DEEP TRANSFORMER for INTENT CLASSIFIER

Connor Lee
BCL9779@nyu.edu
New York University
USA

ABSTRACT

With the large-scale availability of data and the increasing popularity of online service platforms, more and more users are reading reviews of listed items before deciding to make a purchase. To help users make informed decisions, we employ machine learning techniques for sentiment analysis and intent mining to analyze product reviews. In this work, we used two classical machine learning algorithms: the Multinomial Naive Bayes classifier and the XGBoost classifier, and two deep learning algorithms: a fine-tuned BERT classifier and zero-shot inference by utilizing a pre-trained LLM to classify the intent of reviews as positive or negative. These models are applied to the open public Amazon book review data for intent mining. From our extensive experiments, we observed that the fine-tuned BERT model outperforms the other three models, with an F1 score at least 25% higher than the second-highest model (XGBoost). The zero-shot classifier demonstrated better performance than the Naive Bayes classifier, highlighting the superiority of LLMs without fine-tuning. We provide a discussion on the pros and cons of each classifier and offer visualization analysis through some examples using deep learning visualization tools: Transformer Interpreter and TensorBoard Projector.

KEYWORDS

BERT language model, Classification, Natural language processing, zero-shot inference, Multinomial Naive Bayes classifier and XGBoost classifier

ACM Reference Format:
Connor Lee . 2024. DEEP TRANSFORMER for INTENT CLASSIFIER . In *Proceedings of (KDD).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

As online retailers attempt to reign in the digital market by satisfying customer needs, they rely on customer reviews and communication channels for a rating of their products. These reviews not only help retailers but also for customers to gauge the quality of their products. On the customer side, reading reviews about an item gives users an idea of the item's quality. On the retailer side, by analyzing the reviews, businesses can survey the customers' opinions and provide corresponding strategies. By leveraging these public product reviews, companies can build recommendation systems or better-targeted marketing campaigns. These recommendation systems can have distinct classifications: negative, positive, or neutral. This process is called sentiment analysis or intent mining in the machine learning domain [9]. Sentiment analysis uses techniques like natural language processing (NLP) to identify subjective information. In this paper, we create a TF-IDF (term frequency-inverse document frequency) feature vector [4] for traditional machine learning methods and an embedding vector for deep learning models to classify book reviews as positive or negative. The main objective of our work is to present a comparative study on different classifiers based on confusion matrix and accuracy while analyzing the features correlated to sentiment categories to identify the most efficient model in the NLP domain. The organization of our paper is as follows: (1) we first give a brief introduction of feature engineering related to NLP texts that are applied to our models. (2) Next, we list our four machine-learning models for sentiment analysis. (3) Then the detailed performance evaluation metrics and experiments output are presented. (4) Afterwards, we provide further analysis to explain our model results. (5) Finally, we conclude with our key learnings.

KDD, August 25–29, 2024, Barcelona

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 RELATED WORK

Sentiment analysis, a natural language processing task, involves determining the sentiment or emotion expressed in text data. Various machine learning models, including classical ones like Multinomial Naive Bayes classifier and XGBoost classifier, alongside modern approaches like BERT and prompt-based inference, have shown remarkable success in sentiment analysis tasks. Classical models such as Multinomial Naive Bayes classifier leverage probabilistic principles and feature engineering to classify text into sentiment categories. On the other hand, gradient boosting algorithms like XGBoost classifier utilize an ensemble of decision trees to capture complex relationships between words and sentiments. Modern approaches like BERT (Bidirectional Encoder Representations from Transformers) [3] leverage pre-trained transformer models to generate contextualized representations of text, enabling them to capture nuanced sentiment information effectively. Additionally, prompt-based inference techniques, which involve prompting a pre-trained language model with specific instructions to perform sentiment analysis, have emerged as a powerful approach for sentiment analysis tasks.

2.1 Feature Engineering

2.1.1 TF-IDF feature vector. TF-IDF stands for term frequency-inverse document frequency and quantifies the importance of terms in a document within a corpus. It comprises two components: term frequency (TF) and inverse document frequency (IDF).

Term frequency measures the frequency of a term relative to the total terms in a document. IDF measures how common a term is across the corpus. IDF is calculated as follows, where t is the term, d is the document, and D is the corpus:

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D)$$

In our data, review texts are treated as documents within the corpus. For classical models like Multinomial Naive Bayes and XGBoost, we use TF-IDF as input features.

2.1.2 BERT Pretrained Embedding vector . BERT is a deep bidirectional pre-trained encoder model [3]. BERT model is trained on the dataset named as BookCorpus, which contains 800M words, and the English Wikipedia, which holds 2500M words. Using BERT, we can extract features, namely word and sentence embedding vectors, from text data. These embeddings are useful for keyword search, semantic search, and information retrieval. They are also used as high-quality feature inputs to downstream tasks: like clustering or semantic search. In our fine-tuned BERT classification model, we used embedding of the sentence-BERT model [7] as initial input features. It maps sentences paragraphs to a 384-dimensional dense vector space for multi-class classification.

2.2 ML Models

We provide the details of the four models to be applied for: multinomial naive bayes classifier, xgboost classifier, fine-tuning BERT as binary classifier and prompt classifier.

2.3 Multinomial Naive Bayes classifier

The multinomial naive bayes classifier is a probabilistic model used for text classification tasks. It is based on the naive bayes assumption that features are conditionally independent given the class label. For text data, it models the word count vectors using a multinomial distribution. The probability of a document d belonging to class c is calculated as:

$$P(c|\mathbf{d}) = \frac{P(c) \prod_{i=1}^{n} P(w_i|c)^{x_i}}{\sum_{c' \in C} P(c') \prod_{i=1}^{n} P(w_i|c')^{x_i}}$$

where $P(c)$ is the prior probability of class c, $P(w_i|c)$ is the conditional probability of word w_i given class c, x_i is the count of word w_i in the document $\mathbf{d}$, and the denominator is a normalization factor over all possible classes C. (see Figure 1)

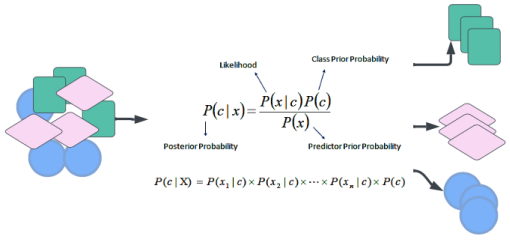

Figure 1: Naive Bayes classifier

2.4 Xgboost classifier

XGBoost is a tree-based sequential ensemble machine learning algorithm [2]. It builds a strong classifier by sequentially adding weak classifiers. Initially, a tree model is built from the training data, followed by subsequent models targeting to correct prediction errors from previous ones. This process continues until the error threshold is met or the maximum number of models is reached.

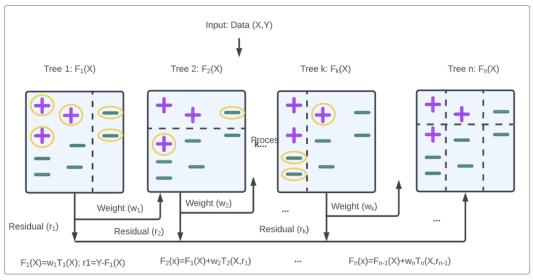

Figure 2: Gradient Boost Tree Model

Figure 2 illustrates the training process of the XGBoost classifier. Initially, the data is trained by the first classifier (F1(x)), which incorrectly predicts some data points. Subsequent classifiers are trained to correct these errors until all data points are predicted correctly or a maximum number of models is reached. The optimal number of models is determined through hyperparameter tuning.

The objective function of the XGBoost model on the t'th iteration is given by:

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

, where $\mathcal{L}(t)$ is the cost function for n training data points, the first term represents the error generated by the previous classifier and the new fitted function, and the second term is a regularizer used to control model complexity.

2.5 Pretrained Language Modeling (BERT)

Recently, transformer-based pretrained language models have demonstrated state-of-the-art performance in natural language processing (NLP). For example, BERT (Bidirectional Encoder Representations from Transformers) [3, 14] has achieved outstanding results through masked self-supervised pre-training and transformer-based modeling. Pretraining with masked language models and transfer learning [10] has significantly improved many natural language understanding (NLU) tasks, such as text classification and natural language inference.

BERT is trained using self-supervised learning, where the model learns the context of a sentence by predicting masked tokens. This method allows BERT to understand word meanings based on context, achieving performance comparable to supervised learning without the need for extensive human labeling. Additionally, BERT's next sentence prediction task enables it to learn relationships between sentences, which is useful for tasks like question answering and natural language inference. In our approach, we use BERT for binary sentiment classification.

2.6 Zero-Shot Inference with LLM

Pre-trained language models, such as GPT and LLMA etc, can be utilized for zero-shot inference by providing them with carefully crafted prompts that guide the model to perform the desired task without updating its weights. In zero-shot inference, the model predicts the class y for a prompt $\mathbf{x}$ by optimizing:

$$\hat{y} = \arg\max_{y} P(y|\mathbf{x};\theta)$$

where θ represents the fixed parameters of the pre-trained model. This method is particularly useful when labeled data is limited, as it avoids the need for extensive fine-tuning. It is generally faster and more computationally efficient since the model weights remain unchanged during inference. The performance of zero-shot inference heavily depends on the quality of the prompt and the model's ability to generalize from the prompt to the actual task. Although fine-tuning usually outperforms zero-shot inference when ample labeled data is available, zero-shot inference can be competitive or even superior in scenarios with limited labeled data or simpler tasks.

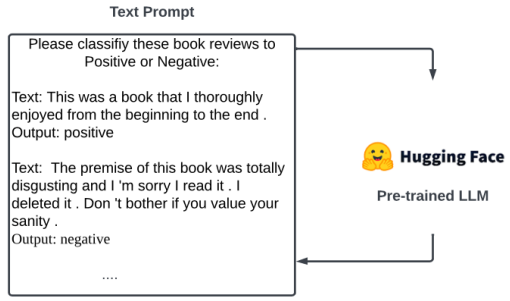

Figure 3: Prompt-based Classifier

3 MODEL APPROACHES

3.1 Multinomial Naive Bayes classifier

We employed the `sklearn.naive_bayes.MultinomialNB` class from the scikit-learn library [11] to implement a Multinomial Naive Bayes classifier for our text classification task. To optimize the model's performance, we performed hyperparameter tuning by leveraging the `GridSearchCV` functionality from scikit-learn. This systematic approach allowed us to search through a predefined grid of hyperparameter values and identify the optimal value of the α parameter, which controls the smoothing factor for the Naive Bayes estimator.

3.2 XGBoost Classifier

We installed the XGBoost library [2] and employed the `GridSearchCV` functionality to perform hyperparameter tuning for the XGBoost classifier. The objective function for the XGBoost model was set to `'binary:logistic'`, as we were addressing a binary classification task. The `GridSearchCV` approach allowed us to systematically search through a predefined grid of hyperparameter values and identify the optimal combination of hyperparameters that maximized the performance of the XGBoost model on our dataset.

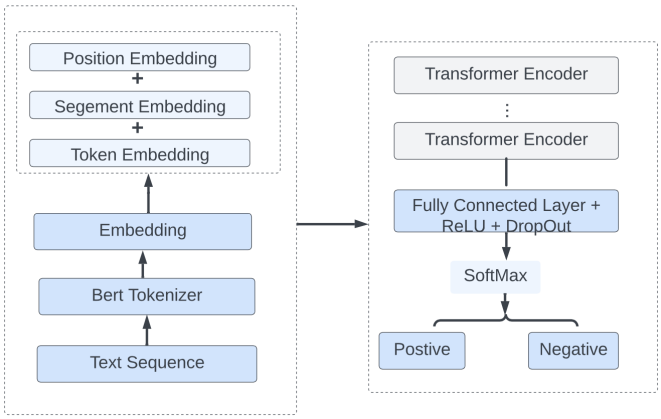

Figure 4: Bert Fine-tuning architecture

3.3 BERT Fine Tuning Classifier

To fine-tune BERT for binary classification, we start with a pre-trained BERT model, which consists of 12 transformer blocks, each with 12 self-attention heads and a hidden size of 768. The model processes input sequences up to 512 tokens, where each sequence begins with the [CLS] token, representing the entire sequence, and uses the [SEP] token to separate segments. We take the final hidden state of the [CLS] token as the aggregate representation of the sequence. This representation is then fed into a binary classification layer, implemented as a dense layer followed by a softmax activation function to predict the probability of each class. Specifically, the probability of class c is given by $p(c|h) = \text{softmax}(Wh)$, where h is the hidden state of the [CLS] token and W is a task-specific parameter matrix. During fine-tuning, we optimize all model parameters, including the parameters of the pretrained BERT and the newly added classification layer, using the ADAM optimizer [5] and binary cross-entropy loss. (see Figure 4)

3.4 Prompt-based Classifier

Pre-trained Language Models (LLMs) have demonstrated remarkable capabilities in various natural language processing tasks, including zero-shot inference. In our experiments, we utilize zero-shot inference techniques to enable the model to predict outputs for tasks or domains on which it has not been explicitly trained.

4 EXPERIMENTS

We evaluated the performance of Naive Bayes Classifier, Xgboost classifier, prompt-based classifier and BERT fine-tuned classifier on an amazon open public book review dataset [1, 12]. Through the experiments of text classification on different models, we aim to find the answers on these key questions:

- How is each model's performance and what can be learned from the outputs?
- Is deep learning model better than classic model?
- Is prompt inference better than fine-tuning pre-trained LLM?

Table 1: Statistics of the dataset

Rating	Population	Sample
5.0	789,450	5885
4.0	259,658	1970
3.0	115,784	883
1.0	90,821	726
2.0	70,178	536

4.1 Data Statistics

The amazon book review dataset [1, 1, 8] contain reviews about 3m user on 212404 unique books. Each review has 3 columns: rating (0-5), reviewText and Summary. These three features contain the most relevant information to the rating classes. Due to the large dataset size, we took the subset of the raw dataset (10K) by using random sample() method. The distribution of rating score for sample and population is shown in Table 1 and Figure 5.

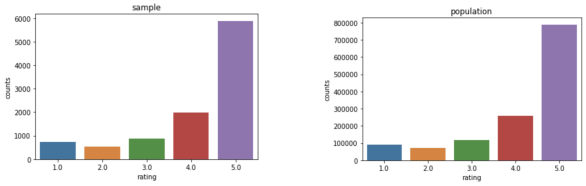

Figure 5: Sample vs. Population Rating Distributions

4.2 Training and Validation

We split the dataset into training and testing sets with a 10:90 ratio for the entire 10K dataset after preprocessing (details in the Appendix). We performed cross-validation with k-fold = 3 and evaluated the model performances on the test set.

4.3 Evaluation metrics

We evaluated the model performance using confusion matrix AUC-ROC and AUC-PR (precision recall) metrics to provide a comprehensive assessment of their classification capabilities.(see Figure 6). Compared to the machine learning classification metrics like "accuracy" give less useful information, which is simply the difference between correct predictions divided by the total number of predictions. Confusion matrix is useful for deep analysis of model performance since it directly shows true positives, false positives, true negatives and false negatives. AUC-ROC and AUC-PR also help interpret the model performance for unbalance dataset.

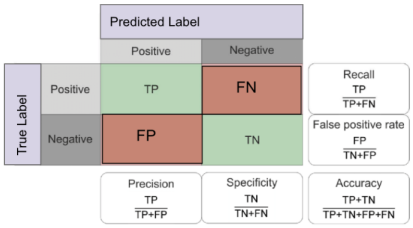

Figure 6: Confusion Matrix

4.4 Results

4.4.1 Multinomial Naive Bayes classifie . We utilized the sklearn packageand grid search to perform hyper-parameter tuning . The grid search is: param_grid = {'alphas': np.arange(1, 10, 0.1)} } and the best *alpha* = 1. (Table 2 , Figure 7)

Table 2: Precision, Recall and F1 score (threshold=0.5)

Class	Precision	Recall	F1-score
0	0.8640	1.0000	0.9270
1	0.0000	0.0000	0.0000

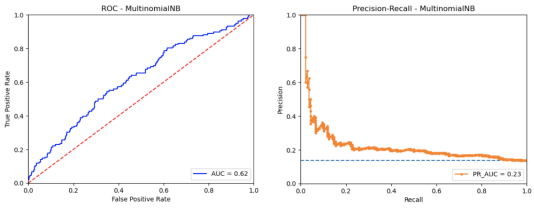

Figure 7: Naive Bayes Classifier AUC-ROC and AUC-PR

4.4.2 XGBClassifier .

`XGBClassifier( learning_rate=0.01, max_delta_step=0, max_depth=4, min_child_weight=5, n_estimators=200, n_jobs=1, nthread=1, objective='binary:logistic', random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1.0)` (Table 3 , Figure 8)

Table 3: Precision, Recall and F1 score (threshold=0.5)

Class	Precision	Recall	F1-score
0	0.9165	0.9780	0.9462
1	0.7564	0.4338	0.5514

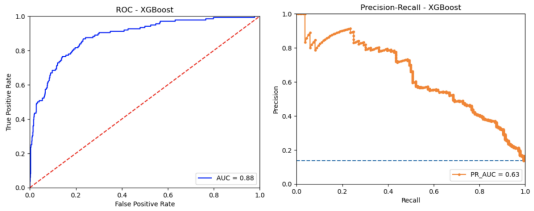

Figure 8: XGB Classifier AUC-ROC and AUC-PR

4.4.3 BertClassifier . In our implementation,we used the transformer library of Hugging Face [15]. We created a BertClassifier class with a BERT model to extract the last hidden layer of the [CLS] token and a single-hidden-layer feed-forward neural network as our classifier. (Table 4 , Figure 9)

Table 4: Precision, Recall and F1 score (threshold=0.5)

Class	Precision	Recall	F1-score
0	0.9484	0.9572	0.9528
1	0.7109	0.6691	0.6894

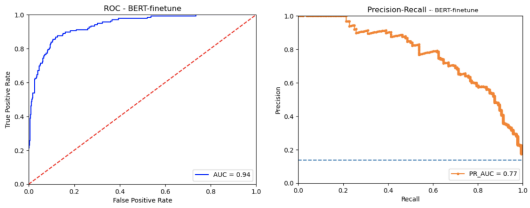

Figure 9: BERT Classifier AUC-ROC and AUC-PR

4.4.4 Zero-shot Inference. We use the huggingface pre-trained model: $falcon-7b-instruct$ and provide prompt template to ask model to generate the sentiment analysis output, for instance:
Prompt: f""" Classify the text into negative or positive, please don't classify into neutral.
Text: {reviews[review]}
Sentiment: """
(Table 5)

Table 5: Precision, Recall and F1 score (threshold=0.5)

Class	Precision	Recall	F1-score
0	0.9390	0.7040	0.8047
1	0.2704	0.7059	0.3910

4.5 Summary

To summarize, from the confusion matrix , AUC-ROC and AUC-PR output, we can see that fine-tuned BERT model outperforms classical Naive Bayesian classifier and xgboost classifier. Zero-shot inference performs better than Naive Bayesian classifier. The result validates the limitation of Naive Bayesian classifier and xgboost classifier which cannot handle large numbers of sparse and high-dimensional features. The BERT-pretrained model was pre-trained on a big corpus and the fine-tuned deep learning architecture can learn non-linearity and high-dimensional vectors very well without overfitting. In real-world scenarios, for intent classification in NLP, the TF-IDF feature vector is sparse and high-dimensional, leading to overfitting for Naive Bayesian classifier. Although Zero-shot inference performs not better than trained model, it is faster and more computationally efficient during inference.

5 FURTHER ANALYSIS

5.1 Transformer Interpreter

To further support the deep learning model's performance, we ran a transformer interpreter [6], which is a model explainability tool designed to work exclusively with the transformers package. It

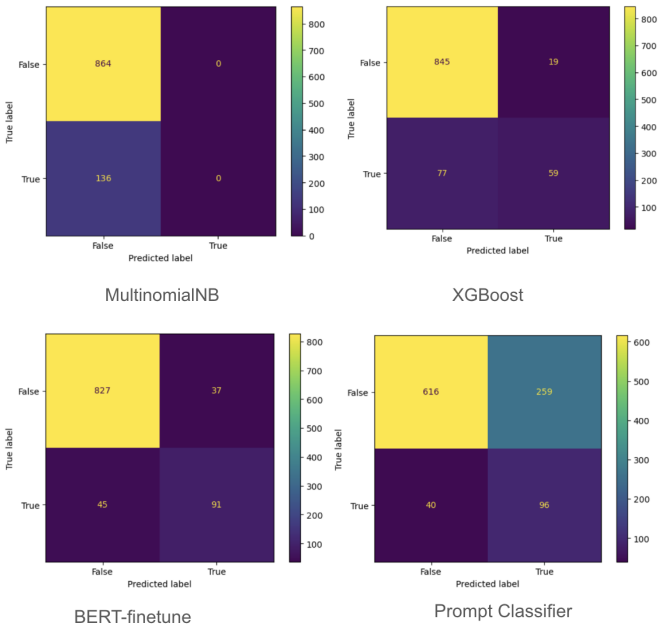

Figure 10: Confusion matrix

produces a dictionary of word attributions mapping labels to a list of tuples for each word and its attribution score. These explainer attribution scores are highlighted and display the visualization in-line with weights.

In the examples below (Figure 11), the review text with the true label is displayed and corresponding attribute with higher-weight related to target variable is highlighted. For instance, for the first review text: "The premise of this book was totally disgusting and I 'm sorry I read it . I deleted it . Don 't bother if you value your sanity". "disgusting" has more weights related to our target rating value (=1, very low raing). It is highlighted to reflect its corresponding low rating and the summary "Disgusting premise" also validates "disgusting" consistent with the content from our ground-truth data. The rest of the examples are the same as this interpretation.

Positive Negative

Rating	ReviewText	Summary
1	The premise of this book was totally disgusting and I 'm sorry I read it . I deleted it . Don 't bother if you value your sanity .	Disgusting premise
2	This is not a good ghost story , nor is it a romance . To me it was a poorly written Har le quin romance with a lot of sm ut thrown in . The author doesn't appear to know what genre she's trying to capture . There were numerous typos which I found annoying . The author also incorporated Elvis ' name , but did it in a way that I found disrespectful as well as inaccurate .	Lousy Read
4	Way too short . It is a good, fun story . It pulls you in and keeps you reading . It 's not that it was really too short , just that it was so fun . I tore through it too fast . It 's well written with very few errors. What few there were did not stop the flow.	Good...but ;)
5	This was a book that I thoroughly enjoyed from the beginning to the end . The story line was full of details and kept me involved and entertained . I loved the characters and their quirks . The story was smoothly written . Secret agents and art . Who would have believed how much fun and add a twist of fate and love . I will read this author again . Enjoy this book and you won 't be disappointed .	Wow and wonderful read with a twist

Figure 11: Transformer Interpreter

5.2 Embedding Visualization

Furthermore, to have a good understanding of the review text, we generate the embeddings from fine-tuneded BERT model and visualize them in tensorboard [13] shown from Figure 12. Figure 12 shows each data point is displayed as an embedding vector learned by model and represents corresponding review text. If the mouse hovers over one particular data point, it also shows its content (i.e. raw review text), as you can see from Figure 12. We list one review text with high rating and its nearest neighbor (shown in Figure 12); same for one review text with low rating and its nearest neighbor (shown in Figure 13). These two examples demonstrate the embedding vectors learned from our deep learning models can capture the content and sentiment very well.

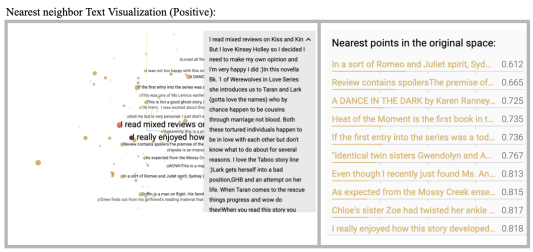

Figure 12: The neighbors of positive review text . On the right panel, it shows the top closest texts near to the given positive review text. The last text has the highest closest score. We can tell the texts close to each other have similar positive sentiments.

Figure 13: The neighbor of negative review text . On the right panel, it shows the top closest texts near to the given negative review text. The last text has the highest closest score. We can tell the texts close to each other have similar negative sentiments.

6 CONCLUSION AND FUTURE WORK

In this study, we investigated the applicability of classical machine learning models and deep learning models, including naive bayes classifier, xgboost classifier, fine-tuned BERT and pre-trained large language models (LLMs), for book review sentiment classification. We evaluated these models through cross-validation and testing. The results show that the fine-tuned BERT model outperforms the other models in terms of classification accuracy, while zero-shot inference demonstrates the potential of LLMs. In future work, we will design few-shot prompt templates and apply the latest pre-trained LLMs to further explore text classification in real-world applications.

REFERENCES

[1] Mohamed Bakhet. 2022. Amazon Books Reviews. https://www.kaggle.com/datasets/mohamedbakhet/amazon-books-reviews Accessed: 2024-05-26.

[2] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[3] 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* (2018).

[4] Lukáš Havrlant and Vladik Kreinovich. 2017. A simple probabilistic explanation of term frequency-inverse document frequency (tf-idf) heuristic (and variations motivated by this explanation). *International Journal of General Systems* 46, 1 (2017), 27–36.

[5] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[6] Vasudev Lal, Arden Ma, Estelle Aflalo, Phillip Howard, Ana Simoes, Daniel Korat, Oren Pereg, Gadi Singer, and Moshe Wasserblat. 2021. InterpreT: An interactive visualization tool for interpreting transformers. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*. 135–142.

[7] Eunchan Lee, Changhyeon Lee, and Sangtae Ahn. 2022. Comparative study of multiclass text classification in research proposals using pretrained language models. *Applied Sciences* 12, 9 (2022), 4522.

[8] Julian McAuley. n.d.. Amazon Product Data. https://cseweb.ucsd.edu/~jmcauley/datasets/amazon/links.html Accessed: 2024-05-26.

[9] A Mounika and S Saraswathi. 2019. Classification of book reviews based on sentiment analysis: A survey. *Int. J. Res. Anal. Rev* 6 (2019), 150–155.

[10] Emilio Soria Olivas, Jos David Mart Guerrero, Marcelino Martinez-Sober, Jose Rafael Magdalena-Benedito, L Serrano, et al. 2009. *Handbook of research on machine learning applications and trends: Algorithms, methods, and techniques: Algorithms, methods, and techniques*. IGI global.

[11] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine Learn-ing in Python, Journal of Machine Learning Re-search, 12. (2011).

[12] Jérémie Rappaz, Julian McAuley, and Karl Aberer. 2021. Recommendation on live-streaming platforms: Dynamic availability and repeat consumption. In *Proceedings of the 15th ACM Conference on Recommender Systems*. 390–399.

[13] Daniel Smilkov, Nikhil Thorat, Charles Nicholson, Emily Reif, Fernanda B Viégas, and Martin Wattenberg. 2016. Embedding projector: Interactive visualization and interpretation of embeddings. *arXiv preprint arXiv:1611.05469* (2016).

[14] 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 (2017).

[15] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771* (2019).

REPRODUCIBILITY

DATA SOURCE

Amazon Book Reviews dataset can be downloaded from [1]. It is composed of two sub-datasets: reviews and books' profile.

PRE-PROCESSING

- After download the datasets, merge reviews and books' profile dataset using 'title' as key.
- Perform: drop duplicates, remove na() records
- the reviews having neutral rating (i.e. 3) are removed.
- Sample 10k records as our dataset
- The review text is chosen as predict variable, and the text is pre-processed by:
  Lowercase the review ; Remove special characters; stemming, lematizing; Remove stop words except "not" and "can"; Remove trailing whitespace
- create target variable for records having rating=1 and 2 as negative , rating=4 and 5 as positive.

DATA SPLIT

The dataset is split into train and test sets with a 90:10 ratio using `train_test_split` from scikit-learn. We use StratifiedKFold=3 for cross validation.

FEATURE ENGINEERING

For review text, we use `sklearn` to create TF-IDF vector $X\_train\_tfidf = tf\_idf.fit\_transform(X\_train\_preprocessed)$ and $X\_val\_tfidf = tf\_idf.transform(X\_val\_preprocessed)$

MODELS

- **Multinomial Naive Bayes classifier** An classifier using TF-IDF vectorized text features.
- **BERT Fine-tuning Classifier**: A fine-tuned BERT model with an added single-layer feed-forward neural network classifier on top of the BERT output.
  - The text data is tokenized and encoded using the BERT tokenizer.
  - PyTorch DataLoader is used to create batches of data for efficient training.
  - The BERT model is fine-tuned using the AdamW optimizer and a linear learning rate scheduler.
  - The model is trained for 2-4 epochs, with evaluation on the validation set after each epoch.
- **XGBoost Classifier**: An XGBoost classifier using TF-IDF vectorized text features.
- **Prompted-based Zero-shot inference**: Load pre-trained $falcon-7b-instruct$ from huggingface and create prompt to ask model to generate positive or negative label.

TRAING AND VALIDATION

Hyper-parameter tuning and cross-validation are performed on these models. The best parameters are chosen to predict on the test set. Results are analyzed by confusion matrix, AUC ROC and Precision-recall ROC.

SOURCE CODE

Ipython notebook can be accessed in: https://github.com/Connorlee487/dlearning