Sentiment Analysis using Naïve Bayes

Sandipbhai Viradiya (19921091)

Western Sydney University, Australia

***Abstract*—Classification techniques are widely used in many fields nowadays for accurate classification. Here we are going to develop Naïve Bayes Classifier to classify the tweet in two classifications positive and negative.**

Core of Naïve Bayes Modelling

We've learned other techniques too for the classification like logistic regression but for this assignment, we are going to use Naïve Bayes Modelling.

Naïve Bayes modelling is basically using conditional probability to find probability of document to lie in particular class. In context of text document, formula would be,

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

Where P(c/d) is probability of class c given document d. P(c) is the prior probability. Here, we do not need to know probability of document d P(d) as we are trying to get maximization over c. Hence, we can ignore the denominator P(d).

Our goal is to identify if the tweet if positive or negative. So, if c belongs to C (positive or negative), we can create below formula:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d)$$

In short, we want most probable class for given document. If we replace P(c/d) by our conditional probability formula, we get below formula.

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d) = \underset{c \in C}{\operatorname{argmax}} P(d|c)P(c)$$

Here, we have P(d/c) which is likelihood means the probability that given a class c, document d belongs to it and P(c) is prior probability which means probability of having document in class c.

Naïve Bayes modelling has one assumption that for given glass c, individual features in document is independent of others.

If we consider each word of the document to be individual feature, we can write the formula as

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i \in positions} P(w_i|c)$$

Here $f_i = w_i$ means word in ith position.

As these numbers are small, we use log of the likelihood function,

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}}(\log P(c) + \sum_{i \in positions} \log P(w_i|c))$$

P(w_i/c) is probability of word w_i in the document of class c. This will be number of time w_i occurs in document of class c, divided by sum of the counts of each word in document of class c. So, formula would be

$$P(w_i|c) = \frac{count(w_i, c)}{\sum_{w \in V} count(w|c)}$$

And P(c) formula would be

$$P(c) = \frac{N_c}{N_{doc}}$$

Here, N_c is number of documents in class c and N_{doc} total number of documents.

Smoothing

Suppose that we are trying to classify the document where we have word that our classifier has not seen before. That means P(w_i/c) will become zero and log of it will make it infinite. To resolve this issue, we use technique called smoothing where we add constant to each count in P(w_i/c) formula. So now, the final formula would be

$$\hat{P}(w_i|c) = \frac{count(w_i, c) + 1}{\sum_{w \in V}(count(w, c) + 1)} = \frac{count(w_i, c) + 1}{(\sum_{w \in V} count(w, c)) + |V|}$$

Implementation

Here, we are going to implement NaiveBayesClassifier to classify the document from the above concept in python. First, we need to train our model. In data science, we normally divide the data into three parts, train, validate and test. Here, we are going to divide that data into train and validation set and test data will be given externally to test our classifier.

To split the data into two parts, we are going to use “sklearn” library. For reading CSV file as dataframe, we are going to use “pandas” library.

Before training, we are going to clean the data, which is called preprocessing.

Preprocessing

In preprocessing, we are going to
- Remove punctuations
- Remove #hashtags
- Remove user tagging or user reference (@USER)
- Lower the texts so that all word treated as same

Functions

processText ← To process text and clean document
countWordInClasses ← Count frequency of word class wise
train ← To train our model
predict ← Predict from provided data. If we pass labels (Known sentiment), it will return accuracy, recall and precision, else it will return sentiment with document.

Training

We need P(c) and P(w_i/c) to find likelihood for each class. We will loop through each tweet and based on class we will put word and count in multiarray where first key is class. In this way we can directly access word from particular class by VARIABLE[CLASS][WORD]. We will have individual function for frequency as we did in nGram.

Important classifier variables

logprior ← This will have prior probability class wise
loglikelihood ← This will be multiarray variable. First key will be class and, in every class, will have word and count.
all_classes← Classes in which we are gonna classify, Here it will be 0 and 1
all_voc ← This will be total words in D

Below is the algorithm for Naïve Byes:

```
$N_{doc}$ ← Number of documents in D
all_classes ← set(Training data)

for each c ∈ C    #We will loop through each class to train
    $N_c$ ← Number of documents in D for class c
    logprior[c] ← log $\frac{Nc}{Ndoc}$

    # all_voc will be used in smoothing
    all_voc ← Total words in D (Vocabulary) #Function will be there

    # All doc will be used in separate function to count V and class wise word count
    alldoc[c] ← append(d) where d ∈ D with class c

    # Calculate P($w_i$/c)
    for each word in all_voc
        count(w/c) ← Number of occurrences of w in alldoc[c]
        loglikelihood(w/c) ← log($\frac{count(w,c)+1}{sum(count(w',c))+V}$)
```

Classifying

So now using above algorithm, we will have probability of each word with respect of class that can be used in prediction of the class.

```
for each c ∈ C
        sum[c] ← logprioir[c] #Above function gives this

        for each word in testdoc
                if word ∈ V
                        sum[c] += loglikelihood[c][word]
        return argmax_c sum[c]
```

Whichever sentiment has max probability, it will be predicted sentiment.

We can predict class from testing dataset using above detail and compare probability of negative and positive tweet. We can compare this with the true result. We will have true positive (TP), false positive (FP), false negative (FN) and true negative (TN) value. We can find precision, accuracy and recall using below formula.

Precision = TP / (TP + FP)
Accuracy = (TP + FN) / (TP + FP + FN + TN)
Recall = TP / (TP + FN)