Sentiment Analysis in Python using NLTK

Shravan I.V.
Cisco Systems Inc, Bangalore

Analysing Sentiments with NLTK

In this article, we explore ways to analyse sentiments from a given text and how some machine learning techniques can help in the process.

Sentiment analysis is used in opinion mining, business analytics and reputation monitoring. It helps businesses understand the customers' experience with a particular service or product by analysing their emotional tone from the product reviews they post, the online recommendations they make, their survey responses and other forms of social media text. Businesses can get feedback on how happy or dissatisfied the customer is, and use this insight to gain a competitive edge.

In this article, we explore how to conduct sentiment analysis on a piece of text using some machine learning techniques. Python happens to be one of the best programming language choices when it comes to machine learning and textual analytics as it is easy to learn, is open source, and is effective in catering to machine learning requirements like processing large data sets and performing mathematical computations. Natural Language ToolKit (NLTK) is one of the popular packages in Python that can aid in sentiment analysis.

About NLTK

NLTK is an open source natural language processing (NLP) platform available for Python. It is capable of textual tokenisation, parsing, classification, stemming, tagging, semantic reasoning and other computational linguistics. NLTK is a community driven project and is available for use on Linux, Mac OS X and Windows.

Let's first get started by installing NLTK to glue with Python using the following steps.

1. NLTK can be installed using Pip, a package management tool that Python users might be familiar with. Pip comes, by default, on Python version 2.7.9 and later. However, if you are using an older version of Python and don't have Pip already installed, use the following command to do so.

On Ubuntu:

```
sudo apt-get install python-pip
```

On Fedora Linux:

```
sudo yum install python-pip
```

2. With Pip, install NLTK using the following command:

```
sudo pip install -U nltk
```

This completes the NLTK download and installation, and you are all set to import and use it in your Python programs.

In this article, we will analyse sentiments from a piece of text using the NLTK sentiment analyser and the Naïve's Bayes Classifier. As a sample, I've taken some user reviews on restaurants in Bengaluru from *www.zomato.com* as shown below.

```
"Great place to be when you are in Bangalore."
"The place was being renovated when I visited so the seating
was limited."
"Loved the ambience, loved the food"
"The food is delicious but not over the top."
"Service - Little slow, probably because too many people."
"The place is not easy to locate"
"Mushroom fried rice was tasty"
```

Analysis using NLTK Vader SentimentAnalyser

NLTK comes with an inbuilt sentiment analyser module – *nltk.sentiment.vader*—that can analyse a piece of text and classify the sentences under positive, negative and neutral polarity of sentiments. A code snippet of how this could be done is shown below:

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

hotel_rev = ["Great place to be when you are in Bangalore.",
"The place was being renovated when I visited so the seating
was limited.”,
“Loved the ambience, loved the food”,
“The food is delicious but not over the top.”,
“Service - Little slow, probably because too many people.”,
“The place is not easy to locate”,
“Mushroom fried rice was tasty”]

sid = SentimentIntensityAnalyzer()
for sentence in hotel_rev:
     print(sentence)
     ss = sid.polarity_scores(sentence)
     for k in ss:
         print(‘{0}: {1}, ‘.format(k, ss[k]), end=’’)
     print()
```

When running the above Python script, the different sentiment proportions for individual sentences are obtained as shown below:

```
Great place to be when you are in Bangalore.
neg: 0.0, neu: 0.661, compound: 0.6249, pos: 0.339,

The place was being renovated when I visited so the seating
was limited.
neg: 0.147, neu: 0.853, compound: -0.2263, pos: 0.0,

Loved the ambience, loved the food
neg: 0.0, neu: 0.339, compound: 0.8316, pos: 0.661,

The food is delicious but not over the top.
neg: 0.168, neu: 0.623, compound: 0.1184, pos: 0.209,

Service - Little slow, probably because too many people.
neg: 0.0, neu: 1.0, compound: 0.0, pos: 0.0,

The place is not easy to locate
neg: 0.286, neu: 0.714, compound: -0.3412, pos: 0.0,

Mushroom fried rice was tasty
neg: 0.0, neu: 1.0, compound: 0.0, pos: 0.0,
```

The compound value here conveys the overall positive or negative user experience.

Analysis using Naïve's Bayes Classifier

Apart from Vader, one can create one's own classification model using Naïve's Bayes Classifier. In the machine learning context, Naïve's Bayes Classifier is a probabilistic classifier based on Bayes' theorem that constructs a classification model out of training data. This classifier learns to classify the reviews to positive or negative using the supervised learning mechanism. The learning process starts by feeding in sample data that aids the classifier to construct a model to classify these reviews.

```
import nltk
from nltk.tokenize import word_tokenize

# Step 1 – Training data
train = [("Great place to be when you are in Bangalore.",
"pos"),
  ("The place was being renovated when I visited so the
seating was limited.", "neg"),
  ("Loved the ambience, loved the food", "pos"),
  ("The food is delicious but not over the top.", "neg"),
  ("Service - Little slow, probably because too many
people.", "neg"),
  ("The place is not easy to locate", "neg"),
  ("Mushroom fried rice was spicy", "pos"),
]

# Step 2
dictionary = set(word.lower() for passage in train for word
in word_tokenize(passage[0]))

# Step 3
t = [({word: (word in word_tokenize(x[0])) for word in
dictionary}, x[1]) for x in train]

# Step 4 – the classifier is trained with sample data
classifier = nltk.NaiveBayesClassifier.train(t)

test_data = "Manchurian was hot and spicy"
test_data_features = {word.lower(): (word in word_
tokenize(test_data.lower())) for word in dictionary}

print (classifier.classify(test_data_features))
```

The output for the above code can be 'pos', denoting positive. The training data here is an array of sentences with corresponding class types – positive (pos) or negative (neg) to train the classifier. The dictionary formed in Step 2 consists of all the words obtained by tokenising or breaking this list of sentences. Step 3 starts constructing the data to be fed to the Naïve Bayes Classifier and Step 4 feeds the data to the classifier. With these steps, you might try out testing the classifier with different sentences. END

References

[1] *http://www.nltk.org/howto/sentiment.html*
[2] *http://www.nltk.org/api/nltk.sentiment.html*
[3] Hutto, C.J. and Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.

By: Shravan I.V.

The author currently works as a software developer at Cisco Systems India. He can be contacted at *iv.shravan@gmail.com*