# DocBERT: BERT for Document Classification

**Ashutosh Adhikari, Achyudh Ram, Raphael Tang,** and **Jimmy Lin**
David R. Cheriton School of Computer Science
University of Waterloo
{adadhika, arkeshav, r33tang, jimmylin}@uwaterloo.ca



## Abstract

We present, to our knowledge, the first application of BERT to document classification. A few characteristics of the task might lead one to think that BERT is not the most appropriate model: syntactic structures matter less for content categories, documents can often be longer than typical BERT input, and documents often have multiple labels. Nevertheless, we show that a straightforward classification model using BERT is able to achieve the state of the art across four popular datasets. To address the computational expense associated with BERT inference, we distill knowledge from BERT$_{large}$ to small bidirectional LSTMs, reaching BERT$_{base}$ parity on multiple datasets using 30× fewer parameters. The primary contribution of our paper is improved baselines that can provide the foundation for future work.

## 1 Introduction

Until recently, the dominant paradigm in approaching natural language processing (NLP) tasks has been to concentrate on neural architecture design, using only task-specific data and word embeddings such as GloVe (Pennington et al., 2014). The NLP community is, however, witnessing a dramatic paradigm shift toward the pre-trained deep language representation model, which achieves the state of the art in question answering, sentiment classification, and similarity modeling, to name a few. Bidirectional Encoder Representations from Transformers (BERT; Devlin et al., 2019) represents one of the latest developments in this line of work. It outperforms its predecessors, ELMo (Peters et al., 2018) and GPT (Radford et al., 2018), by a wide margin on multiple NLP tasks.

This approach consists of two stages: first, BERT is pre-trained on vast amounts of text, with an unsupervised objective of masked language modeling and next-sentence prediction. Next, this pre-trained network is then fine-tuned on task-specific, labeled data.

BERT, however, has not yet been fine-tuned for document classification. Why is this worth exploring? For one, modeling syntactic structure has been arguably less important for document classification than for typical BERT tasks such as natural language inference and paraphrasing. This claim is supported by our observation that logistic regression and support vector machines are exceptionally strong document classification baselines. For another, documents often have multiple labels across many classes, which is again uncharacteristic of the tasks that BERT examines.

In this paper, we first describe fine-tuning BERT for document classification to establish state-of-the-art results on four popular datasets. This increase in model quality, however, comes at a heavy computational expense. BERT contains hundreds of millions of parameters, while the previous baseline uses less than four million and performs inference forty times faster.

To alleviate this computational burden, we apply knowledge distillation (Hinton et al., 2015) to transfer knowledge from BERT$_{large}$, the large BERT variant, to the previous, much smaller state-of-the-art BiLSTM. As a result of this procedure, with a few additional tricks for effective knowledge transfer, we achieve results comparable to BERT$_{base}$, the smaller BERT variant, using a model with 30× fewer parameters.

Our contributions in this paper are two fold: First, we establish state-of-the-art results for document classification by simply fine-tuning BERT; Second, we demonstrate that BERT can be distilled into a much simpler neural model that provides competitive accuracy at a far more modest computational cost.

## 2 Background and Related Work

Over the last few years, neural network-based architectures have dominated the task of document classification. Liu et al. (2017a) develop XML-CNN for addressing this problem's multi-label nature, which they call extreme classification. XML-CNN is based on the popular KimCNN (Kim, 2014), except with wider convolutional filters, adaptive dynamic max-pooling (Chen et al., 2015; Johnson and Zhang, 2015), and an additional bottleneck layer to better capture the features of large documents. Another popular model, Hierarchical Attention Network (HAN; Yang et al., 2016) explicitly models hierarchical information from documents to extract meaningful features, incorporating word- and sentence-level encoders (with attention) to classify documents. Yang et al. (2018) propose a generative approach for multi-label document classification, using encoder–decoder sequence generation models (SGMs) for generating labels for each document. Contrary to the previous papers, Adhikari et al. (2019) propose LSTM$_{reg}$, a simple, properly-regularized single-layer BiLSTM, which represents the current state of the art.

In the current paradigm of pre-trained models, methods like BERT (Devlin et al., 2019) and XL-Net (Yang et al., 2019) have been shown to achieve the state of the art in a variety of tasks including question answering, named entity recognition, and natural language inference. However, these models have a prohibitively large number of parameters and require substantial computational resources, even to carry out a single inference pass. Similar concerns related to high inference latency or heavy run-time memory requirements have led to a myriad of works, such as error-based weight-pruning (LeCun et al., 1990), and more recently, model sparsification and channel pruning (Louizos et al., 2018; Liu et al., 2017b).

Knowledge distillation (KD; Ba and Caruana, 2014; Hinton et al., 2015) has been shown to be an effective compression technique which "distills" information learned by a larger model (the teacher) into a smaller model (the student). KD uses the class probabilities produced by a pre-trained teacher, the soft targets, to train a student model over a transfer set (the examples over which distillation takes place). Being model agnostic, the approach is suitable for our study, as it enables the transfer of knowledge between different types of architectures, unlike most of the other model compression techniques.

## 3 Our Approach

To adapt BERT$_{base}$ and BERT$_{large}$ models for document classification, we follow Devlin et al. (2019) and introduce a fully-connected layer over the final hidden state corresponding to the `[CLS]` input token. During fine-tuning, we optimize the entire model end-to-end, with the additional softmax classifier parameters $W \in \mathbb{R}^{K \times H}$, where $H$ is the dimension of the hidden state vectors and $K$ is the number of classes. We minimize the cross-entropy and binary cross-entropy loss for single-label and multi-label tasks, respectively.

Next, we distill knowledge from the fine-tuned BERT$_{large}$ into the much smaller LSTM$_{reg}$, which represents the previous state of the art (Adhikari et al., 2019). We perform KD by using the training examples, along with minor augmentations to form the transfer set. In accordance with Hinton et al. (2015), we combine the two objectives of classification using the target labels ($\mathcal{L}_{classification}$) and distillation ($\mathcal{L}_{distill}$) using the soft targets, for each example of the transfer set.

We use the target labels to minimize the standard cross-entropy or binary cross-entropy loss depending on the type of dataset (multi-label or single-label). For the distillation objective, we minimize the Kullback–Leibler (KL) divergence KL$(p||q)$ where $p$ and $q$ are the class probabilities produced by the student and the teacher models, respectively. We define the final objective as:

$$\mathcal{L} = \mathcal{L}_{classification} + \lambda \cdot \mathcal{L}_{distill} \qquad (1)$$

where $\lambda \in R$ weighs the losses' contributions to the final objective.

## 4 Experimental Setup

Using Hedwig,[1] an open-source deep learning toolkit with a number of implementations of document classification models, we compare the fine-tuned BERT models against HAN, KimCNN, XMLCNN, SGM, and LSTM$_{reg}$. For simple yet competitive baselines, we run the default logistic regression (LR) and support vector machine (SVM) implementations from Scikit-Learn (Pedregosa et al., 2011), trained on the tf–idf vectors of the documents.

[1] https://github.com/castorini/hedwig

| Dataset | $C$ | $N$ | $W$ | $S$ |
|---|---|---|---|---|
| Reuters | 90 | 10,789 | 144.3 | 6.6 |
| AAPD | 54 | 55,840 | 167.3 | 1.0 |
| IMDB | 10 | 135,669 | 393.8 | 14.4 |
| Yelp 2014 | 5 | 1,125,386 | 148.8 | 9.1 |

Table 1: Summary of the datasets. $C$ denotes the number of classes in the dataset, $N$ the number of samples, and $W$ and $S$ the average number of words and sentences per document, respectively.

We use Nvidia Tesla V100 and P100 GPUs for fine-tuning BERT and run the rest of the experiments on RTX 2080 Ti and GTX 1080 GPUs. We use PyTorch 0.4.1 as the backend framework, and Scikit-learn 0.19.2 for computing the tf–idf vectors and implementing LR and SVMs.

### 4.1 Datasets

We use the following four datasets to evaluate BERT: Reuters-21578 (Reuters; Apté et al., 1994), arXiv Academic Paper dataset (AAPD; Yang et al., 2018), IMDB reviews, and Yelp 2014 reviews. Reuters and AAPD are multi-label datasets while documents in IMDB and Yelp '14 contain only a single label.

For Reuters, we use the standard ModApté splits (Apté et al., 1994); for AAPD, we use the splits provided by Yang et al. (2018); for IMDB and Yelp, following Yang et al. (2016), we randomly sample 80% of the data for training and 10% each for validation and test.

We summarize the statistics of the datasets used in our study in Table 1.

### 4.2 Training and Hyperparameters

While fine-tuning BERT, we optimize the number of epochs, batch size, learning rate, and maximum sequence length (MSL), the number of tokens that documents are truncated to. We observe that model quality is quite sensitive to the number of epochs, and thus the setting must be tailored for each dataset. We train on Reuters, AAPD, and IMDB for 30, 20, and 4 epochs, respectively. Due to resource constraints, we train on Yelp for only one epoch. As is the case with Devlin et al. (2019), we find that choosing a batch size of 16, learning rate of $2\times10^{-5}$, and MSL of 512 tokens yields optimal performance on the validation sets of all datasets. More details are provided in the appendix.

For distillation, we train the LSTM$_{reg}$ model to capture the learned representations from BERT$_{large}$ using the objective shown in Equation (1). We use a batch size of 128 for the multi-label tasks and 64 for the single-label tasks. We find the learning rates and dropout rates used in Adhikari et al. (2019) to be optimal even for the distillation process.

To build an effective transfer set for distillation as suggested by Hinton et al. (2015), we augment the training splits of the datasets by applying POS-guided word swapping and random masking (Tang et al., 2019). The transfer set sizes for Reuters, IMDB and AAPD are $3\times$, $4\times$, and $4\times$ their training splits respectively, whereas only $1\times$ (i.e., no data augmentation) the corresponding training split for Yelp2014 due to computational restrictions. We use a $\lambda$ of 1 for the multi-label datasets and 4 for the single-label datasets.

## 5 Results and Discussion

We report the mean $F_1$ scores for multi-label datasets and accuracy for single-label datasets, along with the corresponding standard deviation, across five runs in Table 2. We copy values for rows 1–9 from Adhikari et al. (2019). Due to resource limitations, we report the scores from only a single run for BERT$_{base}$ and BERT$_{large}$.

Consistent with Devlin et al. (2019), BERT$_{large}$ achieves state-of-the-art results on all four datasets, followed by BERT$_{base}$ (see Table 2, rows 10 and 11). The considerably simpler LSTM$_{reg}$ model (row 9) achieves high scores, coming close to the quality of BERT$_{base}$. However, it is worth noting that all of the models above row 10 take only a fraction of the time and memory required for training the BERT models.

Surprisingly, distilled LSTM$_{reg}$ (KD-LSTM$_{reg}$, row 12) achieves parity with BERT$_{base}$ on average for Reuters, AAPD, and IMDB. In fact, it outperforms BERT$_{base}$ (on both dev and test) in at least one of the five runs. For Yelp, we see that KD-LSTM$_{reg}$ reduces the difference between BERT$_{base}$ and LSTM$_{reg}$, but not to the same extent as in the other datasets.

To put things in perspective, Table 3 reports the inference times on the validation sets of all the datasets. We calculate the inference times with batch size 128 for all the datasets on a single RTX 2080 Ti. The relative speedup achieved by KD-LSTM$_{reg}$ is at least around $40\times$ with respect to

| # | Model | Reuters | | AAPD | | IMDB | | Yelp '14 | |
|---|---|---|---|---|---|---|---|---|---|
| | | Val. $F_1$ | Test $F_1$ | Val. $F_1$ | Test $F_1$ | Val. Acc. | Test Acc. | Val. Acc. | Test Acc. |
| 1 | LR | 77.0 | 74.8 | 67.1 | 64.9 | 43.1 | 43.4 | 61.1 | 60.9 |
| 2 | SVM | 89.1 | 86.1 | 71.1 | 69.1 | 42.5 | 42.4 | 59.7 | 59.6 |
| 3 | KimCNN Repl. | 83.5 ±0.4 | 80.8 ±0.3 | 54.5 ±1.4 | 51.4 ±1.3 | 42.9 ±0.3 | 42.7 ±0.4 | 66.5 ±0.1 | 66.1 ±0.6 |
| 4 | KimCNN Orig. | – | – | – | – | – | 37.6[8] | – | 61.0[8] |
| 5 | XML-CNN Repl. | 88.8 ±0.5 | 86.2 ±0.3 | 70.2 ±0.7 | 68.7 ±0.4 | – | – | – | – |
| 6 | HAN Repl. | 87.6 ±0.5 | 85.2 ±0.6 | 70.2 ±0.2 | 68.0 ±0.6 | 51.8 ±0.3 | 51.2 ±0.3 | 68.2 ±0.1 | 67.9 ±0.1 |
| 7 | HAN Orig. | – | – | – | – | – | 49.4[3] | – | 70.5[3] |
| 8 | SGM Orig. | 82.5 ±0.4 | 78.8 ±0.9 | – | 71.0[2] | – | – | – | – |
| 9 | $\text{LSTM}_{reg}$ | 89.1 ±0.8 | 87.0 ±0.5 | 73.1 ±0.4 | 70.5 ±0.5 | 53.4 ±0.2 | 52.8 ±0.3 | 69.0 ±0.1 | 68.7 ±0.1 |
| 10 | $\text{BERT}_{base}$ | 90.5 | 89.0 | 75.3 | 73.4 | 54.4 | 54.2 | 72.1 | 72.0 |
| 11 | $\text{BERT}_{large}$ | **92.3** | **90.7** | **76.6** | **75.2** | **56.0** | **55.6** | **72.6** | **72.5** |
| 12 | $\text{KD-LSTM}_{reg}$ | 91.0 ±0.2 | 88.9 ±0.2 | 75.4 ±0.2 | 72.9 ±0.3 | 54.5 ±0.1 | 53.7 ±0.3 | 69.7 ±0.1 | 69.4 ±0.1 |

Table 2: Results for each model on the validation and test sets. Best values are bolded. *Repl.* reports the mean of five runs from our reimplementations; *Orig.* refers to point estimates from [†]Yang et al. (2018), [‡]Yang et al. (2016), and [††]Tang et al. (2015). $\text{KD-LSTM}_{reg}$ represents the distilled $\text{LSTM}_{reg}$ using the fine-tuned $\text{BERT}_{large}$.

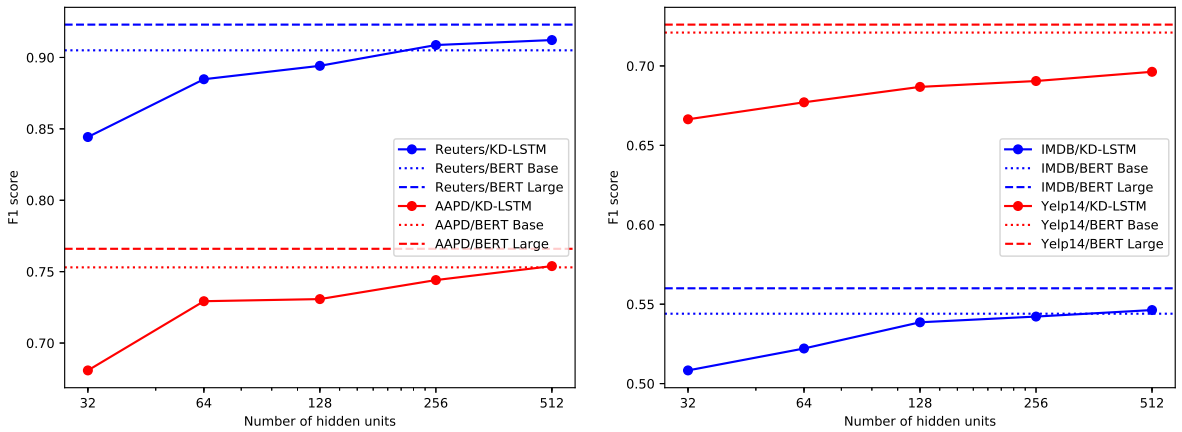


Figure 1: Effectiveness of $\text{KD-LSTM}_{reg}$ vs. $\text{BERT}_{base}$ and $\text{BERT}_{large}$

| Dataset | $\text{LSTM}_{reg}$ | $\text{BERT}_{base}$ |
|---|---|---|
| Reuters | 0.5 (1×) | 30.3 (60×) |
| AAPD | 0.3 (1×) | 15.8 (50×) |
| IMDB | 6.8 (1×) | 243.6 (40×) |
| Yelp'14 | 20.6 (1×) | 1829.9 (90×) |

Table 3: Comparison of inference latencies (seconds) on validation sets with batch size 128.

$\text{BERT}_{base}$. Additionally, Figure 1 shows the comparison between the number of parameters and prediction quality on the validation sets. These plots convey the effectiveness of the $\text{KD-LSTM}_{reg}$ model with different numbers of hidden units: 32, 64, 128, 256, and 512. We find that $\text{KD-LSTM}_{reg}$, with just 256 hidden units (i.e., $\sim$ 1% parameters of $\text{BERT}_{base}$) attains parity with $\text{BERT}_{base}$ on Reuters, while for AAPD, 512 hidden units ($\sim$ 3% parameters of $\text{BERT}_{base}$) are enough to overtake $\text{BERT}_{base}$.

## 6 Conclusion and Future Work

In this paper we improve the baselines for document classification by fine-tuning BERT. We also use the knowledge learned by BERT models to improve the effectiveness of a single-layered lightweight BiLSTM model, $\text{LSTM}_{reg}$, using knowledge distillation. In fact, we show that the distilled $\text{LSTM}_{reg}$ model achieves $\text{BERT}_{base}$ parity on a majority of datasets, resulting in over 30× compression in terms of the number of parameters and at least 40× faster inference times.

For future work, it would be interesting to study

the effects of distillation over a range of neural-network architectures. Alternatively, formulating specific model compression techniques in the context of transformer models deserves exploration.

## Acknowledgments

This research was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada, and enabled by computational resources provided by Compute Ontario and Compute Canada.

## References

Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. 2019. Rethinking complex neural network architectures for document classification. In *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)*, pages 4046–4051.

Chidanand Apté, Fred Damerau, and Sholom M. Weiss. 1994. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12(3):233–251.

Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? In *Advances in Neural Information Processing Systems 27*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 167–176, Beijing, China.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *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)*, pages 4171–4186.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *arxiv/1503.02531*.

Rie Johnson and Tong Zhang. 2015. Effective use of word order for text categorization with convolutional neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 103–112.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751.

Yann LeCun, John S. Denker, and Sara A. Solla. 1990. Optimal brain damage. In *Advances in Neural Information Processing Systems 2*, pages 598–605.

Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017a. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 115–124.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. 2017b. Learning efficient convolutional networks through network slimming. *2017 IEEE International Conference on Computer Vision*, pages 2755–2763.

Christos Louizos, Max Welling, and Diederik P. Kingma. 2018. Learning sparse neural networks through $l_0$ regularization. *arxiv/1712.01312*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432.

Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019. Distilling task-specific knowledge from BERT into simple neural networks. *arxiv/1903.12136*.

Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu, and Houfeng Wang. 2018. SGM: sequence generation model for multi-label classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3915–3926.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: generalized autoregressive pretraining for language understanding. *arxiv/1906.08237*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

# A Appendix

## A.1 Hyperparameter Analysis

**MSL analysis.** A decrease in the maximum sequence length (MSL) corresponds to only a minor loss in $F_1$ on Reuters (see top-left subplot in Figure 2), possibly due to Reuters having shorter documents. On IMDB (top-right subplot in Figure 2), lowering the MSL corresponds to a drastic fall in accuracy, suggesting that the entire document is necessary for this dataset.

On the one hand, these results appear obvious. Alternatively, one can argue that, since IMDB contains longer documents, truncating tokens may hurt less. The top two subplots in Figure 2 show that this is *not* the case, since truncating to even 256 tokens causes accuracy to fall lower than that of the much smaller LSTM$_{reg}$ (see Table 2). From these results, we conclude that any amount of truncation is detrimental in document classification, but the level of degradation may differ.

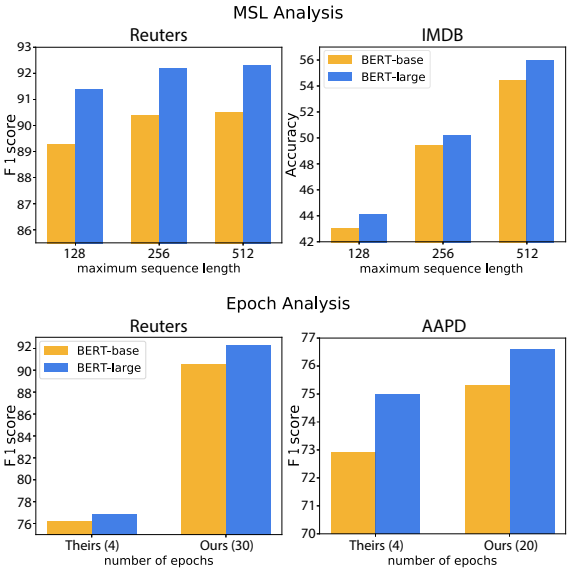


Figure 2: Results on the validation set from varying the MSL and the number of epochs.

**Epoch analysis.** The bottom two subplots in Figure 2 illustrate the $F_1$ score of BERT fine-tuned using different numbers of epochs for AAPD and Reuters. Contrary to Devlin et al. (2019), who achieve the state of the art on small datasets with only a few epochs of fine-tuning, we find that smaller datasets require many more epochs to converge. On both the datasets (see Figure 2), we see a significant drop in model quality when the BERT models are fine-tuned for only four epochs, as suggested in the original paper. On Reuters, using four epochs results in an $F_1$ worse than even logistic regression (Table 2, row 1).