RoBERTa and Transfer Learning to Predict Review Helpfulness

Ruth Ashford & Toby Petty

Abstract

Customers are often overwhelmed with information when seeking feedback on products, services, or businesses. Predicting which reviews will be most helpful to customers makes it easier and faster for them to make informed decisions. Existing literature on this task creates a generalizable solution using a fine-tuned BERT model, outperforming the previous bag-of-word approaches. Our research finds that a fine-tuned RoBERTa model marginally outperforms BERT, and we discover potentially promising future advancements using transfer learning.

1. Introduction

Digital PR has become increasingly powerful, with as many as 97% of people checking online reviews before visiting a local business[1]. In 2018 the popular business review site, Yelp, claimed to have 174 million reviews[2]. Ever-growing numbers of reviews can be both a blessing and a curse to potential customers. Sifting through reviews to find useful information can be overwhelming, and the majority of reviews written are positive (excellent scores making up 50% of all reviews), whereas one-star reviews make up only 16% of reviews[3]. Our focus in this report is not on a review's star rating but on how helpful the review is to a potential customer. Helpfulness is difficult to measure as it is subjective; a review may be deemed helpful if it provides insights into matters such as the best dish, if the business is dog friendly, or if you need to make a reservation. By being able to surface the most useful information to customers, they will be able to make more informed decisions on the next business they interact with. At the time of writing, fine-tuned BERT models had proven to be most successful at predicting the helpfulness of a review. Research has shown that RoBERTa has outperformed BERT on similar classification problems [8], which is our first experiment in this report. Our second experiment is centered on applying transfer learning, where we explore the viability of training models on reviews in one product domain to make predictions of helpfulness in another. One potential application of this could be to train a helpfulness classifier that can be used in settings where there is little or no training data available, for example, when launching a new product type.

2. Background

Using statistical and machine learning methods to predict the helpfulness of product reviews goes back almost twenty years, and progress in tackling this problem largely mirrors developments in machine learning and deep learning. In 2006 Kim et al. formalized a mathematical definition of helpfulness as the ratio of helpful to unhelpful ratings, framing the task as a regression problem [11]. Prior to the onset of deep learning and large language models, researchers often attempted to model helpfulness using combinations of text-based features (n-grams, TFIDF, hand-crafted importance features, readability, spelling errors, etc.) and non-text metadata features (review length, HTML tags, product features and ratings, reviewer-related features, etc.) and used techniques including SVMs, tree-based models, text-mining, sentiment analysis, and econometrics [11, 12, 13, 14, 15, 16, 18]. As

[1] https://review42.com/resources/online-reviews-statistics/

[2] https://www.searchenginejournal.com/web-directories-list/287799/

[3] https://www.yelp.com/factsheet

recently as 2017, a meta-analysis of research progress in the problem emphasized the importance of reviewer and metadata features over the actual text contents of reviews [17].

In recent years the task has been reframed as a binary classification of reviews with an arbitrary number of helpful votes (driven by websites starting to hide numbers of unhelpful votes), and researchers have explored deep learning methods. In 2018 Chen et al. showed that CNN-based models could achieve SOTA performance [19]. In 2020 Olatunji et al. experimented with context-aware methods using the attention mechanism [20], and Shuzhe et al. experimented with BERT on Amazon product reviews [21]. Reviewing the literature in 2022, Bilal and Almazroi noted: *"performance of existing solutions [...] has varied considerably and reported contradictory results on the effectiveness of various approaches"* [1]. Accordingly, they set out to benchmark the performance of cutting-edge solutions using BERT fine-tuning, which is the starting point for our research.

3. Methods

Data

Our primary dataset comes from the open source customer reviews data published by Yelp [5]. It consists of 6 million reviews across multiple categories of businesses on Yelp. Metadata for each review is provided, which indicates the number of times a review has been voted for as being helpful by readers of the review. To convert these votes into a binary label indicating 'helpful' or 'not-helpful', we used the same methodology as the existing literature [1, 6, 7], where we discarded reviews that only had one, two, or three votes and then marked reviews with no votes as being unhelpful and reviews with four or more votes as being helpful. This was done to avoid class overlap.

In order to work with this data on limited resources, we randomly select 600,000 reviews of restaurants and then further sample this to create a balanced data set across positive and negative labels of 60,000. From this, we carve out 20% for testing. We will refer to this primary data set as the 'Yelp data' in this paper.

Metrics

We found from our research [8] that it was common practice to use accuracy as the topline metric when reporting results of binary classification models. Given that we ensured our data sets had a balanced number of positive and negative examples, we also decided to use accuracy as our topline metric.

Baseline

Since our starting point was Bilal et al.'s work using fine-tuned BERT models, we initially trained a baseline model using the parameters they specified and trained it on their balanced dataset of 10k Yelp reviews [1], to confirm that the results were reproducible (we refer to this data as the "Bilal data" elsewhere). The authors used Hugging Face's open-source `BertForSequenceClassification` model, a pre-trained implementation of BERT with a sequence classification/regression head on top[4]. We used the Tensorflow version whereas the authors used PyTorch, but otherwise we followed their parameterization based on the original BERT paper's recommendations for fine-tuning (dropout 0.1, batch size of 16 or 32, learning rate of 2e-5, and epsilon of 1e-8). Bilal et al. experimented with different sequence lengths and so we used the value of 320, which achieved the best results on their test set[5]. However, we did reduce the batch size from the 32 they used to 16 due to the constraints of limited GPU availability. After training this model for 4 epochs, we achieved a test set accuracy of 0.696, very close to the authors' result of 0.707.

[4] https://huggingface.co/docs/transformers/model_doc/bert#transformers.TFBertForSequenceClassification

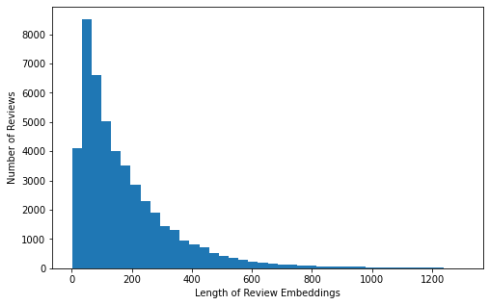
[5] https://link.springer.com/article/10.1007/s10660-022-09560-w/tables/6

Having validated that the paper's results were reproducible, we then trained the same model on the Yelp data and obtained a test set accuracy of 0.767 to serve as the baseline.

Experiment 1: RoBERTa

For our first experiment, we investigate applying RoBERTa, which is an extension to BERT that has proven to outperform the original model on multiple types of language tasks [8]. RoBERTa has the same underlying architecture as BERT but is trained on a large data corpus and has optimized a number of key parameters. We use Hugging Face's `TFRobertaForSequenceClassification` and recommendations from the original RoBERTa paper [8] for our hyperparameters when fine-tuning, which are detailed in Appendix A.2. Due to compute constraints we only used 4 epochs, and we experimented with both the linear decay with warm-up learning rate from the paper and with a fixed learning rate. To create embeddings we used `RobertaTokenizer` and found that the embedding lengths resulted in a long-tailed distribution [Fig. 1], with an average length in our training data of 179. Based on this finding, we experimented with a number of fixed embedding lengths {64, 128, 256, 320, 384, 448, 512} when training the model.

Figure 1: Embedding lengths of training data using RoBERTa tokenization



Experiment 2: Transfer Learning

Transfer learning in NLP via fine-tuning pre-trained BERT models has been applied with success in a number of tasks and domains. Mozafari et al. outperformed previous results in the task of classifying hate speech in social media using a dataset of less than 50k annotated tweets [9]. Peng et al. introduced a set of benchmark tasks for using pre-trained language models on biomedicine tasks and found that fine-tuning BERT models on PubMed extracts and clinical notes, which ranged in size from hundreds to low thousands of examples, was able to outperform SOTA in some tasks [10]. Most relevant to our research, Chen et al. demonstrated using transfer learning to predict helpfulness in Amazon product categories with limited data, but whereas they trained CNN models from scratch, we leverage pre-trained BERT models [19].

For our experiment, we explore transfer learning by testing whether review datasets from one set of product categories can be used to classify helpfulness in a different product category. The hypothesis being tested is that there are linguistic patterns that characterize "helpfulness" that are common across reviews of different product types, which a model could learn from one domain and apply to make predictions in another. We test this with both a "zero-shot" approach of classifying product data from a completely unseen domain, and an intermediate approach between zero-shot and full fine-tuning where we apply additional lightweight training of just the classifier layers on examples from the new domain. The null hypothesis is that these methods are not able to outperform the relevant baseline model.

This experiment uses a dataset of product reviews from the Amazon website[6] [3, 4]. The reviews are bucketed into categories such as "Books", "Movies and TV", "Home and Kitchen", "Toys and Games", etc. First, we randomly sample 50k reviews from the "Books" category, evenly split between helpful and unhelpful (using the same criteria for classifying helpfulness

[6] Data made publicly available by Julian McAuley of UCSD http://jmcauley.ucsd.edu/data/amazon/

described in the *Data* section). The rationale for choosing “Books” was twofold:

1. “Books” is the largest category by volume of the dataset (more than 8m records in the “small sample” dataset alone). Therefore if transfer learning is successful, this produces a good opportunity for future training of a better classifier with a much larger dataset.
2. A book is a distinctly different type of product to a restaurant, therefore this serves as a good benchmark to test the extent to which patterns learned from reviews in one product domain can be used to classify helpfulness in a completely different, unrelated product domain.

Next we scaled up the dataset in both size and variety of product domains, selecting an additional 50k records from each of the next four largest product categories by volume (“Electronics”, “Movies and TV”, “CDs and Vinyl”, and “Kindle Store”), and added them to the “Books” dataset, to create a larger Amazon product review dataset of 250k samples. (The two datasets will be referred to as “Amazon small” and “Amazon large”.)

With both small and large datasets we performed full fine-tuning of a BERT classifier using the Bilal parameters, and then tested its performance on a test subset of the Amazon dataset (to compare its same-domain performance with the baseline). Next we used these Amazon-trained models to make immediate predictions on both Bilal and Yelp test datasets with no additional tuning, to see the potential for “zero-shot cross-domain” transfer learning. Finally, with both models we performed additional extra fine-tuning stages, where we froze most or all layers of the BERT model, and just trained either the final classification head, or the final classification head plus the BERT pooler layers, on the Bilal/Yelp training datasets. To benchmark these models’ scores on the Bilal/Yelp test datasets we also applied the same lightweight extra fine-tuning procedure to a base BERT model with the same specifications but no full fine-tuning on the Amazon data.

4. Results and discussion

Experiment 1: Alternative approaches

When fine-tuning RoBERTa on the Yelp data set, we found that an embedding length of 384 with a fixed learning rate yielded the best results, with a test accuracy of 0.7751 (full results in Appendix A.2), which is a relative improvement of 1% on the baseline BERT model.

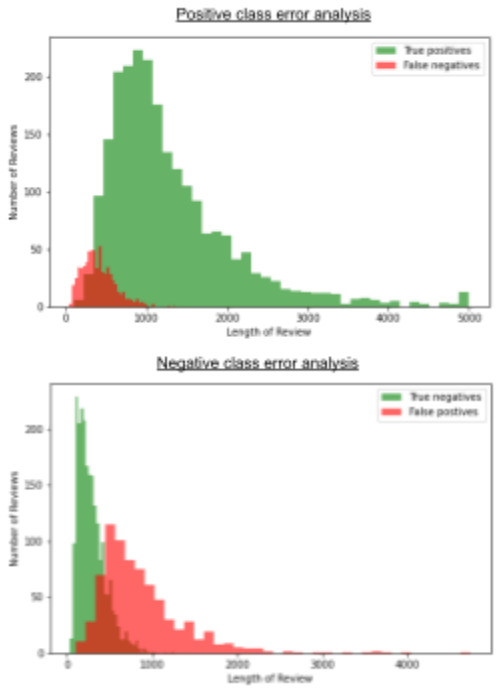
Table 1: RoBERTa: Best Fined-Tuned Model Comparison

Model	Bilal test accuracy	Yelp test accuracy
BERT baseline	**0.696**	0.767
RoBERTa: fixed learning rate	0.695	**0.775**
RoBERTa: linear decay learning rate	0.671	0.767

Although in the RoBERTa paper they found that a linear decay learning rate worked best, due to the fact that we were fine tuning with a much smaller data set, this approach led to overfitting and slightly worse results than with a fixed learning rate (see Fig. 3 in Appendix A.2).

Reading the review text of the false positive and false negative results was not very useful in our error analysis due to how subjective helpfulness is, but inspecting the length of the reviews that were incorrectly classified did provide some insights. We found that the model was frequently incorrectly classifying shorter reviews as unhelpful and longer reviews as helpful, as seen in Fig. 2.

Figure 2: Comparison of review text length across correct and incorrect predictions



To counter this, we tried two text trimming strategies. First, we used only the first 500 characters of the review and created our embeddings using this. Next, we tried using only the middle 500 characters of the review text to create embeddings. Although this led to more of the unhelpful reviews being classified correctly, it performed worse at classifying helpful reviews correctly (full results in Appendix A.2).

Experiment 2: Transfer Learning

The results show that the models which are fine-tuned on Amazon data are able to achieve significantly better than random zero-shot prediction accuracy on Bilal/Yelp data from an unseen product domain. However, they also show that full fine-tuning on Amazon data followed by additional lightweight fine-tuning on the different domain Bilal/Yelp data does not outperform simply using a base BERT model with the lightweight fine-tuning stage. To avoid confusion, in the table and discussion which follows we use the following terminology:

- *"Full fine-tuning"* means training all layers of BERT plus the classification layers.
- *"Extra-tuning"* means training classification layers only - either the head (2 layers), or the head and the BERT pooler (4 layers).

Table 2: Transfer Learning Results - Test Set Accuracy

Model training	Bilal Data	Yelp Data
Full fine-tuning on same domain dataset	**0.696**	0.756
No fine-tuning, 2 layers extra-tuned	0.657	0.744
No fine-tuning, 4 layers extra-tuned	**0.696**	**0.765**
Full fine-tuning on Amazon small (50k) ("zero-shot")	0.600	0.600
Full fine-tuning on Amazon large (250k) ("zero-shot")	0.653	0.654
Full fine-tuning on Amazon small (50k), 2 layers extra-tuned	0.626	0.722
Full fine-tuning on Amazon large (250k), 2 layers extra-tuned	0.645	0.728
Full fine-tuning on Amazon small (50k), 4 layers extra-tuned	0.686	0.757
Full fine-tuning on Amazon large (250k), 4 layers extra-tuned	0.683	0.753

In Table 2 we see that just extra-tuning 4 layers of the BERT model outperforms each attempt to use additional transfer-learning to first learn a general model of "helpfulness" by fine-tuning with Amazon data. It also even matches or outperforms full fine-tuning on the same domain training sets. We offer several possible explanations for these results.

Both Amazon datasets are potentially too small to represent the concept of "helpfulness" in a way that can be learned within our training schedule. Since we find that extra-tuning only classification layers beat even full fine-tuning on same-domain data, this suggests that all the datasets are too small to effectively finetune all of BERT's parameters for this task, and seem to degrade the richer representations learned from the much larger datasets used in BERT pretraining. Evidence of the importance of dataset size overall is shown in the fact that Bilal dataset scores are consistently lower than the Yelp dataset scores, which is 5x the dataset size. Although we were conscious of the need to

scale up from the training data used in the original Bilal paper (since it had shown clear signs of overfitting), and had increased to 10x Bilal data size in Amazon small, and 25x in Amazon large, it may be this is still insufficient.

Another explanation could be that our hyperparameters were incorrectly tuned for the task, since we did not adjust these for this experiment and just used those specified in the Bilal and BERT papers. However, we note that the BERT paper's recommendations regarding fine-tuning stated: "*large data sets (e.g., 100k+ labeled training examples) were far less sensitive to hyperparameter choice than small data sets*", so we expected the suggested hyperparameters to be reasonable, at least for Amazon large. We suggest that future experiments could be improved by one or both of: training on a much larger dataset (there are tens of millions of records available in the Amazon dataset), or by exploring different hyperparameters for the learning rate and regularization techniques such as dropout.

As with our first experiment, analyzing the predictions of the models showed that they seemed to strongly associate review length with helpfulness (where short reviews were more likely to be classified as unhelpful and vice-versa). One idea to deal with this bias in future experiments would be to train an ensemble of classifiers, where each model is trained only on a subset of (balanced) data with similar review lengths, so that it could not just learn to predict based on the number of padding tokens at the end of the sequence.

Also noticeable is that from the two different versions of extra-tuning, all our experiment results show that unfreezing the BERT pooler layer significantly outperforms just unfreezing the classification head, by an accuracy score of at least 0.03 in all cases.

Finally, our results show the viability of a zero-shot transfer learning approach to building a classifier for product domains with little or no data, in the performance of the fine-tuned Amazon models when predicting on the unseen Bilal/Yelp test sets. As shown in Table 2, both Amazon models performed significantly better than random (0.5 on a balanced dataset) when applied to test data with no additional extra-tuning. Furthermore, on both test sets we see significant improvement (accuracy increase of roughly 0.05) in the scores when the data size and number of categories increases five-fold between Amazon small and large. It is also noticeable that the scores on both unseen test sets are very similar for both models, suggesting that the models have to some extent learned a general representation of "helpfulness" (even if it just relates to review length, as discussed previously). For future experiments to extend this work we would seek to repeat the training process with larger subsets of the Amazon dataset comprising additional product categories (or with alternative datasets), to see how zero-shot prediction accuracy increases in response.

5. Conclusion

Our experiments demonstrate that the fine-tuned RoBERTa model results in a small but measurable improvement on the fine-tuned BERT model for the classification of helpful customer reviews. The transfer learning results show the viability of training a "zero-shot" classifier for predicting helpfulness in unseen product domains, using widely available data from different product types. But we find that full fine-tuning BERT using data from one domain to apply to another does not outperform using a base BERT model with lightweight classification layer fine-tuning on the same domain's data. We open-source all of our code for preprocessing datasets, training models, results, and evaluation procedures at: https://github.com/toby-p/nlp-bert-predicting-helpfulne

References

[1] Muhammad Bilal, Abdulwahab Ali Almazroi. 2022. *Effectiveness of Fine-tuned BERT Model in Classification of Helpful and Unhelpful Online Customer Reviews*

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee Kristina Toutanova. 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

[3] R. He, J. McAuley. 2016. *Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering*

[4] J. McAuley, C. Targett, J. Shi, A. van den Hengel. 2015. *Image-based recommendations on styles and substitutes*

[5] Yelp review data

https://www.yelp.com/dataset

[6] Bilal, M., Marjani, M., Hashem, I. A. T., Malik, N., Lali, M. I. U., Gani, A. 2021. *Profiling reviewers' social network strength and predicting the "helpfulness" of online customer reviews. Electronic Commerce Research and Applications, 45, 101026.*

[7] Ge, S., Qi, T., Wu, C., Wu, F., Xie, X., Huang, Y. 2019. *Helpfulness-aware review based neural recommendation. CCF Transactions on Pervasive Computing and Interaction, 1*(4), 285–295.

[8] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov. 2019. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*

[9] Marzieh Mozafari, Reza Farahbakhsh, and Noël Crespi. 2019. *A BERT-Based Transfer Learning Approach for Hate Speech Detection in Online Social Media*

[10] Yifan Peng, Shankai Yan, Zhiyong Lu. 2019. *Transfer Learning in Biomedical Natural Language Processing: An Evaluation of BERT and ELMo on Ten Benchmarking Datasets*

[11] Soo-Min Kim, Patrick Pantel, Tim Chklovski, Marco Pennacchiotti. 2006. *Automatically Assessing Review Helpfulness*

[12] Anindya Ghose, Panagiotis G. Ipeirotis. 2011. *Estimating the Helpfulness and Economic Impact of Product Reviews: Mining Text and Reviewer Characteristics*

[13] Qing Caoa, Wenjing Duan Qiwei Gana. 2009. *Exploring determinants of voting for the "helpfulness" of online user reviews: A text mining approach*

[14] Mohammad Salehana, Dan J. Kim. 2016. *Predicting the performance of online consumer reviews: A sentiment mining approach to big data analytics*

[15] Jyoti Prakash Singh, Seda Irani, Nripendra P. Rana, Yogesh K. Dwivedi, Sunil Saumya, Pradeep Kumar Roy. 2016. *Predicting the "helpfulness" of online consumer reviews*

[16] Yang Liu, Xiangji Huang, Aijun An, Xiaohui Yu. 2008. *Modeling and Predicting the Helpfulness of Online Reviews*

[17] Hong Hong, Di Xu, G. Alan Wang, Weiguo Fan. 2017. *Understanding the determinants of online review helpfulness: A meta-analytic investigation*

[18] Thomas L.Ngo-Yea, Atish P.Sinha. 2014. *The influence of reviewer engagement characteristics on online review helpfulness: A text regression model*

[19] Cen Chen, Yinfei Yang, Jun Zhou, Xiaolong Li, Forrest Sheng Bao. 2018. *Cross-Domain Review Helpfulness Prediction based on Convolutional Neural Networks with Auxiliary Domain Discriminators*

[20] Iyiola E. Olatunji, Xin Li, Wai Lam. 2020. *Context-aware Helpfulness Prediction for Online Product Reviews*

[21] Shuzhe Xu, Salvador E. Barbosa, Don Hong. 2020. *BERT Feature Based Model for Predicting the Helpfulness Scores of Online Customers Reviews*

Appendix A: Full Results

1. Transfer Learning

All models used Hugging Face's `TFBertForSequenceClassification` as the base pre-trained model, with the following parameters:

- Sequence max length = 320
- Batch size = 16
- Training epochs = 4
- Learning rate = 2e-5
- Epsilon = 1e-8

Extra fine-tuning involved training the final hidden layers for an additional 4 epochs on the Bilal/yelp training set. Two different extra fine-tuning schemes were tried - first training just the final 2 hidden classifier layers, and then also training the final 2 BERT pooler layers (4 layers total). Training time for extra fine-tuning was orders of magnitude faster than training the full BERT model, usually less than 5 minutes per epoch for the Bilal dataset and roughly 20 minutes per epoch for the larger Yelp dataset.

To benchmark the effectiveness of transfer learning, the first row in each table below shows the performance of a model in which all layers were fully trained on the training dataset paired with the test set (i.e. trained on the Bilal/Yelp datasets). The second row of each table is the benchmark of transfer learning performance; using the base BERT model with *only extra fine-tuning* on Bilal/Yelp. The final two rows evaluate the transfer learning experiment; we use the models which have undergone full pre-training on the Amazon small/large datasets. The first column shows the "zero-shot" results when the model directly makes predictions on the test set with no fine-tuning. The last two columns show the performance of the models after they undergo extra fine-tuning on 2 or 4 layers. (Best scores highlighted in bold.)

Table 3: Bilal data test set accuracy

Training dataset	Full fine-tuning	Fine-tuned classifier (2 layers)	Fine-tuned classifier and BERT pooler (4 layers)
Bilal data (10k)	**0.696**	-	-
None (BERT base)	-	0.657	**0.696**
Amazon small (50k)	0.600	0.626	0.686
Amazon large (250k)	0.653	0.645	0.683

Table 4: Yelp data test set accuracy

Training dataset	Full fine-tuning	Fine-tuned classifier (2 layers)	Fine-tuned classifier and BERT pooler (4 layers)
Yelp data (50k)	0.756	-	-
None (BERT base)	-	0.744	**0.765**
Amazon small (50k)	0.600	0.722	0.757
Amazon large (250k)	0.654	0.728	0.753

2. RoBERTa

All models used Hugging Face's `TFRobertaForSequenceClassification`, loading weights from 'roberta-base' as the base pre-trained model. The following parameters were used in fine-tuning:

- Batch size = 32
- Training epochs = 4
- Epsilon = 1e-8
- Fixed learning rate = 2e-5

All RoBERTa models fine-tuned the classification layers using the final 4 hidden layers of the pre-trained model.

To benchmark the learning, we also fine-tuned the model using the data set used in the Bilal paper [1], the results of which are included in Table 5. To examine the impact of increasing the data set, we constructed a larger data set from the Yelp data consisting of 10 times the training examples, although this led to ~1% improvement on our best fine-tuned RoBERTa model using the smaller Yelp data set, it took around 10 times as long to run, based on this we don't think the increased computation time is worth the performance gain.

Table 5: Results for RoBERTa with fixed learning rate across multiple embedding lengths

Training dataset	Embedding size	Test loss	Test accuracy	Fine tuning training time per epoch
Yelp (47k)	64	0.5791	0.7128	161s
Yelp (47k)	128	0.5464	0.7465	302s
Yelp (47k)	256	0.5287	0.7677	607s
Yelp (47k)	320	0.5212	0.7703	782s
Yelp (47k)	**384**	0.5214	**0.7715**	976s
Yelp (47k)	448	0.5193	0.7706	1154s
Yelp (47k)	512	0.5183	0.7710	1375s
Yelp (470k)	384	0.4713	**0.7852**	9207s
Bilal data (10k)	384	0.6384	0.6950	134s

Taking the best model from the above result with a max embedding length of 384, we produced results with a linear decay learning rate with warm up, the following hyperparameters were used:

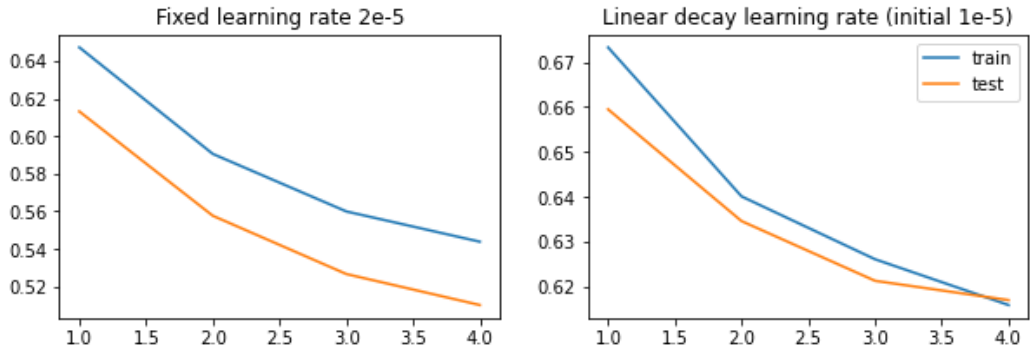
- Batch size = 32
- Training epochs = 4
- Linear decay learning rate with warm up over 6% of training steps, with initial learning rate in {1e-5, 2e-5, 3e-5}
- Embedding length = 384

Table 6: Results for RoBERTa with linear decay learning rate across various initializations

Training dataset	Initial learning rate	Test loss	Test accuracy	Fine tuning training time per epoch
Yelp (47k)	1e-5	0.6213	0.7603	920s
Yelp (47k)	2e-5	0.5704	0.7615	920s
Yelp (47k)	3e-5	0.5402	**0.7667**	921s
Bilal (10k)	3e-5	0.6598	0.6705	132

We found that the linear decay learning rate with warm up was overfitting during training in comparison to training using a fixed learning rate, as can be seen in Fig. 3.

Figure 3: Training and validation loss for fixed vs. decaying learning rates



Text trimming

We trimmed the review text in two ways:

1. Take the first 500 characters only of the review text to generate the embedding
2. Take the middle 500 characters of the review text to generate the embedding

In either of these cases if the original review text was less than 500 characters then we just used the review text without any trimming applied.

We used the same hyperparameters as in the fixed learning rate model above:

- Batch size = 32
- Training epochs = 4
- Epsilon = 1e-8
- Fixed learning rate = 2e-5

Again, fine-tuning on the final four classification layers only.

Table 7: Results for RoBERTa across text trimming strategies

Training dataset	Text trimming	Embeddi ng size	Test loss	Test accuracy	Fine-tuning training time per

					epoch
Yelp (47k)	First 500 characters	64	0.5790	0.7126	154s
Yelp (47k)	First 500 characters	128	0.5441	0.7396	287s
Yelp (47k)	First 500 characters	192	0.5404	**0.7508**	435s
Yelp (47k)	Middle 500 characters	64	0.5946	0.6992	214s
Yelp (47k)	Middle 500 characters	128	0.5786	0.7148	359s
Yelp (47k)	Middle 500 characters	192	0.5784	0.7139	519s