Bag of Words Implementation for Sentiment Analysis

Apurba Das
40263612
contactapurbadas@gmail.com

October 5, 2024

Abstract. This report describes a Bag of Words (BoW) implementation for sentiment analysis [1, 2, 3] in Amazon product evaluations. It corresponds to a real life use-case where the marketing team wants to gain insights into customer sentiment regarding their latest products. Specifically, they were interested in understanding whether customers were generally satisfied or dissatisfied with their purchases. Our goal is to use the Naive Bayes classification algorithm to determine if reviews are positive or negative. This document describes the experiments, results, and major discoveries, as well as some additional modifications and an analysis of their impact on the model's predictions.

Contents

1 Introduction

This assignment focuses on using a Bag of Words (BoW) model to analyze sentiment on Amazon product reviews. The major purpose is to categorize customer feedback as good or negative. The report analyzes the model's performance and the effectiveness of the Naive Bayes classifier.

2 Experiments

The following section mentions details of what we did: motivation, methodology, results , analysis and limitations.

2.1 Experiment 1 - BoW and Naive Bayes classification

2.1.1 Motivation

The goal of employing a Bag of Words (BoW) model is to turn textual data into numerical features that may be used to categorize sentiment. This basic yet effective method aids in capturing word frequency to represent text. Next, we use Naive Bayes as it is a simple and effective probabilistic classifier for text classification tasks, suited for handling large datasets.

2.1.2 Methodology

We used data from Amazon product reviews. The text was pre-processed to construct a BoW representation. A Naive Bayes classifier was then used to predict positive or negative sentiment.

To elaborate on the methodolgy a little further - As a first step, we got all of our data ready. Our texts have the format such that the first word refers to the label, __label__1 corresponds to 1 and 2-star reviews(negative sentiment) and __label__2 corresponds to 4 and 5-star reviews (positive sentiment).

Normally reviews are very messy including words with incorrect spelling, different formats and a lot of words that do not add any value to our classification. So we created a method to clean them.

We also need to have our data organized and ready to access. To achieve this, we used the pandas library. We first created a dataframe containing all the data provided in the training document, and then created a sorted dictionary by frequency from most frequent to less frequent tokens. In order to see Zip's law first hand, we plotted the first 30 most frequent words of the vocabulary.

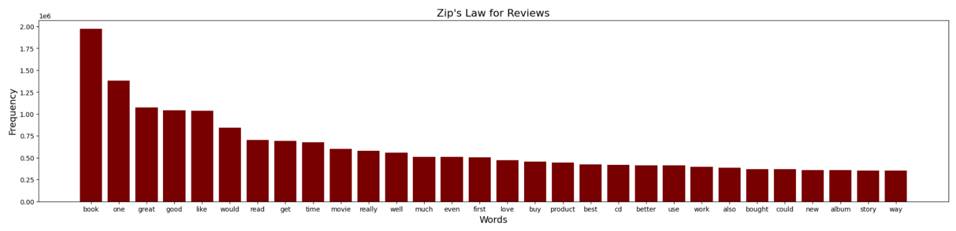
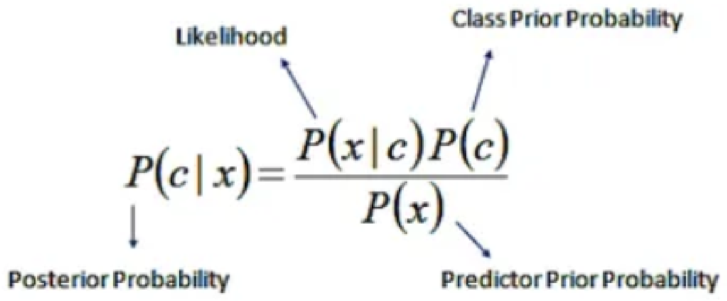


Figure 1: Zipś Law for Reviews


Likelihood
Class Prior Probability
$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$
Posterior Probability
Predictor Prior Probability

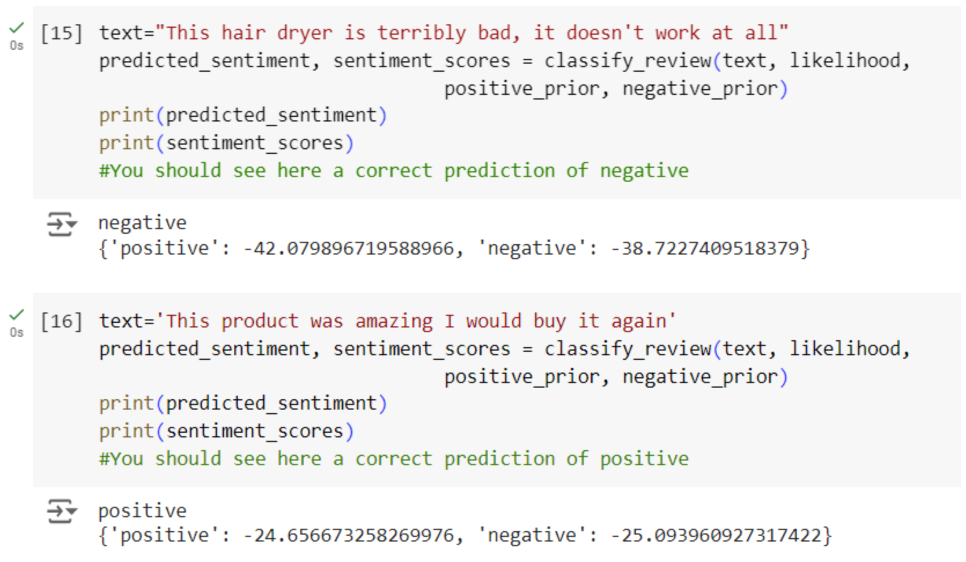
$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \cdots \times P(x_n|c) \times P(c)$$

Once our reviews are "clean", we start the real work. Naive Bayes includes three important components, as shown in the equation.

We need to calculate two things for the posterior probability: the likelihood and the class prior probability (since it's naive bayes we ignore the predictor prior probability). This means we calculate the probability that a review is positive or negative without taking into account the content, and then create a method to calculate the log likelihood of each word in our vocabulary. For this, we use a smoothing factor of 0.2. We tested out our model on a sample text for both positive and negative, and then ran the actual testing on the provided data.

2.1.3 Results

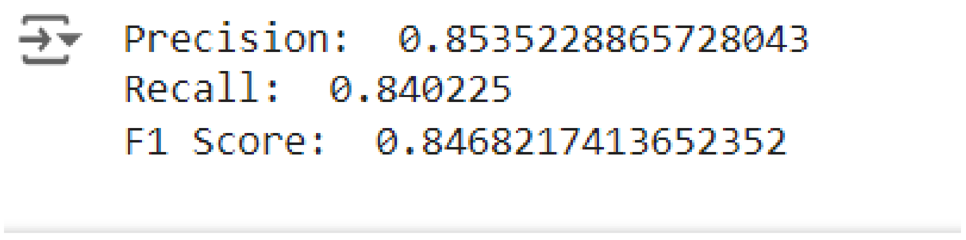
For our previous testing on the sample text, we were able to classify it in the correct category. Next, we built a review classification system and updated the confusion matrix values.



```
[15] text="This hair dryer is terribly bad, it doesn't work at all"
     predicted_sentiment, sentiment_scores = classify_review(text, likelihood,
                                              positive_prior, negative_prior)
     print(predicted_sentiment)
     print(sentiment_scores)
     #You should see here a correct prediction of negative

     negative
     {'positive': -42.079896719588966, 'negative': -38.7227409518379}

[16] text='This product was amazing I would buy it again'
     predicted_sentiment, sentiment_scores = classify_review(text, likelihood,
                                              positive_prior, negative_prior)
     print(predicted_sentiment)
     print(sentiment_scores)
     #You should see here a correct prediction of positive

     positive
     {'positive': -24.656673258269976, 'negative': -25.093960927317422}
```

Figure 2: Classification of the sample text

Then, we calculated precision, recall, and F1 score. For our use-case, we get the following result:



```
Precision:  0.8535228865728043
Recall:  0.840225
F1 Score:  0.8468217413652352
```

2.1.4 Analysis

Based on the obtained metrics, our model performs well in this context. Our model's high precision of 0.854 indicates its ability to predict favorable outcomes 85.4% of the time, reducing false positives, a crucial factor in minimizing costly false alarms. It also has a high recall (0.840) indicates it accurately identifies 84% of positive cases, but it may still miss some, potentially leading to false negatives. The F1 score, a harmonic mean of precision and recall, is a good metric. In our case, that is also high at 84% which indicates a good balance between recall and precision.

2.1.5 Limitations

As with any AI classification model, this model too has its limitations. The BoW model in general may not capture contextual or semantic information effectively. This can hinder our ability to perceive nuances in data, particularly in activities involving natural language processing. Also, metrics such as accuracy, recall, and F1 score do not provide information about how the model performs in real-world circumstances. So we can't confidently say how accurate our model will be with other data unless we test it out with much more unseen test data.

2.2 Experiment 2 - Modification of smoothing factor and no preprocessing of data

2.2.1 Motivation

We have already created the regular bag of words model and tested it out. Now, we want to experiment further to understand the impact of the parameters and process we chose to follow and apply in our previous model creation process. Earlier we chose a smoothing factor of 0.2, and did preprocessing for the text. This time, we try out:
1. To modify the smoothing factor (use three different values)
2. To use the raw texts without any pre-processing techniques

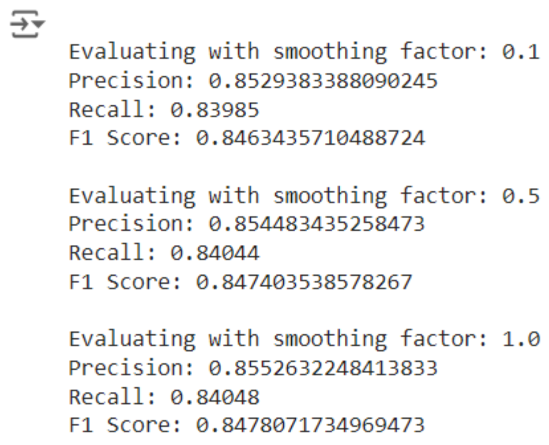
2.2.2 Methodology

For testing with 3 different values of smoothing factors, we use the same method as before i.e., we calculate the log likelihood and classify the review. But this time, instead of just 1 constant smoothing factor, we take 3 values, and loop over them to find its impact on our classification model. The values of smoothing factor we used were [0.1, 0.5, 1.0]. The result of this experiment is documented in the next section.

Apart from this, we also wanted to see the impact of using raw text without any preprocessing. For this, we kept the smoothing factor constant at 0.2 to compare with our previous model, and just split the text into tokens without any pre-processing. Post this, we created a function called to classify the raw text and printed the same metrics of precision, recall and f1 score for a fair comparison with the original model.

2.2.3 Results

Below are the results of our two experimentation. The first is with different smoothing values.



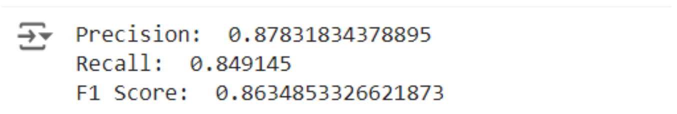
```
Evaluating with smoothing factor: 0.1
Precision: 0.8529383388090245
Recall: 0.83985
F1 Score: 0.8463435710488724

Evaluating with smoothing factor: 0.5
Precision: 0.854483435258473
Recall: 0.84044
F1 Score: 0.847403538578267

Evaluating with smoothing factor: 1.0
Precision: 0.8552632248413833
Recall: 0.84048
F1 Score: 0.8478071734969473
```

Figure 3: Results using different smoothing factors

Table 1: Comparison using Different Smoothing Factors

Smoothing Factor	Precision	Recall	F1 Score
0.1	**0.8529**	**0.8399**	**0.8463**
0.5	**0.8545**	**0.8404**	**0.8474**
1.0	**0.8553**	**0.8405**	**0.8478**

Next, we have the result using raw text.



```
Precision:  0.87831834378895
Recall:  0.849145
F1 Score:  0.8634853326621873
```

Figure 4: Results of the evaluation metrics without preprocessing of the text

2.2.4 Analysis

When the smoothing factor rises from 0.1 to 1.0, the model's precision gradually increases, indicating a marginal improvement in positive predictions.

All smoothing variables maintain recall consistency, indicating a balanced distribution of classes. As the smoothing factor increases, the F1 score likewise somewhat climbs. But, diminishing returns indicate that the model is already optimized at lower smoothing levels, and we don't have to risk increasing the smoothing value beyond the generally acceptable limits.

Meanwhile, in case of not applying any preprocessing, we find that even using raw text, the model performs comparatively at par with the earlier model that used preprocessing. In fact, it actually performs a little better than the earlier model. This could indicate that the preprocessing operations aren't substantially changing the feature space in a way that has a positive impact on the performance of the model. This can be because the model can already extract the relevant information from the unprocessed input. Another possibility is that the model is sufficiently stable to handle changes in data processing without appreciably affecting the overall result. This can imply that broadly speaking, the data's underlying patterns are robust.

Overall, we do not see any significant changes to our final prediction and evaluation metrics of precision, recall and F1 score due to our experimentation.

2.2.5 Limitations

There are certain restrictions when it comes to analyzing the outcomes in terms of the smoothing variables and performance measures. A comprehensive understanding cannot be obtained by depending only on Precision, Recall, and F1 Score. Additional insights can be obtained from other measures, particularly in datasets that are imbalanced. The relationship between the smoothing factor, preprocessing and model performance may not be straightforward, and the findings may be specific to the dataset and model used. Additionally, the model's performance may reflect biases in the training data. To improve the analysis, it is essential to incorporate a broader range of metrics, conduct experiments with different datasets, and employ robust validation techniques.

3 Conclusion and Future Work

This section details the final conclusion, findings and future work in this study of using a Bag of Words (BoW) model for sentiment analysis.

3.1 Summary of your work and your findings

The Bag of Words (BoW) model for sentiment analysis on Amazon product evaluations was implemented using the Naive Bayes classification technique. The major findings show that the model is effective at classifying feelings, with a high precision of 0.854 and a recall of roughly 0.840, resulting in a good F1 Score that nicely balances these measures. Experimenting with different smoothing values resulted in marginal improved precision which varied, although using raw text without preprocessing produced equivalent performance to processed data. These findings indicate that the model is robust enough to manage differences in input processing while capturing key patterns in consumer opinion.

3.2 Limitations

The analysis was mostly based on precision, recall, and F1 Score, which may not accurately reflect the model's performance in a variety of real-world circumstances. Also, our conclusions are dependent on the dataset and model utilized, limiting their applicability to different circumstances. The model's robustness can also be trained and tested using new datasets which are more diverse, to remove a lot of the limitations.

3.3 Future Work

Future research can explore alternative models like Support Vector Machines or deep learning for better context and semantic capture. Including a wider range of assessment metrics will provide a more comprehensive understanding of model performance. A lot of constraints were identified, including the reliance on a small set of performance criteria, potential biases in the training data, etc. which can be improved upon going forward. Hyperparameter tuning, cross-validation, handling bias, and temporal analysis can improve model accuracy and provide valuable insights for understanding customer feedback on Amazon products.

References

[1] Galli, K. Real-world python machine learning tutorial w/ scikit learn (sklearn basics, nlp, classifiers, etc)s. https://www.youtube.com/watch?v=M9Itm95JzL0, 2019.

[2] Scikit-learn. Machine learning in python. https://scikit-learn.org/stable/index.html.

[3] Shaik, S. Sentiment analysis with bag of words. https://medium.com/swlh/sentiment-analysis-with-bag-of-words-4b9786e967ca, 2020.