Overview of NLTK and Usage in Text Analytics

Introduction

Natural Language Toolkit (NLTK) is a string processing library for Python program to work with human language data to enable many real-life applications and implementations in text analysis. There are several NLP libraries in Python like spaCy, CoreNLP, PyNLPI, Polyglot. NLTK and spaCy are most widely used libraries. Spacy works well with large information and for advanced NLP.

These days there are tons of both, structured and unstructured data which is getting generated online through social media and other online platforms. The unstructured data amounts for total 95% of the data and remaining is structured data. NLP comes to the rescue to handle growing amount of unstructured data, as it helps in classifying text and remove the unnecessary stopwords. It accelerates the text analytics and helps in normalizing the data along with the desired output which is basically getting some idea on human behavior and their emotions. For example, sentiment analysis is a classic application of NLP. The emotions attached to a person's tweet or opinions, whether positive or negative, is very well expressed with the help of NLP with its inbuilt libraries.

NLTK was introduced with the following goals in mind

1. Simplicity: To provide a framework in order to get rid of tedious work-load of text classification and make this process more intuitive.
2. Consistency: To provide a framework with consistent interfaces and data structures.
3. Modularity: To provide components that can be used and understood easily without the need to go through the entire toolkit!

NLTK has been implemented for a lot of real-world applications which are instrumental for their growth. Text classification is commonly in use and helps in getting rid of redundant data and retain the useful stuff.

Body

We will focus on NLTK usage to perform text analysis. Some of the main steps involved for text processing using NLTK are discussed below:

1. **Tokenization**

The breaking down of text into smaller units is called tokens. Tokens are a small part of that text. If we have a sentence, the idea is to separate each word and build a vocabulary such that we can represent all words uniquely in a list. Numbers, words, etc. all fall under tokens. There are different tokenizer which can be used, e.g. word_tokenize, TweetTokenizer, regexp_tokenize.

```
text = "Hi! I Am Learning Natural Language Processing."
 print(nltk.tokenize.word_tokenize(text))
```

#output

```
[‘Hi’,
 ‘!’,
 ‘I’,
 ‘Am’,
 ‘Learning’,
 ‘Natural’,
 ‘Language’,
 ‘Processing’
]
```

2. Case Normalization

All words can be converted into lower case to avoid redundancy in the token list and to not let model confused by seeing same word with different cases like one in uppercase and one without to interpret differently. As python is a case sensitive language so it will treat NLP and nlp differently. One can easily convert the string to either lower or upper by using: str.lower() or str.upper(). For example, while checking for the punctuation one can convert character to either lower case or upper case.

```
import string

text = "Hi! I Am Learning Natural Language Processing."

text_normalize_and_clean = "".join([i.lower() for i in text if i
not in string.punctuation])
```

```
text_normalize_and_clean
```

#output

```
'hi i am learning natural language processing'
```

3. Removing Stop words

Stopwords include he, she, I, but, was were, and, being, have, etc., which do not add meaning to the data. So, these words should be removed which helps reduce the features from our data. These are removed after tokenizing the text.

```
stopwords = nltk.corpus.stopwords.words('english')

text = "Hi!! I'm Learning Natural Language Processing!!!."

text_new = "".join([i for i in text if i not in string.punctuation])

print(text_new)

words = nltk.tokenize.word_tokenize(text_new)

print(words)

words_new = [i for i in words if i not in stopwords]

print(words_new)
```

#output

```
['Hi',

 'Im',

 'Learning',

 'Natural',

 'Language',

 'Processing'

]
```

4. Stemming

In given text we may find many words like exciting, excited etc. which have a root word, excite, all of these convey the same meaning. So, we can simply extract the root word and remove the rest. Here the root word formed is called 'stem'. It is also not necessary that stem needs to exist and should have a meaning. Just by removing the suffix and prefix, we generate the stems.

NLTK provides us with PorterStemmer LancasterStemmer and SnowballStemmer packages.

```
from nltk.stem.porter import PorterStemmer

text = "Natural language processing is an exciting area."

# Reduce words to their stems

stemmed = [PorterStemmer().stem(w) for w in text]

print(stemmed)
```

#output

```
['natur', 'languag', 'process', 'excit', 'area']
```

5. **Lemmatization**

Lemmatization is to extract the base form of the word. The word extracted is called Lemma and it is available in the dictionary. Lemmatization is usually more sophisticated than stemming. Stemmer works on an individual word without knowledge of the context. For example, the word "better" has "good" as its lemma. This thing will miss by stemming because it requires a dictionary look-up. Lemmatization is more accurate as it uses more informed analysis to create groups of words with similar meanings based on the context, so it is complex and takes more time. NLTK provides WordNet Lemmatizer that makes use of the WordNet Database to lookup lemmas of words.

```
stopwords = nltk.corpus.stopwords.words('english')

text = "Hello! How are you!! I'm very excited that you're going
for a trip to India!! Yayy!"

text_new = "".join([i for i in text if i not in string.punctuation])

print(text_new)

words = nltk.tokenize.word_tokenize(text_new)

print(words)

words_new = [i for i in words if i not in stopwords]

print(words_new)

word_lemma = nltk.WordNetLemmatizer()

w = [word_lemma.lemmatize(word) for word in words_new]

print(w)
```

#output

```
[‘Hello’,
 ‘How’,
 ‘excited’,
 ‘youre’,
 ‘going’,
 ‘trip’,
 ‘India’,
 ‘Yayy’
]
```

6. Syntax Tree generation

We can define grammar and then use NLTK RegexpParser to extract all parts of speech from the sentence and draw functions to visualize it.

```
# Import required libraries
import nltk
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
from nltk import pos_tag, word_tokenize, RegexpParser
# Example text
sample_text = "The quick brown fox jumps over the lazy dog"
# Find all parts of speech in above sentence
tagged = pos_tag(word_tokenize(sample_text))
#Extract all parts of speech from any text
chunker = RegexpParser("""
                NP: {?*} #To extract Noun Phrases
                P: {}    #To extract Prepositions
                V: {}    #To extract Verbs
                PP: {}   #To extract Prepositional Phrase
                VP: { *} #To extract Verb Phrases
                “””)
# Print all parts of speech in above sentence
```

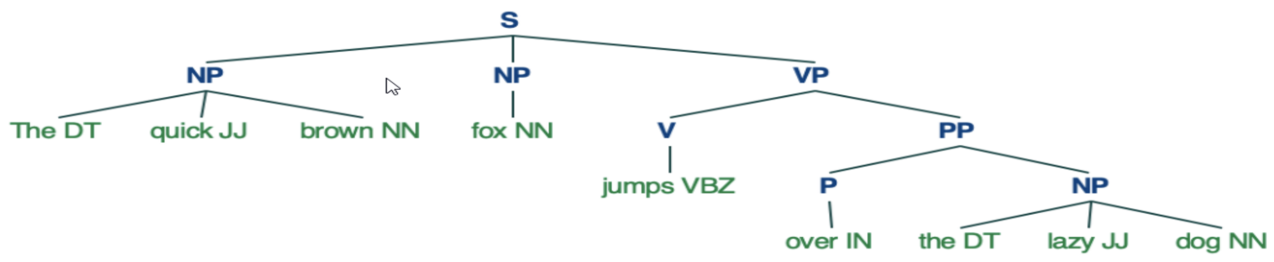
```
output = chunker.parse(tagged)

print("After Extracting", output)

output.draw()
```

#output

```
After Extracting
 (S
  (NP The/DT quick/JJ brown/NN)
  (NP fox/NN)
  (VP (V jumps/VBZ) (PP (P over/IN) (NP the/DT lazy/JJ dog/NN))))
```



7. POS Tagging

A part-of-speech tagger, or POS-tagger, processes a sequence of words, and attaches a part of speech tag to each word. It gives each word a particular tag and process them.

```
text = word_tokenize("And now for something completely different")

nltk.pos_tag(text)
```

#output

```
[('And', 'CC'), ('now', 'RB'), ('for', 'IN'), ('something',
'NN'),('completely', 'RB'), ('different', 'JJ')]
```

Conclusion

NLTK is very good tool for natural language processing and has many modules needed to process unstructured data. However, it has a long way to go as more and more unstructured data is increasing everyday and there are more complex datasets to process and analyze which need more modules and libraries to deal with the issues

References

1. https://www.nltk.org/
2. https://blog.eduonix.com/artificial-intelligence/natural-language-processing-using-natural-language-toolkit-nltk/
3. https://www.analyticsvidhya.com/blog/2021/07/nltk-a-beginners-hands-on-guide-to-natural-language-processing/

4. https://www.guru99.com/nltk-tutorial.html
5. https://www.analyticssteps.com/blogs/what-natural-language-toolkitnltk-nlp