Comparative Study of Binary Sentiment Analysis for the Product Review dataset with the Explainable AI framework

Dimitra Muni
732A92 - Text Mining
Linköping University
dimmu472@student.liu.se

Abstract

This project aimed to analyze binary sentiment for the Amazon Fine Food Reviews dataset[1] using different vectorization and word embedding techniques and evaluated the performance on the gold standard dataset. In the first part of the project, we utilized two different word vectorization (i.e. count and TF-IDF vectorizers) techniques in conjunction with three linear classifiers (i.e. Multinomial Naive Bayes, Logistic Regression, and Support Vector Machine (SVM)) to create six experimental pipelines. Furthermore, we used the *Halving Grid Search* technique to find the best-performing model with five-fold cross-validation. Here SVM slightly outperformed the other two models with an accuracy of 93%. In the second part of the project, we utilized pre-trained DistilBERT to perform sentiment classification on a smaller subset of original data; without fine-tuning, it achieved 85% accuracy, but the performance was not as high as the SVM model. In the third part of the project, we performed LIME (Ribeiro et al., 2016) analysis for a subset of misclassified samples identified from the final SVM model and documented actionable insights.

1 Introduction

In the last decade, massive development in Masked Language Models has opened the floodgates of possibility in Natural Language Processing (NLP) domain. **B**idirectional **E**ncoder **R**epresentations from **T**ransformers (BERT), developed by (Devlin et al., 2018) at Google, is one such model. The base version of BERT has 110 million parameters and is computationally expensive to train. Several variants of this model, such as DistilBERT (Sanh et al., 2019) with 66 million parameters, is significantly faster than the base version of BERT; however, with a decline in performance.

In this project, we focus on the problem of binary sentiment analysis for the Amazon Fine Foods dataset, which contains product reviews collected over a decade. We intend to address the following research objectives,

1. Experiment with word vectorizer-based technique with machine learning classifiers to perform sentiment classification.

2. Investigate the applicability of the compact Masked Language Model (MLM) for sentiment classification.

3. Inference about misclassified samples obtained from the best-performing model, generating actionable insight using an explainable AI technique.

In the initial part of this project, we created six different vectorizer-classifier pipelines, using two vectorizers and three classifiers; we utilized a halving grid search technique with five-fold cross-validation to tune these pipelines and evaluated the performance on the test set. In the second part of the project, we trained the DistilBERT model for ten epochs and evaluated the performance on the test set. In the last part of the project, we focus on the misclassified samples (from the test set) identified using the Linear SVM model. We used a popular explainable AI technique called LIME analysis (Ribeiro et al., 2016) to perform this study.

2 Theory

2.1 Vectorization

The vectorization technique converts the corpus of text data into a matrix representation; one such technique is Count Vectorizer, which uses a *bag-of-word* representation (Jurafsky and Martin, 2023). Here the corpus is assumed to be a collection of words, where a matrix represents the frequency of the word w in document d. A major constraint

[1] https://snap.stanford.edu/data/web-FineFoods.html

of this representation is that the word's position in a sentence is not inconsequential; rather, the frequency of the word is important (Jurafsky and Martin, 2023). Another more advanced method of vectorization is *TF-IDF*; here, *TF* refers to the **term frequency**, which accounts for the occurrence of the term t in document d, while *IDF* is **inverse document frequency**, a measure of the number of documents a term t appears. One noteworthy advantage of TF-IDF representation is that it assigns relevance to rare occurring words by higher IDF value (Schütze et al., 2008).

2.2 Classifiers

2.2.1 Multinomial Naive Bayes

Multinomial Naive Bayes classifier is based on Bayes' rule (Bayes, 1763)[2] about the conditional probability of two events. This classifier is often a preferred baseline evaluation method due to its simplicity. According to Jurafsky and Martin (p.61, 2023), there is a **naive** assumption about conditional independence of the feature probability given the class.

2.2.2 Logistic Regression

According to Bishop and Nasrabadi (p.205, 2006), the logistic regression classifier is one of the linear classifier models that uses *logistic sigmoid* function[3] for binary classification, minimizing cross-entropy loss. One advantage of this method is that for the correlated features in the dataset, the logistic regression classifier is preferred over naive Bayes because the logistic regression can assign the weights more efficiently amongst the correlated features; while this mechanism is not present in the naive Bayes model (Jurafsky and Martin (p.86, 2023)); due to this reason, we intend to use this model in our experiments.

2.2.3 Support Vector Machines

Support Vector Machine (SVM) classifier originally presented by (Vapnik, 1999) belongs to the class of *maximum margin classifiers* (Bishop and Nasrabadi, 2006), where the objective is to find a hyperplane that separates two classes with the highest margin. There are several variations of SVM classifiers based on the kernel function. However, we only consider the Linear SVM classifier; this method is appropriate for datasets with high dimensions if the dataset has considerably more samples than the number of features; for this reason, we include this method in our analysis.

2.3 Masked Language Model

According to (Devlin et al., 2018), the BERT model is trained using a *masked language modeling* technique, where a model is trained by hiding randomly chosen tokens and is asked to predict that token. This model introduces bidirectional context to the training of transformer-based architecture using the attention mechanism (Vaswani et al., 2017). The BERT model is trained in two iterations; the first iteration is called **pre-training**, where the model is trained on unlabeled data, and the second is **fine-tuning**, where the model is tuned on labeled data. This model uses WordPiece embedding (Wu et al., 2016), which converts a word into sub-word units.

In figure [1], the pre-training and fine-tuning procedure is represented, where a sentence pair is separated by `[SEP]` token, and before each sentence, there is a classification token `[CLS]`. Input embedding is denoted by $\boldsymbol{E}$, and $\boldsymbol{C}$ represents the final vector corresponding to `[CLS]` token, and $T_1, T_2, ..., T_N$ represents tokens' final vector representation, where *N* = *768* for the BERT-base model.

2.3.1 DistilBERT

DistilBERT, introduced by (Sanh et al., 2019), is another example of a compact model, with the number of layers reduced by half[4], and has a similar architecture to that of a BERT-base model, as indicated in the table [1]. According to (Turc et al., 2019), knowledge distillation is the technique (Hinton et al., 2015), where a bigger teacher model transfers the knowledge to a more compact model.

Model	No. of Layers	Hidden Size	Attention Heads	Parameters (millions)
BERT-base	12	768	12	110
DistilBERT	6	768	12	66

Table 1: BERT and DistilBERT model Architecture Comparision

Triple Loss: Authors trained DistilBERT using a linear combination of three loss functions, cross-entropy distillation loss (L_{ce}), masked language modeling loss (L_{mlm}), and cosine embedding loss (L_{cos}) calculated by performing cosine operation on a hidden vector of student and teacher.

[2]Bayes' rule for two events x and y is presented as in term of conditional probability, $p(x|y)p(y) = p(y|x)p(x)$.

[3]logistic sigmoid: $\sigma(a) = \frac{1}{1+\exp(-a)}$

[4]Additionally token-type embedding and pooler are removed as well.

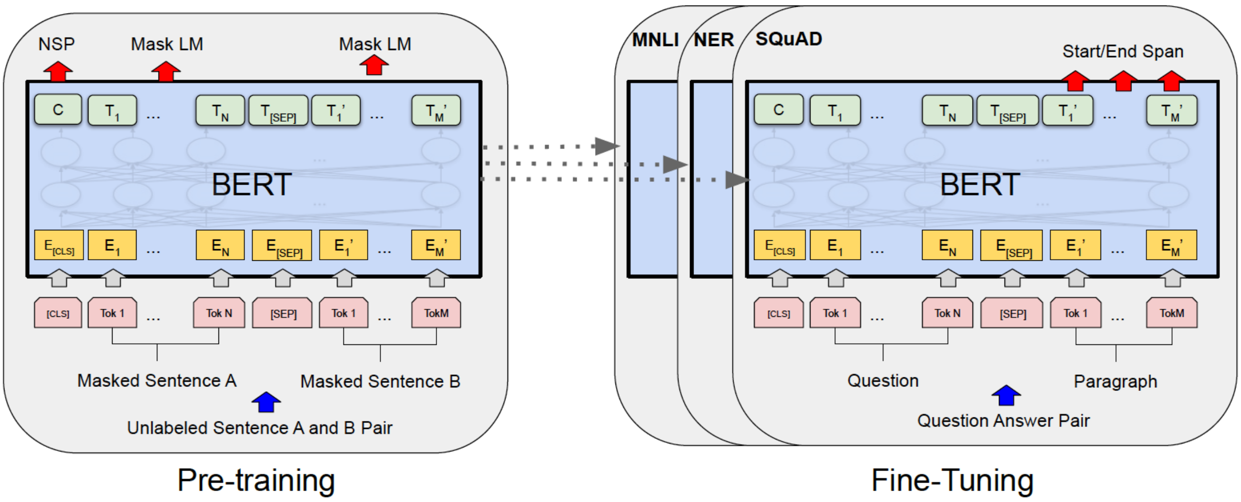

Figure 1: Pre-training and Fine tuning procedure in BERT, the figure taken from Devlin et al. (p.3, 2019)

According to (Sanh et al., 2019) **DistilBERT** is significantly faster (60%) and smaller (40%) than BERT-base. The authors evaluated DistilBERT for the sentiment classification task on the IMDb dataset, and it performed almost at par (accuracy of 92.82) with BERT-base (accuracy of 93.46). We intend to investigate the usage of DistilBERT for a similar task but with the Amazon Fine Foods dataset.

2.4 LIME

Local **I**nterpretable **M**odel agnostic **E**xplanation (LIME) (Ribeiro et al., 2016) is a popular explainable AI framework to understand the underlying pattern that black box models are trained with and for the inference about the predictions. LIME perturbs the black-box model, observes local changes, and provides a visual explanation for greater understanding. LIME is model agnostic, i.e. it can be used for any model and provide explanations.

3 Data

We have chosen the Amazon Fine Foods dataset hosted on the SNAP library affiliated with Stanford University. This dataset contains reviews of 74,258 unique products and 568,454 product reviews by 256,059 users, collected between October 1999 and October 2012.

The features of this dataset are described in the table [2]. Our analysis focused on the *text* and *score* variables. Firstly, in figure [2], the distribution of review score is presented, where the dataset had a class imbalance, with a higher number of reviews with a score of 5.

Feature	Description
productId	ASIN - Amazon Standard Identification Number
userId	reviewer identification
profileName	reviewer profile name
helpfullness	fraction of users who found the review helpful
score	score from 1 to 5
time	time of review post
summary	review summary
text	review text

Table 2: Feature Description for Amazon Fine Food dataset

Figure 2: Distribution of Review Scores

As we intend to perform a binary classification of the review dataset, the newly engineered dataset disregards the reviews with a score of 3, and the reviews with scores 1 and 2 are encoded with binary value '1' and reviews with scores 4 and 5 are

	Before Undersampling		After Undersampling	
Type	**Positive**	**Negative**	**Positive**	**Negative**
Train	371989	68965	68965	68965
Test	65646	12170	65646	12170

Table 3: Distribution of Positive and Negative Class before and after Undersampling

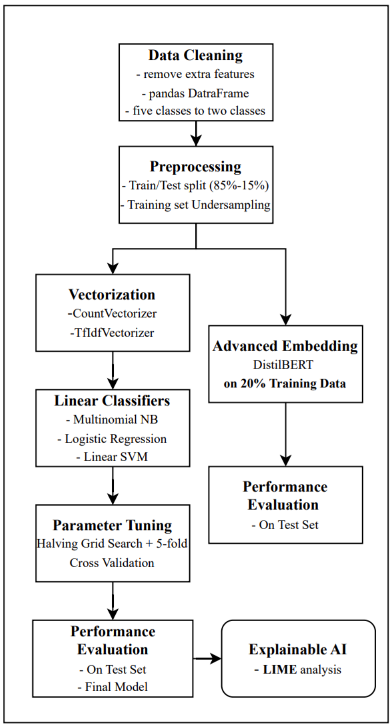

Figure 3: Workflow of the Project

encoded with '0'. After performing the aforementioned encoding, the dataset has 371,989 reviews with values '1' (positive reviews) and 68,965 reviews with values '0' (negative reviews); if we train our model with the imbalanced dataset, the results would be incorrect and untrustworthy. To mitigate this problem, we perform random undersampling using the `imblearn` python package, so the final dataset would have 68,965 reviews with positive and negative reviews classes each, as described in the table [3].

4 Method

The workflow of this project is visualized in figure 3. We have utilized python packages such as **scikit-learn** (Pedregosa et al., 2011), **imbalanced-learn** (Lemaître et al., 2017), **NumPy** (Harris et al., 2020), **Pandas** (Wes McKinney, 2010) (pandas development team, 2020) and **Matlplotlib** (Hunter, 2007).

4.1 Data Cleaning

The FineFoods dataset was scraped by (McAuley and Leskovec, 2013). The dataset is available in `txt.gz` format. The data-cleaning procedure is described as follows,

1. Created a raw list[5] from the textual data. Converted this list of rows into a dictionary, disregarded extra features if the raw has more than eight features. The dictionary was converted to a `pandas` data frame.

2. Finally, we converted the five-class classification problem into a binary classification problem, as documented in section [3].

4.2 Preprocessing

In the preprocessing part, first, we converted the dataset from the previous step into training and test sets using `train_test_split` [6] function from `sklearn` library, here the train-test split was selected to be 85%-15%. We used *stratify* option to maintain the same proportion of class division in training and test sets.

Furthermore, the class imbalance problem in the training dataset was addressed using `RandomUnderSampler`[7] from the `imblearn` library. The final training dataset has a 50%-50% division between positive and negative classes.

4.3 Vectorizer-Classifier Pipeline

We have used two vectorizers and three classifiers, as shown in the table [4] and [5], and set up six $(2 * 3)$ pipelines using `Pipeline` function from the `sklearn.pipeline` module.

Vectorizer	Parameters
Count Vectorizer and TF-IDF Vectorizer	ngram_range: [(1,1),(1,2),(1,3)] encoding:'latin-1' stop_words: 'english'

Table 4: Vectorizer with the parameters

We utilized the functions `CountVectorizer` and `TfidfVectorizer` from `sklearn.feature_extraction` module. We chose three different **n-gram** variations as indicated in table [4], these are the *unigram*, *unigram* + *bigram* and *unigram* + *bigram* + *trigram*. Additionally, we choose `'latin-1'` encoding and remove the stop words using `stop_words` command. We have utilized three

[5]Using Google Colab with python3 distribution.
[6]Available from `sklearn.model_selection`
[7]Available from the `imblearn.under_sampling` module

Classifier	Parameter
Multinomial NB	alpha: [1e-3,0.4, 0.8, 1,10]
Logistic Regression	C: [1e-3, 1e-2, 1e-1, 1, 10]
Linear SVM	C: [1e-3, 1e-2, 1e-1, 1, 10]

Table 5: Classifier with the grid of parameters

iteration	n_resource	no. of candidates
1	34,480	15
2	68,960	8
3	137,920	4

Table 6: Halving Grid Search Parameters

classifiers functions from `sklearn` library using `MultinomialNB`, `LogisticRegression` and `LinearSVC` functions[8].

Here `MultinomialNB` function is optimized for the smoothing parameter `alpha`. While `LogisticRegression` (LR) is optimized on the regularization parameter `C`, furthermore, as this model utilizes the gradient descent method to update the weight vectors, we set the maximum number of iterations (`max_iter=100,000`) [9]. We utilize `lbfgs` solver (Byrd et al., 1995) for its robustness[10]. Thirdly we employ the `LinearSVC` function with the maximum number of iterations as `max_iter=100,000` with setting the seed[11].

4.4 Halving Grid Search with Cross Validation

Tuning the parameters for six vectorizer-classifier pipelines is an expensive task, considering the fact that the training dataset is relatively large, with 137,930 training samples; furthermore, we intend to perform a grid search on several combinations. To give a perspective, these six pipelines perform a grid search on 15 combinations of parameters.

Here, a conventional grid search technique might not be a time-optimal solution. Therefore, we approach this problem by utilizing the Halving Grid Search method, based on the Successive Halving (SH) technique ((Jamieson and Talwalkar, 2016) and (Li et al., 2016). This method is based on the tournament of candidates. Initially, all the candidate models are trained on a smaller dataset; out of these, the best-performing candidates survive and are further trained on a larger dataset iteratively, and the parameter space of the candidates keeps shrinking. Finally, the best-performing model is found.

[8]These functions are available from `naive_bayes`, `linear_model` and `svm` modules accordingly.

[9]Based on our primary experiments we realized that this model takes the considerably higher number of iterations before it converges.

[10]`http://users.iems.northwestern.edu/~nocedal/lbfgsb.html`

[11]In order to reproduce the same results, we set the seed variable as `random_state=1729` for `LogisticRegression` and `LinearSVC` functions.

We have used the `HalvingGridSearchCV` function available from `sklearn.model_selection` module; this function has parameters such as `factor` and `min_resource`. According to the documentation, the parameter `factor` is the rate at which the number of training samples grows or the rate of reduction for the candidates. Here `min_resource` refers to the number of available samples in the first iteration.

In our experiments, we have chosen the value for `min_resource = 34,480` and `factor = 2` because in 3^{rd} iteration value of **n_resource** will grow to be **137,920** training samples, while the total training samples is **137,930**, so in the last iteration, the model will be able to perform the evaluation on almost entire training dataset to find the best-performing candidate out of the four models.

We have chosen to five-fold cross-validation for better regularization. Finally, we re-train the best-performing model obtained on the entire data set with the corresponding parameters for further inference in the next stage; here, we use the `CalibratedClassifierCV` function, as the `LinearSVC` does not provide probabilities associated with each class in the prediction, we utilized the `CalibratedClassifierCV` function to obtain the probability associated with each class for the LIME analysis.

4.5 DistilBERT Training

DistilBERT model is faster to train than base-BERT, but still, it is time expensive computation even on TPU v2, so we first train these models on the 20% subset of training data. This 20% subset is further divided into train-validation sets with a split of 75%-25% to compute the epoch-wise loss and accuracy. Pre-trained DistilBERT is used to build a classifier, using TensorFlow interface (Abadi et al., 2015)and Keras (Chollet et al., 2015) package on the Google Colab environment with python3 distribution. We list the steps involved in this process here,

1. We installed the `tensorflow`, and `tensorflow-text` version `2.9.0` in the

python3 environment[12]. The pre-trained DistilBERT model is used to build the classifier using the corresponding `preprocesser` and the `encoder` on TensorFlow Hub.

2. Next, the `pooled_output` from encoded text is passed through a `Dropout` layer with a rate of 0.1 to control overfitting. The last layer is a `Dense` layer with a neuron that uses a `sigmoid` activation function, which has output 0 or 1.

3. We trained the models for `epochs=10` with `batch_size=256`. We trained the models by using adam optimizer (Kingma and Ba, 2014) with 0.001 learning rate and `binary_crossentropy` loss. To minimize the training time, the training was performed on a subset (20%) of the original training data, which was further split into training and validation sets with a split of (75%-25%). Finally, we evaluated the performance of this model on the test set.

4.6 LIME Inference

The misclassified samples identified by the tuned Linear SVM model using are utilized for the LIME inference. We utilized `LimeTextExplainer` module from the `lime_text` and `lime` modules.

5 Results

5.1 Vectorizer-Classifier Pipeline

The classification report for the pipelines created using Count Vectorizer is represented in table [7]; here, all three classifiers seem to perform equally well. However, Linear SVM has slightly higher precision for the negative class.

Table [8] presents the classification report for pipelines created using TF-IDF Vectorizer. Here Linear SVM and Logistic Regression seem to perform equally well, but Linear SVM has slightly higher precision for the negative class.

5.2 DistilBERT Results

The evaluation result DistilBERT is displayed in table [9]. Next, the evaluation of DistilBERT on the test set is documented in table [10].

5.3 Inference using LIME Analysis

This section presents the confusion matrix for the final SVM model in figure [4], where labels 0 and 1 refer to the negative and positive classes, respectively.

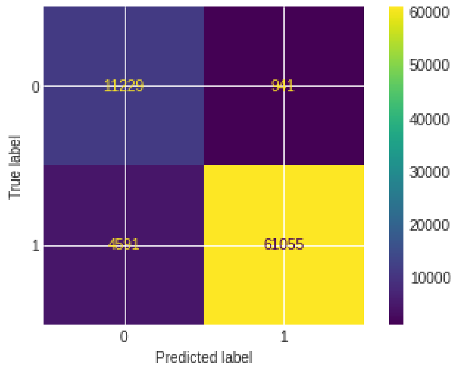

Figure 4: Confusion Matrix for the Final SVM model

We provide an example of a LIME text explainer for a false negative sample in figures [5] and a false positive sample in figure [7], where one can observe the prediction probabilities for each class. Furthermore, the vertical graph presents the contribution of the top ten words regarding class probabilities; these words are also highlighted in the adjacent review text. The blue highlighted words correspond to the probabilities of a negative sentiment of the review, while the words in orange represent probabilities corresponding to the positive sentiment predictions.

6 Discussion

In the first part of the project, we experimented with different classifiers with pipelines created with Count Vectorizer and TfidfVectorizer. These vectorizers are based on the *bag-of-word* idea for the corpus, where the context in the sentence is not considered. However, one can improve the understanding of context by incorporating the **n-gram** mechanism. In our experiments, with both the vectorizers, using a combination of unigram, bigram (and trigram) instead of just the unigram has improved the model's performance, which can be observed in tables [7] and [8].

According to Jurafsky and Martin (p.86, 2023), the Logistic Regression model performs better than the Multinomial NB model in larger datasets. The results in the table [7] and [8] demonstrate it to be valid to a certain extent. Although the overall accuracy for Multinomial NB and Logistic Regression is almost similar, it is evident that the Precision (and F1-score) for the *negative class* is 2% to 3% higher for the Logistic Regression model. As the Precision for the negative class is a ratio of True

[12]Ancilliary packages such as `tensorflow_text` and `tensorflow_hub` are installed.

Classifier	Class	Precision	Recall	F1-score	Accuracy	Tuned Parameters
Multinomial NB	negative	0.67	0.91	0.78		n_gram: (1,3)
	positive	0.98	0.92	0.95	0.92	alpha: 0.8
Logistic Regression	negative	0.69	0.93	0.79		n_gram: (1,3)
	positive	0.99	0.92	0.95	0.92	C:1
Linear SVM	negative	0.70	0.92	0.79		n_gram: (1,3)
	positive	0.99	0.93	0.95	0.93	C: 0.1

Table 7: Classification Report for the experiment with Count Vectorizer Pipeline + Classifier Pipelines, with Halving Grid Search CV

Classifier	Class	Precision	Recall	F1-score	Accuracy	Tuned Parameters
Multinomial NB	negative	0.65	0.93	0.76		n_gram: (1,3)
	positive	0.99	0.91	0.94	0.91	alpha: 0.4
Logistic Regression	negative	0.70	0.93	0.80		n_gram: (1,2)
	positive	0.99	0.93	0.95	0.93	C: 10
Linear SVM	negative	**0.71**	0.93	**0.80**		n_gram: (1,2)
	positive	**0.99**	0.93	**0.96**	0.93	C : 1

Table 8: Classification Report for the experiment with TF-IDF Vectorizer + Classifier Pipelines, with Halving Grid Search CV

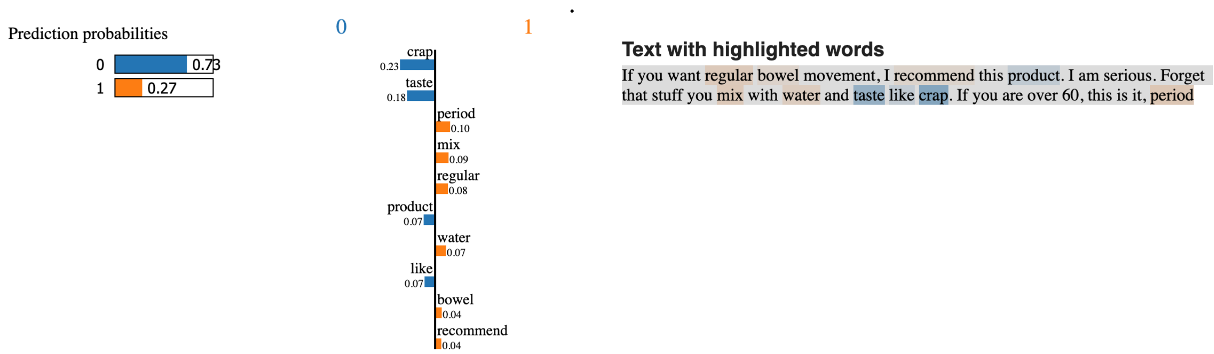

Figure 5: False Negative example, LIME Text Explainer Visualization

	productID	userID	profileName	helpfulness	score	time	summary	text
184729	B000FDMLUE	A2A5GOTBTDHJPF	William Clay	3/5	5	1196208000	I love this stuff . I eat it every day and goo...	If you want regular bowel movement, I recommen...

Figure 6: False Negative Example, Row Entry from the Test Set

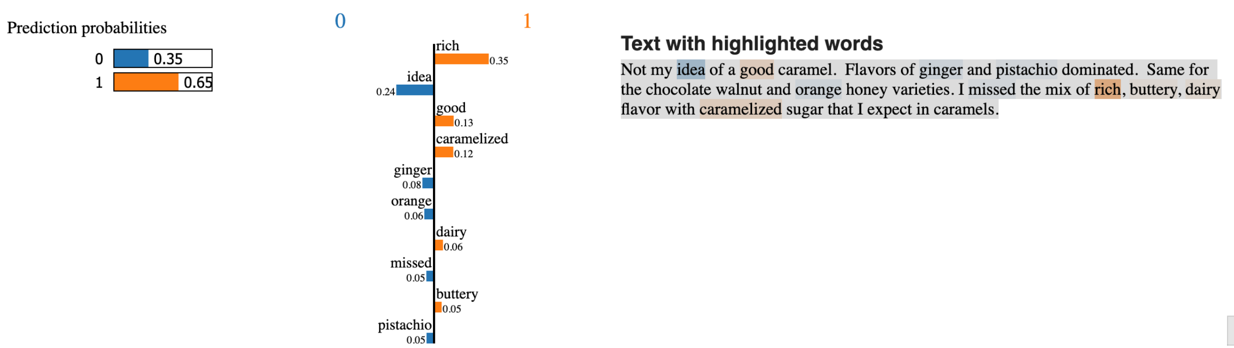

Figure 7: False Positive Example, LIME Text Explainer Visualization

	productID	userID	profileName	helpfulness	score	time	summary	text
829	B000UWSQT0	A1LO4ME566EKLC	Show Me	0/1	1	1227916800	not for traditional caramel lovers	Not my idea of a good caramel. Flavors of gin...

Figure 8: False Positive Example, Row Entry from the Test Set

Model	Training Loss	Validation Loss	Training Accuracy	Validation Accuracy
DistilBERT	0.3592	0.3552	0.8463	0.8507

Table 9: Evaluation of DistilBERT trained on the 20% of data, results are based on the last five epochs out of ten

Model	Class	Precision	Recall	F1-score	Accuracy
DistilBERT	negative	**0.51**	0.86	0.64	
	positive	**0.97**	0.84	0.90	**0.85**

Table 10: Classification Report for DistilBERT, on the test set (without tuning)

Negative to the sum of True Negative and False Negative, it can be inferred that the Logistic Regression model is better at minimizing false negatives. The linear SVM model performs slightly better when comparing it with Logistic Regression for both pipelines; it can be observed that Precision for the negative class for linear SVM is 1% higher than Logistic Regression for both cases. One major limitation of the prediction was relatively poor Precision for the negative reviews across all the experiments caused by higher false negatives. It is important to note that the test set has a class imbalance, with a *positive-to-negative ratio* of approximately $5:1$.

In the next part of the project, we trained the DistilBERT model on the subset of the original training set (20%) for ten epochs. As per the evaluation metrics in tables [9] and [10], training and validation accuracies were 0.84 and 0.85 respectively, while the test accuracy was 0.85. However, the F1-score for the negative class was poor (around 0.64), caused by 0.51 precision. The relatively poor evaluation results revealed that the pre-trained DistilBERT model, without the parameter tuning, might not be an ideal choice for sentiment classification. Furthermore, we did not utilize the entire training set for the downstream task, another limitation of this study.

In the third part of the project, we performed LIME analysis for the False Negatives and False Positive samples identified using the tuned Linear SVM model at the end of section [4.4]. Firstly in the figures [5] and [6], an example of a false negative sample is demonstrated; it can be observed that the words such as *crap* and *taste* overpowers the positive sentiment in the text, resulting in the misclassification. This particular review has a positive summary, and out of five users, three found this helpful, leading us to believe that *summary* of the review could be valuable in the disambiguation of the edge cases. Next, the figures [7] and [8]) demonstrate the first False Positive sample, where the model was unable to interpret *Not my idea of a good caramel* as a sentence with negative sentiment, and assigned higher probabilities to words *good*, *rich*, and *caramelized* which classified to be a positive review.

7 Conclusion

In this project, we performed a Binary Sentiment Analysis for the Amazon Fine Food Review dataset. In the first part of the project, we experimented with different vectorization techniques and machine-learning classifiers to find a model that delivered the highest performance based on the model evaluation metrics. We used the Halving Grid Search technique with five-fold cross-validation to find the optimum parameters in relatively less time. We observed the Linear SVM model with the TF-IDF vectorization model yielded 93% accuracy. However, the significant limitation of this method was the lower Precision (0.71) for the negative class (higher false negatives).

Next, we experimented with DistilBERT for the sentiment classification task and achieved 85% accuracy. However, the major limitation of this work was the lack of fine-tuning for the DistilBERT model; due to this reason, the Precision for the negative class is relatively poor (0.51). In similar work, the BERT model achieved an accuracy of 79%. However, it was a five-class classification problem (Zhao and Sun, 2022). In future works, we propose to include optimizer scheduling using the `AdamW` (Loshchilov and Hutter, 2017) technique, which could efficiently slow down the learning processing for the initial epochs, leveraging the pre-trained model's knowledge[13].

Lastly, based on the final Linear SVM model, we performed the LIME technique on the six samples from the test set identified from the misclassified predictions. Several words incorrectly predicted the sentiment of the review *text*, so we propose to use the review *summary* as *prior* class probabilities using the Multinomial NB classifier. Additionally, the review's *helpfulness* could be incorporated as utilized in the works of (Yang et al., 2015). However, we need to conduct more work to evaluate the effectiveness of these changes.

[13] https://www.tensorflow.org/text/tutorials/classify_text_with_bert

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Thomas Bayes. 1763. Lii. an essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfr s. *Philosophical transactions of the Royal Society of London*, (53):370–418.

Christopher M Bishop and Nasser M Nasrabadi. 2006. *Pattern recognition and machine learning*, volume 4. Springer.

Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. 1995. A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing*, 16(5):1190–1208.

François Chollet et al. 2015. Keras. `https://keras.io`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature*, 585(7825):357–362.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network.

J. D. Hunter. 2007. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95.

Kevin Jamieson and Ameet Talwalkar. 2016. Non-stochastic best arm identification and hyperparameter optimization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 240–248, Cadiz, Spain. PMLR.

Dan Jurafsky and James H Martin. 2023. *Speech and Language Processing*, 3rd edition.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. 2017. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2016. Hyperband: A novel bandit-based approach to hyperparameter optimization.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization.

Julian John McAuley and Jure Leskovec. 2013. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *Proceedings of the 22nd international conference on World Wide Web*, pages 897–908.

The pandas development team. 2020. pandas-dev/pandas: Pandas.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?": Explaining the predictions of any classifier.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper, and lighter. *arXiv preprint arXiv:1910.01108*.

Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962*.

Vladimir Vapnik. 1999. *The nature of statistical learning theory*. Springer science & business media.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Wes McKinney. 2010. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*, pages 56 – 61.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Yinfei Yang, Yaowei Yan, Minghui Qiu, and Forrest Bao. 2015. Semantic analysis and helpfulness prediction of text for online product reviews. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 38–44.

Xinyue Zhao and Yuandong Sun. 2022. Amazon fine food reviews with bert model. *Procedia Computer Science*, 208:401–406.