Natural Language Processing - Sentiment Analysis

Team: Mona Gandhi, Jeffrey Li, Henil Satra**. Project Mentor TA:** Yufei Wang

1) Abstract

For this project, we study the problem of sentiment analysis of product reviews, which is important for ecommerce sites in achieving higher customer satisfaction and sales. A positive review has a rating of 4 or 5 and a negative review has a rating of 1, 2 or 3. Text cleaning and feature extraction techniques were performed on the reviews data, namely removing special characters and numbers, removing stop words, and tokenizing the data. Other forms of text vectorizations were also examined, including Word2Vec, GLOVE and Bag-of-Words. We developed baseline models using simple machine learning classifiers, like Naive Bayes, Logistic Regression, and XGBoost, before moving to Long short-term memory (LSTM) models. Finally we also evaluated BERT embeddings with LSTM. We evaluated our models in binary classification tasks through different performance metrics, namely through computing the confusion matrix, the AUC score, and the F1 score. From our baseline results, we observed that Bag-of-Words feature extraction obtained the best performance on the original dataset and the dataset shift. We also observe that LSTM+BERT performs the best on the training data followed by LSTM+GLOVE and then LSTM+Word2Vec. The number of dimensions of features play a big role in extracting information. We see they do not perform very well in baselines as taking an average loss of the information extracted.

2) Introduction:

The Amazon Fine Food Reviews dataset from Kaggle contains approximately 50000 reviews from over a ten year period. Each entry has product and user information, ratings, and a text review. We primarily focus on ratings, an integer value ranging from 1 to 5, and the text review of the product or service. For our binary classification problem, we divided the reviews between positive and negative sentiment. A positive review has a rating of 4 or 5 and a negative review has a rating of 1, 2 or 3. Our eventual machine learning model will be able to take an input text review and output whether the review contains positive or negative sentiment.

Text cleaning and feature extraction techniques were performed on the data, namely removing special characters and numbers, removing stop words, and tokenizing the data. Other forms of text vectorizations were also examined, including Word2Vec, GLOVE and Bag-of-Words. Downsampling was performed to account for the large class imbalance between the two classes. We developed baseline models using simple machine learning classifiers, like Naive Bayes, Logistic Regression, and XGBoost, before moving to Long short-term memory (LSTM) models. We also tested on BERT embeddings with LSTM. The same tokenizers and models trained on Amazon Fine Food Reviews were used for our dataset shift with Stanford's Large Movie Review Dataset. **(http://ai.stanford.edu/~amaas/data/sentiment/).**

We evaluated our models in binary classification tasks through different performance metrics. Firstly, we obtained the accuracy score as an initial metric, knowing that accuracy score may not be as trustworthy given the nature of our data and problem. Secondly, we computed the confusion matrix to account for the predictions from the different classes, to obtain other measurements like precision and recall, and to build our ROC/AUC curve. Our final performance metric is the F1 score. However, for evaluating LSTMs with different embeddings, we use accuracy as our metric.

3) How We Have Addressed Feedback From the Proposal Evaluations:

One of the key feedbacks from our proposal evaluation was that we should focus on feature extraction. As a result, we incorporated feature extraction techniques like basic vectorization, Word2Vec, GLOVE and Bag-of-Words, and trained various baseline and more sophisticated machine learning models on them. Through this, we assessed the results from the different forms of vectorization on the model performances. We did not do transfer learning for BERT. Rather, we used pre-trained BERT features to train an LSTM. We compared BERT features with other feature extraction techniques. Due to shortage of time, as pointed out by our mentor, we did not consider prompting GPT-3. Another feedback was to experiment with upsampling. However, there was no significant difference in performance improvement between upsampling and downsampling, but there was an noticeable increase in computational runtime. Results for upsampling can be found in the baseline notebooks.

4) Background:

We are building from existing GitHub and Kaggle repositories, created by users who have previously built and trained models on the Amazon Fine Food Reviews dataset.

A. GitHub Repository: Amazon_Fine_Food_Reviews-sentiment_analysis. https://github.com/arunm8489/Amazon_Fine_Food_Reviews-sentiment_analysis.
The user's objective was to classify whether a review is positive (rating of 4 or 5) or negative (rating of 1 or 2). User uses Word2Vec and KeyedVectors for feature preprocessing and used several baseline machine learning models, namely Naive Bayes, Logistic Regression, and XGBoost, before building and implementing an LSTM model. Performance metrics used included confusion matrix and AUC/ROC curve.

B. Kaggle Notebook: Sentiment Analysis: Universal Sentence Encoder 91%. https://www.kaggle.com/code/kshitijmohan/sentiment-analysis-universal-sentence-encoder-91/notebook.
The user's objective was to classify whether a review is positive (rating of 4 or 5) or negative (rating of 1, 2 or 3). User used Universal Sentence Decoder for feature preprocessing and addresses class imbalance through downsampling. User built and implemented a sequential model with softmax activation function.

5) Summary of Our Contributions

1. Implementation contribution(s):

As baseline models, we will be comparing various feature extraction techniques for basic models like Naive Bayes, logistic regression and XGBoost. Then we move on to more complex neural networks for natural language tasks like LSTMs and transformer models like BERT. We will be using vectorization, Word2Vec, and Bag-of-Words as feature extraction techniques. Through this, we want to compare feature extraction techniques on model performance and also train and assess different models in the sentiment analysis problem.

2. Evaluation contributions:

We used our baseline models as benchmarks for our more complex models and to better understand our data. The two metrics we considered for this discussion are the AUC score and F1 score. We selected these

metrics as they can account for class imbalance and misclassifications better than accuracy scores. We repeated the training and assessment on the Large Movie Review Dataset as our dataset shift.

6) Detailed Description of Contributions

6.1 Implementation Contributions

Preprocessing:

- We visualize the number of samples for each rating, and find that there are a lot more samples with a rating of 5. Thus, to create a binary classification task of positive and negative sentiments, samples rated 1, 2 and 3 were grouped as negative samples and rations of 4 and 5 were grouped as positive samples. We visualize the number of samples in the two groups following the change and observe that a class imbalance persists, as there were more positive sentiments than negative sentiments..
- ***Text Cleaning***: We take the text for the review and perform some basic cleaning on it like making all the text in lowercase, get rid of all special characters and numbers, and remove the stop words using NLTK library.

Feature Extraction:

- ***Vectorization*:** Using Keras preprocessing tool, we create tokens for the review text after cleaning, thus indexing each token. These indexes are used to create features. We have implemented this technique for both the baseline machine learning models and neural-network-based models.
- ***Word2Vec:*** We used NLTK tokenizer to tokenize the data followed up with using Gensim to create Word2Vec features. We use pretrained `fasttext-wiki-news-subwords-300` word2vec vectors, which helps in handling subwords (and unknown words). They have a dimension of 300.
- ***Bag-of-Words:*** We count the frequency of occurrence of each word and represent it as a vector of word counts. The resulting vector represents the features and can be used as input to our models.
- ***Glove*:** We use this pre-trained word embedding model that represents words as dense vectors in a high-dimensional space. Glove is trained on a large corpus of text data(here wiki-gagword) and captures semantic and syntactic relationships between words. We have a 300 dimensional pretrained vector that we feed into our models later.
- ***BERT***: We use bert-base-uncased to get a tokenizer and pretrained model. For sentences in training data, we use the tokenizer and the model to get an embedding of 768 for each word.

Sampling Data:

For training and testing, we utilize an 80:20 training-testing split using Scikit-learn's *train_test_split* function. Given the class imbalance, we examined random undersampling and oversampling techniques to mitigate biases stemming from size differences between the minority class and majority class in the training data. The performance of oversampling and undersampling were similar despite the addition of more data instances (and subsequently higher runtime), so we remained with undersampling for the rest of the project.

For Word2Vec, GLOVE and BERT we use a subset of the dataset (10k samples) which we further split in an 80:20 for train and test, in such a way that we have a balanced training data and testing data as well.

ML Models

- ***Baselines*:** As baseline models, we consider logistic regression, Naive Bayes, and XGBoost classifiers. We trained these baseline models using features generated from each of the vectorization techniques. Further hyperparameter tuning was performed using GridSearch Cross-Validation to optimize hyperparameters and the results for the best performing models were reported.

- ***Neural Networks:*** We looked into training LSTMs with our vectorized and BOW data. We fine-tuned the number of layers for the LSTMs, the hidden size, and embedding size. However, we did not have good results as there is no sentence structure in these features. Thus we moved to Word2Vec, GLOVE and BERT embeddings. We create a different LSTM model to train and test data for each of these features. We also tune hyperparameters like learning rate, number of epochs, batch size and so on and save the best model. We test the best model on the test data.

Dataset Shift: We want our final model to be generalizable on unseen data. Thus, after training our model with the Amazon Fine Food Reviews dataset for sentiment analysis tasks, we want to assess its performance on a different dataset but with a similar problem regarding reviews and sentiment. We selected the Large Movie Review Dataset from Stanford as our dataset shift. The dataset contains 50000 movie reviews, equally divided between positive and negative movie reviews. The nature of this dataset made it ideal for binary sentiment classification tasks.

6.2 Evaluation Contribution

The goal for our project was to compare various feature extraction methods and train them on different machine learning models. We considered different forms of feature extraction including regular vectorization using Keras preprocessing tool, Word2Vec, Glove, and Bag-of-Words. We implemented baseline models Naive Bayes, Logistic Regression, and XGBoost and neural-network-based models, namely LSTM and BERT. We used our baseline models as benchmarks for our more complex models and to better understand our data. The two metrics we considered for this discussion are the AUC score and F1 score. We repeated the assessment on the Large Movie Review Dataset as our dataset shifted.

Our baseline results are shown in the supplementary. Of our baseline results, bag-of-words feature extraction was the most successful in obtaining the best AUC score and F-1 score for both the Amazon Fine Food Reviews dataset and dataset shift. Logistic regression with hyperparameter tuning yielded the highest scores, with 0.870 AUC score and 0.868 F-1 score for the original dataset and 0.756 AUC score and 0.756 F-1 score with the dataset shift. This can be shown in Figure 7 and Figure 8. Other feature extraction techniques did not yield as high metrics. One possible explanation for poor performance is the reduced size of the training and testing samples of models for Word2Vec and Glove feature extraction as we were constrained by time.

Our results for LSTM (that captures sentence semantics) are shown in the supplementary. We observe BERT performing the best among all the feature extraction techniques that we used to train LSTM. We also did some hyperparameter tuning for which we add the results, plots and graphs in the GitHub Repository. BERT being trained on transformers does capture more features than Word2Vec and GLOVE.

7) Compute/Other Resources Used:

We relied on Google Colaboratory as the main Python environment for creating and executing our models. We utilized Nvidia CUDA for training and testing our larger neural networks-based models.

8) Conclusions

Through working on this project, we were able to develop and implement useful and practical models for sentiment analysis tasks. Bag-of-Words feature extraction yielded the best baseline model performances for both the Amazon dataset and the Movies dataset. We also observed that BERT performs the best on the training data followed by LSTM GLOVE and LSTM Word2Vec. This project allowed us to achieve a greater

understanding of the differences between traditional and deep learning natural language processing (NLP) pipelines. Our results and our final models may be useful for companies who want to derive insights from reviews left by consumers. In addition, our work overall details the steps we took from simple baseline models to more complex models and may be useful for ambitious students (or Kagglers) who want to learn and explore NLP/ML applications in text analysis.

Through working on the project, the focus of our project gradually evolved. Rather than solely designing and implementing machine learning models to achieve optimal performances, we also considered different forms of feature extraction and vectorization. Due to shortage of time, as pointed out by our mentor, we were not able to prompt GPT-3. Other feedback from our mentor was also quite useful as it introduced us to new ideas on sampling due to our class imbalance as well as areas and model results in which we may need to prioritize.

Sentiment analysis of these reviews can help ecommerce sites achieve higher customer satisfaction and sales, while also saving time. Additionally, it can provide customers with ways of making informed purchasing decisions. However, we need to account for potential biases in predictions, especially since reviews may be artificial or fake. Since the data is public but also made by users, it should be collected and handled responsibly.

9) Roles of team members:

Mona Gandhi: Mona worked on final pre-processing, GLOVE, Word2Vec, Bag-of-Words, and feature extraction/preprocessing. She also worked on implementing the LSTM model.

Jeffrey Li: Jeffrey contributed to implementing and performing hyperparameter tuning on baseline models, and model assessment metrics for both sets of data.

Henil Satra: Henil worked on the initial pre-processing of the data and implemented the BERT model.

Github Repository Link:

https://github.com/henilsatra/Sentiment-Analysis-of-Amazon-Fine-Food-Reviews

---

Other Prior Work / References (apart from Sec 3) that are cited in the text:

1. G. S. N. Murthy, Shanmukha Rao Allu, Bhargavi Andhavarapu, Mounika Bagadi, Mounika Belusonti, “Text based Sentiment Analysis using LSTM”, IJERT, 2020

https://www.ijert.org/text-based-sentiment-analysis-using-lstm

2. B. Selvakumar, B. Lakshmanan, “Sentimental analysis on user’s reviews using BERT”, Materials Today: Proceedings, 2022

https://doi.org/10.1016/j.matpr.2022.03.678

Supplementary Materials:

The results from our experiments are shown in the figures below. Models were trained and tested on the respective datasets mentioned.

Fig 1. Model Performance: Fine Food Reviews Dataset (Vectorized)

	Naive Bayes	LogReg	XGBoost
AUC Score	0.540	0.534	0.717
F-1 Score	0.722	0.484	0.786

Fig 2. Model Performance: Large Movie Review Dataset (Vectorized)

	Naive Bayes	LogReg	XGBoost
AUC Score	0.512	0.503	0.557
F-1 Score	0.204	0.0826	0.413

Fig 3. Model Performance: Fine Food Reviews Dataset (Word2Vec - 1600 Test Samples)

	Naive Bayes	LogReg	XGBoost
AUC Score	0.6715	0.749	0.745
F-1 Score	0.673	0.753	0.743

Fig 4. Model Performance: Large Movie Review Dataset (Word2Vec - 1600 Test Samples)

	Naive Bayes	LogReg	XGBoost
AUC Score	0.5	0.495	0.491
F-1 Score	0.00332	0.610	0.0813

Fig 5. Model Performance: Fine Food Reviews Dataset (Glove - 1600 Test Samples)

	Naive Bayes	LogReg	XGBoost
AUC Score	0.6715	0.749	0.745
F-1 Score	0.673	0.753	0.743

Fig 4. Model Performance: Large Movie Review Dataset (Glove - 1600 Test Samples)

	Naive Bayes	LogReg	XGBoost
AUC Score	0.666	0.704	0.675
F-1 Score	0.585	0.701	0.662

Fig 7. Model Performance: Fine Food Reviews Dataset (Bags-of-Words)

	Naive Bayes	LogReg	XGBoost
AUC Score	0.791	0.870	0.834
F-1 Score	0.895	0.868	0.883

Fig 8. Model Performance: Large Movie Review Dataset (Bags-of-Words)

	Naive Bayes	LogReg	XGBoost
AUC Score	0.670	0.756	0.756
F-1 Score	0.580	0.756	0.731

Fig 9. Model Performance: Fine Food Reviews Dataset (LSTM)

	Word2Vec(300)	GLOVE(300)	BERT(768)
Train Accuracy	0.931	0.975	0.981
Test Accuracy	0.762	0.802	0.812