Amazon Review Classification with Machine Learning and Deep Learning Models

Siqi Li

Abstract

Deep learning models have surpassed traditional machine learning approaches in various text classification tasks in recent years. In this project, I trained and fine-tuned the parameters for three machine learning models and two deep learning models and provided a quantitative analysis of their performance based on Amazon video review data.

1 Introduction

Text classification is the process of assigning tags or categories to text based on its content. It's one of the fundamental and classic tasks in natural language processing (NLP) and has a wide range of applications such as sentiment analysis and email spam detection. This project trained and fine-tuned five text classification models (logistic regression, support vector machine, multinomial naive bayes, BERT and LSTM) and evaluated their performances based on multi-class classification accuracy.

2 Related Work

In the field of natural language processing, there are two main types of models for doing text classification: traditional machine learning models and deep learning models.

2.1 Traditional Machine Learning Models

Models such as logistic regression, Naïve Bayes, and k-nearest neighbor are traditional machine learning approaches but still commonly used. Support vector machine models are also broadly used to do text classification. In addition, tree-based classification algorithms including XGBoost and random forests are fast and accurate for document categorization (Kowsari, et al., 2019). These models leverage bag-of-words with TF-IDF score or word embedding with Word2Vec feature representations (Mikolov, et al., 2013).

2.2 Deep Learning Models

Deep learning models include recurrent neural networks (RNNs), convolutional neural networks (CNNS), Long short-term memory (LSTM), hierarchical attention networks (HAN) and language models such as bidirectional encoder representations from transformers (BERT).

Languages models, for example, overcome the biggest limitation of the classic word embedding approach: polysemy disambiguation, a word with different meanings (e.g. “bank” or “stick”) is identified by just one vector.

ELMO, developed in 2018, doesn’t apply a fixed embedding but instead looks at the entire sentence and then assigns an embedding to each word using a bidirectional LSTM (Peters, et al., 2018).

Google’s BERT (2018) further combines ELMO context embedding and several Transformers. Moreover, BERT is bidirectional, which was a big advance for Transformers. The vector BERT assigns to a word is a function of the entire sentence. As a result, a word can have different vectors based on its contexts (Devlin, et al., 2018).

3 Dataset

3.1 Data Source

The dataset chosen for this project is Amazon video review data from Amazon Review Data. The raw dataset contains information such as reviews (ratings, text, helpfulness votes) and product metadata (descriptions, category information, price, brand, and image features) for 717,651 reviews. Due to computational limitation, only the first 500,000 records were used in this text classification project. Each pre-process record has

the overall review score (integer from 1 to 5) and the review text (string).

3.2 Dataset Statistics

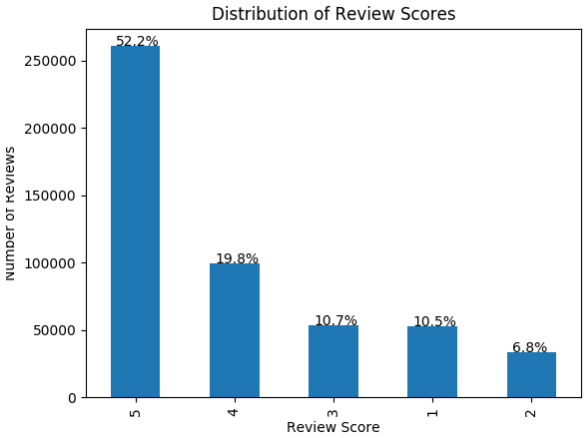

Figure 1: Distribution of Review Scores

Summary Statistics	
Number of Reviews	500,000
Average Length of Reviews	775.50
Standard Deviation of Length of Reviews	984.28
Minimum Length of Reviews	1
Length of Reviews (25 percentiles)	191
Length of Reviews (50 percentiles)	422
Length of Reviews (75 percentiles)	948
Maximum Length of Reviews	31,713

Table 1: Dataset Statistics

4 Methods

4.1 Text Preprocessing

Text preprocessing is an essential step for natural language processing (NLP) tasks. It transforms raw text inputs into a more digestible form so that machine learning algorithms can perform better.

The raw reviews were cleaned by removing special characters, removing numbers, removing English stop-words, removing extract spaces, and converting texts to lower cases

4.2 Model Candidates

This project trained and compared five kinds of text classification models. For each kind of model, I experimented with different model parameter combinations.

The five kinds of models are:

1. Logistic regression model
2. Support vector machine model
3. Multinomial Naïve Bayes model
4. BERT model
5. LSTM model (one-directional LSTM, bidirectional LSTM, and bidirectional LSTM with self-attention layer)

4.3 Model Training and Evaluation

For traditional machine learning models (logistic regression, support vector machine, and naïve bayes), TF-IDF score was used as the feature representation. For deep learning models, BERT model used its own feature representation and LSTM models tokenized texts by turning each text into a sequence of integers with each integer being the index of a token in a dictionary.

In terms of training and test set split, 80% of the data was used for training the model and the rest 20% was used as test set for evaluating model performance. Multi-class classification accuracy on the test set was used as the evaluation metric, and confusion matrices were also generated for understanding how model perform in each class.

5 Results

5.1 Logistic Regression

#	Penalty	tf–idf n_gram range	C	Accuracy
1	L2	(1,1)	1.0	0.6545
2	L2	(1,2)	1.0	0.6827
3	L1	(1,1)	1.0	0.6491
4	L2	(1,1)	2.0	0.6591

Table 2: Logistic Regression Model Results

5.2 Support Vector Machine

#	Penalty	tf–idf n_gram range	Loss	Dual	Accuracy
1	L2	(1,1)	squared hinge	True	0.6659
2	L2	(1,2)	squared hinge	True	0.7227
3	L2	(1,1)	squared hinge	False	0.6659
4	L2	(1,1)	hinge	True	0.6553
5	L2	(1,3)	squared hinge	True	0.7284

Table 3: Support Vector Machine Model Results

5.3 LSTM

#	Num Epoch	Max Num Words	Max Input Length	Embedding Dimension	Bidirectional	Self-Attention Layer	Batch Size	Accuracy
1	9	50,000	250	100	False	False	256	0.6932
2	9	50,000	350	100	False	False	256	0.6914
3	9	50,000	428	100	False	False	256	0.6980
4	9	50,000	256	100	True	False	512	0.6897
5	9	50,000	256	100	True	False	256	0.6948
6	10	50,000	350	100	True	False	512	0.6859
7	9	50,000	350	100	True	False	256	0.6970
8	11	50,000	512	100	True	False	256	0.6957
9	7	50,000	256	100	True	True	128	0.6948
10	7	50,000	350	100	True	True	128	0.6955
11	7	50,000	512	100	True	True	64	0.6986
12	7	100,000	512	256	True	True	64	0.7091

Table 4: LSTM Model Results

5.4 Multinomial Naïve Bayes

#	tf–idf n_gram range	fit_prior	alpha	Accuracy
1	(1,1)	True	1.0	0.5304
2	(1,2)	True	0	0.6290
3	(1,1)	True	0	0.6111
4	(1,2)	False	0	0.6197

Table 5: Multinomial Naïve Bayes Model Results

5.5 BERT

#	Number of Epochs	Max Input Length	Batch Size	Accuracy
1	15	25	256	0.6552
2	9	50	256	0.6836
3	10	75	128	0.7077
4	8	150	64	0.7344
5	9	200	64	0.7410
6	7	250	16	0.7487
7	10	300	16	0.7512
8	8	350	8	0.7552
9	10	428	8	0.7557
10	7	512	8	0.7580

Table 6: BERT Model Results

6 Discussions

Among traditional machine learning models, support vector machine model out-performed logistic regression model and multinomial naïve bayes models. The best SVM model used L-2 penalty, square-hinge loss, TF-IDF feature representation of unigram, bigram, and trigram, and achieved a multi-class accuracy of 0.7284 on the test set. In addition, increasing n-gram range of TF-IDF from using only 1-gram to using 1-gram to 3-gram significantly helped boost the performance for logistic regression model (0.6548 to 0.6827) and SVM models (0.6659 to 0.7284).

Various LSTM models were trained and evaluated, but they all have similar results (~69%). One thing I noticed was that batch size seemed to be related to model accuracy. Specifically, a smaller batch size seemed to lead to a higher accuracy for bidirectional LSTM model. This reflects Kevin's (2018) finding that higher batch sizes leaded to lower asymptotic test accuracy with a multi-layer perceptron (MLP) model. Moreover, adding a self-attention layer didn't improve model performance a lot in this work. Other model parameters or structure are needed to be fine-tuned in order to boost LSTM's performance. Overall, the bidirectional LSTM model with an embedding size of 256, a max number of words of 100,000, a batch size of 64 and a self-attention layer had the best accuracy of 0.7091 on the test set.

For BERT model, due to computational limits, only max input length was fine-tuned. The results show that the greater the input length, the higher the classification accuracy. This could be explained by the fact that the average review length is 775.5, and with longer input length, BERT models can extract more information to make a better classification. The best BERT model had max length of input equal to 512 and a batch size equal to 8. It achieved an accuracy of 0.75802 in the test set, which was an 3.91% increase from the best. SVM model.

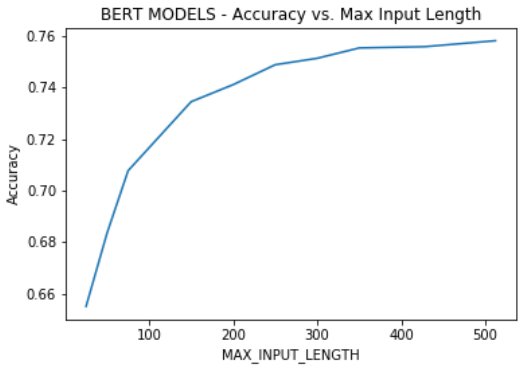

Figure 2: Accuracy vs. max input length for BERT models

The following graph shows the normalized confusion matrix for the best BERT model. While the overall accuracy is satisfying, the model still has a potential for improvement, especially for correctly distinguishing reviews with 4 and 5 stars.

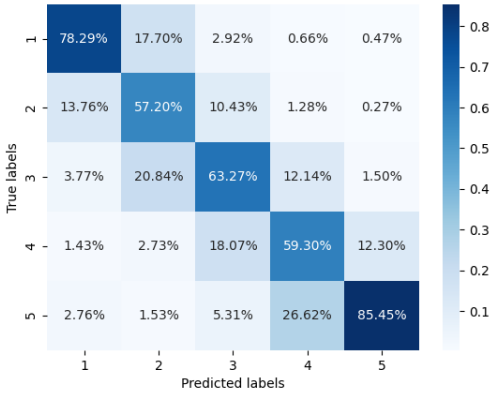

Figure 3: Normalized Confusion Matrix for the Best BERT Model

One limitation is that while BERT model had the best text classification performance, it took a much longer time to train, compared to LSTM and other traditional machine learning models. The best BERT model, for instance, needed more than 6 hours to train an epoch using a NVIDIA GeForce RTX 2080 Ti GPU, while the best SVM model took only 35 minutes to train on an Intel 4-Core i7 CPU. If hardware permits, training BERT model with a higher max input length may lead to an even better performance than what I currently had in this work.

7 Conclusions

In conclusion, BERT model had the best performance among the five models trained and evaluated in this work. Support vector machine model had the second-best performance, but its run-time was significantly less than that for the BERT model, and thus may be more practical in industry. In the future, if hardware permits, training BERT model with a higher max input length may lead to an even better performance than what I had in this work.

References

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Kevin, S. (2018). Effect of batch size on training dynamics. *Medium*. Available at: https://medium.com/mini-distill/effect-of-batch-size-on-training-dynamics-21c14f7a716e

Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., & Brown, D. (2019). Text classification algorithms: A survey. *Information*, *10*(4), 150.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119).

Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., & Gao, J. (2020). Deep learning based text classification: A comprehensive review. *arXiv preprint arXiv:2004.03705*.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*

Pietro, M., (2020). Text Classification With NLP: Tf-Idf Vs Word2vec Vs BERT. *Medium*. Available at: https://towardsdatascience.com/text-classification-with-nlp-tf-idf-vs-word2vec-vs-bert-41ff868d1794

Sun, C., Qiu, X., Xu, Y., & Huang, X. (2019, October). How to fine-tune bert for text classification?. In *China National Conference on Chinese Computational Linguistics* (pp. 194-206). Springer, Cham.

Wang, S. I., & Manning, C. D. (2012, July). Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (pp. 90-94).

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Advances in neural information processing systems* (pp. 649-657).