Published in Towards Data Science

Raymond Cheng Follow

Jun 29, 2020 · 2 min read ★ · Listen

Text Preprocessing With NLTK

Common Preprocessing Methods for NLP

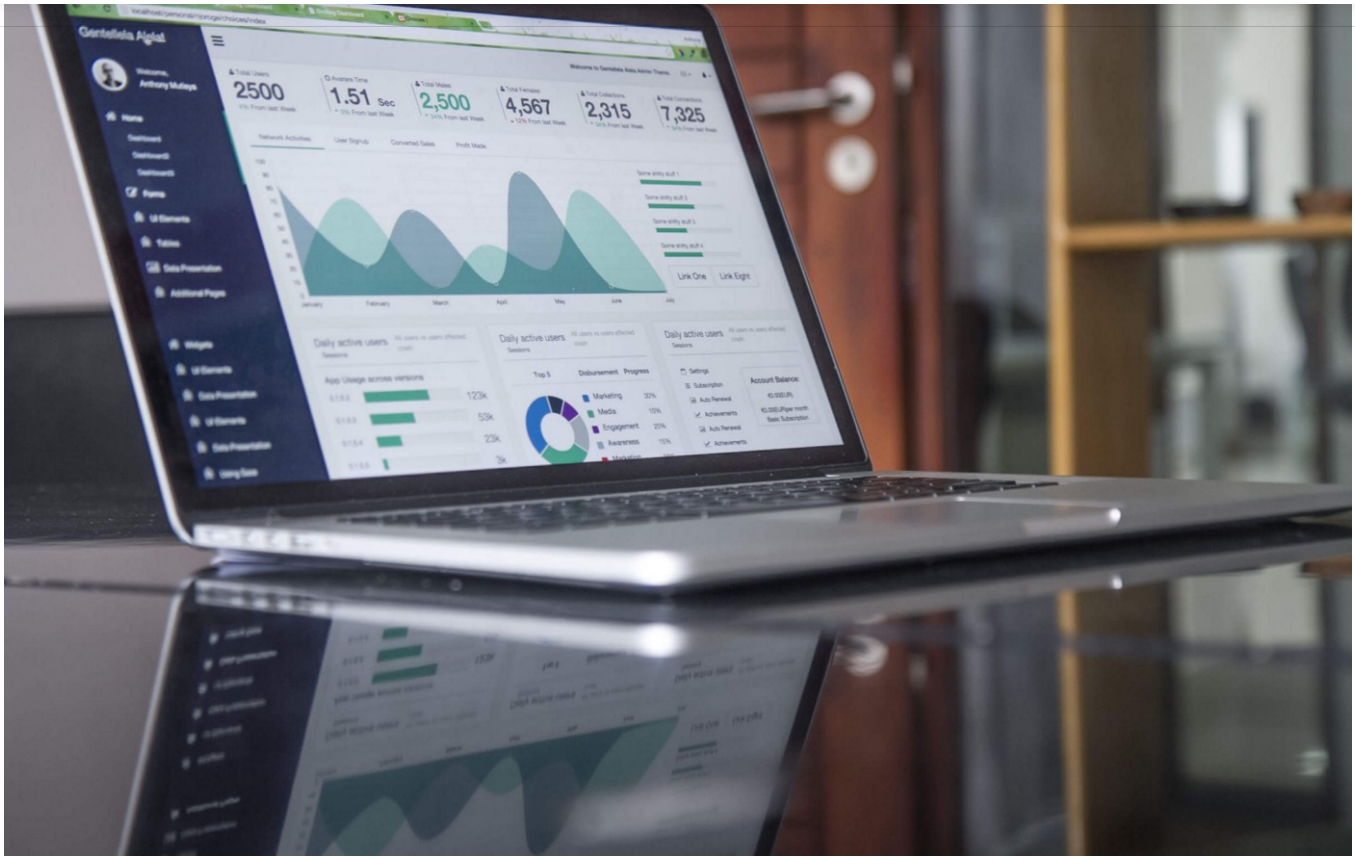

Photo by Carlos Muza on Unsplash

Intro

Almost every **Natural Language Processing (NLP)** task requires text to be preprocessed before training a model. Deep learning models cannot use raw text directly, so it is up to us researchers to clean the text ourselves. Depending on the nature of the task, the preprocessing methods can be different. This tutorial will teach the most common preprocessing approach that can fit in with various NLP tasks using **NLTK (Natural Language Toolkit)**.

Why NLTK?

- **Popularity**: NLTK is one of the leading platforms for dealing with language data.
- **Simplicity**: Provides easy-to-use APIs for a wide variety of text preprocessing methods
- **Community**: It has a large and active community that supports the library and

Now you know the benefits of NLTK, let's get started!

Tutorial Overview

1. Lowercase
2. Removing Punctuation
3. Tokenization
4. Stopword Filtering
5. Stemming
6. Part-of-Speech Tagger

All code displayed in this tutorial can be accessed in my Github repo.

Import NLTK

Before preprocessing, we need to first download the NLTK library.

```
pip install nltk
```

Then, we can import the library in our Python notebook and download its contents.


Lowercase

As an example, we grab the first sentence from the book *Pride and Prejudice* as the text. We convert the sentence to lowercase via `text.lower()` .


Removing Punctuation

To remove punctuation, we save only the characters that are not punctuation, which can be checked by using `string.punctuation` .


Tokenization

Strings can be tokenized into tokens via `nltk.word_tokenize` .

tokenization.ipynb hosted with ❤️ by **GitHub** **view raw**

Stopword Filtering

We can use `nltk.corpus.stopwords.words('english')` to fetch a list of stopwords in the English dictionary. Then, we remove the tokens that are stopwords.


stopword.ipynb hosted with ❤️ by GitHub view raw

Stemming

We stem the tokens using `nltk.stem.porter.PorterStemmer` to get the stemmed tokens.

stemming.ipynb hosted with ❤️ by **GitHub** **view raw**

POS Tagger

Lastly, we can use `nltk.pos_tag` to retrieve the part of speech of each token in a list.

The full notebook can be seen here.

Combining all Together

We can combine all the preprocessing methods above and create a `preprocess` function that takes in a *.txt* file and handles all the preprocessing. We print out the tokens, filtered words (af stopword filtering), stemmed words, and POS, one of which is usually passed on to the model c for further processing. We use the *Pride and Prejudice* book (accessible here) and preprocess it.

This notebook can be accessed here.

Conclusion

Text preprocessing is an important first step for any NLP application. In this tutorial, we discuss several popular preprocessing approaches using NLTK: lowercase, removing punctuation, tokenization, stopword filtering, stemming, and part-of-speech tagger.