IMDB Movie Review Sentiment Classification

Eric Li
Department of Machine Learning
Carnegie Mellon University
Pittsburgh, PA 15213
eli1@andrew.cmu.edu

Varun Gudibanda
Department of Mathematical Sciences
Carnegie Mellon University
Pittsburgh, PA 15213
vgudiban@andrew.cmu.edu

1 Introduction

Current state of the art models in Natural Language Processing such as BERT and XLNet employ a process of massive pre-training and subsequent model finetuning on downstream tasks. Finetuning provides a method by which large neural network architectures can be warm-started in their training process by using weights learned from training on related tasks. This allows for a significant reduction in computation resources and computation time. That being said, these models still have the issue of requiring high-end GPU or TPU access in order to perform finetuning in a reasonable amount of time.

This paper aims to address this hardware issue caused by fine-tuning, specifically by using a base XLNet/BERT model to extract features from a text classification problem and incorporating these features with those of a less expensive learning model in order to reduce training time without sacrificing model performance.

2 Data

Our primary dataset is the IMDB Movie Review Dataset[6] used for text classification. The dataset is composed of IMDB movie reviews along with an assigned sentiment score of 0 (negative review) or 1 (positive review). The goal is to a classify a given review as positive or negative based solely on the content of the review. The dataset is comprised of 50,000 movie reviews along with rating labels on a 1 to 10 scale. A review is labeled if it has a score of ≤ 4 and positive if it has a score ≥ 7. In the dataset, exactly half the reviews are positive and half are negative and no neutral reviews are present (i.e. those with a score of 5 or 6). No more than 30 reviews are allowed for any given movie and the train and test sets contain a disjoint set of movies.

Sentiment analysis is a key aspect of NLP and serves as a benchmark of similarity to human understanding. The movie review dataset is similar to more formalized GLUE tasks, albeit less advanced. The dataset and problem are generally easier to score well on as compared to more recently developed GLUE tasks for sentiment extraction. That being said, the dataset serves a strong initial assessment for investigating new sentiment extraction models as the results are often transferable to similar sentiment tasks. Although sentiment analysis is not the only aspect of NLP which contributes to robust language models, it is a very central part of any well-performing model.

Although the dataset contains legitimate movie reviews, it may be biased due to the fact that they are all from IMDB. Furthermore, it is unclear how the reviews for each movie are sampled. It is possible that these are acquired in a biased manner which would affect learning of each review. Thus, all results should be compared with results drawn from other datasets.

3 Background

Our midway report explored the usage of a logistic regression used alongside feature extraction methods of bag of words and bigram/unigram mixed modeling. Bag of words is the simplest approach

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

to feature extraction whereby a count of each word is included as a feature in the final feature vector. Bigram/unigram mixed modeling uses the same approach, however introduces bigrams (two words) as features as well. These methods netted a final testing accuracy of 0.88 and 0.89 respectively with a logistic regression model tuned with an inverse regularization strength of 0.05.

Further testing revealed that lemmatization of words, a method which removes inflectional endings of words to acquire a smaller dictionary size, had little effect on accuracy, however greatly increased the training time required. Due to this, we opted to work on non-lemmatized data.

4 Related work

A large amount of research has already been done on sentiment analysis of this dataset. We describe the various models that we combine in our methods in sufficient detail to get an idea of the mathematics at play in our feature extraction.

4.1 Doc2Vec

The approach for learning paragraph vectors is heavily inspired by the methods for learning word vectors which have been discussed in lecture (word2vec). Recall, that given a sequence of training words $w_1, w_2, ..., w_T$, the word vector model attempts to maximize the average log probability

$$\frac{1}{T}\sum_{t=k}^{T-k} \log p(w_t|w_{t-k}, ..., w_{t+k})$$

Prediction is then performed to predict w_t using a softmax classifier.

In contrast to word embeddings, a paragraph vector involves learning both the word embeddings as well as a paragraph vector. The paragraph vector can be thought of as another word that is learned. The algorithm is employed as follows:

1. Train to get word vectors W, softmax weights U, b and paragraph vectors D
2. Given new paragraph, new paragraph vector is inferred by performing gradient descent on D while holding W, U, and b fixed

This is known as paragraph vector with distributed memory (PV-DM).

Another form of paragraph vectors is with distributed bag of words (PV-DBOW). This is done by modeling each paragraph as just the paragraph vector, which is then used to predict randomly sampled words from the paragraph.[5]

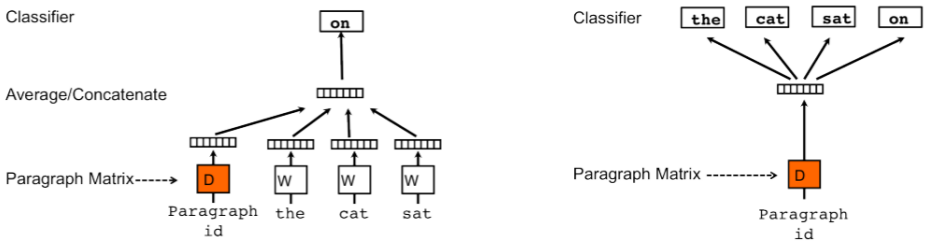


Figure 1: PVDM and PVDBOW visualization respectively

PV-DM feature extraction in conjunction with a logistic regression yields decent results of 82.7% accuracy [7]. Furthermore, it has been shown that concatenating PV-DM and PV-DBOW produces the best results for sequence classification accuracy, around 89% [8].

4.2 BERT

Bidirection Encoder Representations from Transformers (BERT) is a language representation model capable of obtaining state of the art results on the majority of NLP tasks [3].

The model architecture is a multi-layer bidirectional transformer similar to those discussed in lecture [2].

There are two steps to using BERT, pre-training and fine-tuning. Pre-training is done using two unsupervised tasks: masked logistic model (MLM) and next sentence prediction (NSP). MLM extracts a deep bidirectional representation by randomly masking some percentage of the tokens and attempting to predict these tokens. NSP is used primarily to model sentence relationships for question answering and natural language inference whereby a model is is given sentence pairs (some which are real and some which are randomly paired) and is asked to identify which of them are actually pairs.

For a text sequence x, BERT first constructs a corrupted version $\hat{x}$ by randomly setting a portion of tokens in x to be masked. If we let the masket tokens be $\bar{x}$, then the training objective is given as

$$\max_{\theta} \log p_\theta(\bar{x}|\hat{x}) \approx \sum_{t=1}^{T} m_t \log p_\theta(x_t|\hat{x})$$

where p_θ is a softmax function on the hidden vectors of the transformer.

Fine-tuning simply involves encoding the inputs and outputs and training the model on these. Fine-tuned BERT can achieve an accuracy of 95.4% when using the large model.

4.3 XLNet

XLNet is a pretraining method for NLP which combines ideas from autoregressive language modeling and autoencoding language modeling [9]. Similar to BERT, the model requires fine-tuning for downstream tasks.

One of the issues with BERT is that the autoencoder partially masks input tokens during pretraining which results in a pre-training-finetune discrepancy due to the fact that mask tokens do not appear in most real world NLP tasks. This is not a huge issue for our purposes due to the fact that we are not employing any finetuning. That being said, BERT also assumes independence on predicted tokens which is not necessarily true. XLNet deals with possible dependencies such as this, allowing for training objectives to be computed as follows.

XLNet circumvents this by utilizing a permutation language modeling objective. Let Z_t bt eh set of all possible permutations fo the length T index sequence. Let z_t and $z_{<t}$ denote the t-th element and the first $t-1$ elements of the permutation set. Then the objective is expressed as

$$\max_{\theta} \mathbb{E}_{z \sim Z_t} \left[\sum_{t=1}^{T} \log p_\theta(x_{z_t}|x_{z_{<t}}) \right]$$

As you can see, the training objective of XLNet and of BERT are quite similar. However, this small change allows XLNet to model bidirectional dependencies without the shortcomings of BERT.

When fine-tuned, the model can give 96.8% accuracy.

4.4 GloVe

Global vectors for word representation (GloVe) is an unsupervised training algorithm used to develop word vector embedding dictionaries by training on various large-scale corpuses such as Wikipedia and Twitter [4].

Applications of GloVe embeddings to the IMDB movie review dataset result in 82% classification accuracy [1].

5 Methods

Our initial approach involves the combination of Doc2Vec feature extraction alongside the hidden layer outputs of an untrained XLNet. Prior to testing this implementation, we developed custom implementations of both methods. From there, we explored additional methods including the usage of BERT and GloVe embeddings.

Our parameters in each case are raw text files. These are read in and tokenized/cleaned in some form that is specific to the feature extraction method. Our training and testing sets are both composed of 25000 reviews, each of which is extracted into some number of continuous features based on the methods used. These features are used to classify binary labels (0 or 1) in the final top layer model. These include logistic regression, random forest, and a custom neural network. The parameters for each of these top layers were tuned using a grid search on concatenated Doc2Vec features and are listed below. The parameters achieved similar performance across a large range, however these were selected based on highest accuracy on a held-out validation set.

1. Logistic regression: regularizer parameter = 1.0, loss = l2, maximum iterations = 500
2. Random forest: number of trees = 100, minimum samples per node = 10
3. Neural network: number of hidden units = 600, dropout probability = 0.1

5.1 Doc2Vec

Our initial investigation into the use of Doc2Vec shown in our poster presentation was found to be flawed. Testing examples with the same sentiment label as those of training examples were given the same document ID. This led to an extremely inflated testing accuracy.

After accounting for this, we reevaluated testing accuracy of Doc2Vec word embeddings. We found that there was a significant increase in accuracy when concatenating the embeddings of two forms of Doc2Vec, PV-DBOW and PV-DM with averaging.

We employed an adaptive learning rate training method for Doc2Vec. The issue with the contemporary training method is that LabeledSentences which only occur once will be trained with a fixed learning rate. Shuffling and adapative learning rate have shown to have better results [8]. The training algorithm is given as follows:

1. Iterate over data.
2. Shuffle the data differently for each iteration.
3. Manually control learning rate and reduce each iteration.

Refer to related work for more details regarding training Doc2Vec. The gensim package was used to generate Doc2Vec features. Each learning method results in 100 features; therefore, the concatenation of PV-DBOW and PV-DM is 200 total features for each review.

5.2 Untrained XLNet

In order to supplement Doc2Vec paragraph vectors, we chose to concatenate these features with those of an untrained XLNet. We chose to use HuggingFace along with its pytorch transformers package to load in a base XLNet model (cased). Unfortunately, the current iteration of HuggingFace has a configuration bug. It is necessary to change the `modeling_xlnet.py` function manually and replace `config.n_tokens` to the number of tokens specified for the configuration (32000). We selected two forms of the model, XLNet model without any top layer and XLNet for sequence classification.

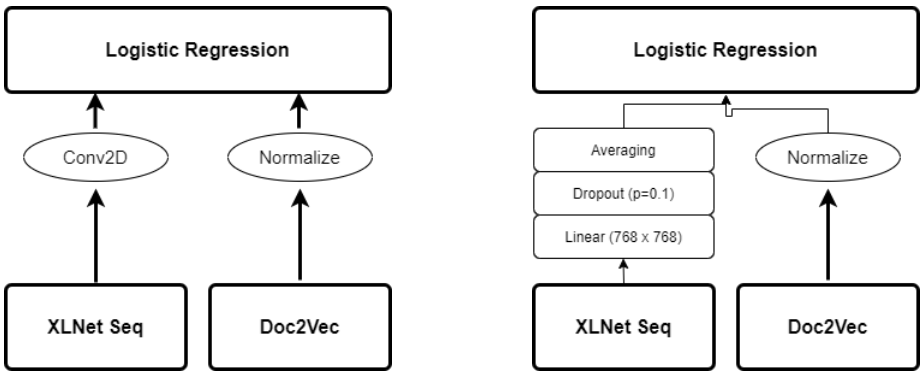
XLNet for sequence classification outputs one value for each class the model is attempting to classify for. These features were added as is to the final feature vector.

The XLNet model without any top layer required significant transformation in order to produce a reasonable number of features from the 768 hidden states in the final hidden layer. Given that we implemented a max sequence length of 128 tokens, this was far too many features for our final regression model (128×768). We experimented with three different methods for feature size reduction.

1. Additional linear layer to transform into 100 final features per token in the sequence.
2. Additional linear layer projection with normally initialized weights followed by a dropout layer for model pruning. The final result was then averaged across each axis such that each token had a single value associated with it. This was done as an adaptation of the source code for XLNet arbitrary application purposes resulting in 128 features for each review [9].

3. Convolutional layer of size 16×96 with stride equal to window size applied to 128×768 features to reduce to an 8 by 8 window of 64 flattened features.

The last two architectures had the best performance and are shown below.



(a) Convolutional summarizer (b) Neural Network summarizer

Figure 2: Untrained XLNet concatenated with Doc2Vec features

Following concatenation, we experimented with three different forms of normalization for Doc2Vec features.

1. No normalization: Features are kept as they are from both models and simply concatenated.
2. Constant normalization: Doc2Vec features are multiplied by a constant factor such that they have order similar to that of XLNet features.
3. Adaptive normalization: Doc2Vec features are scaled according to each mini-batch such that the maximum feature has value 1.

5.3 Untrained BERT

Following feedback received from our presentation, we experimented with feature extraction on untrained BERT due to its extensive library support and similarity to XLNet in feature extraction efficacy. Furthermore, we decided to extract features from the final four hidden layers of the network in order to obtain higher-level features (instead of only fine-grain features). 100 of the initial features were extracted from each layer and concatenated, leading to 400 total features per review. These were then fed into the same model architecture as shown above.

5.4 Doc2Vec w/ Glove

Given the strong performance of Doc2Vec on its own, we considered attempting to supplement Doc2Vec performance. Using the idea of hierarchical architecture network implemented by a different project group, we considered combining the feature representations of Glove and PV-DBOW to represent the high level document structure in addition to individual word embeddings.

We employed the 50 dimensional glove embedding, however this still amounts to around 50,000 features for any given review. As a result, we considered reducing the size of these features by averaging each quartile of words in order to derive an eventual 200 features. Although this severely dilutes the representative capabilities of our embeddings, this was necessary in order to maintain feature density ratio between PV-DBOW features and Glove features.

These features were subsequently concatenated and fed through a custom neural network. Two linear layers were used along with a dropout layer in between in order to improve generalization. Tanh acivations were used to mirror the activation layers used broadly in other NLP networks (XLNet, BERT, various RNNs, etc.). Backpropagation was executed using cross-entropy loss. We selected stochastic gradient descent with a learning rate of 0.1 and momentum of 0.9 as our optimizer after parameter fine-tuning. The full architecture is shown below.

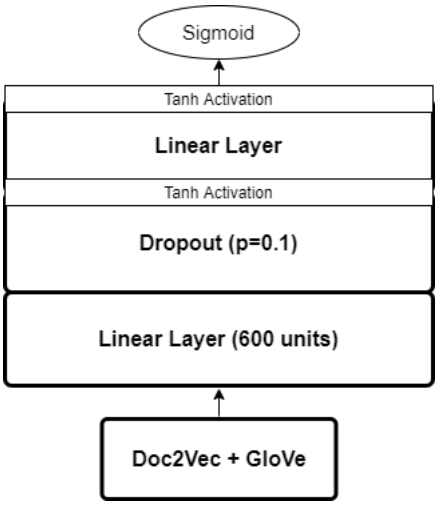


Figure 3: Neural network for Doc2Vec+GloVe classification

Additional methods were tested including random forest classifier, logistic regression, and support vector machines. Similar results were obtained from all of these implementations.

Using our intuition regarding hierarchical feature embeddings, we also considered employing only PV-DBOW concatenated with GloVe features (as PV-DM is primarily a Word2Vec encoding). This would give us a specially trained paragraph vector along with a more robust word encoding. That being said, the GloVe encoding is not trained specifically on the data and may not be as directly applicable as a result.

5.5 Additional methods

Two additional methods that were unable to be completed were also explored along the way.

- Experimentation with hierarchical attention networks using Doc2Vec features for initialization failed to capture fine-grain features as HAN generates attention vectors that are used to create a hierarchy which Doc2Vec already attempts to describe.
- Changing feature extraction layers within XLNet was unable to be completed within the given timeframe.

6 Results

Before exploring the actual results, we would like to compare a few simple runtime metrics to warrant this exploration in the first place. The entire Doc2Vec training process took 9.8255 minutes without GPU acceleration. In contrast, fine-tuning BERT on the IMDB movie review dataset took approximately 14 minutes when using Google colab research notebooks which utilize an NVIDIA Tesla K80 GPU for GPU accelerated processing. XLNet had a similar computation time. Thus, it is clear that Doc2Vec features can be extracted significantly faster, especially if done with parallelization techniques for each word.

We consider model accuracy on a held out test set. Results were found to be most effective when adaptive normalization was employed for Doc2Vec and XLNet/BERT features, thus all those shown use adaptive normalization. Concatenated feature models generally used the most successful variation of which ever features were being incorporated (e.g. PV-DM + PV-DBOW and XLNet with convolutional averaging). Standard error was a derived as a result of the random shuffling used to train Doc2Vec and Random Forest and derived by testing multiple iterations. Some results were not calculated due to calculation cost or if the model was used primarily as a base for further investigation.

Table 1: Various Doc2Vec Supplementation Methods

	Accuracy		
Method	Logistic	Random Forest	NN
PV-DM	0.7934 $\pm$0.002	NA	NA
PV-DBOW	0.8957 $\pm$0.0012	NA	NA
PV-DM + PV-DBOW	0.8958 $\pm$ 0.0009	0.8566 $\pm$ 0.0013	NA
XLNet Sequence	0.5164	NA	NA
XLNet Conv	0.5276	0.5100 $\pm$0.03	NA
BERT Selected	0.6850	0.6550 $\pm$ 0.024	NA
PV-DM + PV-DBOW + XLNet	0.8960 $\pm$0.001	0.8500 $\pm$0.006	NA
PV-DM + PV-DBOW + BERT	0.8979 $\pm$ 0.008	0.8535 $\pm$ 0.003	NA
PV-DBOW + GloVe	0.8938 $\pm$0.002	0.82 $\pm$0.004	0.8946
PV-DM + PV-DBOW + GloVe	0.8944 $\pm$0.0011	0.8391 $\pm$0.01	0.8934

Additional tests were performed to measure the convergence of the neural network methods explored. These plots are shown below in figure 4.

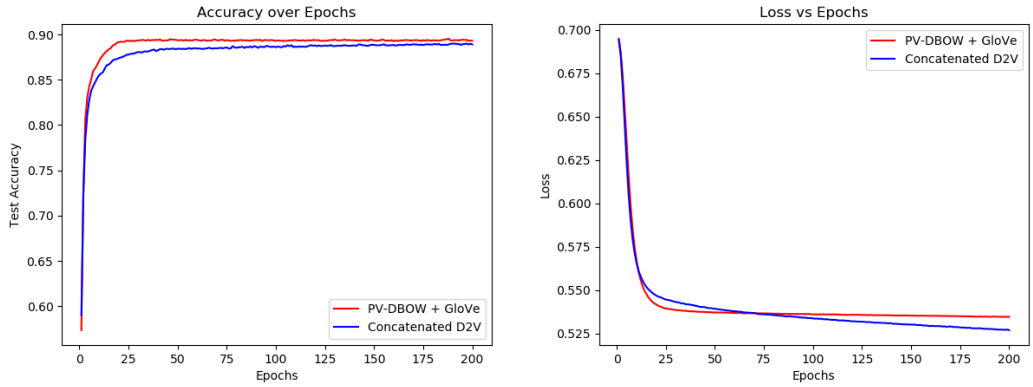


Figure 4: Accuracy and loss plots for custom neural network and Doc2Vec features

Convergence plots of the neural network indicate that PV-DBOW converges more quickly than concatenated Doc2Vec. Even though concatenated Doc2Vec achieves lower loss, PV-DBOW is still able to achieve better generalization accuracy. In general, both methods converge very quickly.

The Doc2Vec results are very similar to those found in previous Doc2Vec explorations. The results would suggest that supplementation of Doc2Vec with the methods shown above have no statistically significant impact on classification of the IMDB movie review dataset. Unfortunately, this would imply that the addition of various supporting features adds very little to improve accuracy which runs directly counter to our expectations. Furthermore, XLNet alone has extremely poor performance. Although it was expected for it to not do extremely well, the results indicate that the features have no predictive value.

The final best accuracy was achieved by the Doc2Vec and BERT combined model, however the standard error makes it clear that these results are not statistically significant.

7 Discussion and Analysis

Employing a base XLModel/BERT without any fine-tuning made the assumption that models trained on an expansive corpus can extract textual features of passages regardless of classification task. This assumption was immediately challenged when we explored the accuracy of these features when fed into a logistic regression model. XLNet features on their own produced a measly 52.76 percent classification accuracy. In contrast, BERT was able to produce a more respectable 68.5 percent classification accuracy which is closer to what we would expect.

Outside of the pre-training model feature extraction, our other features were extremely limited. Doc2Vec and Glove embeddings are based entirely on word vector construction which is good for semantic similarities but can have trouble distinguishing entirely opposite words (for example "good" and "bad" have similar vector representations). Furthermore, these models have trouble dealing with words outside of the model dictionary, though this was likely a smaller issue than the one that was previously stated. Unfortunately, it appears that the expressive capabilities of Doc2Vec are too limited to learn deep semantic meaning from them.

The results also suggest that feature concatenation may not improve the descriptiveness of the feature set. Every one of these models has accuracy similar to that of PV-DBOW on its own. This may be a result of the fact that many of these methods generate features that have high correlation. For example, Doc2Vec concatenation results in many very similar features. In an attempt to promote feature selection we experimented with L1 loss, however this provided minimal to no model improvements.

Ideally we could employ a supervised learning method that generates similarly compact representations of language as Doc2Vec or Glove, but is also capable of learning context and meaning in the way that XLNet and BERT excel. That being said, it does not appear that this can be achieved through simple word embedding methods such as Doc2Vec.

References

[1] Analysing imdb reviews using glove and lstm. `https://github.com/LJANGN/Analysing-IMDB-reviews-using-GloVe-and-LSTM`.

[2] Niki Parmar Jakob Uszkoreit Llion Jones Aidan N Gomez Lukasz Kaiser Ashish Vaswani, Noam Shazeer and Illia Polosukhin. Attention is all you need. *In Advances in Neural Information Processing Systems*, page 6000–6010, 2017.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[4] Richard Socher Jeffrey Pennington and Christopher D. Manning. Glove: Global vectors for word representation. 2014.

[5] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. *Proceedings of the 31 st International Conference on Machine Learning*, 2014.

[6] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[7] Alejandro Pelaez, Talal Ahmed, and Mohsen Ghassemi. Sentiment analysis of imdb movie reviews. *Rutgers University*, 2015.

[8] Mohit Rathore. Gensim doc2vec imdb sentiment dataset.

[9] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237, 2019.