Sentiment Classification for Amazon Product Reviews Based on Deep Learning Method

1st Yunqi Song
Department of Statistics
University of Michigan
Ann arbor, America
syunqi@umich.edu

***Abstract*—This project explores sentiment classification of Amazon product reviews using deep learning methods. Using the 'amazon-polarity' dataset, the project trains and evaluates the performance of a baseline TF-IDF+LinearSVC model, GloVe-based LSTM and TextCNN models, and a BERT model. The results demonstrate that deep learning methods significantly outperform traditional machine learning approaches across accuracy, F1 score, and AUC metrics. Among all models, BERT achieves the highest performance, confirming its strength in capturing contextual and semantic nuances in text. While BERT incurs the highest computational cost, the LSTM model emerges as a strong alternative for resource-constrained scenarios. This study emphasizes the practical effectiveness of modern NLP architectures for real-world text classification tasks.**

***Index Terms*—sentiment analysis, deep learning methods**

I. Introduction

A. Project Background and Motivation

Sentiment analysis is one of the most practical and widely applied tasks in Natural Language Processing (NLP). It has driven the development of various fields, such as public opinion monitoring, automatic summarization, recommendation systems, and dialogue systems. For e-commerce platforms, sentiment analysis can be used to evaluate user reviews and identify positive and negative feedback on products, which helps improve product quality and better customer experience. In my project, a dataset 'amazon-polarity' is obtained from Hugging Face. This dataset consists of 35 million Amazon product reviews, spanning a period of 18 years up to March 2013. For each review, not only is its text content shown, but a label ('positive' or 'negative') is given to represent its sentiment. This dataset is very suitable for a text sentiment classification task, and the motivation of my project is to build a model that can classify these reviews with high accuracy.

In recent years, deep learning models have become popular and achieved great success in NLP. In my project, I will mainly focus on how to transform our text into data that can be handled by computers and implement classification models based on deep learning method. I hope it can help me learn about modern word embedding techniques and deep learning models for text classification, and be familiar with implementing deep learning models with pytorch. Also, I want to see how deep learning methods can improve our task compared with traditional text feature engineering (e.g., TF-IDF) and machine learning models (e.g., SVM). Therefore, I will also build a baseline model based on traditional methods in my project and I expect a significant improvement for deep learning models.

B. Literature Review

Sentiment classification is a common task in Natural Language Processing (NLP). Early sentiment classification methods relied on sentiment lexicons and rule-based approaches, such as SentiWordNet [1] and VADER [2]. These methods used predefined sentiment word lists and sentiment polarity calculations but lacked adaptability to different domains and languages. As machine learning models become popular, they are also applied in sentiment classification. As proposed in [3], there are two main steps for sentiment classification using machine learning techniques. First, one should conduct feature engineering such as TF-IDF [4] and N-gram [5]. Second, one should use those features to train machine learning models such as SVM and Naïve Bayes. However, the requirement for handcrafted features makes machine learning methods highly dependent on data quality and feature selection.

Today, deep learning methods have been widely used in sentiment classification and have achieved great success. The first to be mentioned is the word embedding techniques, which aim to represent words as dense vectors in a continuous space, capturing semantic and syntactic relationships between words. The first word embedding model based on deep learning is Word2Vec [6]. Following Word2Vec, many word embedding models have been proposed, such as GloVe [7] and FastText [8]. These word vectors can then be used as input for many deep learning classification models. Some of them, which are suitable for NLP problems, are TextCNN [9] and LSTM [10]. TextCNN is good at capturing local semantic information, while LSTM is good at capturing contextual dependencies in text.

In 2017, the Transformer architecture [11] was proposed. The Transformer uses self-attention mechanisms to model global dependencies in sequences and enables parallel computation, significantly improving efficiency and scalability for large datasets. It has since become the foundational architecture for many state-of-the-art models in NLP. Among them, Bidirectional Encoder Representations from Transformers (BERT) [12], introduced in 2018, revolutionized NLP and significantly improved performance across a wide range of

NLP tasks, including sentiment classification. In the following years, we also see many variants and extensions of BERT, such as RoBERTa [13] and ALBERT [14].

With the development of deep learning, transfer learning has also become widely used. Without training a whole new model, one can just fine-tune a large-scale pretrained model and transfer it to a particular task. As stated in [15], transfer learning offers significant advantages in low-resource languages, fine-tuning flexibility, and multi-task generalization.

II. Method

A. Data Description

The dataset used in the project is 'amazon-polarity'. It provides a training set (3.6 million reviews) and a test set (0.4 million reviews). Each review is given a label representing its sentiment ('0' means negative and '1' means positive). Due to the limitation of my computing resources, I will only use part of the whole dataset. That is, I will randomly select 0.5 million positive reviews and 0.5 million negative reviews from training set, and 0.1 million reviews from test set. From now on, when we mention 'training set' or 'test set', we mean the new constructed ones.

Here we have an overview of the training set. A vocabulary is created based on the training set and it shows that the training set contains 426,803 different words. Fig. 1 shows the distribution of the review length in the training set. Most reviews are no longer than 200 words, which means those reviews are not too long.

Below, we will build several models to classify the sentiment of reviews. These models will take text reviews as input and output the probability that the review is positive.

B. Baseline Model

To show the power of deep learning methods, a traditional model is used as a comparison. This model first transforms reviews into TF-IDF features. TF-IDF can identify keywords that are representative in a specific document but uncommon across the entire corpus. Next, the TF-IDF features will be used to train a linear SVC.

This model will be built on the basis of the Python module'sklearn'. 'sklearn' provides 'TfidfVectorizer' and 'LinearSVC' that helps us implement the baseline model easily.

C. Models Based on GloVe

As mentioned in the literature review, transfer learning is powerful. Here we use a pre-trained model GloVe from Hugging Face. GloVe captures both global and local semantic relationships. The pre-trained GloVe model from Hugging Face can transform any word in its vocabulary into a 300-dimensional vector. If there is a new word not in its vocabulary, we will randomly generate one vector.

GloVe integrates well with deep learning models. The word embeddings it produces can be used as input to LSTM and TextCNN. LSTM is a kind of recurrent neural network (RNN) that utilizes cell state and gates to model long-term dependencies, while TextCNN is a kind of convolution neural network

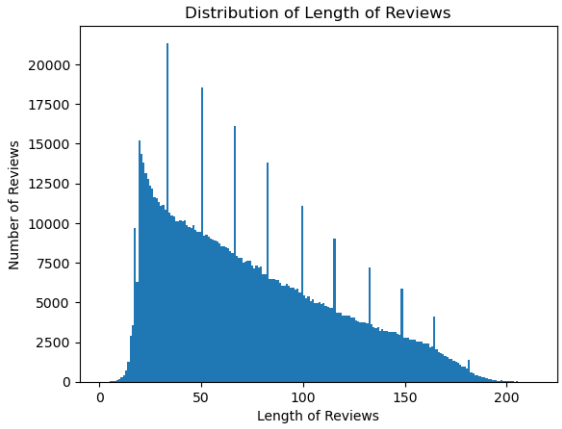

Fig. 1. The distribution of the review length in the training set.

(CNN) that uses 1-D convolution kernels with different sizes to capture local patterns. Finally, the LSTM (or TextCNN) output will go into a linear layer and a sigmoid function to become a probability.

In Python, we will use the module 'gensim' to load the pre-trained GloVe model, create embedding matrix and tokenize text. We use the 'torch' module to construct the LSTM and TextCNN models. Due to a requirement for fixed-length input, we will pad or truncate these reviews. During the training process, the pre-trained model will also be fine-tuned.

D. BERT

BERT is a state-of-the-art model that has a strong performance in many tasks. BERT's unified transformer-based design allows a single architecture to handle multiple tasks via simple output layer changes. To perform text classification, BERT adds a special token [CLS] at the beginning of every input sequence. After passing through the BERT model, this token can give a vector that is used to summarize the entire input. Then we simply pass this vector through a linear layer and a sigmoid function to obtain a probability.

In Python, we use the module 'transformers' to load a pre-trained BERT model from Hugging Face. This module also provides a tokenizer designed for BERT. We will build a model based on 'torch' to pass tokenized input through BERT and use the vector given by [CLS] to produce a probability via linear layer and sigmoid function.

III. Results

A. Model Specification

To optimize model hyperparameter settings and model structures, we split the training set into a training set and a validation set. The split ratio is 8:2. Models with different settings will be trained on the training set and evaluated on the validation set. The best model will then be selected. For the baseline model, we will select the one with highest validation accuracy. For deep learning models, we will select those with lowest validation loss.

TABLE I
MODEL EARLY STOP EPOCH AND TRAINING TIME

Model Names	Early Stop Epoch	Training Time
TF-IDF+LinearSVC	–	56s
GloVe+LSTM	6	15min 48s
GloVe+TextCNN	6	6min 6s
BERT	3	137min 9s

TABLE II
MODEL ACCURACY AND F1 SCORE

Model Names	Evaluation Metrics	
	Accuracy	*F1 score*
TF-IDF+LinearSVC	91.13%	91.09%
GloVe+LSTM	92.67%	92.88%
GloVe+TextCNN	91.89%	91.91%
BERT	95.07%	95.13%

As described before, we implement four models in our project: one baseline model (TF-IDF+LinearSVC), and three deep learning models (GloVe+LSTM, GloVe+TextCNN, BERT). Below, we report model specification for these four models after validation.

- TF-IDF+LinearSVC: When producing TF-IDF features, we will use 1-gram and 2-gram, and only consider the top 20 thousand frequent grams in the training corpus. In LinearSVC, the penalty is 0.1.
- GloVe+LSTM: The length of reviews will be padded or truncated to 150. The LSTM is set to be a bidirectional one with 512 hidden dimensions and 2 hidden layers.
- GloVe+TextCNN: The length of reviews will be padded or truncated to 150. The size of convolution kernel is 3,4,5, each with 200 filters.
- BERT: The length of reviews will be padded or truncated to 200. No other changes are made to the pre-trained BERT model from Hugging Face.

B. Model Training

The training of the baseline model is based on the Python module 'sklearn', while deep learning models are trained on the Pytorch framework. We use a GPU to accelerate the training process.

When training deep learning models, the criterion is the cross-entropy loss and the optimizer is the Adam optimizer. The learning rate is 1×10^{-4} and the maximum training epoch is 10. The training batch size for LSTM and TextCNN is 258, while for BERT is 128.

To avoid overfitting of deep learning models, we use the dropout technique. In our model, the dropout layer is placed between the LSTM hidden layers and behind the linear layer. The dropout rate is 0.3. In addition, early stops are added to the training process. That is, if the validation loss increases during training, then the training process will immediately stop.

In Tab. I, we report the early stop epoch and training time for each model. We see that compared to the baseline model, deep learning models need much longer time to train. The training of BERT lasts for more than 2 hours even with GPU acceleration.

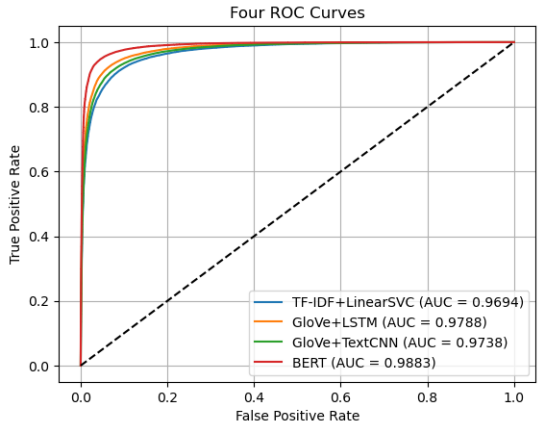

Fig. 2. ROC curves for each model.

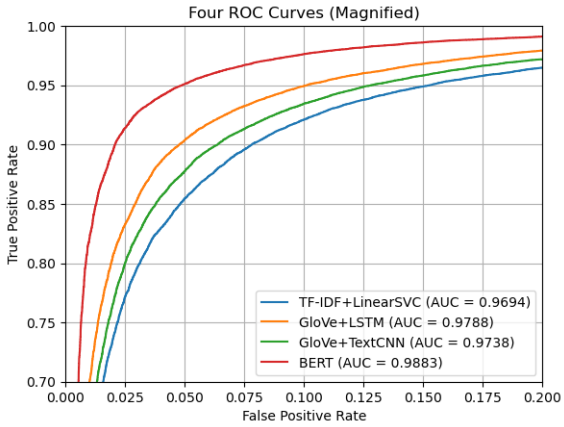

Fig. 3. ROC curves for each model (magnified).

C. Model Evaluation

After training, the models are evaluated on the test set. As mentioned above, these models output the probability that a review is positive. Then we first predict a review to be positive when $p > 0.5$. In Tab. II, we report the test accuracy and the F1 score. The three deep learning models all achieve better performance than the baseline model, and the BERT model demonstrates significant advantages compared to other models.

Next, we further analyze the false positive ones. Fig. 2 shows the four ROC curves of each model and their AUCs are also calculated. To see the difference more clearly, we magnify a section of Fig. 2 and it is shown as Fig. 3. The AUCs of deep learning models are all better than the baseline model, and the BERT model is the best. In addition, it can be observed that any two ROC curves do not intersect. The ROC curves of deep learning models are always above that of the baseline model, which means that the advantage of deep learning models over the baseline model is uniform.

IV. Conclusion

In my project, I successfully implement three deep learning models with Pytorch framework and pre-trained models from Hugging Face to make sentiment classification for the dataset 'amazon-polarity' from Hugging Face. The project results show that deep learning methods perform better than traditional methods in terms of the test accuracy, F1 score and AUC. Observation of ROC curves also shows a uniform advantage of deep learning models over traditional models in terms of true positive identification.

As a state-of-the-art model, the BERT model again shows its powerful capability of sentiment classification on this 'amazon-polarity' dataset. Without changing its model structures, the BERT model has the best performance in terms of all model evaluation methods in my project.

However, the only disadvantage of the BERT model is its long training time, even with GPU acceleration. If computing resources are constrained and the BERT model is unable to implement, according to the results of my project, an LSTM model is suggested since its training time is not too long and its performance is the second best in terms of all model evaluation methods in my project.

Code

The code used in this project has been uploaded to GitHub. The repository link is: https://github.com/SONG-Yunqi/stats-507-Final-Project.

References

[1] Hutto, Clayton, and Eric Gilbert. "Vader: A parsimonious rule-based model for sentiment analysis of social media text. Proceedings of the international AAAI conference on web and social media. Vol. 8. No. 1. 2014.

[2] Esuli, Andrea, and Fabrizio Sebastiani. "Sentiwordnet: A publicly available lexical resource for opinion mining." LREC. Vol. 6. 2006.

[3] Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan. "Thumbs up? Sentiment classification using machine learning techniques." arXiv preprint cs/0205070 (2002).

[4] Sparck Jones and Karen. "A statistical interpretation of term specificity and its application in retrieval." Journal of documentation 28.1 (1972): 11-21.

[5] Shannon and Claude E. "A mathematical theory of communication." The Bell system technical journal 27.3 (1948): 379-423.

[6] Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global vectors for word representation." Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.

[7] Bojanowski, Piotr, et al. "Enriching word vectors with subword information." Transactions of the association for computational linguistics 5 (2017): 135-146.

[8] Yoon Kim. "Convolutional Neural Networks for Sentence Classification." Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). 2014.

[9] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

[10] Bojanowski, Piotr, et al. "Enriching word vectors with subword information." Transactions of the association for computational linguistics 5 (2017): 135-146.

[11] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

[12] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers). 2019.

[13] Liu, Yinhan, et al. "Roberta: A robustly optimized bert pretraining approach." arXiv preprint arXiv:1907.11692 (2019).

[14] Lan, Zhenzhong, et al. "Albert: A lite bert for self-supervised learning of language representations." arXiv preprint arXiv:1909.11942 (2019).

[15] Ruder, Sebastian, et al. "Transfer learning in natural language processing." Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: Tutorials. 2019.