COL341

Assignment 2- Naive Bayes

Prabhat Kanaujia
2016CS50789

Principles used -

The Naive Bayes classifier was modelled as follows:

1. Given a document **X = (**x_1, x_2, …. , x_n), the probability that it belongs to class '**K'** is given by:

$$P(C_K \mid X) = P(C_K) \cdot \prod_{i=1}^{n} P(x_i \mid C_K)$$

2. And $P(x_j \mid C_K) = (t_{jK}) / (\sum_{i=1}^{n} t_{iK})$

   , where t_{iK} = frequency of word **i** in all documents of class **K.**

PART A:

1. Unigrams were used as features. A regex ("**\w+**") was used to extract the features from the text, which were used to build a vocabulary as well as to train the classifier and predict classes.
2. Following equation was used to include laplace smoothing in the probability calculation:

$$P(x_j \mid C_K) = (t_{jK} + 1) / (\sum_{i=1}^{n} t_{iK} + 1 + \text{distinct words in vocab})$$

3. Log values of probabilities were used to avoid underflow.

Accuracy: 0.6033257093404216
Macro F1-score: 0.32664389581071535

PART B:

1. Stopwords were removed by comparing to a predefined set from the **NLTK** package .
2. Porter Stemmer(**NLTK** package) was used to stem the the unigrams to create new features.

Stop-words were first removed and then the other words were stemmed. It was not done the other way round because then some of the good words would have been stemmed to give stop words and would have been removed as well, leading to loss of information.

Accuracy: 0.6000494529269951
Macro F1-score: 0.3552347526902032

Observation: Stemming leads to a very minor increase in performance and the time taken is more than double of that without stemming.

PART C:

Following feature engineering techniques were employed:

1. **Bigrams:**
    a. A set of bigrams was created, along with the set of unigrams.
    b. Out of these bigrams, only those bigrams were selected whose frequency in the entire dataset was more than a certain threshold. This threshold is variable and was tuned to get maximum F-score.
    c. The set of bigrams was then appended to the unigrams to from the new vocabulary.

2. **TF-IDF:**
    a. It stands for term frequency-inverse document frequency. This technique is a good way to assign high weightage to words that are unique to a certain documents and in high frequency and low weightage to words which are common across the entire dataset.
    b. First, a data structure is generated which stores the frequency of each word in each document. Then, an array, equal to the size of the vocabulary is constructed, where each entry is given by:

    $$IDF_i = \log(\text{(total no. of documents)/(number of documents containing word i)})$$

    c. This row is then multiplied to every row of above matrix and summed over the respective classes.
    d. Then we proceed in the same way as before.

3. **Trigrams**
    a. A set of trigrams was created, along with the set of unigrams.
    b. Out of these trigrams, only those were selected whose frequency in the entire dataset was more than a certain threshold. This threshold is variable and was tuned to get maximum F-score.
    c. The set of trigrams was then appended to the existing vocabulary to from the new vocabulary.

The above features were combined to obtain the best accuracy, after removing stopwords and without word-stemming.
The **Best-Performing Model is:**

Use **TF-IDF,** combined with **unigrams, bigrams**(which occur more than **20** times in the training data) and **trigrams**(which occur more than **10** times in the training data). This model had the following performance parameters:

Accuracy: 0.5943623663225567
Macro F1-score: 0.4329116069683171
Time taken: 55 seconds

Stemming actually gives a better performance:

Accuracy: 0.5953514248624591
Macro F1-score: 0.4338259414846176
Time taken: 124 seconds

But it takes more than double the time, while increasing the F1-score by 0.0009. Hence, it is not an acceptable trade-off.