HW-1

Name: Wang, Jue **NetID:** jwr0983

Relevant Code

Github Link: https://github.com/MSIA/jwr0983_msia_text_analytics_2020/tree/homework1

Problem 1

For this problem I compared tokenization, stemming and POS tagging in two libraries: NLTK and Spacy. In addtion, I compared the use of parallelization in both libraries when doing word and sentence level stemming.

In terms of the time taken for each task with and without parallelization, here is a list of all the times taken in seconds:

- NLTK
  - Tokenization: 4.29
  - Tokenization with Parallelization: 2.38
  - Stemming: 8.43
  - POS Tagging: 18.2
- Spacy
  - Tokenization: 83.18
  - Tokenization with Parallellization: 199.8
  - Lemmatization: 88.59
  - POS Tagging: 89.37

Spacy currently doesn't support built-in stemming, so we are using lemmatization instead. From the above list of times, we see that NLTK is quicker in completing these tasks than Spacy, and is better paired with parallelization when doing tokenization, since we do see a drastic decrease in time for tokenization with parallelization in NLTK than without parallelization. Also, Spacy does not support operation for word length > 100000, and we had to manually reset the max limit for certain operations to work under Spacy.

Problem 2

For this problem we are writing two regex expressions that match email address and dates. For the first part on matching email addresses, I'm using the following expression:

```
__r'[A-Za-z0-9_\-\.]+\@[A-Za-z0-9_\-\.]+\.[A-Za-z0-9_\-\.]+__.
```

This expression makes sure that the matched string has upper/lower case letters, dash/underscores, or dots prior to the @ symbol, the same sets of characters before and after the "." in the email address.

Some examples parsed:

- 'p00261@psilink.com'
- '930416.141520.7h1.rusnews.w165w@mantis.co.uk'
- '2944079995.1.p00261@psilink.com'
- 'usenet@worldlink.com'
- 'mathew@mantis.co.uk'
- 'decay@cbnewsj.cb.att.com'

For the second part on matching dates, I'm mathing for types of date strings separately before combining them into a single list of matched date strings.

- To match dates with format "dd/mm/yyyy" or "dd/mm/yy":

```
r'[0-9]+/[0-9]+/[0-9]+'
```

- To match dates with format "yyyy-mm-dd":

```
r'[0-9]{4}\-[0-9]{2}\-[0-9]{2}'
```

- To match dates with format "dd MONTH yyyy":

```
r'\d+ [JFAMSOND]\w+ \d+'
```

- To match dates with format "MONTH dd[st/th] yyyy":

```
r'[JFAMSOND]\w+ \d+[th|st]+ \d+'
```

Some examples parsed:

- '2001-01-02'
- '3/18/93'
- '3/31/93'
- '16 Apr 1993'
- 'February 17th 1992'