abbreviations are handled in spacy**

Spacy has inbuilt abbreviation model based on wordnet corpus which is a part of special case. It the abbr. is present in exception,it will preserve it or it will perform tokenization.

How to provide domain specific or customised abbreviation?

We can add our customised abbreviation to spacy as a component as shown below

import spacy

from scispacy.abbreviation import AbbreviationDetector

nlp = spacy.load("en\_core\_sci\_sm")

# Add the abbreviation pipe to the spacy pipeline.

abbreviation\_pipe = AbbreviationDetector(nlp)

nlp.add\_pipe(abbreviation\_pipe)

Adding special tokenization case -refer document