Part of speech tagging, what it is and how can you implement it (on python)

1. What it is

Before we look at a couple of libraries that can help us implement part of speech (POS) tagging we should first define what it is and why do we need it.

In a language, we have grammatical labels for words such as *noun*, *adjective*, *verb*, *etc*. In computer science, these labels are called part of speech tags, and there's a larger pool of these labels. POS tagging is the process of marking up a word in a corpus to a corresponding part of a speech tag, based on its context and definition. For example, lets consider the following sentence: *All hotel rooms are pretty much the same, although the room number might change*. Here's a part-of-speech tagged version:

All$_{DT}$ *hotel*$_{NN}$ *rooms*$_{NNS}$ *are*$_{VBP}$ *pretty*$_{RB}$ *much*$_{RB}$ *the*$_{DT}$ *same*$_{JJ}$*,* *although*$_{IN}$ *the*$_{DT}$ *room*$_{NN}$ *number*$_{NN}$ *might*$_{MD}$ *change*$_{VB}$*.*

Where VBP and VP are different types of verbs, NN and NNS are singular and plural nouns. This is just a subset of about 80commonly used tags. (ChengXiang Zhai, 2016)

2. Why we need it

POS tagging is used to understand the meaning of any sentence or to extract relationships and build a knowledge graph. If we think of the useful Bag of Words model in NLP, these fail to capture the syntactic relations between words. This is in part because there might be ambiguity in what a word might represent. For example, in the sentence "Watch the play", *play* is a Noun, on the other hand if we say "Play with us", *play* is a verb.

3. How we can implement it

3.1 NLTK

NLTK is a leading platform for building Python programs to work with human language data. (NLTK 3.5 documentation, n.d.)

Here's a sample code on how you could do it

```
import nltk
from nltk.tokenize import word_tokenize

text = word_tokenize("This is a test on how part of speech tagging works")
print(nltk.pos_tag(text))
```

and here’s the output:

```
[('This', 'DT'), ('is', 'VBZ'), ('a', 'DT'), ('test', 'NN'), ('on', 'IN'), ('how', 'WRB'), ('part', 'NN'),
('of', 'IN'), ('speech', 'NN'), ('tagging', 'NN'), ('works', 'NNS')]
```

An array of POS tagged words.

3.2 spaCy

spaCy is one of the best text analysis libraries. spaCy excels at large-scale information extraction tasks and is one of the fastest in the world.

Here’s a sample of how you would do the same tagging using spaCy

```
import spacy

nlp = spacy.load("en_core_web_sm")
text = ("This is a test on how part of speech tagging works")
doc = nlp(text)
for token in doc:
    print(token, token.pos_)
```

in this case we use a model “en_core_web_sm”. This is a core model, a general-purpose pretrained models to predict named entities, part-of-speech tags and syntactic dependencies. Can be used out-of-the-box and fine-tuned on more specific data. (spaCy Models, n.d.)

This is the output:

```
This DET
is AUX
a DET
test NOUN
on ADP
how ADV
part NOUN
of ADP
speech NOUN
tagging VERB
works NOUN
```

here we get a document with tokens that have a pos property.

3.3 StanfordNLP

StanfordNLP is the combination of the software package used by the Stanford team in the CoNLL 2018 Shared Task on Universal Dependency Parsing, and the group's official Python interface to the Stanford CoreNLP software.

Now we try the same text with StanfordNLP:

```
import stanfordnlp


stanfordnlp.download('en')

nlp = stanfordnlp.Pipeline(processors = "tokenize,mwt,lemma,pos")

doc = nlp("""This is a test on how part of speech tagging works""")

def extract_pos(doc):

    parsed_text = {'word':[], 'pos':[], 'exp':[]}

    for sent in doc.sentences:

        for wrd in sent.words:

            parsed_text['word'].append(wrd.text)

            parsed_text['pos'].append(wrd.pos)

    return parsed_text

extract_pos(doc)
```

And the output is the following:

```
{'word': ['This', 'is', 'a', 'test', 'on', 'how', 'part', 'of', 'speech', 'tagging', 'works'], 'pos': ['DT', 'VBZ', 'DT', 'NN', 'IN', 'WRB', 'NN', 'IN', 'NN', 'NN', 'NNS'], 'exp': []}
```

This one was the hardest to setup but also potentially the fastest. You need to install the py package torch which is a pain, and it only seems to work as of now with Python 3.7. You also need to download the language model for English which is 1.96 GB.

4. Conclusions

There are several packages to handle POS tagging, many more just for python than the ones covered in this paper. But these are considered the most important ones (the established, the newcomer and the high-end). And fortunately for the most part they are easy to use.