Project Report

Sentiment Analysis using Naive Bayes

Submitted to: Submitted by:

Prof. Mithun Balakrishnan Aditya Mahajan(AXM156630)

Harshil Shah (HXS155030)

**Problem description:**

The problem is to detect whether a given sentence indicates happiness or sadness.

**Proposed Solution:**

We trained a naïve bayes model using different attributes to indicate the sentiment of a sentence.

**Full Implementation Details:**

The formula for naïve bayes is

*P*(*y*∣*x*1,…,*x**n*)=*P*(*y*)∏n*i*=1 *P*(*x**i*∣*y*)/*P*(*x*1,…,*x**n*).

Here y is sentiment of the sentence and xi's are the attributes of the given sentence. So P(x1x2.....xn) is the probability of all the attributes for a given type of a sentence.

P(xi|y) is the probability of the word occurring given the sentiment was y and P(y) is the probability of the sentiment which is simply the ratio of the count of the sentences with a given sentiment and the total number of sentences.

Now for a given sentence we calculate the probability of the sentence being positive given the words and also calculate the probability of the sentence being negative given the words, whichever probability is higher we assign that tag to the sentence.

**If P(Positive| words)>P(Negative| words):**

**Sentence will be classified as positive**

**else:**

**Sentence will be classified as negative**

1. **Baseline Strategy:**

Steps followed:

a) Created attributes of each sentence by treating each word in a sentence as a separate

attribute.

b) Converted those attributes into lowercase form.

c) Trained a naïve bayes model on the resulting dataset.

Using this baseline strategy we got an accuracy of 65% on the testing database

2. **Improvement Strategy**:

Steps followed:

a) After removing the stopwords we used the six NLP features (lemmas, bigrams, POStags, Dependency Parser, HyperNym and SynNyms) in a sentence to form the features of the model

b) Converted all the features of the sentence into attributes of the model

c) Trained the naïve bayes model on the resulting dataset

Using this strategy the accuracy improved to 92% on the testing dataset

**Examples**:

**Baseline Strategy**:

Consider a sentence like: I am so happy.

The above sentence

After passing through step1 of the baseline strategy, the attributes created for the sentence will be:

1. I

2. am

3. so

4. happy

Now when we pass these attributes to the step2 of the baseline strategy,the attributes modified will be:

1. i

2. am

3. so

4. happy

So if the P(Positive| ( i,am, so, happy))>P(Negative| ( i,am,so,happy))

The sentence will be classified as positive

else:

will be classsified as negative

**Improved Strategy**:

In this strategy we convert the tokens in to bigrams, so the new attributes generated will be:

1. happy
2. (i, am)
3. (am,so)
4. (so, happy)
5. VBP
6. NN
7. RB
8. JJ
9. cop
10. advmod
11. nsubj
12. be
13. digit
14. some more hypernym …
15. iodine
16. then
17. some more synonyms

Using the improved model the accuracy increased to 92%.

**Programming tools (including third party software tools used):**

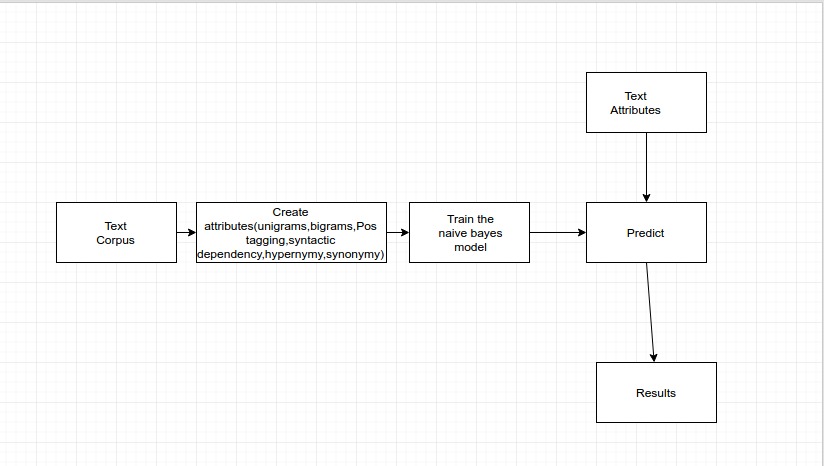
**Language used:** Python

External API and libraries used:

1. NLTK module

2. CMU's syntactic parsing api

Architectural Diagram:



Results:

1. By using baseline model, we get an accuracy of about 65%

2. By incorporating the improved model, the accuracy increased to 92%

Analysis:

The improved strategy was indeed better because the features extracted were better to train the model. If each of the six features were given as a feature itself we got the following accuracies

Lemma = 0.84

Bigrams = 0.61

PosTaging = 0.53

Syntactic Dependency Parsing = 0.73

HyperNym = 0.96

SynNyms = 0.88

From the above results we can say that hypernyms help to classify the sentence better compared to other features as PosTagging and bigrams.

Problems encountered:

Some of the features took too long to extract from the input sample like Syntactic Dependency Parsing, which took too long to train the model with this features. Also the feature of dependency parser was alone not that of an important feature compared to others. To get the results faster we trained just few samples from the whole data set.

Pending issues:

To get the better results more training samples are to be given but because the feature extraction is taking too long we are unable to take all training samples into consideration.

Potential improvements:

The classifier is naïve Bayes right now if we take some other more sophisticated classifier the results could have improved.

Other methods like clustering would help cluster the two classes (happy, sad).