Natural Language Processing using NLTK

Rachel Buttry, Keziah Sheldon, Eesha Das Gupta

1 Introduction

Natural Language Processing, or NLP, is a mechanism for computers to analyze human spoken and written languages using machine learning algorithms. NLP finds it's uses in sentiment analysis, spam detection, artificial intelligence, text generators, etc.

Natural Language Toolkit, or NLTK, is a Python module for Natural Language Processing that can extract features from text via methods like tokenization, stemming, lemmatization and classify them using numerous machine learning algorithms. Additionally, it provides an interface to over 50 text corpora and lexical resources for text analysis and also includes wrappers for other Python modules such as Scikit Learn.

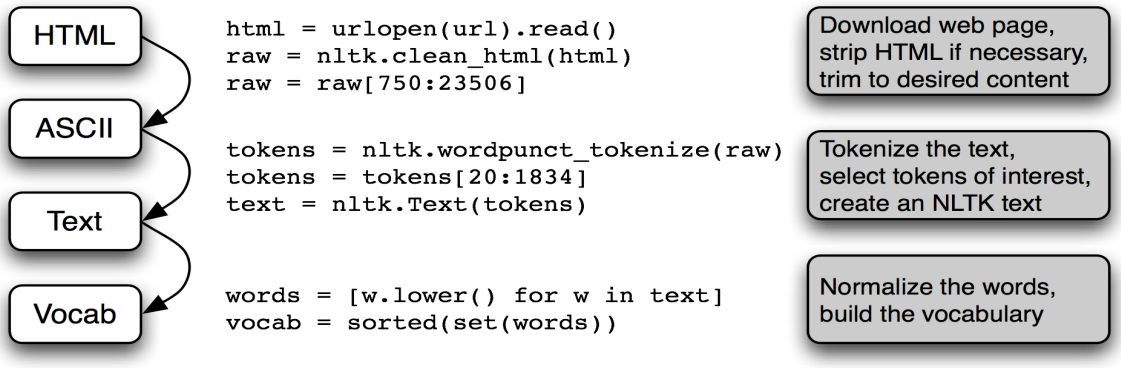

Figure 1: The NLP Pipeline for NLTK

2 Tokenizing

Tokenizing is the first step of NLP. Tokenization takes the input text string and splits the string into substrings as groups of sentences and words. Separating sentences may seem simpler than it actually is–it can't just depend on splitting according a period or punctuation mark since especially in recent decades that is often used ambiguously (For example, emoticons, abbreviations etc).

In addition to tokenizing by sentence, it is also possible to tokenize by word. Depending on what you are trying to achieve, you can tokenize by word or sentence. NTLK contains many different importable tokenizers for your use, one of the foremost being the Punkt word and sentence tokenizer, that was pre-trained on a large corpus from many European languages, such as English, German, Finnish, Estonian etc (a total of 17 languages)[1].

The Punkt tokenizer utilized an unsupervised multilingual sentence boundary detection system that did not depend on any additional resources besides the corpus it was supposed to segment into sentence.

It is also possible to have a supervised machine learning system to train the tokenizer. One of the other tokenizer algorithms, such as the one suggested by Riley (1998) had a supervised decision tree for sentence boundary detection, and exhibited a very low error rate of 0.2%, which was 0.82% better than Punkt. However, the caveat was that his tokenizer relied on having a huge test training data set, which is not readily available in other languages, as well has requiring a lexical intuition about the dataset you use it on which is realistically not always possible[3].

3 Part of Speech Tagging

Part-of-speech tagging (POS tagging or POST), also called grammatical tagging or word-category disambiguation, is the process of marking up a word in a text as corresponding to a particular part of speech as well as context. A simplified form of this is in the identification of words as nouns, verbs, adjectives, adverbs, etc. Tags give each word the usage of its part of speech and gives it a identifier, for example: 'JJ: adjective or numeral, ordinal' or 'IN: preposition or conjunction, subordinating'.

The default tagger for NLTK is a 'maxent treebank pos', which is a maximum entropy model to determine the parts-of-speech within the text.

Additionally, it is possible to train your own tagger using the treebank data from the `nltk.corpus` module.

4 Stemming and Lemmatization

Stemming is the the process of reducing a word to its stem through basic suffix stripping. For instance, the stem of the word "lively" is "live", so running a the statement "She acted lively while helping me today." through a stemmer (after tokenizing), would return the stems 'act', 'live', and 'help' with their respective words, while the rest of the words would just return the initial input (the words were already at their stem).

A commonly known algorithm for stemming is the Porter Algorithm which runs a given word through a 5 phase process. In each of the phases, a multitude of different rules are applied to the word. For instance, the stemmer will look at the end of the word and see if it matches any of the pre-set conditions. For example, if a word ends in IES, the stemmer replaces it with I.

A problem that arises is that, for many cases, the stemmers will not return actual words. For instance, the stem of the words "excitement" and "excited" are both "excit". Stemming is sufficient in many cases of interpreting the meaning of texts, but a better option in terms of meaning, is lemmatization.

Lemmatization, on the other hand, is the reducing of a word to a common base word, known as the lemma. For instance, the word "excitement" reduces

to the lemma “excite”. When lemmatizing, we also want to take into account the part of speech of the word as different rules apply to different parts of speech. However, the key goal remains the same for both stemming and lemming. We want to reduce different forms of a word to a common root such that we can apply meaning to the word.

5 Text Classification

Text classification is an application of NLTK classifiers to categorize text. NLTK's `classify` package has built in modules for text classification. Some modules within `nltk.classify` are :

- Naive Bayes Classifier
- Maximum Entropy Model
- Decision Tree
- Scikit Learn Wrapper

and others.

This paper discusses classification using Naive Bayes Classification and Maximum Entropy Model.

5.1 Naive Bayes Classification

Conditional probability according to Bayes' Theorem is

$$P(C|x) = \frac{P(x|C)P(C)}{P(x)}$$

where, C is a category and $x = x_1, x_2, ..., x_i$

Then, Naive Bayes makes the assumption that each element x_i is independent of other elements of x for a given category C. That implies,

$$P(x_i|x_1, x_2, .., x_(i-1), x(i+1), ..., C) = P(x_i|C)$$

This assumption, when applied to Bayes' Theorem, gives us

$$P(C|x) = \frac{P(C)\prod_{i=1}^{n} P(x_i|C)}{P(x_1, x_2, ..., x_n)}$$

The above can be used as a probability distribution model to classify text.

5.2 Maximum Entropy Model

For a probability space consisting of P(C) and P(D), where D is a huge in comparison to C, Maximum Entropy Model looks at P(D) that match the most with empirical data. It then calculates entropy in the distribution, and picks out the model with the highest entropy, on the assumption that maximum entropy implies maximum uniformity. Maximum entropy model is also used for logistic regression.

5.3 Applications

Text classification can be used for many practical purposes. It's applications range from security to industry and academia. It can also be used to incorporate Natural Language Processing into Artificial Intelligence. Text Classification methods use tokenizing, stemming and lemmatization to extract features from the text and classify the piece of text using the above algorithms.

A major field of study that exploits text classification is sentiment analysis. Sentiment analysis often uses Naive Bayes Classifiers and lexical corpora to classify words in a social media post, for instance, into positive and negative and using it to judge moods and sentiments of the author. In addition, spam detectors use Naive Bayes Classifiers and Maximum Entropy Model as well.

6 Conclusion

Natural Language Toolkit is a nifty way of utilizing Natural Language Processing with Python. It offers tools for both data acquisition and classification and comes with corpora which is available in multiple human languages. Its classifier module has methods based on different algorithms that can also be used other machine learning methods. These features make NLTK capable of practical uses in sentiment analysis, artificial intelligence, spam detection, and more.

7 Citations

1 http://textminingonline.com/dive-into-nltk-part-i-getting-started-with-nltk

2 http://www.nltk.org/index.html

3 Kiss, Tibor, and Jan Strunk. "Unsupervised Multilingual Sentence Boundary Detection — Computational Linguistics — MIT Press Journals." Unsupervised Multilingual Sentence Boundary Detection — Computational Linguistics — MIT Press Journals. MIT Press Journals, Dec. 2006. Web. 30 Nov. 2016.

4 https://github.com/nltk/nltk/wiki/FAQ

5 http://snowball.tartarus.org/algorithms/porter/stemmer.html