Rachel Rosenberg
MSiA 490: Text Analytics
October 5, 2019

Comparison of Two Libraries: NLTK and Spacy

The libraries I completed this analysis in were NLTK and SpaCy. I downloaded the "20 Newsgroups 18828" dataset, which as duplicates removed and contains only "TO" and "FROM" subject headers. I decided to download this version of the data because it was the cleanest; I did not want to spend extra time cleaning the data, removing duplicates, or removing TO and FROM tags. I completed all of the analysis on one file in the dataset: file 59846.txt from the "Sci.Space" library. This was a challenging dataset because it contained many numbers, equations, and non-UTF8 characters (Roman numerals, Greek alphabet, etc). To begin, for all datasets, I replaced numbers with "0" (I did not care about the numbers themselves), converted everything to lowercase, dropped blank words (''), and removed punctuation. I primarily cleaned with the String package in Python. All of these cleaning steps made it much easier to process and read the rest of the data.

The NLTK package had great functionality for everything I wanted to do. It was simple and quick to tokenize with the word and sentence tokenizer functions – the hardest part for me was keeping track of where breaks were in lists (for a while it was giving POS tags for every character, etc). Stemming and finding POS tags were also very straightforward; NLTK has great built-in functions for both that allow these functionalities in just a few lines of code.

Even with as easy as NLTK made it to munge the data, I prefer SpaCy for analysis. SpaCy is easy to use, it plays nicely with list comprehensions, it is very fast, and the syntax is intuitive. Another reason why I prefer SpaCy is that the POS tags it gave made much more sense; rather than gibberish letters like "CD" and "NN" for words, it gives human-readable tags like "VERB" and "ADV". However, I had an issue with my package installation, so it ended up being somewhat more difficult to get the tags out of the data than it was using the NLTK package.

The main difference between NLTK and SpaCy is the algorithm used; NLTK comes with a variety of algorithms, while SpaCy comes with only one state-of-the-art algorithm that is consistently updated as NLP technology improves. This makes SpaCy faster, but NLTK more versatile. The two languages also treat text differently; SpaCy uses a purely object-oriented approach (which matches Python well), while NLTK deals exclusively in strings. This is a style difference and does not significantly affect processing time; it is the user's choice which style they prefer.

I did not try to parallelize my work in either case, because I was working with such small amounts of data that it was not necessary. However, the multiprocessing package in Python is package-agnostic; both SpaCy and NLTK functionalize well, so it should be very simple to functionalize regardless of analysis package used.

Rachel Rosenberg
MSiA 490: Text Analytics
October 5, 2019

Write and test regular expressions:

Text used: We are very excited for the Christmas party next month on 12-12-19. As such, we need to prepare food and decorations for the event. Debbie will be heading up food preparation; if you would like to be a part of her team, please contact debbie101@gmail.com by December 4 2019. Judy will be leading decorations for the event; if you would like to work with her, please contact judybee43@aol.net by 11/29/19. Alan will take the lead on the White Elephant portion of the event; if you would like to help Alan out, please reach out to alanjones@jones.jo by December 3, 2019. If you don't have time but would like to donate money to the event, please contact the studio at admin@namaskaryoga.com after November 30, 2019 and before 12-11-19 so that we have time to appropriately allocate funds between Debbie, Judy, and Alan. Thank you!

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

Code:

```
emails = re.compile(r"\b\w*@\w*.\w*\b")
print(emails.findall(text))
```

Answer:

```
['debbie101@gmail.com', 'judybee43@aol.net', 'alanjones@jones.jo', 'admin@
namaskaryoga.com']
```

Rachel Rosenberg
MSiA 490: Text Analytics
October 5, 2019

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

Code:

```
dates = re.compile(r'''[0-9]{2}/[0-9]{2}/[0-9]{2}|
                |[0-9]{2}-[0-9]{2}-[0-9]{2}|
                |January\s[0-9]{1,2},{0,1}\s[0-9]{0,4}|
                |February\s[0-9]{1,2},{0,1}\s[0-9]{0,4}|
                |March\s[0-9]{1,2},{0,1}\s[0-9]{0,4}|
                |April\s[0-9]{1,2},{0,1}\s[0-9]{0,4}|
                |May\s[0-9]{1,2},{0,1}\s[0-9]{0,4}|
                |June\s[0-9]{1,2},{0,1}\s[0-9]{0,4}|
                |July\s[0-9]{1,2},{0,1}\s[0-9]{0,4}|
                |August\s[0-9]{1,2},{0,1}\s[0-9]{0,4}|
                |September\s[0-9]{1,2},{0,1}\s[0-9]{0,4}|
                |October\s[0-9]{1,2},{0,1}\s[0-9]{0,4}|
                |November\s[0-9]{1,2},{0,1}\s[0-9]{0,4}|
                |December\s[0-9]{1,2},{0,1}\s[0-9]{0,4}''', flags = re.DOTALL)

print(dates.findall(text))
```

Answer:

```
['12-12-19', 'December 4 2019', '11/29/19', 'December 3, 2019', 'November
30, 2019', '12-11-19']
```

Rachel Rosenberg
MSiA 490: Text Analytics
October 5, 2019

Main ideas of Charniak paper:

Explain in 2 to 3 paragraphs the main ideas of the required reading paper (Charniak E. A maximum-entropy-inspired parser (Links to an external site.).). The explanation needs to target a general audience, and be as simple as possible, use your own words. Assume your audience is somewhat technical, but knows nothing about parsers or machine learning. Your work will be judged on how well 1) you understood the material and 2) more importantly, how well are you able to communicate the ideas.