Assignment 2 Part 2: Algorithm Description and Results

Implementing Naive Bayes Classifier using Spark MapReduce

Naïve Bayes Algorithm description:

function TRAIN NAIVE BAYES(D, C) **returns** log $P(c)$ and log $P(w|c)$

for each class $c \in C$ # Calculate $P(c)$ terms
   N_{doc} = number of documents in D
   N_c = number of documents from D in class c
   $logprior[c] \leftarrow \log \frac{N_c}{N_{doc}}$
   $V \leftarrow$ vocabulary of D
   bigdoc[c] ← **append**(d) **for** d ∈ D **with** class c
   for each word w in V   # Calculate $P(w|c)$ terms
      count(w,c) ← # of occurrences of w in *bigdoc*[c]
      $loglikelihood[w,c] \leftarrow \log \frac{count(w,c) + 1}{\sum_{w' \text{ in } V} (count(w',c) + 1)}$
return *logprior*, *loglikelihood*, V

function TEST NAIVE BAYES(*testdoc*, *logprior*, *loglikelihood*, C, V) **returns** best c

for each class $c \in C$
   $sum[c] \leftarrow logprior[c]$
   for each position i in *testdoc*
      word ← *testdoc[i]*
      if *word* ∈ V
         $sum[c] \leftarrow sum[c] + loglikelihood[word,c]$
return $\text{argmax}_c \; sum[c]$

Figure 4.2 The naive Bayes algorithm, using add-1 smoothing. To use add-α smoothing instead, change the $+1$ to $+\alpha$ for loglikelihood counts in training.

Map Reduce Algorithm Description:

1. Log prior calculation:
   For each label, the calculate the prior probability by dividing the count of the label by the total count of all labels.
   The prior probability is added as a new column (prior) and the log prior is added as a new column (log_prior).

2. Log Likelihood calculation:
   - Count Smoothing: The count of each word is incremented using the incr function, which likely adds a smoothing value (like Laplace smoothing) to avoid zero probabilities. The result is stored in a new column count_smoothed.

- Total Word Count Smoothing: Similarly, the total word count for each sentiment is incremented using the incr_v function, and the result is stored in a new column total_word_count_smoothed.
- The likelihood of each word given its sentiment is calculated by dividing the smoothed word count (count_smoothed) by the smoothed total word count (total_word_count_smoothed). This is stored in a new column likelihood.
- The logarithm of the likelihood is computed to facilitate calculations in logarithmic space, improving numerical stability and efficiency. This is stored in a new column log_likelihood.

3. Model and result creation:
   - The test_df DataFrame is exploded so that each word in the words_stemmed column gets its own row. This helps in calculating the probabilities for each word separately.
   - The exploded DataFrame is joined with the priors DataFrame on the sentiment column to add the log prior probabilities.
   - It is then joined with the likelihood DataFrame on both the sentiment and word columns to add the log likelihood probabilities for each word.
   - The data is grouped by words_stemmed, sentiment, and log_prior.
   - The sum of the log_likelihood values for all words in each review is calculated and stored in a new column sum_log_likelihood.
   - The log-probability for each review is computed by adding the log_prior to the sum_log_likelihood. This value is stored in a new column log_probability.
   - intermediate columns log_prior and sum_log_likelihood is dropped for clarity.
   - The resulting DataFrame, which contains words_stemmed and their corresponding log_probability, is joined back with the original test_df (with sentiment renamed to label) to retain the original structure of the DataFrame.

Priors for each class:

	ABC sentiment	1.2 prior	1.2 log_prior
1	positive	0.5023891993022022	-0.6883801622634533
2	negative	0.4976108006977979	-0.6979370322103389

Evaluation: The accuracy is low as the implementation of Naïve Bayes classifier is very simple without any optimizations. For comparison, SKlearn's vanilla NB gives accuracy of 73% with TFIDF vectorization.

Accuracy	*0.66*
Precision	*0.59*
Recall	*0.68*
F1 measure:	*0.64*