Natural Language Processing
Sentiment Analysis of Amazon Reviews

Team: Federico Cimini (CIS 5190), Liang-Yun Cheng (CIS 5190), Samuel Thudium (CIS 5190)
Project Mentor TA: David Yan
GitHub repository: AmazonFineFoodReviews

1) Abstract

Sentiment analysis is a popular introduction to natural language processing (NLP). This problem can be broken down into three simple steps: (1) clean input text, (2) learn embeddings for the words in present data, and (3) use these embeddings to classify the overall sentiment of new input text. We sought to build and improve upon the existing corpus of sentiment analysis notebooks that are prevalent on sites such as Kaggle. The sheer volume of these notebooks suggests that there are many ways to approach this problem, but typically there is no systematic comparison of the performance that these various methodologies have on a common dataset.

Thus, here we present a side-by-side performance of NLP and classification models on over 500,000 Amazon food reviews. Namely, we compare (1) NLP models with increasing fidelity to language structure, which are coupled with (2) classification models of increasing complexity. Overall, we were able to achieve quality results even using simpler models (TF-IDF + Logistic Regression), but saw an expected improvement using more complex models that take sequence information into account (word2vec + LSTM). This report aims to improve and expand upon the existing space of sentiment analysis projects.

2) Introduction

To approach this problem, we used the “Amazon Fine Food Reviews”[1] dataset from Kaggle to conduct model training and testing. This dataset consists of 568,454 Amazon reviews of food items; of interest to this study was the full review text, review summary, and star rating of each review. The raw dataset contains star-review ratings from 1 to 5; these are skewed dramatically toward 5-star reviews (**Fig. 1A**). Given our knowledge of review systems, we opted to pool the five-scale rating classes into two, which we deemed “positive” and “negative”. Pooling slightly improved the balance between positive and negative classes, but the positive class still was much more abundant than the negatives (**Fig. 1B**). The imbalanced classes were handled with upsampling of the minority class in the training phase.

In addition to the baseline input of cleaned review text, we wanted to test the robustness of our chosen models to datashift. In particular, given models trained on whole reviews, we sought to interrogate the NLP and classification frameworks ability to predict sentiment given inputs from shifted distributions such as review summaries and random word dropout from review text. These data shifts provide interesting challenges for sentiment analysis because they test the capacity of NLP models’ ability to capture features of user sentiment in a more restricted setting than the one on which the models were initially trained. An overview of summary statistics of the differences between the full review texts and review summaries is provided (**Fig. 1C**).

Based on our review of existing resources, we proceeded to implement our systematic analysis of sentiment analysis methodologies into several discrete steps. The contributions made to the implementation and evaluation of this topic are detailed in section 5.

2) How We Have Addressed Feedback From the Proposal Evaluations

We have gotten positive feedback on our previous milestones. The most substantial feedback that we got in milestone 2 was the suggestion that we incorporate more significant hyperparameter tuning using grid search. We have incorporated this into each phase of the project as We have incorporated hyperparameter tuning and streamlined our datashift processes after evaluations.

3) Background

For the baseline model, our work closely references the Kaggle notebook "*Predicting Sentiment and Helpfulness*" authored by Eugen Anghel [2]. Anghel conducted a simple text processing, followed by vectorizing the documents with 1-4 n-grams. Some improvements considered include: more robust text cleaning, using full review text instead of summary text for training, incorporation of random up-sampling of minority class, and use of Multinomial Naive Bayes classifier for prediction.

In the next phase, to explore a more complex NLP model, we used "*Amazon Fine Food Reviews: Sentiment Analysis*" [3] by Chirag Samal as the base for the "*Word2Vec + LSTM*" model. Unlike the author, we explored the possibility of training our own Word2Vec embedding, instead of using pre-trained embeddings. For training the embeddings, we referenced "*Gensim Word2Vec Tutorial*" [4] by Pierre Megret and "*word2vec and random forest classification*" [5] by Arunava.

Unlike most Kaggle notebooks where authors do not explore the effect of hyperparameters on model performance, we investigated the effect of tuning some hyper-parameters in logistic regression, Multinomial Naive Bayes classifier, LSTM, TF-IDF, and Word2vec. Moreover, instead of using one single metric to evaluate the performance of the models, we consider AUC, accuracy, and F1 of the test set. Lastly, we tested the robustness of the models by introducing data shifts in the test set, where summary and reviews with 50%, 75%, and 90% random dropouts are used to evaluate the performance of the models.

4) Summary of Our Contributions

4. 1 - Implementation contribution(s):

<u>Pre-Processing:</u> First, in order to reduce noise in the review text, the following text pre-processing steps were included: lemmatization, tokenization, and removal of stopwords, punctuation, hyperlinks, and special characters. This step is performed on both the full review text and summary text. The target column is simplified into +1 positive and -1 negative as described in the previous section. Before training the models, a random up-sampler was used to handle imbalance data, where minority class (target = -1) is up-sampled to match the majority class.

<u>Modeling</u>: Then, three text embedding techniques were explored, TF-IDF, Word2Vec, and pre-trained BERT. Lastly, logistic regression, Multinomial Naive Bayes classifier, and LSTM were used to compare the performance of various classification techniques.

4.2 - Evaluation contributions:

The performance of all models was compared, followed by the performance of these models after significant data shifts: word dropout and summary text instead of full review text. The aim of this comparison was to determine the robustness of the implemented models. We monitored the efficacy of these models using two primary metrics: classification accuracy and ROC AUC.

5) Detailed Description of Contributions

5.1 - Implementation contributions:

Text cleaning: Before we could pass text data into our models, the review text had to be cleaned. We opted to set a stricter threshold for a word to be deemed 'cleaned'. To this end, we removed the following elements from the review texts: hyperlinks, digits, apostrophes, punctuation, and stopwords. Additionally, we converted all words to lowercase and performed lemmatization to truncate or swap words into their base forms. Using the cleaned text data, we generated wordclouds to visualize the words used with greatest frequency and grouped on sentiment class (positive or negative) (**Fig. 2**). Based on the wordclouds generated, we could not determine differences between sentiment classes.

5.1.2 - TF-IDF

We began our comparison of NLP and classification models by defining our baseline performance. To do so, we trained a TF-IDF count-vectorizer and logistic regression model. To determine the optimal parameters for TF-IDF, we performed Grid Search cross validation on different ngram_ranges. The relative performance of the vectorizer with each parameter was determined by coupling the cross validation process with a OLS-logistic regression model. After performing cross validation using this pipeline, we observed that the optimal configuration for the ngram_ranges parameter was (1,2), (which means unigrams and bigrams). This optimized model was used as our baseline.

5.1.3 - Logistic Regression

We next sought to find the optimal parameters for the logistic regression model used to make the sentiment prediction given the best TF-IDF model parameters discovered. Again, cross validation was performed, this time on the regularization parameter, C. Furthermore, the model regularization was set to the L2-norm. From the scikit-learn documentation [6], the regularization parameter, C, is inversely proportional to the regularization strength. A range of values were investigated which corresponded to very strong regularization up to very weak regularization. Results will be summarized in the contribution section.

5.1.4 - Multinomial Naive Bayes

In our review of the literature surrounding sentiment analysis, we found that Naive Bayes Classifiers are commonly used. In particular, Multinomial Naive Bayes has been shown to perform well in sentiment classification tasks [7]. As before, we performed GridSearch cross validation to tune this model; the hyperparameters of the scikit-learn implementation are limited [8], but the 'fit_prior' parameter, which is a boolean value telling the model whether or not to learn class prior probabilities, and the 'alpha' parameter, which controls additive smoothing of the model to handle the issue of zero probability common to Naive Bayes. After cross validation, the best parameters were chosen and used to refit a new model to the entire training set. This trained model was used to test on the held out review texts as well as shifted test sets.

5.1.5 - Word2Vec

Next, we explored another word embedding method Word2Vec, which aims to capture the semantic closeness of words. The two parameters we explored were *min_count* and *window*. The parameter *min_count* denotes how many times the word should appear in the entire training corpus in order to be considered a vocabulary word in the model. The values explored were 10, 20, and 50. The parameter *window* is the maximum distance between the current and predicted words in a sentence. The values tested were 2 and 4. Unlike the previous model, since scikit-learn's GridSearch API for Word2Vec has been deprecated, we manually constructed 6 models and compared its performance. The Word2Vec embedding was coupled with OLS logistic regression to predict review sentiment.

5.1.6 - LSTM

With the best result from Word2Vec, the trained embedding was used to train the LSTM model. Again, it is expensive to tune the numerous hyper-parameters available for an LSTM model. We explored one hyperparameter *SpatialDropout1D(x)*. Unlike the traditional dropout technique that considers element-wise random dropout, *SpatialDropout1D(x)* randomly drops out a dimension of the feature map of all the channels. SpatialDropout is an effective means of encouraging the learning of independent feature maps [9]. We considered values [0.2, 0.3, 0.4].

5.1.7 - BERT

An additional model we tried running was a classification model using a pre-trained version of BERT: BERT_base_cased. This model was trained on BookCorpus, a dataset consisting of 11,038 unpublished books and English Wikipedia (excluding lists, tables and headers) [10]. We intended to use this state-of-the-art technique as a way of having an additional baseline to compare our other, more fine-tuned models on. This notebook from Kaggle titled "Sentiment Analysis using BERT"[11] was adapted for our dataset.

5.2 - Evaluation contributions:

5.2.1 - Tuning TF-IDF n-gram parameter

Using a logistic regression with standard parameters, Grid Search with cross validation was applied to try out different ngram ranges. The best ngram range was (1,2), which includes unigrams and bigrams, contrasting with our expectations (the bigger the ngrams, the more context and the better the predictions), but it appears that in this case having less context around each word proved to be more informative. This improves the performance of the notebook we were basing our modeling on, since they used (1,4) for their ngram ranges (**Fig. 3**). Other hyperparameters of TF-IDF could have been analyzed and tuned, such as maximum and minimum document frequency. Due to time and resources constraints we decided to use the default parameters. The accuracy of our baseline model is therefore 0.85.

5.2.2 - Tuning Logistic Regression on optimal TF-IDF embeddings

Using the best parameters for TF-IDF defined above, count-vectorization was performed and used as input to cross validation of the Logistic Regression model. As mentioned above, the regularization parameter was tuned in conjunction with L2. This resulted in better-than-expected performance given the models used. Using accuracy and ROC AUC metrics, we saw that the best regularization parameter for logistic regression was 10 (Figure 4). More regularization (as C → 0) resulted in poorer performance, while the improvements gained from less regularization (as C → infinity) diminished above C=10. After refitting to the entire training set using the best parameter, the accuracy on held out test data was: 0.895 and ROC AUC was 0.856 [12]. It is interesting to note that while both the training and validation curves for accuracy and ROC AUC show near optimal results, that the final test accuracy was much lower. This may suggest that these plots show overfitting to the training data, though the fact that the cross validation metrics do not show a similar decrease.

5.2.3 - Pairing TF-IDF with Multinomial Naive Bayes

Logistic regression did surprisingly well, but as mentioned previously one of the commonly used classification models used for sentiment analysis is Naive Bayes. More specifically, under naive bayes we make the assumption that the class can be predicted using the product of individual conditional probabilities [13]. However, even using tuned parameters MultinomialNB (**Fig. 5**), the performance actually declined slightly in comparison to logistic regression; the accuracy on held out test data was: 0.819 and ROC AUC was 0.874. We hypothesize that the random split that was used to produce the training and testing sets happened to partition the data such that the samples in the test set were notably more difficult than in the training set.

5.2.4 - Word2Vec

Next, unlike TF-IDF which only captures the significance of words in a corpus, Word2Vec aims to capture the semantic closeness of words. With the hyper-parameters tested, we can observe the *window* of 4 on average performed better than *window* of 2. With being able to use more words in the prediction, the model is able to better capture the context. Moreover, by having more words in the vocabulary (with a lower *min_count*), the model generally performed better. However, the trend is not conclusive. Overall, we found that the best parameter combinations would be *min_count* = 10 and *window* = 4 (**Fig. 6)**.

Interestingly, Word2Vec embeddings with logistic regression did not perform better than TF-IDF with logistic regression. It is possible that we have not found the optimal hyper-parameters (**Fig. 7)**. It will be good to set up a more extensive search for optimal hyper-parameter combinations.

5.2.5 - LSTM

Next, using the best Word2Vec embedding obtained from the previous section, we can already observe a significant improvement with the incorporation of bi-directional LSTM (**Fig. 8**), which uses both previous words to predict the next word and the latter words to predict the previous word. With almost identical LSTM structure but with our own Word2Vec embedding, we were able to obtain an accuracy of 0.90 on the test data; whereas the Kaggle notebook we referenced only obtained an accuracy of 0.83 on the train data. With the *SpatialDropout1D( )* hyperparameter, we can observe that while it can improve the model performance (**Fig. 9**), the trend is not conclusive. It will be interesting to compare the effect of the *SpatialDropout1D(x)* layer compared to the *dropout* parameter within the LSTM layer.

5.2.6 - BERT ('bert-base-cased')

When applying BERT to our dataset we got a test accuracy of 0.8712 and AUC score of 0.8205 (**Fig. 10**). While this is an adequate accuracy, it is a lower result than expected, and we believe it was due to three reasons: (1) In order to manage the memory requirement from notebooks we had to limit our entire database to 50,000 reviews (12.5%). With more resources available the model should be trained on the entire dataset. (2) No hyperparameter tuning was applied due to the extreme time duration of processes. By adjusting hyperparameters and improving the architecture of the model better results could be obtained. (3) The pre-trained model was trained on books and wikipedia articles that can be quite different from the language used on user reviews. A different pre-trained BERT model should be used, maybe one trained on Google News which could have more "sentimental" vocabulary.

Effect of Data Shifts:

The effect of data shift was rather interesting. We hypothesized that "summary" text would generally yield lacking results due to the limited number of words that the model can use to predict the sentiment. This hypothesis is reflected in the decreasing AUC and accuracy that we observe in the random dropout samples. However, surprisingly, “summary” text was able to generate similar AUC and accuracy results as random dropout of 50% of the full review text (**Fig. 11, Fig. 12**). In retrospect, while the summary text is generally concise, customers often use precise words to capture their feelings about the products, such as “best food”, “worst product”, etc; therefore, despite having limited words, overall, summary text is able to provide enough context for the models to make good predictions.

6) Compute/Other Resources Used
We utilized AWS resources for data storage as well as model training and testing. Raw reviews data was cleaned in Google Colab, which is then exported and uploaded to an Amazon S3 bucket. We trained, tuned, and tested our models on Amazon Sagemaker notebook instances.

7) Conclusions
The “Amazon Fine Food Reviews” dataset was uploaded at least 6 years ago. Over the years, many have attempted to analyze the data with a variety of NLP models. Upon closer inspection of the Kaggle notebooks, one can soon realize that most notebooks explore only 1 model with limited hyperparameter tuning. We initially only planned on exploring the use of TF-IDF and LSTM for conducting sentiment analysis. However, overtime, we learned of the various models that can be implemented, and were able to deploy a workable Word2Vec, Multinomial Naive Bayes classifier, and BERT. While a lot of resources are available on Kaggle, we still spent a lot of time updating the code (some packages were updated) and fitting the code to our dataset. In addition, as the model complexity increased, the training time also increased. Also, since the more complex models were not compatible with the scikit-learn GridSearch method, we were unable to experiment with more hyperparameter combinations.

In the future, one can explore the use of pre-trained BERT embeddings that may be more closely aligned with review data, such as Google news [14]. Given the polarized nature of news content, it is possible that a model trained on this corpus of data would perform better than the BERT model we picked initially. Further, future work can be focused on hyperparameter tuning with the existing models as most of the complex models have myriads of hyperparameters that can be explored.

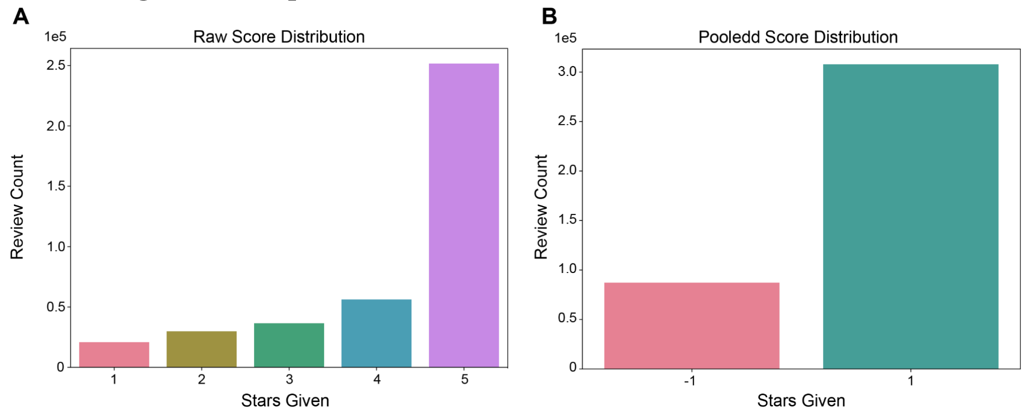
NLP models can have social and environmental impact. They are significantly vulnerable to bias in data. These models could treat certain words or language expressions related to vulnerable populations as having a negative sentiment, which can impact the performance of users and businesses that buy and sell these food products if implemented unchecked. Additionally, state of the art transformers and other deep learning techniques release a significant amount of carbon dioxide into the atmosphere. Sharing and using pre-trained models like word2vec and BERT helps reduce the impact of this activity. The graph on **Fig 13** shows CO2 emissions of ML training versus other human activities [15].

8) Roles of team members:
Federico Cimini: Federico helped build the initial notebook for data cleaning and preprocessing, and the word clouds. He tuned hyperparameters of TF-IDF and ran the BERT model.
Sam Thudium: Sam produced the visualizations for label distributions and text summary statistics. He also set up the AWS environment, tuned hyperparameters for logistic regression and multinomial naive bayes, and wrote helper functions used in several notebooks.
Liang-Yun Cheng: Liang worked on using various Kaggle notebooks to set up the Word2Vec and LSTM model. She also explored the effect of various hyperparameters on each of those models.

References:

[1] Amazon Fine Food Reviews;
https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews
[2] Eugen Anghel:
https://www.kaggle.com/code/eugen1701/predicting-sentiment-and-helpfulness
[3] Chirag Samal:
https://www.kaggle.com/code/chirag9073/amazon-fine-food-reviews-sentiment-analysis
[4] Pierre Megret: https://www.kaggle.com/code/pierremegret/gensim-word2vec-tutorial
[5] Arunava:
https://www.kaggle.com/code/arunava21/word2vec-and-random-forest-classification
[6] Sklearn log reg:
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
[7] Multinomial NB paper: http://paper.ijcsns.org/07_book/201903/20190310.pdf
[8] Sklearn mulitnomialNB:
https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
[9] "Efficient Object Localization Using Convolutional Networks" for *SpatialDropout1D(x)*:
https://arxiv.org/pdf/1411.4280.pdf
[10] bert-base-cased: https://huggingface.co/bert-base-cased
[11] Prakhar Pathi https://www.kaggle.com/code/prakharrathi25/sentiment-analysis-using-bert
[12] See model results file for model:
https://github.com/sthudium25/CIS5190-AmazonFoodReviews
[13] "Speech and Language Processing", Daniel Jurafsky & James H. Martin. 2023.
https://web.stanford.edu/~jurafsky/slp3/4.pdf
[14] Google News BERT pre-trained: https://huggingface.co/fse/word2vec-google-news-300
[15] How do Transformers work? https://huggingface.co/learn/nlp-course/chapter1/4

APPENDIX 1 - Images and Graphs



C

Metric	Full Review	Summary
Mean	39.56	2.83
Minimum	1	1
Maximum	1951	24
Standard Deviation	39.81	1.55
Median	28	2

Figure 1.

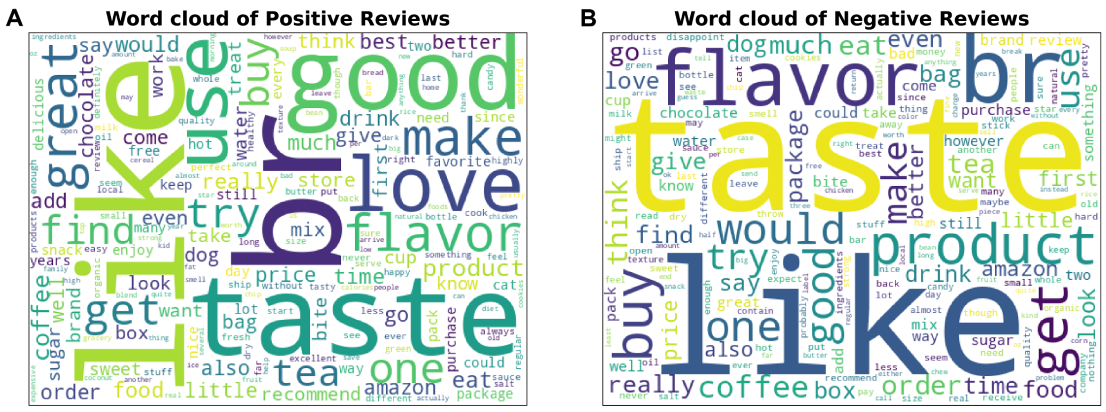


Figure 2.

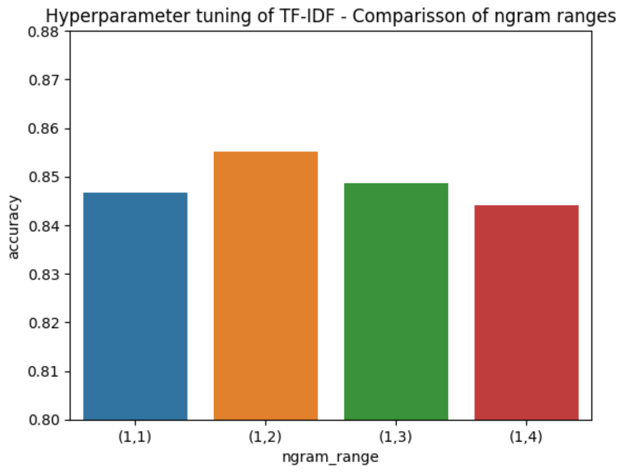


Figure 3.

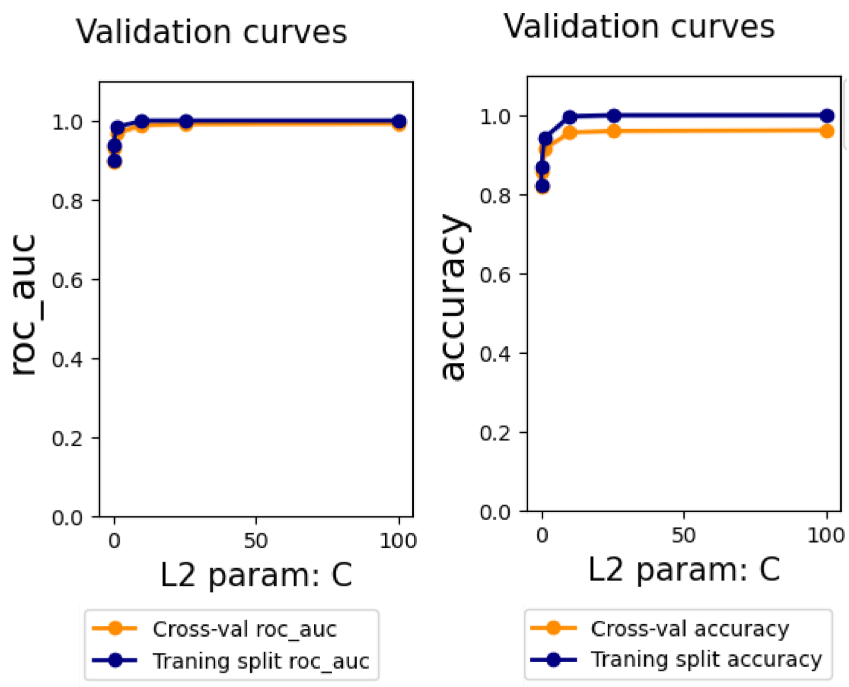


Figure 4.

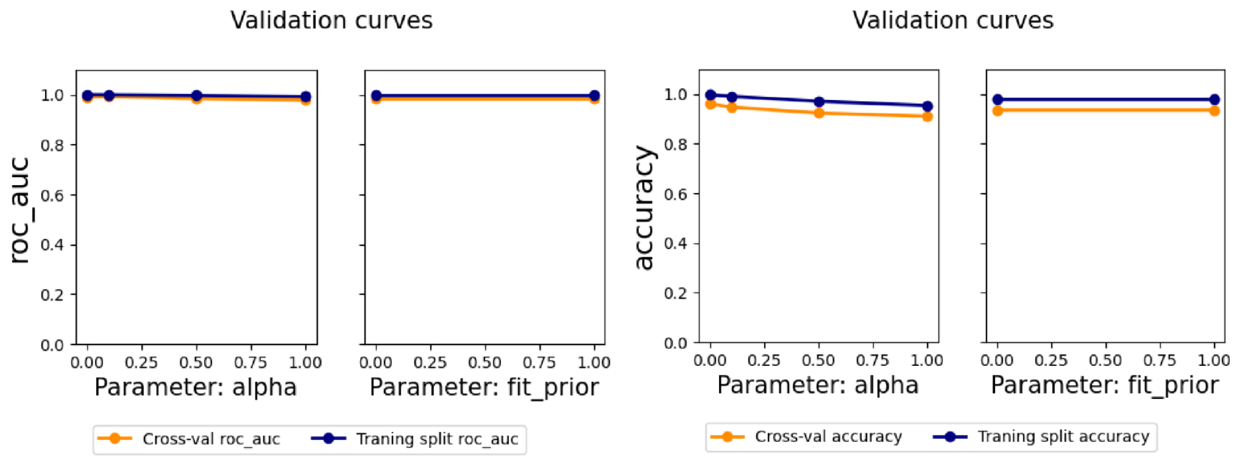


Figure 5.

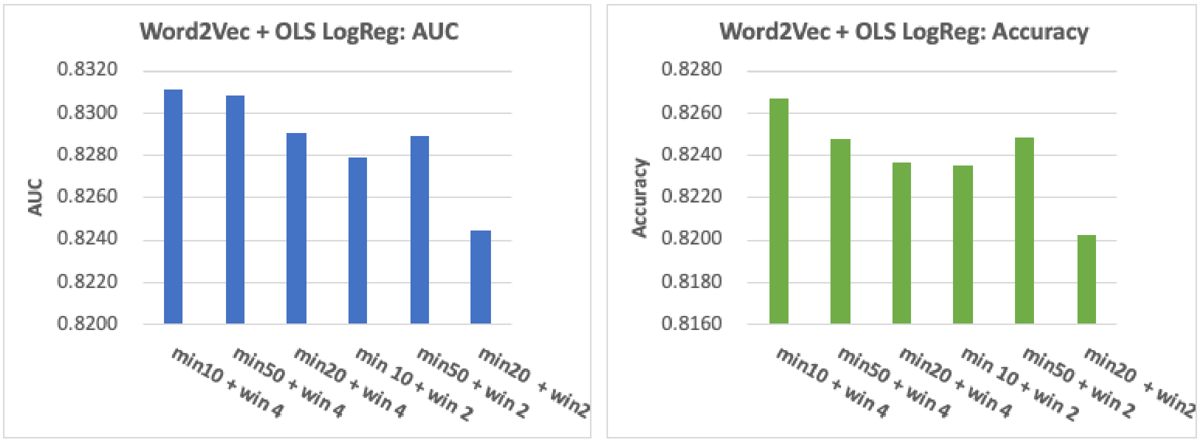


Figure 6.

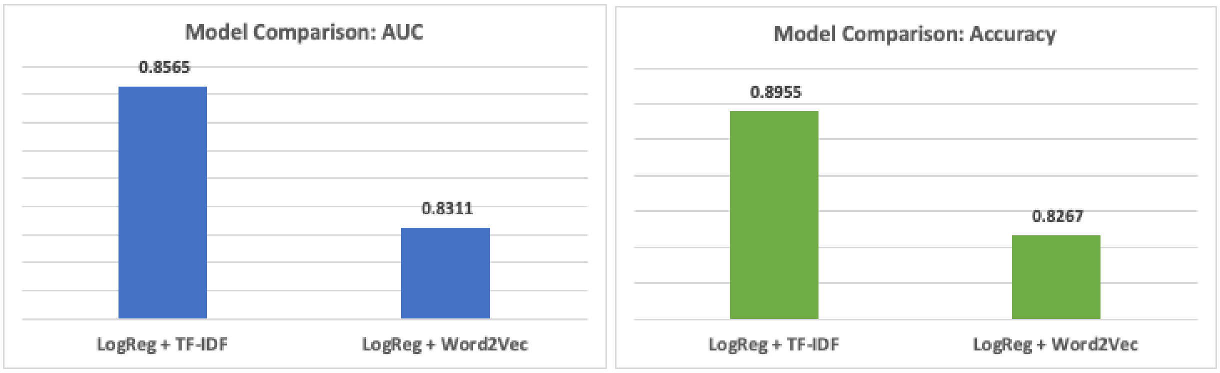


Figure 7.

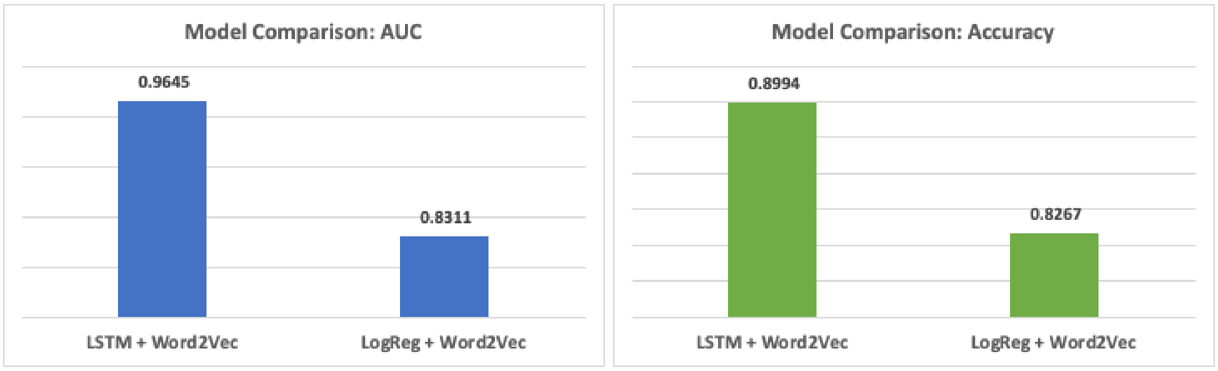


Figure 8.

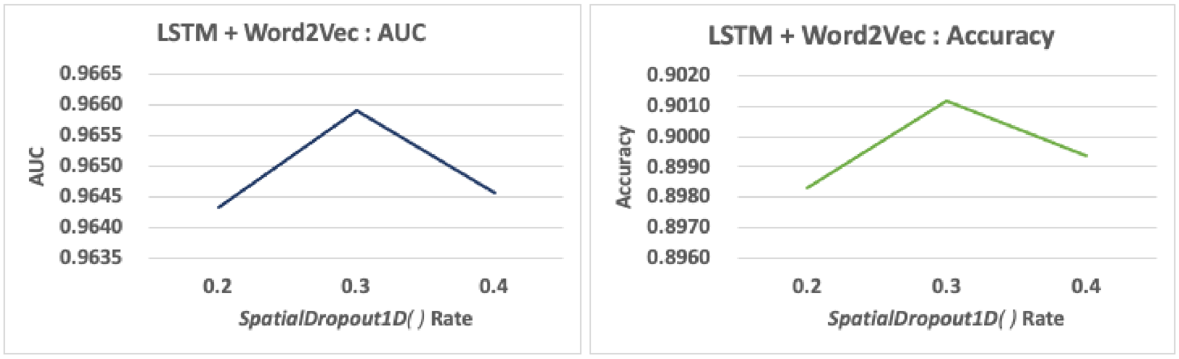


Figure 9.

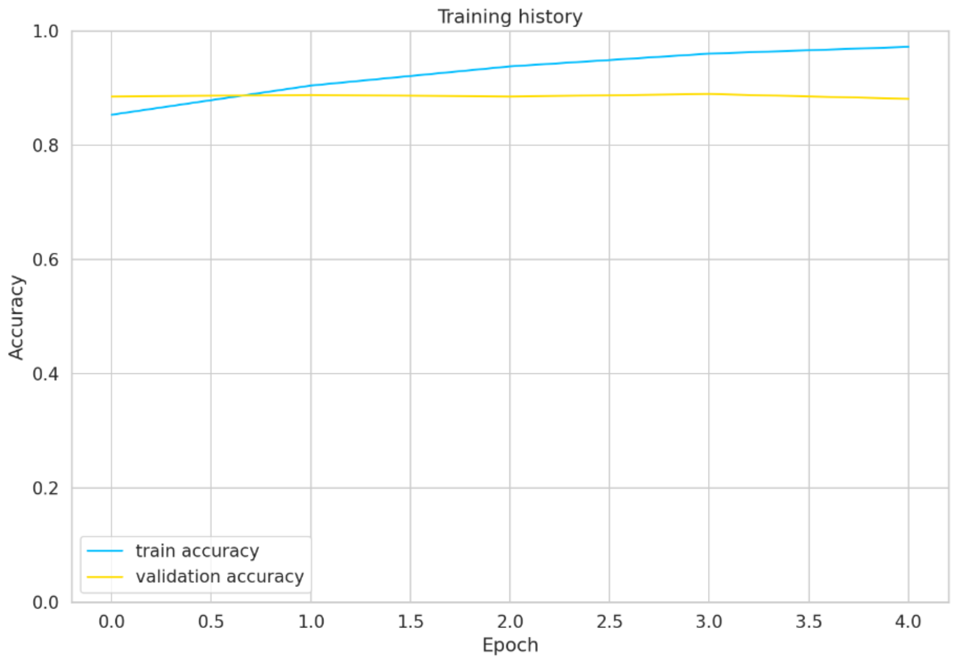


Figure 10.

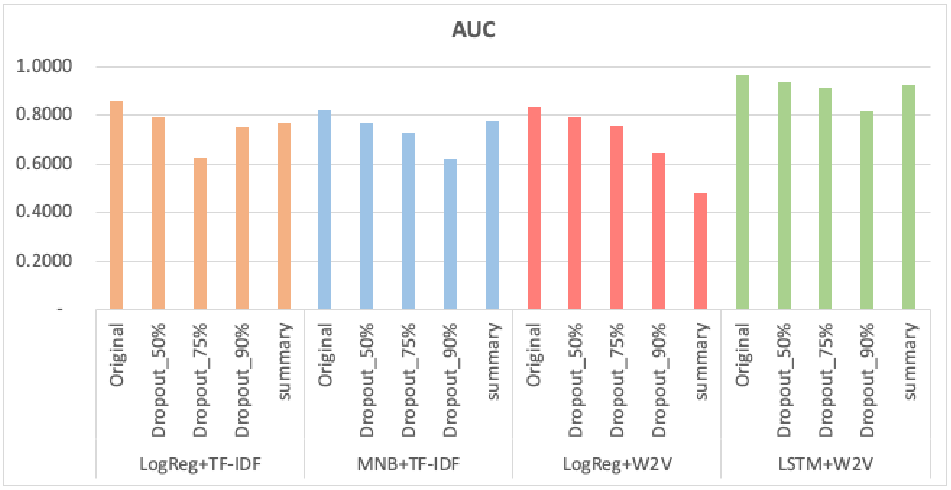


Figure 11.

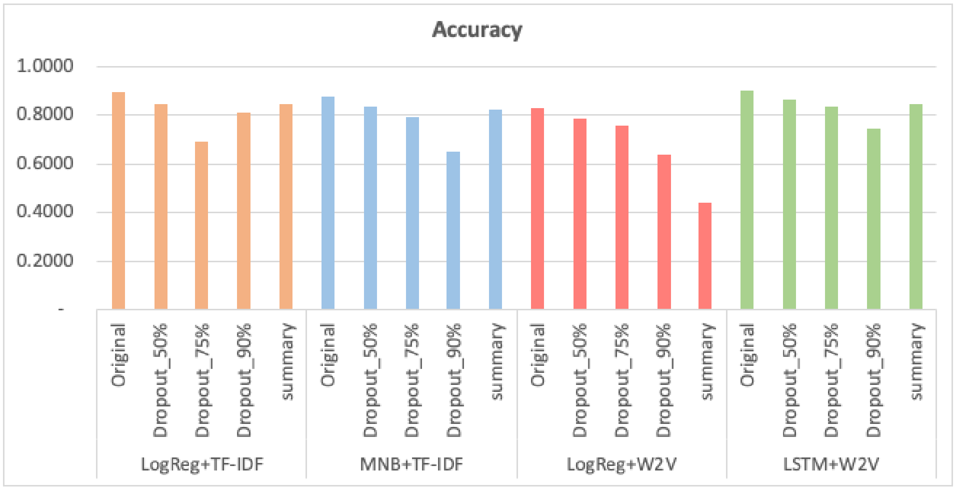


Figure 12.

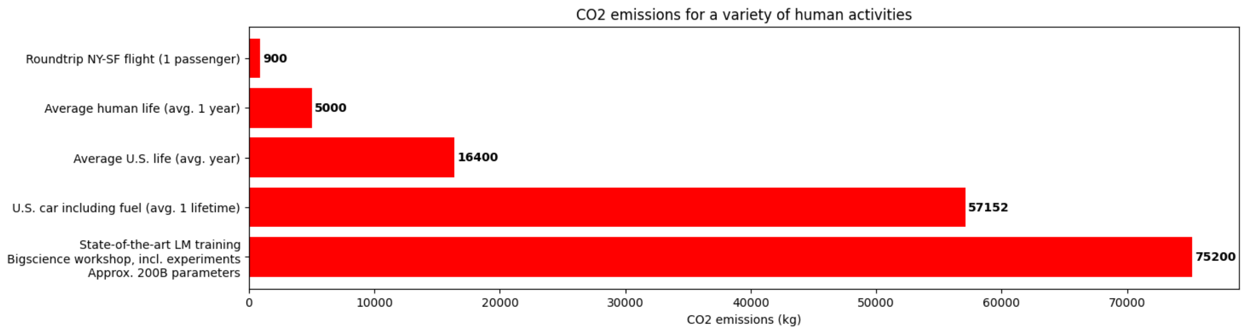
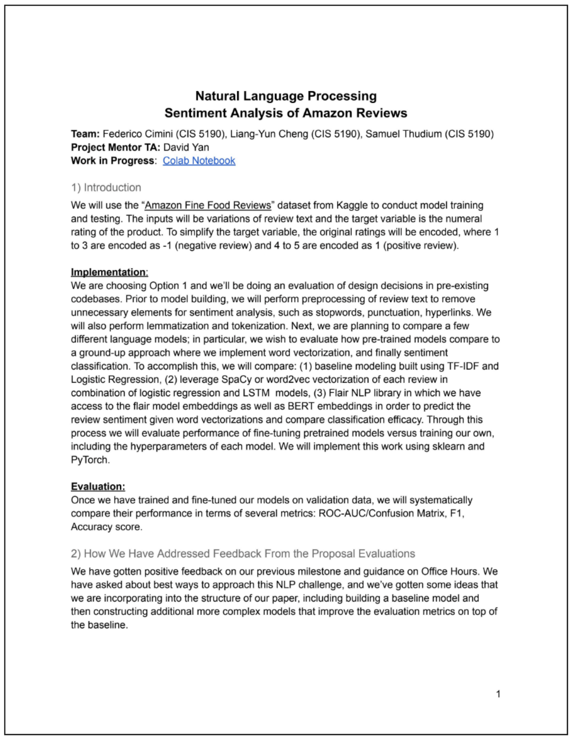


Figure 13.

APPENDIX 2: Key Prior Works

A. Predicting Sentiment and Helpfulness. Provides the initial configurations and logic of our baseline model, which we tuned.
https://www.kaggle.com/code/eugen1701/predicting-sentiment-and-helpfulness
B. Gensim Word2Vec Tutorial. Provides the basic guidelines for Word2Vec
https://www.kaggle.com/code/pierremegret/gensim-word2vec-tutorial
C. word2vec and random forest classification. Provides the basic guidelines for Word2Vec
https://www.kaggle.com/code/arunava21/word2vec-and-random-forest-classification
D. Amazon Fine Food Reviews: Sentiment Analysis. Provides the basic guidelines for LSTM.
https://www.kaggle.com/code/chirag9073/amazon-fine-food-reviews-sentiment-analysis

APPENDIX 3 - Midway report (Milestone 2)



Natural Language Processing
Sentiment Analysis of Amazon Reviews

Team: Federico Cimini (CIS 5190), Liang-Yun Cheng (CIS 5190), Samuel Thudium (CIS 5190)
Project Mentor TA: David Yan
Work in Progress: Colab Notebook

1) Introduction

We will use the "Amazon Fine Food Reviews" dataset from Kaggle to conduct model training and testing. The inputs will be variations of review text and the target variable is the numeral rating of the product. To simplify the target variable, the original ratings will be encoded, where 1 to 3 are encoded as -1 (negative review) and 4 to 5 are encoded as 1 (positive review).

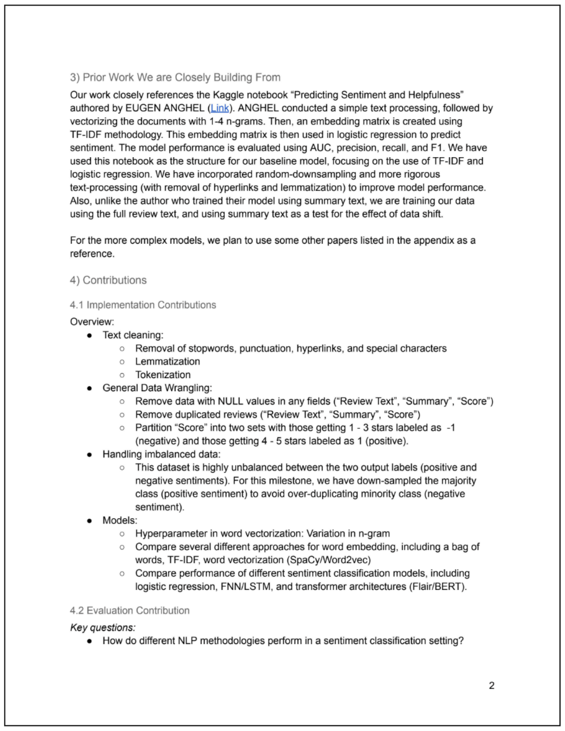
Implementation:
We are choosing Option 1 and we'll be doing an evaluation of design decisions in pre-existing codebases. Prior to model building, we will perform preprocessing of review text to remove unnecessary elements for sentiment analysis, such as stopwords, punctuation, hyperlinks. We will also perform lemmatization and tokenization. Next, we are planning to compare a few different language models; in particular, we wish to evaluate how pre-trained models compare to a ground-up approach where we implement word vectorization, and finally sentiment classification. To accomplish this, we will compare: (1) baseline modeling built using TF-IDF and Logistic Regression, (2) leverage SpaCy or word2vec vectorization of each review in combination of logistic regression and LSTM models, (3) Flair NLP library in which we have access to the flair model embeddings as well as BERT embeddings in order to predict the review sentiment given word vectorizations and compare classification efficacy. Through this process we will evaluate performance of fine-tuning pretrained models versus training our own, including the hyperparameters of each model. We will implement this work using sklearn and PyTorch.

Evaluation:
Once we have trained and fine-tuned our models on validation data, we will systematically compare their performance in terms of several metrics: ROC-AUC/Confusion Matrix, F1, Accuracy score.

2) How We Have Addressed Feedback From the Proposal Evaluations

We have gotten positive feedback on our previous milestone and guidance on Office Hours. We have asked about best ways to approach this NLP challenge, and we've gotten some ideas that we are incorporating into the structure of our paper, including building a baseline model and then constructing additional more complex models that improve the evaluation metrics on top of the baseline.

1



3) Prior Work We are Closely Building From

Our work closely references the Kaggle notebook "Predicting Sentiment and Helpfulness" authored by EUGEN ANGHEL (Link). ANGHEL conducted a simple text processing, followed by vectorizing the documents with 1-4 n-grams. Then, an embedding matrix is created using TF-IDF methodology. This embedding matrix is then used in logistic regression to predict sentiment. The model performance is evaluated using AUC, precision, recall, and F1. We have used this notebook as the structure for our baseline model, focusing on the use of TF-IDF and logistic regression. We have incorporated random-downsampling and more rigorous text-processing (with removal of hyperlinks and lemmatization) to improve model performance. Also, unlike the author who trained their model using summary text, we are training our data using the full review text, and using summary text as a test for the effect of data shift.

For the more complex models, we plan to use some other papers listed in the appendix as a reference.

4) Contributions

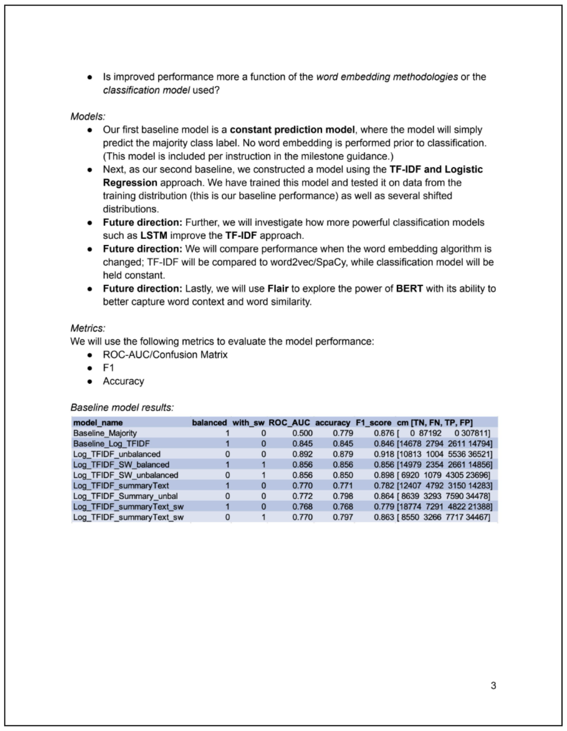
4.1 Implementation Contributions

Overview:
- Text cleaning:
    - Removal of stopwords, punctuation, hyperlinks, and special characters
    - Lemmatization
    - Tokenization
- General Data Wrangling:
    - Remove data with NULL values in any fields ("Review Text", "Summary", "Score")
    - Remove duplicated reviews ("Review Text", "Summary", "Score")
    - Partition "Score" into two sets with those getting 1 - 3 stars labeled as -1 (negative) and those getting 4 - 5 stars labeled as 1 (positive).
- Handling imbalanced data:
    - This dataset is highly unbalanced between the two output labels (positive and negative sentiments). For this milestone, we have down-sampled the majority class (positive sentiment) to avoid over-duplicating minority class (negative sentiment).
- Models:
    - Hyperparameter in word vectorization: Variation in n-gram
    - Compare several different approaches for word embedding, including a bag of words, TF-IDF, word vectorization (SpaCy/Word2vec)
    - Compare performance of different sentiment classification models, including logistic regression, FNN/LSTM, and transformer architectures (Flair/BERT).

4.2 Evaluation Contribution

Key questions:
- How do different NLP methodologies perform in a sentiment classification setting?

2



- Is improved performance more a function of the *word embedding methodologies* or the *classification model* used?

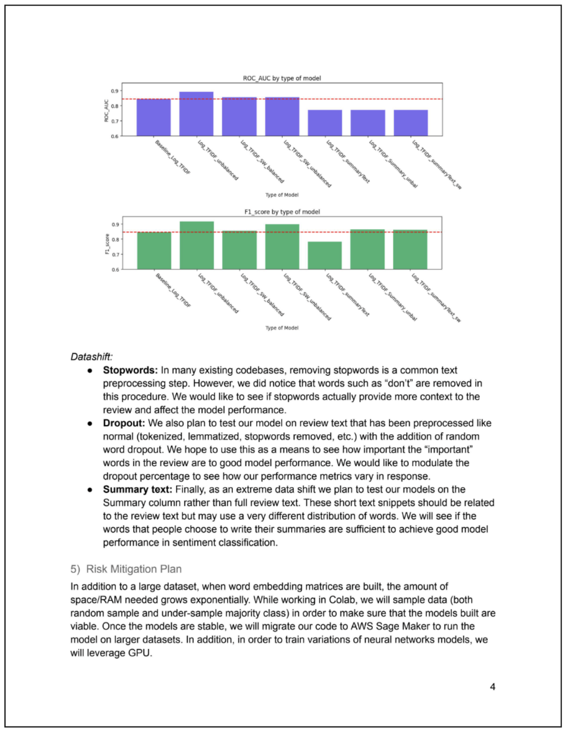
Models:
- Our first baseline model is a **constant prediction model**, where the model will simply predict the majority class label. No word embedding is performed prior to classification. (This model is included per instruction in the milestone guidance.)
- Next, as our second baseline, we constructed a model using the **TF-IDF and Logistic Regression** approach. We have trained this model and tested it on data from the training distribution (this is our baseline performance) as well as several shifted distributions.
- **Future direction:** Further, we will investigate how more powerful classification models such as **LSTM** improve the **TF-IDF** approach.
- **Future direction:** We will compare performance when the word embedding algorithm is changed; TF-IDF will be compared to word2vec/SpaCy, while classification model will be held constant.
- **Future direction:** Lastly, we will use **Flair** to explore the power of **BERT** with its ability to better capture word context and word similarity.

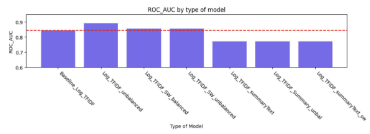
Metrics:
We will use the following metrics to evaluate the model performance:
- ROC-AUC/Confusion Matrix
- F1
- Accuracy

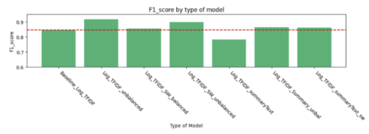
Baseline model results:

model_name	balanced	with_sw	ROC_AUC	accuracy	F1_score	cm [TN, FN, TP, FP]
Baseline_Majority	1	0	0.500	0.779	0.876	[0 87192 0 307811]
Baseline_Log_TFIDF	1	0	0.845	0.845	0.846	[14678 2794 2611 14794]
Log_TFIDF_unbalanced	0	0	0.892	0.879	0.918	[10813 1004 5536 36521]
Log_TFIDF_SW_balanced	1	1	0.856	0.856	0.856	[14979 2354 2661 14856]
Log_TFIDF_SW_unbalanced	0	1	0.856	0.850	0.898	[6920 1079 4305 23696]
Log_TFIDF_summaryText	1	0	0.770	0.771	0.782	[12407 4792 3150 14283]
Log_TFIDF_Summary_unbal	0	0	0.772	0.798	0.864	[8639 3293 7590 34478]
Log_TFIDF_summaryText_sw	1	0	0.768	0.768	0.779	[18774 7291 4822 21388]
Log_TFIDF_summaryText_sw	0	1	0.770	0.797	0.863	[8550 3266 7717 34467]

3





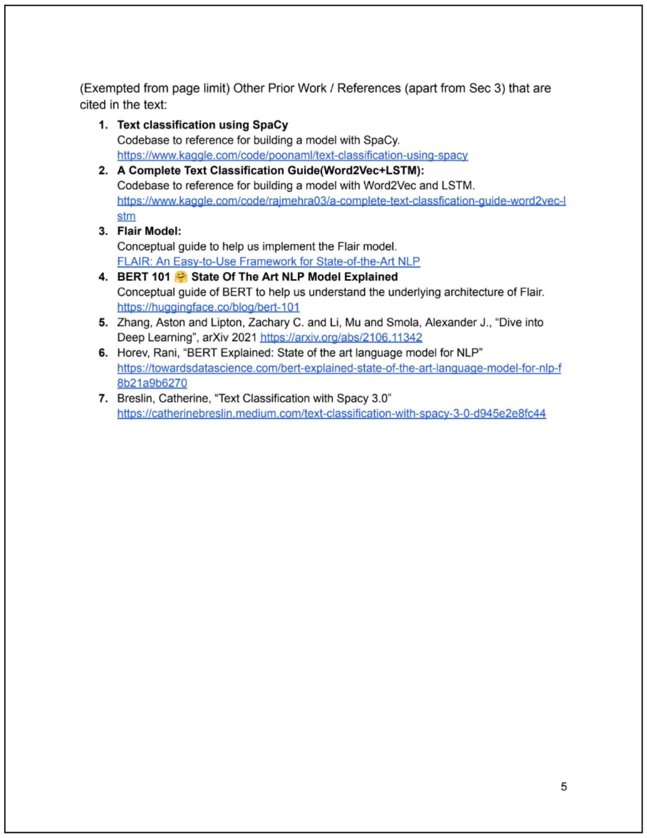


Datashift:
- **Stopwords:** In many existing codebases, removing stopwords is a common text preprocessing step. However, we did notice that words such as "don't" are removed in this procedure. We would like to see if stopwords actually provide more context to the review and affect the model performance.
- **Dropout:** We also plan to test our model on review text that has been preprocessed like normal (tokenized, lemmatized, stopwords removed, etc.) with the addition of random word dropout. We hope to use this as a means to see how important the "important" words in the review are to good model performance. We would like to modulate the dropout percentage to see how our performance metrics vary in response.
- **Summary text:** Finally, as an extreme data shift we plan to test our models on the Summary column rather than full review text. These short text snippets should be related to the review text but may use a very different distribution of words. We will see if the words that people choose to write their summaries are sufficient to achieve good model performance in sentiment classification.

5) Risk Mitigation Plan

In addition to a large dataset, when word embedding matrices are built, the amount of space/RAM needed grows exponentially. While working in Colab, we will sample data (both random sample and under-sample majority class) in order to make sure that the models built are viable. Once the models are stable, we will migrate our code to AWS Sage Maker to run the model on larger datasets. In addition, in order to train variations of neural networks models, we will leverage GPU.

4



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

1. **Text classification using SpaCy**
   Codebase to reference for building a model with SpaCy.
   https://www.kaggle.com/code/poonaml/text-classification-using-spacy
2. **A Complete Text Classification Guide(Word2Vec+LSTM):**
   Codebase to reference for building a model with Word2Vec and LSTM.
   https://www.kaggle.com/code/rajmehra03/a-complete-text-classfication-guide-word2vec-lstm
3. **Flair Model:**
   Conceptual guide to help us implement the Flair model.
   FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP
4. **BERT 101 🤗 State Of The Art NLP Model Explained**
   Conceptual guide of BERT to help us understand the underlying architecture of Flair.
   https://huggingface.co/blog/bert-101
5. Zhang, Aston and Lipton, Zachary C. and Li, Mu and Smola, Alexander J., "Dive into Deep Learning", arXiv 2021 https://arxiv.org/abs/2106.11342
6. Horev, Rani, "BERT Explained: State of the art language model for NLP"
   https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270
7. Breslin, Catherine, "Text Classification with Spacy 3.0"
   https://catherinebreslin.medium.com/text-classification-with-spacy-3-0-d945e2e8fc44