Text Analysis Homework 1
Megan Hazlett
September 28, 2020

Github : https://github.com/MSIA/mlh9188_msia_text_analytics_2020/tree/homework1

1. After completing the lab assignment, write a 1-page (or less) comparison report for the libraries you used during the lab, including comparison of running times, ease of parallelization, performance review, etc. Your work will be judged based on how useful and thorough your report is.

For this assignment I decided to compare the text analysis python packages, NLTK and Spacy. I evaluated each language based on tokenization (sentiments and words), stemming/lemmatization, and part of speech tagging. I used the following article, "Florida Man Sues Dating Site Over Refund After Doctor Ordered Him to Stay Home" to inform my analysis. The results are as follows.

For tokenization, NLTK and Spacy have similar timing. For NLTK, the performance is moderate. It sometimes considers punctuation as whole words and often breaks apart links. The sentiments also tended to be lengthy. I found Spacy's tokenization more useful because the sentiments were less lengthy and more like separate human thoughts.

NLTK uses stemming, while Spacy uses lemmatization. While the two techniques have the shared goal to find root words, stemming takes a crude approach and simply chops off the end of a word, whereas lemmatization takes more time and employs grammar to find the root. When comparing the actual results, I found that Spacy's lemmatization was much more accurate and useful. NLTK's stemming turned common words like "February" into "Februari" and "newletter" into "newslett". In addition, it was obvious that the software was not programmed to recognize newer words (i.e. turned "coronavirus" into "coronaviru"). Spacy's lemmatization, on the other hand, had far less of these issues and even recognized "coronavirus" as its own word.

For part of speech tagging, again, NLTK was faster than Spacy. While the accuracy seemed to be similar, I liked the layout of Spacy's results better. I appreciated how it wrote out when a word was a noun, rather than using a code for me to look up.

As far as parallelization, NLTK and Spacy both provide fairly simple options. With NLTK, one can use python's multiprocessing library and employ the pooling function. On the other hand, Spacy has a built in parallelization function in its library.

In conclusion, although NLTK is usually faster, I believe that Spacy is better in terms of accuracy and usefulness. At a glance, lemmatization seems to have better root identification as well as new word recognition than stemming. Spacy's sentiments in tokenization also tend to be more useful and break apart separate thoughts.

2. Write and test regular expressions for the following tasks:

2.1 Match all emails in text and compile a set of all found email addresses.

See attached Jupyter Notebook for running code.

I used the following article to find email addresses: I used the following code to find all emails in the text corpus: Breastfeeding and Breast Cancer Risk Reduction: Implications for Black Mothers

I used the following python code to find all emails:

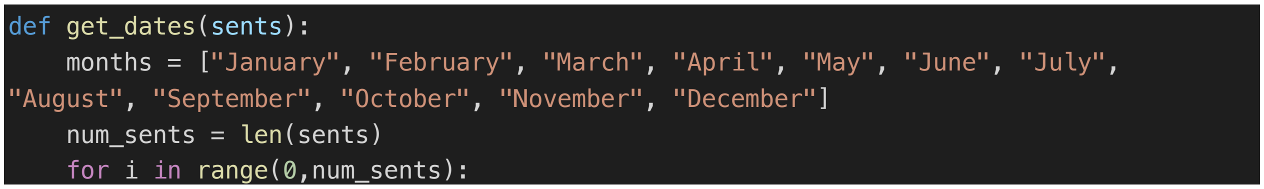
```
maternity_sents = sent_tokenize(maternity_content)

maternity_words = word_tokenize(maternity_content)

num_sents = len(maternity_sents)

for i in range(0,num_sents):

    line = maternity_sents[i]

    match = re.search(r'[\w\.-]+@[\w\.-]+', line)

    if match != None:

        print(match)
```

2.2 Find all dates in text (e.g. 04/12/2019, April 20th 2019, etc).

See attached Jupyter Notebook for running code.

I used the following article to find email addresses: "Florida Man Sues Dating Site Over Refund After Doctor Ordered Him to Stay Home"

I used the following python code to find all dates:



```
def get_dates(sents):
    months = ["January", "February", "March", "April", "May", "June", "July",
"August", "September", "October", "November", "December"]
    num_sents = len(sents)
    for i in range(0,num_sents):
        line = sents[i]
        match = re.search(r'\d{1,2}/\d{1,2}/\d{4}', line)
        match2 = re.search(r'\d{1,2}-\d{1,2}-\d{4}', line)
        match3 = re.search(r'\d{4}/\d{1,2}/\d{1,2}', line)
        match4 = re.search(r'\d{4}-\d{1,2}-\d{1,2}', line)
        if match != None:
            print(match)
        if match2 != None:
            print(match2)
        if match3 != None:
            print(match3)
        if match4 != None:
            print(match4)
        month_key = any(months in line for months in months)
        if month_key != False:
            match5 = re.search(r'\b[A-Z].*?\b[" "]{1}\d{1,2}[\,\.]{1}[" "]{1}\d{4}',
line)
            if match5 != None:
                print(match5)
```