NATURAL LANGUAGE PROCESSING

Sarthak Bhan

60004200005

1. Introduction

Natural Language Processing (NLP) is a tract of Artificial Intelligence and Linguistics, devoted to make computers understand the statements or words written in human languages. Natural language processing came into existence to ease the user's work and to satisfy the wish to communicate with the computer in natural language. Since all the users may not be well-versed in machine specific language, NLP caters those users who do not have enough time to learn new languages or get perfection in it.

A language can be defined as a set of rules or set of symbols. Symbol are combined and used for conveying information or broadcasting the information. Symbols are tyrannized by the Rules. Natural Language Processing basically can be classified into two parts i.e., Natural Language Understanding and Natural Language Generation which evolves the task to understand and generate the text (Figure 1).

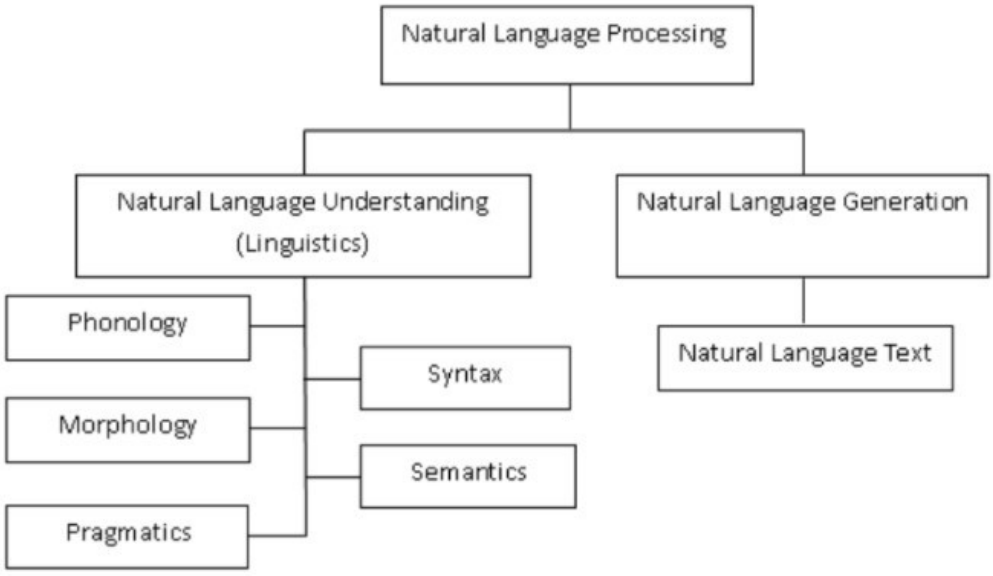

Figure 1. Broad Classification of NLP

We use NLTK Classifiers to search a predefined library for positive or negative words in the review statements and give rating accordingly.

2. Objectives

This program shows the probability of a review being positive or negative and rate the restaurant on the basis of predefined *movie_review* library, which contains positive and negative sample sentences, and compare the review provided by user by classifying them into 2 groups, one for validation and other one for training in the ratio of 80:20.

3. Operational Definitions

- Classifiers: Classification is the task of choosing the correct class label for a given input. In basic classification tasks, each input is considered in isolation from all other inputs, and the set of labels is defined in advance. A classifier is called supervised if it is built based on training corpora containing the correct label for each input.

- Naïve Bayes Classifier: Naive Bayes classification makes use of Bayes theorem to determine how probable it is that an item is a member of a category. If I have a document that contains the word "trust" or "virtue" or "knowledge," what's the probability that it falls in the category "ethics" rather than "epistemology?" Naive Bayes sorts items into categories based on whichever probability is highest.

  Bayes Theorem
  Bayes theorem tells us that the probability of a hypothesis given some evidence is equal to the probability of the hypothesis multiplied by the probability of the evidence given the hypothesis, then divided by the probability of the evidence.

  $$Pr(H|E) = Pr(H) * Pr(E|H) / Pr(E)$$

- Data Splits:

  a. Training data set: When you use the entire data set for training the model, what you have is just the training data set. You train the model using the entire data set and test the model performance on the random data set taken from the entire training sample data set.

  b. Validation data set: When you split the data set into two splits where the one split is called a training data set and another split is called a validation data set. You train the model using the training data set and evaluate the model performance using the validation data set. Generally, the training and validation data set is split into an 80:20 ratio. Thus, 20% of the data is set aside for validation purposes. The ratio changes based on the size of the data. In case, the data size is very large, one also goes for a 90:10 data split ratio where the validation data set represents 10% of the data.

4. Methodology

What is sentiment analysis?

Sentiment analysis the process of computationally identifying and categorizing opinions expressed in a piece of text, especially in order to determine whether the writer's attitude towards a particular topic, product, etc. is positive, negative, or neutral. In simpler words let's say that it is when you have a text of review as input and as the output you have to predict the class of sentiment either positive? or negative?

For example, A positive review contains something like this:

The hotel is really beautiful. Very nice and helpful service at the front desk.

And for a negative review:

We had problems with the Wi-fi. The food was also not so great.

For us, it is easy to read this and understand whether this is a positive or a negative review. But for computers, it is somewhat harder than that. I'm using the ***movie_reviews*** corpus in the ***nltk*** library for this process. A corpus is simply a large collection of texts. It is a body of written or spoken material upon which a linguistic analysis is based. I'm using the ***Naive Bayes classifier*** as the text classification algorithm.

Sentimental Detection:

This stage aims at detecting customers' sentiments of different restaurant aspect embedded in their reviews. First, each of the review is cut into sentences by SentiStrength, and each sentence then is assigned with a tuple of negative value and positive value, since in reality, one sentence may contain positive sentiments and simultaneously, negative ones. the SentiStrength also fixedly scores the dictionary tokens that include regular emoticons. For instances, "good" is scored {3, −1}, and {1, −4} for the "bad". Note that merely when a word presents within the dictionary, it is characterized by a single score. In addition, additional marks or attributive terms may lead to score change, such as the score of "goood" equals that of "good!!!", and they possibly extend the dictionary. Feature sentiments were calculated by applying SentiStrength as follows. Denote the collection of reviews by $R = \{r_1, r_2, ..., r_n\}$ and the collection of obtained aspects by $T = \{t_1, t_2, ..., t_m\}$. LDA outputs a matrix $W_{n\times m}$, of which the entry $w_{i,j}$ represents the number of times a feature from *i*th review associating to *j*th aspect.

Subsequently, the sentiment score attached to given aspect is the weighted average over the reviews. For every aspect t_j, we calculate the aspect sentiment score t_{sj} as noted i_x Equation (1):

$$ts_j = \frac{\sum_{i=1}^{n} w_{i,j} \times s_i}{\sum_{i=1}^{n} w_{i,j}}$$

where $S = \{s_1, s_2, ..., s_l\}$ denotes the sentiment score of each feature associated with aspect t_j .

Classifier:

NB classifier. NB was defined as a classifier on the basis of Bayes' rule. NB is a scheme based on statistics. Under its assumption, attributes are of equal independence and importance. For classifying an unknown cast, NB selects classes that are the most likely to contain evidence in test cast. NB is widely applied in classifying sentiments for the classification of a given review document d to class c as noted in Equation.

$$p(x_i|c) = \frac{count\ of\ x_i\ in\ document\ d\ of\ class\ c}{total\ number\ of\ words\ in\ document\ d\ of\ class\ c}$$

Based on Bayesian law, the likelihood that any given document being a member of class ci is implied by Equatio .

$$p(c_i|d) = \frac{p(d|c_i) \times p(c_i)}{p(d)}$$

In our context, the hypothesis of conditional independence that gives the particular class (yes or no) is adopted, and no independence exists between words. This is the reason why the model is called "naïve").

$$p(c_i|d) = \frac{\left(\Pi p\left(x_i|c_j\right)\right) \times p\left(c_j\right)}{p(d)}$$

Vader

VADER (Valence Aware Dictionary for Sentiment Reasoning) is a model used for text sentiment analysis that is sensitive to both polarity (positive/negative) and intensity (strength) of emotion. It is available in the NLTK package and can be applied directly to unlabelled text data.

VADER sentimental analysis relies on a dictionary that maps lexical features to emotion intensities known as sentiment scores. The sentiment score of a text can be obtained by summing up the intensity of each word in the text.

For example- Words like *'love', 'enjoy', 'happy', 'like'* all convey a positive sentiment. Also VADER is intelligent enough to understand the basic context of these words, such as *"did not love"* as a negative statement. It also understands the emphasis of capitalization and punctuation, such as *"ENJOY"*

Rating

To calculate rating, positive sentimental rating of a sentence given by the Vader is considered and is stored in a temporary variable *temp*, it then gets iterated through all the sentence to given sum of all the positive rating of all the reviews. This then is processed through a very basic formula to calculate rating. Formula:

$$Rating = \left(\frac{temp}{count}\right) \times 10$$

Where, temp is sum of all positive rating of all the reviews and count is number of reviews. This then is rounded of to 1 decimal place to match the rating scheme which is generally followed.

5. Code

Step 1: Create a python file and import the following packages.

```
import nltk.classify.util
from nltk.classify import NaiveBayesClassifier
from nltk.corpus import *
from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

Step 2: Defining function to extract features and Sentiment Intensity using Vader.

```
def features(word_list):
    return dict([(word, True) for word in word_list])
def sentiment_scores(sentence):
    sid_obj = SentimentIntensityAnalyzer()
    sentiment_dict = sid_obj.polarity_scores(sentence)
    return(sentiment_dict['pos'])
```

Step 3: Getting training data by using movie reviews from NLTK.

```
if __name__=='__main__':
    positive_fileids = movie_reviews.fileids('pos')
    negative_fileids = movie_reviews.fileids('neg')
```

Step 4: Separate the positive and negative reviews.

```
features_positive=[(extract_features(movie_reviews.words(fileids=[f])),'p') for f in positive_fileids]
features_negative=[(extract_features(movie_reviews.words(fileids=[f])),'n') for f in negative_fileids]
```

Step 5: Dividing the data into 2 datasets for the process, namely training and validating

```
threshold_factor = 0.8
threshold_positive = int(threshold_factor * len(features_positive))
threshold_negative = int(threshold_factor * len(features_negative))
```

Step 6: Extracting the features.

```
features_train = features_positive[:threshold_positive]+features_negative[:threshold_negative]
features_val = features_positive[threshold_positive:]+features_negative[threshold_negative:]
```

Step 7: Using Navie Bayes Classifier. Defining the object and training it.

```
classifier = NaiveBayesClassifier.train(features_train)
```

Step 8: Creating some random possible reviews.

```
input_reviews=[
        "The most amazing food ever! And also the staff is so nice to everyone. I highly recommend buying food from here. The best pizza ever",
        "My lunch libre was barely palatable because the meat was so salty that I could barely eat it inside the taco and not at all on its own",
        "Sleepy service, poor food quality, and when we asked why it was like this they stated that their kitchen was backed up, yet the restaurant was damn near empty",
        "I liked the food.",
        "I hate this Resturant so much. It has the worst staff ever!!",
        "It was nice resturant. I liked the food.",
        "The food was bad and the staff was not that friendly. Poor service",
        "Actually it's kinda good.",
        "Not so good restaurant"
]
```

Step 9: Running the classifier on input sentences and obtaining the possible sentiments along with the probability of that being the possible sentiment.

```
print("Sentiments: ")

count = temp = 0
sid = SentimentIntensityAnalyzer()
for review in input_reviews:
        count = count +1
        print("\nReviews:", review)
        probdist = classifier.prob_classify(extract_features(review.split()))
        pred_sentiment = probdist.max()
        print("Predicted sentiment: ", pred_sentiment)
        temp += sentiment_scores(review)
        print("Probability: ", round(probdist.prob(pred_sentiment), 4))
```

```
rating = round((temp/count)*10, 1)

print("\nOverall Restaurant Rating is: ", rating)
```

6. Output

```
Sentiments:

Reviews: The most amazing food ever! And also the staff is so nice to everyone. I highly recommend buying food from here. The best pizza ever
Sentiment:  p
Probability:  0.9856

Reviews: My lunch libre was barely palatable because the meat was so salty that I could barely eat it inside the taco and not at all on its own
Sentiment:  n
Probability:  0.7098

Reviews: Sleepy service, poor food quality, and when we asked why it was like this they stated that their kitchen was backed up, yet the restaurant was damn near empty
Sentiment:  n
Probability:  0.772

Reviews: I liked the food.
Sentiment:  p
Probability:  0.537

Reviews: I hate this Resturant so much. It has the worst staff ever!!
Sentiment:  n
Probability:  0.6632

Reviews: It was nice resturant. I liked the food.
Sentiment:  p
Probability:  0.5741

Reviews: The food was bad and the staff was not that friendly. Poor service
Sentiment:  n
Probability:  0.5019

Reviews: Actually it's kinda good.
Sentiment:  p
Probability:  0.5625

Reviews: Not so good resturant
Sentiment:  n
Probability:  0.5155

Overall Restaurant Rating is:  2.6
```

7. Conclusion

In this program, reviews are analysed and are awarded rating according to their sentiment. Only positive rating is taken for any given sentence and then rating is passed through a basic formula to calculate rating of the restaurant, which follows trend of 1 decimal places.

8. References

- https://www.nltk.org/
- https://www.nltk.org/api/nltk.classify.html#module-nltk.classify
- https://www.nltk.org/api/nltk.corpus.html#module-nltk.corpus
- https://en.wikipedia.org/wiki/Naive_Bayes_classifier
- https://www.nltk.org/_modules/nltk/sentiment/vader.html
- https://vitalflux.com/machine-learning-training-validation-test-data-set/
- https://towardsdatascience.com/naive-bayes-document-classification-in-python-e33ff50f937e
- https://towardsdatascience.com/sentimental-analysis-using-vader-a3415fef7664
- https://www.researchgate.net/publication/319164243_Natural_Language_Processing_State_of_The_Art_Current_Trends_and_Challenges
- https://towardsdatascience.com/sentimental-analysis-using-vader-a3415fef7664#:~:text=VADER%20(%20Valence%20Aware%20Dictionary%20for,intensity%20(strength)%20of%20emotion.&text=The%20sentiment%20score%20of%20a,each%20word%20in%20the%20text