Non-English Text Mining using NLTK and comparison with spaCy

Xueyang Li(xl112@illinois.edu)

University of Illinois at Urbana-Champaign

Introduction:

The Natural Language Toolkit (NLTK) is a popular module suite that allows users to work with human language data to implement natural language processing. However, the majority of modules in NLTK target English and there are few documents which introduce how to use NLTK for non-English text mining. Therefore, I will present some modules for non-English languages in this article. Personally speaking, our group project's topic is "Sentiment analysis for Chatbot". I chose this tech review topic since I want to explore the possibility of enabling multi-language function for our Chatbot, such as English and Chinese.

The structure of this article is as follows. In section 1, I will briefly introduce basic information and applications of NLTK. Then I will discuss why it does not have many packages for non-English languages and present some of those non-English modules in section 2. I will emphasize the module for Chinese text mining by comparing it with other Chinese text mining modules in Section 3.

1: Introduction to NLTK

NLTK is a tool kit based on Python used for natural language processing. With this tool kit, users can implement classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum ("NLTK Documentation"). We can simply import it by executing the command "import nltk".

This toolkit includes many independent modules and those modules mainly belong to 6 general types: parsing (*parser, chunkparser, srparser, and so on*), tagging (*tagger*), Finite State Automata (*fsa*), type checking, visualization (*draw.tree, draw.fsa, and so on*), and text classification (*classifier*) (Loper and Bird, p3-p4). Among all those 6 types, Parsing, Visualization, and Classification are more popular compared with others. Parsing modules take text as input and build data structures. Visualization, not surprisingly, converts text data to visual graphs, such as histograms or pie charts. Classification modules are based on supervised machine learning methods and classify different sentences or documents with different labels.

2: Non-English text mining modules in NLTK

The majority of NLTK modules are designed for English. NLTK does not have many module targets at a specific non-English language since many other languages' structures are similar to English. For example, the sentence "I like science" will be "Ich mag science" in German. As we can see, both sentences' words are separated by space which means it's very easy to tokenize them. The word "I" in English is a Noun (NN) and is the subject of this sentence. The word "like" is a base-form verb (VB) and the word "science" is another Noun that acts as the object of this sentence. In German, those rules still apply. It just replaces "I" with "Ich" and replaces "like" with "mag". This phenomenon also happens to other English-like languages such as Dutch and French. Therefore, NLTK does not have many modules for non-English text mining since many of the popular languages are similar to English and can be processed with those English text mining modules. Also, some modules in NLTK, such as stemming and Lemmatization, stopwords, and tokenization, support not only English. For example, the "stopwords" module, supports 24 different languages. That means we can import

stop words from those different languages without using additional non-English modules. However, although we can tokenize and remove stop words for non-English languages very easily. Those languages do have some different details from English, such as different word tags. Therefore, some modules are developed for non-English text mining. I will introduce nltk_french and nltk.tokenize.stanford_segmenter here.

2.1 nltk_french

The module nltk_french is developed by Christopher M. Church from University of California, Berkeley, and posted on GitHub. Although it hasn't been accepted by the NLTK official website, it does include many functions for French text mining. Instead of using stopwords from NLTK French stopwords library, the author modified the library by adding Véronis stopwords to the existing library. Also, the most important part of the source code is that the author re-defined the word's POS tag based on Stanford's POS tagger. He added some tags which do not exist in English, such as 'PRO' ('strong pronoun') and 'CS' ('subordinating conjunction'). The modified tag list is as follow.

```
tag_abbreviations = {
                    'A': 'adjective',
                    'Adv': 'adverb',
                    'CC': 'coordinating conjunction',
                    'Cl': 'weak clitic pronoun',
                    'CS': 'subordinating conjunction',
                    'D': 'determiner',
                    'ET': 'foreign word',
                    'I': 'interjection',
                    'NC': 'common noun',
                    'NP': 'proper noun',
                    'P': 'preposition',
                    'PREF': 'prefix',
                    'PRO': 'strong pronoun',
                    'V': 'verb',
                    'PONCT': 'punctuation mark',
                    'N': 'noun'}
```

In general, this package integrates French modules in NLTK and improved them. It supports French tokenization, stemming, and tagging. After all those steps, the classification for French text will be the same as English text classification. To use this module, please see the links to the GitHub repo attached in the Reference part.

2.2 nltk.tokenize.stanford_segmenter

The module nltk.tokenize.stanford_segmenter is a module on NLTK official website and it targets Chinese and Arabic text mining. I mentioned in previous section that the many popular languages are similar to English since they are also separated by space and have similar sentence structures as English. However, when it comes to Chinese, things are different. The most difference between English text and Chinese text is the words in Chinese are not separated by spaces. The sentence "I like science" would be "我爱科学" in Chinese. As we can see, different characters are connected to each other without space between them. Thus, it's a big problem to tokenize Chinese. In other languages, we may tokenize the sentences first and then implement POS tagging. However, in Chinese, we may have to tag different parts of the sentences first and then tokenize them, since if we do not know what part each character in a sentence belongs to there would be no way for us to split them. Therefore, the author developed the tokenization function called "StanfordSegmenter()" based on STANFORD_SEGMENTER and STANFORD_MODELS. The example tokenization result is attached below:

If stanford-segmenter version is older than 2016-10-31, then path_to_slf4j should be provieded, for example:

```
seg = StanfordSegmenter(path_to_slf4j='/YOUR_PATH/slf4j-api.jar')
```

```
>>> from nltk.tokenize.stanford_segmenter import StanfordSegmenter
>>> seg = StanfordSegmenter()
>>> seg.default_config('zh')
>>> sent = u'这是斯坦福中文分词器测试'
>>> print(seg.segment(sent))
这 是 斯坦福 中文 分词器 测试

>>> seg.default_config('ar')
>>> sent = u'هذا هو تصنيف ستانفورد العربي للكلمات'
>>> print(seg.segment(sent.split()))
هذا هو تصنيف ستانفورد العربي ل الكلمات
```

Actually, this module only provides the function of Chinese text tokenization, but it would be sufficient to do Chinese text mining based on this module as Verbs in Chinese do not have different forms, so we do not have to perform stemming or lemmatization. The **only problem** with this module is that it will **not** display the words' tags automatically, even though it splits the sentences based on different POS tags. The solution to this issue will be discussed later in **Section 3**. To use this module in NLTK, just simply load the downloaded package into the variable "*seg*" (just like the example above) and execute "*from nltk.tokenize.stanford_segmenter import StanfordSegmenter*". The source code of this function is also attached to the Reference part.

3. Compare "nltk.tokenize.stanford_segmenter" with "spacy.lang.zh" from spaCy

"spaCy" is another text mining toolkit based on Python (and Cython). It supports tokenization for over 65 languages. As to our specific situation, text mining for Chinese, it has a module named "spacy.lang.zh" which supports tokenization and tagging in a simple one-line

code: "*print(token.text, token.pos_, token.dep_)*". This module automatically implements tokenization and tagging in the backend, and all we have to do is retrieve the results by typing "*token.pos*". It can also mark different words' components in this sentence by the variable "*token.dep_*". Below is an example of tokenization and tagging using this module.

```
: import spacy
  import spacy.lang.zh
  nlp = spacy.load("zh_core_web_sm")
  doc = nlp("我爱科学")
  print(doc.text)
  for token in doc:
      print(token.text, token.pos_, token.dep_)
```

```
我爱科学
我 PRON nsubj
爱 VERB ROOT
科学 NOUN dobj
```

As we can see from the picture, it's extremely simple and only needs one line of code.

On the other hand, the "nltk.tokenize.stanford_segmenter" in NLTK, as mentioned before, it can tokenize the sentences but will not automatically tag words like spaCy. However, word tagging can still be done by additional steps. First, we will have to download the Stanford POS tagger package. Then we can execute the following commands:

```
>>> from nltk.tag import StanfordPOSTagger
>>> st = StanfordPOSTagger('chinese-distsim.tagger')
>>> st.tag(word_tokenize(u'客户 表示 自己 有 3 辆 宝马 车 ， 自己 非常 认可 宝马 的 品牌 也 非常 认可 经销店 的 服务'))
```

In general, unlike spaCy, when implementing tokenization and tagging for Chinese with NLTK, we have to do those works in two separate steps, and two packages should be downloaded to complete corresponding steps. It's obviously more complex compared to spaCy.

Conclusion:

In this article, I introduced basic information of NLTK, which is a comprehensive text mining tool based on Python. There are two reasons of why it does not have many modules for

specific non-English language text mining: First, many popular languages are similar to English; second, some modules in it support different languages. As to the comparison of Chinese text mining tools between NLTK and spaCy, spaCy has better performance due to its simple, one-line-code implementation. Also, the modules/source code mentioned in this article are attached to the Reference part.

References

1. Church , Christopher, “nltk_french”, https://github.com/cmchurch/nltk_french/blob/master/french-nltk.py
2. “Chinese pos tag issue #1102”, *GitHub*, https://github.com/nltk/nltk/issues/1102
3. “Installing Third Party Software”, *GitHub*, https://github.com/nltk/nltk/wiki/Installing-Third-Party-Software#stanford-tagger-ner-tokenizer-and-parser
4. Loper, Edward and Bird, Steven, “NLTK: The Natural Language Toolkit”, *arXiv*, 17 May 2002, https://arxiv.org/pdf/cs/0205028.pdf
5. “NLTK Documentation”, *Natural Language Toolkit*, NLTK project, 19 Oct 2021, https://www.nltk.org/
6. “Source code for nltk.tokenize.stanford_segmenter”, *NLTK 3.2.4 documentation*, https://docs.huihoo.com/nltk/3.2/_modules/nltk/tokenize/stanford_segmenter.html