NLP Topic Classification - A Deep Learning Approach

Andreea Zelko
s4311833
a.c.zelko@student.rug.nl

Matej Kucera
s4551192
m.kucera@student.rug.nl

Himawan Indra Bayu
s5719895
h.i.bayu@student.rug.nl

Abstract

Deep learning language models have been the state of the art in natural language processing for some time. They achieve high accuracy even in complicated tasks which require true understanding of written text. Such models include LSTMs as well as pre-trained language models. In this paper, these are compared using a text classification task.

The LSTM model performs worse than pre-trained models as it only achieves an F1-score of 90.8%. This is still a rather impressive score considering the efficiency of a LSTM model. In comparison, pre-trained models end up achieving F1-scores of close to 96% after fine-tuning.

Multiple pre-trained language models were also compared to see whether any of them performed significantly better on the classification task. The results show that the base BERT model performs the best along with FinBERT, a financially focused model. Smaller models like DistilBERT only performed slightly worse than BERT and FinBERT even though they are computationally much more efficient and contain fewer parameters.

1 Introduction

Analyzing text from documents or social media can be helpful when it comes to identifying unseen patterns, interesting insights, or trends. There are various methods of analysis such as topic classification or sentiment analysis. This paper focuses on the multi-class problem of topic classification for textual product reviews.

To classify text, two advanced natural language processing (NLP) techniques will be used. An LSTM model will be trained while fine-tuning the hyperparameters to see what the best performance it can achieve is. Secondly, a variety of pre-trained language models (LM) will be tested. Pre-trained language models will be evaluated by fine-tuning the hyperparameters for a BERT base model (Devlin et al., 2019). Thirdly, various proposed improvements of the BERT model will be compared to see whether they can achieve better performance using the same hyperparameters.

1.1 Research question

The chosen research question is:

Which deep learning based language model can achieve the best F1-score in a review classification task?

2 Related Work

In the last few decades, many researchers explored different approaches to improve the accuracy of text classification by increasing the dataset quality (Esuli and Sebastiani, 2013; Banerjee, 2008; Ge and Moh, 2017). Other researchers have also tried to combine several classifiers to achieve higher accuracy (Danesh et al., 2007; Yuan et al., 2008). It was also found that deep learning methods give promising results compared to traditional machine learning methods (J and U., 2022; Kansara and Sawant, 2020; Dhola and Saradva, 2021). This section explores previous studies and findings about using Long Short-Term Memory and Bidirectional Encoder Representations from Transformers (BERT) in natural language processing.

Mienye et al. conducted a study giving a comprehensive explanation of architecture, variants and various applications of LSTM (Mienye et al., 2024). Koroteev gave an overview of BERT (Koroteev, 2021) while Mohammed and Ali provided

further insight into different types of BERT (Mohammed and Ali, 2021).

In 2020, Adamuthe et al. worked on improving LSTM performance by varying its architecture and word embedding method (Adamuthe, 2020). The study used five different datasets with different numbers of classes. It was noticed that when different numbers of LSTM layers were added, the model with 2 LSTM layers achieved the highest accuracy for all datasets. In addition, it also found that an LSTM model incorporating a Bidirectional Layer had higher accuracy than traditional machine learning and Unidirectional LSTM (Liu and Guo, 2019).

A recent study compared three different models: BERT, Robustly optimized BERT approach (RoBERTa), and A Lite BERT (ALBERT) in a classification task for evaluating the Universal Access to Quality Tertiary Education (UAQTE) (Sy et al., 2024). The models were tested on five different categories. It was found that BERT and RoBERTa yielded the best performance using a small batch size, with optimal learning rates of 3e-5 and 5e-5, respectively, while the ALBERT model had the least accuracy on validation, dev set and test set. Since tuning hyper-parameters is challenging, a separate study showed that adding a superficial dense layer to the pre-trained BERT can outperform other models on the required tasks (Mohammadi and Chapon, 2020).

3 Data

In this research, a dataset containing 6000 reviews was used. The reviews belong to six categories: books, camera, DVD, health, music, and software. The reviews consist of positive and negative comments, but in this experiment, sentiment analysis is not considered. In addition, this dataset was not pre-processed in any way.

To carry out the experiments, the dataset was split into three parts: train, dev, and test set with a ratio of 80/10/10, respectively. During the tuning phase, the train set is used to fit the classifier, while the dev set is used to validate the models' performance. Afterwards, the test set is used to examine the final performance of the models. The distribution of the dataset is shown in Table 1

Category	Train	Dev	Test
Books	801	90	102
Camera	782	112	94
Dvd	801	101	110
Health	787	94	105
Music	822	114	91
Software	807	89	98

Table 1: The distribution of reviews within the dev, train and test set, grouped by category.

4 Method

For the text classification task as outlined in section 3, multiple deep learning NLP techniques can be used. A comparison can be made between using older techniques like the LSTM and state of the art methods, namely pre-trained language models like BERT and its successors.

The experiment was conducted using the `Keras` and `tensorflow` python libraries. These provided easy to use implementations of all the NLP models like LSTM and pre-trained LMs.

4.1 Long Short-Term Memory

Long Short-Term Memory (LSTM) is an improved version of Recurrent Neural Network (RNN) due to its increased memory power that leads to the ability to handle long-dependencies in sequential data (Hochreiter, 1997). For the classification task, the Keras implementation of LSTM was used (Gulli and Pal, 2017).

The architecture of an LSTM consists of an embedding layer, a dense layer, one or more LSTM layers and an output layer. The embedding layer converts every input token into a dense vector, while the dense layer supports the model during the learning process to capture complex patterns. The LSTM layers can be considered the most important part of the model. In this layer, the model learns any long-term dependencies from the data. The output layer generates a probability distribution for target classes. An early stopping criterion was implemented so that training is stopped after three consecutive epochs without improvement of validation loss.

The LSTM layer consists of RNN-like cells with a key difference. The difference is that the LSTM carries an extra information vector down the layer, a so-called cell state. This allows it to store long-term information for additional context, so that it can not only predict what kind of a word should be next, but also retrieve context for that word from earlier input. This makes the LSTM an incredibly powerful NLP model as it is able to in-

fer text very well using both short- and long-term memory (as implied by the name).

Several hyperparameters were tuned to find the optimal architecture of LSTM, such as learning rate, batch size, and epochs. Besides, choosing a suitable optimizer affects the performance of the models. For the architecture, a dense layer was introduced, along with different types of LSTM layers. Dropout was also enabled in the LSTM layers. The hyperparameter values which were used during the tuning phase are shown in Table 2.

Hyperparameter	Values
learning_rate	0.01 - **0.001**
optimizer	SGD, Adam, **RMSProp**
dense_layer	64, **128**
activation_dense_layer	**relu**
batch_size	16, **32**
epochs	10 - **50**
lstm_layer	**bidirectional**, unidirectional
dropout	0.1 - **0.5**
recurrent_dropout	0.1 - **0.5**
trainable	**True**, False

Table 2: Hyperparameter Values of LSTM

4.2 Pre-trained language models

Pre-trained language models are neural networks which are trained on large textual datasets. They learn to understand language during their training and therefore are very effective at NLP tasks. To use them on a task like text classification, one can take a pre-trained model, add a final layer to achieve the desired output shape, and fine-tune it by training on a dataset specific for the task at hand. During the pre-training stage, the model learns to understand language and it can therefore easily adapt to NLP tasks during the fine-tuning step with much less training required as compared to a randomly initialized model. The intuition for this comes from the fact that starting with pre-trained weights in the model is much better than starting with random initialization. The architecture of the model is important as well and many different ones have been proposed.

Due to their complexity, language models take quite a long time to train. The training can be accelerated with a GPU. In order to make it possible to run many iterations for the sake of comparison, the Hábrók cluster operated by the University of Groningen was used for the majority of experiments. Besides the cluster, Google Colab was also used to run tests using the learning rate scheduler. This is because Hábrók proved difficult to set up for the learning rate scheduler due to versioning conflicts.

4.2.1 Fine-tuning the BERT model

In order to see how well the BERT model could truly perform for this task, parameters were fine-tuned through a process of manual experimentation.

The BERT model was tuned using the learning_rate, batch_size, epochs and max_sequence_length. Besides this, a learning rate scheduler was also tried. The parameters were tried with values from the ranges shown in Table 3.

Hyperparameter	Values
learning_rate	**1e-5**, 2e-5, 3e-5, 5e-5
batch_size	8, **16**, 32, 64
epochs	1, **2**, 3
max_sequence_length	100, 256, **512**, 800

Table 3: Hyperparameter Values of BERT

There is a nuanced relation between the learning rate and the batch size. Generally speaking, when the learning rate is decreased, the model makes smaller updates to its weights, so increasing the batch size can help stabilize the training. Reversely, if the learning rate is increased, the model takes bigger steps, so a smaller batch size can help prevent the model from overshooting.

The number of epochs determines how many times the model iterates over the whole dataset. A good number of epochs should be high enough to prevent underfitting while staying low enough to avoid overfitting. The maximum sequence length determines the number of tokens allowed in one sequence. If a sequence is originally longer, it gets truncated. If it is originally shorter, it gets padded. A high length value allows the model to capture more context, but it requires more memory and computation. The BERT model has a quadratic time complexity with respect to the sequence length, so doubling the sequence length quadruples the computation time.

Alongside the manual parameter testing, a learning rate scheduler was also implemented. Unlike the rest of the experiments, this was carried out using Google Colab. Unfortunately, Google Colab offered a limited amount of memory compared to the Hábrók cluster. This meant the max sequence length had to be lowered, therefore im-

Model	LSTM Layer	Dropout	Recurrent Dropout	Trainable
Model A	Bidirectional	0.5	0.5	True
Model B	Bidirectional	0.2	-	True
Model C	Unibidirectional	0.5	0.5	True
Model D	Bidirectional	0.3	0.3	True
Model E	Bidirectional	0.5	-	False

Table 4: Detailed architectures of the top 5 LSTM models

pacting the performance of the learning rate scheduler model. Learning rate schedulers help to escape local minima in the early stages of training (when the rate is kept high), while also avoiding overfitting by lowering the rate in the later stages of training.

5 Results

The results of experiments on the classification task are given below.

5.1 LSTM

After collecting the results of the tuning phase, the top 5 LSTM models with the highest accuracy with different architectures of layers or parameters were selected. The chosen models were named for clarity: Model A, Model B, Model C, Model D, and Model E. The architecture details of each of these models are shown in Table 4. After testing all possible hyperparameters that are shown in Table 2, it was found that the best hyperparameter settings for the LSTM model are

```
learning_rate = 0.001,
optimizer = RMSProp,
dense_layer = 128,
batch_size = 32,
epoch = 50
```

Since each model shown in Table 5 has a different type of LSTM layer, different dropout rate, recurrent rate and trainable value, the details settings can be seen in Table 4.

The train and dev sets were used to tune the LSTM models in the first stage. Train and dev were used to provide example data in the learning process and validate the performance of the models. Model A had the highest accuracy at 90.8%. To ensure the performance of the models, a hidden test was introduced in the second stage. Model A achieved even higher accuracy on the test set at 92% as shown in the classification report, shown in Table 8.

Model	F1-score
Model A	90.8%
Model B	90.2%
Model C	89.8%
Model D	89.7%
Model E	89.5%

Table 5: The accuracy score of all chosen models on dev set

5.2 Pre-trained language models

Pre-trained models can achieve high accuracy on NLP tasks even without fine-tuning. It was found that they consistently outperform LSTM models even before any fine-tuning of hyperparameters. The results of running BERT and tuning its hyperparameters as well as running a variety of other models are reported below. The highest achieved F1-score on unseen data is close to 0,96, which is quite impressive for a complex NLP task.

5.2.1 Fine-tuned BERT model

Many combinations of the possible values of BERT hyperparamaters were tried. Some of the most notable results are presented in the section 8. In the end, it was found that the best parameters are

```
max_seq_len=512,
learning_rate=1e-05,
batch_size=16,
epochs=2
```

This combination produced a validation accuracy of 0.95, with very high precision, recall and f1-score results on the dev dataset. On the test dataset, the model obtained an accuracy of 0.96 and an average F1-score of 0.96.

Besides just manually trying parameter values, a learning rate scheduler was also tested. It was found that an initial learning rate of 0.0001 produced the best values. The other parameters of the

model were set to the best performing values (i.e. 2 epochs and a batch size of 16). However, the maximum sequence length had to be lowered to avoid getting an Out Of Memory error on Google Colab. The scheduler was able to produce results comparable to the one obtained with a fixed learning rate. On the dev set, the accuracy and f1-score average were 0.95, while on the test set the accuracy and f1-score average were 0.96. Since the model trained with the scheduler was forced to use a smaller sequence length, it is believed that under identical circumstances, it would outperform the best BERT model trained with fixed parameters.

5.2.2 Comparing multiple base models

Since there is an abundance of pre-trained models available, it is interesting to try and find which model achieves the best performance on the chosen dataset. All models were used without fine-tuning to level the playing field. A basic Adam optimizer was used with a learning rate of 0.00005 with a categorical cross-entropy loss function. The class labels were one-hot encoded. Models were trained for 1 epoch with a batch size of 8. The results are shown in Table 6. The classification report for each model can be found in section 8. The corresponding confusion matrices are omitted as they provide no further insight over the classification report which contains precision, recall and F1-scores for each class.

Model	F1-Score (avg)
BERT-base-uncased	0.92
RoBERTa-base	0.91
RoBERTa-large	0.88
DistilBERT-base-uncased	0.90
CamemBERT-base	0.80
ALBERT-base-v2	0.88
DeBERTa-base	0.91
BERT-large-uncased	0.90
FinBERT (ProsusAI)	**0.92**
TinyBERT	0.87

Table 6: Average F1-Scores of Various Models

6 Discussion

In this section, the results will be discussed and compared. As multiple models were used to classify the data, a model will be selected as best suited for this classification task according to the results.

6.1 LSTM fine-tuning

In general, tuning the hyperparameters of the LSTM model is challenging since the model is sensitive to changes. Besides choosing the proper value of learning rate, batch size, or epochs, setting the **Trainable = True** also has considerable impact on the performance of model. This is because it allows the model to update the embedding weights during the training phase. In addition, adding a dense layer and LSTM layers also improves the accuracy of the model because the model can capture more complicated patterns.

Both dropout and recurrent dropout improve the percentage accuracy of the models. However, the models which used recurrent dropout took longer to train.

Regarding optimizers' performance, RMSProp outperformed SGD and Adam optimizer. RMSProp offered better stability by maintaining the proportion of fast convergence and smooth weight update.

When it comes to the types of LSTM layers, 4 out of 5 models incorporated bidirectional LSMT layers. This means that bidirectional layers enhance accuracy by allowing the model to memorize information from the past and future time steps.

6.2 BERT fine-tuning

As expected, finding a good balance between the learning rate and the batch size proved to be the trickiest part. It required a lot of trial and error. However, the results are quite satisfactory. It is unsurprising that the fine-tuned BERT model performers better than other models used straight out of the box. This exercise proves the importance of understanding the dataset and customizing the model parameters to best suit the task at hand.

Setting up the learning rate scheduler proved a little more complicated, but manageable. This presented an overhead which, in the end, was not worth it. The results of the scheduler model were nearly identical to the results of the best fine-tuned model with a fixed learning rate. As mentioned before, this might be caused by the memory limitations imposed by Google Colab. However, it could also be the case that the dataset simply does not benefit from a learning rate scheduler. If there are no problematic local minima, then the scheduler only really helps speed up the training.

As future research, it would be interesting to

compare the scheduler against a fix learning rate in an environment where both have access to the same resources.

6.3 Model comparison

As shown in Table 6, multiple models were tested and compared. Below, the models are listed and elaborated upon.

- BERT (Devlin et al., 2019). The BERT model was used as the baseline. This is because it was the model which started the language model revolution and all other models are based on it at least to some extent. Both the `base` and `large` versions of BERT were used. The `base` version is used as a baseline to compare against. Notably, the `large` version performs worse than the base version on this task. This may possibly be caused by over-fitting due to the rather small size of the dataset. It is possible that with some tuning, the large BERT would outperform the base BERT, but that is beyond the scope of this comparison.

- RoBERTa (Conneau et al., 2019). RoBERTa is an improved version of BERT according to literature. However, it is actually found on the task at hand that it performs only marginally better or equal to BERT. Both the base and large versions of RoBERTa were used. Notably, the base version again outperforms the large model.

- DistilBERT (Sanh et al., 2019). DistilBERT is a ”smaller, faster, cheaper and lighter” (Sanh et al., 2019) version of BERT. It only has 67M parameters, which is close to half that of BERT. Considering this, it is rather impressive that it achieves nearly identical performance on this task. The trade-off in accuracy may or may not be worth it for the decreased size depending on context.

- CamemBERT (Martin et al., 2020). CamemBERT is a french language model based on RoBERTa. It was added to the list to see how much of an effect the language barrier will have. The results show that it performs about 10% worse than the other models listed. While it is clearly the worst performer out of the lot, it is still impressive that it can achieve a relatively high accuracy considering it was trained on a completely different language.

- ALBERT (Lan et al., 2019). ALBERT is much smaller than BERT with only 11M parameters. This makes it more feasible for applications where efficiency, speed and memory footprint are key. It is rather impressive to see it achieve an accuracy which is very close to that of the other models while being much smaller.

- DeBERTa (He et al., 2021). DeBERTa is a model with further improvements over BERT and RoBERTa. It uses disentangled attention and an enhanced mask decoder. According to the paper it is published in, it should outperform BERT and RoBERTa. In this task, DeBERTa achieved comparable results to BERT and RoBERTa.

- FinBERT (Araci, 2019). FinBERT is a financially-focused BERT-based model trained specifically on financial language for sentiment analysis. It was included in this comparison because even though it is not focused on this task at all, it outperforms all other models except the baseline BERT by a slim margin.

- TinyBERT (Jiao et al., 2019). TinyBERT is another model with fewer parameters than BERT. The authors claim that it is 7.5x smaller and 9.4x faster than BERT. Similarly to ALBERT, it achieves a level of performance close to that of the other models while being significantly easier to run.

7 Conclusion

It was found that both LSTM models and pre-trained LMs perform very well in the classification task used by this paper. However, pre-trained LMs outperform LSTM models by a significant margin. After fine-tuning, the LSTM models only achieved a 91% F1-score while the pre-trained models obtained a much higher score.

Pre-trained models achieved very high F1-scores on this task. The BERT model was able to achieve a 96% F1-score with fine-tuned hyperparameters. Other pre-trained models were compared with baseline hyperparameter settings, as it in unfeasible to fine-tune each model on this task.

In the comparison, it was found that BERT and FinBERT performed equally at 92%, which is impressive given the lack of fine-tuning. From the small and efficiency-focused models, DistilBERT performed the best at 90% F1-score which is also impressive considering the model size and speed, as well as the lack of fine-tuning.

Overall, it is clear that pre-trained models perform better than LSTM models on the chosen task. BERT achieves an impressively high performance and is chosen as the best performing model, answering the research question posed in section 1.

References

[Adamuthe2020] Amol Adamuthe. 2020. Improved text classification using long short-term memory and word embedding technique. *International Journal of Hybrid Information Technology*, 13:19–32, 03.

[Araci2019] Dogu Araci. 2019. Finbert: Financial sentiment analysis with pre-trained language models.

[Banerjee2008] Somnath Banerjee. 2008. Improving text classification accuracy using topic modeling over an additional corpus. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 867–868.

[Conneau et al.2019] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *CoRR*, abs/1911.02116.

[Danesh et al.2007] Ali Danesh, Behzad Moshiri, and Omid Fatemi. 2007. Improve text classification accuracy based on classifier fusion methods. In *2007 10th International Conference on Information Fusion*, pages 1–6. IEEE.

[Devlin et al.2019] 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, Minneapolis, Minnesota, June. Association for Computational Linguistics.

[Dhola and Saradva2021] Kaushik Dhola and Mann Saradva. 2021. A comparative evaluation of traditional machine learning and deep learning classification techniques for sentiment analysis. In *2021 11th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, pages 932–936.

[Esuli and Sebastiani2013] Andrea Esuli and Fabrizio Sebastiani. 2013. Improving text classification accuracy by training label cleaning. *ACM Transactions on Information Systems (TOIS)*, 31(4):1–28.

[Ge and Moh2017] Lihao Ge and Teng-Sheng Moh. 2017. Improving text classification with word embedding. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 1796–1805. IEEE.

[Gulli and Pal2017] A. Gulli and S. Pal. 2017. *Deep learning with Keras*. Packt Publishing Ltd.

[He et al.2021] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

[Hochreiter1997] S Hochreiter. 1997. Long short-term memory. *Neural Computation MIT-Press*.

[J and U.2022] Sangeetha J and Dr. Kumaran U. 2022. Comparison of sentiment analysis on online product reviews using optimised rnn-lstm with support vector machine. *Webology*.

[Jiao et al.2019] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*.

[Kansara and Sawant2020] Dhvani Kansara and Vinaya Sawant. 2020. Comparison of traditional machine learning and deep learning approaches for sentiment analysis. In Hari Vasudevan, Antonis Michalas, Narendra Shekokar, and Meera Narvekar, editors, *Advanced Computing Technologies and Applications*, pages 365–377, Singapore. Springer Singapore.

[Koroteev2021] Mikhail V Koroteev. 2021. Bert: a review of applications in natural language processing and understanding. *arXiv preprint arXiv:2103.11943*.

[Lan et al.2019] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. *CoRR*, abs/1909.11942.

[Liu and Guo2019] Gang Liu and Jiabao Guo. 2019. Bidirectional lstm with attention mechanism and convolutional layer for text classification. *Neurocomputing*, 337:325–338.

[Martin et al.2020] Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric Villemonte de la Clergerie, Djamé Seddah, and Benoît Sagot. 2020. Camembert: a tasty french language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

[Mienye et al.2024] Ibomoiye Domor Mienye, Theo G. Swart, and George Obaido. 2024. Recurrent neural networks: A comprehensive review of architectures, variants, and applications. *Information*, 15(9).

[Mohammadi and Chapon2020] Samin Mohammadi and Mathieu Chapon. 2020. Investigating the performance of fine-tuned text classification models based-on bert. In *2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 1252–1257.

[Mohammed and Ali2021] Athar Hussein Mohammed and Ali H. Ali. 2021. Survey of bert (bidirectional encoder representation transformer) types. *Journal of Physics: Conference Series*, 1963(1):012173, jul.

[Sanh et al.2019] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

[Sy et al.2024] Christian Y. Sy, Lany L. Maceda, Mary Joy P. Canon, and Nancy M. Flores. 2024. Beyond bert: Exploring the efficacy of roberta and albert in supervised multiclass text classification. *International Journal of Advanced Computer Science and Applications*, 15(3).

[Yuan et al.2008] Pingpeng Yuan, Yuqin Chen, Hai Jin, and Li Huang. 2008. Msvm-knn: Combining svm and k-nn for multi-class text classification. In *IEEE international workshop on Semantic Computing and Systems*, pages 133–140. IEEE.

8 Appendix

The rest of the page is intentionally left blank due to formatting.

Class	Precision	Recall	F1-Score	Support
Books	0.87	0.94	0.90	90
Camera	0.89	0.94	0.91	112
DVD	0.95	0.83	0.89	101
Health	0.96	0.84	0.90	94
Music	0.89	0.96	0.92	114
Software	0.91	0.92	0.92	89
Accuracy		0.91 (600)		
Macro Avg	0.91	0.91	0.91	600
Weighted Avg	0.91	0.91	0.91	600

Table 7: Classification Report LTSM Model A on dev set

Class	Precision	Recall	F1-Score	Support
Books	0.88	0.93	0.90	102
Camera	0.94	0.97	0.95	94
DVD	0.91	0.85	0.88	110
Health	0.95	0.92	0.94	105
Music	0.90	0.96	0.93	91
Software	0.95	0.91	0.93	98
Accuracy		0.92 (600)		
Macro Avg	0.92	0.92	0.92	600
Weighted Avg	0.92	0.92	0.92	600

Table 8: Classification Report LTSM Model A on test set

Class	Precision	Recall	F1-Score	Support
Books	0.97	0.97	0.97	90
Camera	0.94	0.90	0.92	112
DVD	0.98	0.94	0.96	101
Health	0.90	0.94	0.92	94
Music	0.97	0.96	0.97	114
Software	0.88	0.93	0.91	89
Accuracy		0.94 (600)		
Macro Avg	0.94	0.94	0.94	600
Weighted Avg	0.94	0.94	0.94	600

Table 9: Classification Report for BERT (max_seq_len=256, learning_rate=1e-05, batch_size=64, epochs=2)

Class	Precision	Recall	F1-Score	Support
Books	0.90	0.97	0.93	90
Camera	0.93	0.92	0.92	112
DVD	0.96	0.96	0.96	101
Health	0.93	0.91	0.92	94
Music	0.97	0.97	0.97	114
Software	0.94	0.90	0.92	89
Accuracy		0.94 (600)		
Macro Avg	0.94	0.94	0.94	600
Weighted Avg	0.94	0.94	0.94	600

Table 10: Classification Report for BERT (max_seq_len=256, learning_rate=3e-05, batch_size=16, epochs=3)

Class	Precision	Recall	F1-Score	Support
Books	0.98	0.98	0.98	90
Camera	0.91	0.95	0.93	112
DVD	0.97	0.96	0.97	101
Health	0.96	0.90	0.93	94
Music	0.99	0.97	0.98	114
Software	0.90	0.93	0.92	89
Accuracy		0.95 (600)		
Macro Avg	0.95	0.95	0.95	600
Weighted Avg	0.95	0.95	0.95	600

Table 11: Classification Report for BERT (max_seq_len=512, learning_rate=1e-05, batch_size=16, epochs=2)

Class	Precision	Recall	F1-Score	Support
Books	0.95	0.97	0.96	102
Camera	0.98	0.88	0.93	94
DVD	0.98	0.97	0.98	110
Health	0.91	0.99	0.95	105
Music	0.97	0.96	0.96	91
Software	0.96	0.96	0.96	98
Accuracy		0.96 (600)		
Macro Avg	0.96	0.96	0.96	600
Weighted Avg	0.96	0.96	0.96	600

Table 12: Classification Report for BERT on the Test set (max_seq_len=512, learning_rate=1e-05, batch_size=16, epochs=2)

Class	Precision	Recall	F1-Score	Support
Books	0.98	0.97	0.97	90
Camera	0.93	0.92	0.92	112
DVD	0.96	0.95	0.96	101
Health	0.94	0.95	0.94	94
Music	1.00	0.98	0.99	114
Software	0.90	0.94	0.92	89
Accuracy		0.95 (600)		
Macro Avg	0.95	0.95	0.95	600
Weighted Avg	0.95	0.95	0.95	600

Table 13: Classification Report for BERT using a scheduler (max_seq_len=256, learning_rate=0.0001, batch_size=16, epochs=2)

Class	Precision	Recall	F1-Score	Support
Books	0.96	0.97	0.97	102
Camera	0.93	0.96	0.94	94
DVD	0.96	0.97	0.97	110
Health	0.96	0.94	0.95	105
Music	0.98	0.98	0.98	91
Software	0.97	0.94	0.95	98
Accuracy		0.96 (600)		
Macro Avg	0.96	0.96	0.96	600
Weighted Avg	0.96	0.96	0.96	600

Table 14: Classification Report for BERT using a scheduler on the Test set (max_seq_len=256, learning_rate=0.0001, batch_size=16, epochs=2)

Class	Precision	Recall	F1-Score	Support
Books	0.93	0.91	0.92	90
Camera	0.83	0.96	0.89	112
DVD	0.91	0.85	0.88	101
Health	0.98	0.87	0.92	94
Music	0.92	0.95	0.94	114
Software	0.90	0.89	0.89	89
Accuracy		0.91 (600)		
Macro Avg	0.91	0.90	0.91	600
Weighted Avg	0.91	0.91	0.91	600

Table 15: Classification Report for RoBERTa-base

Class	Precision	Recall	F1-Score	Support
Books	0.90	0.93	0.92	90
Camera	0.87	0.86	0.86	112
DVD	0.92	0.82	0.87	101
Health	0.89	0.86	0.88	94
Music	0.92	0.94	0.93	114
Software	0.78	0.88	0.83	89
Accuracy		0.88 (600)		
Macro Avg	0.88	0.88	0.88	600
Weighted Avg	0.88	0.88	0.88	600

Table 16: Classification Report for RoBERTa-large

Class	Precision	Recall	F1-Score	Support
Books	0.89	0.92	0.91	90
Camera	0.86	0.95	0.90	112
DVD	0.94	0.89	0.91	101
Health	0.96	0.84	0.90	94
Music	0.89	0.96	0.93	114
Software	0.89	0.83	0.86	89
Accuracy		0.90 (600)		
Macro Avg	0.91	0.90	0.90	600
Weighted Avg	0.91	0.90	0.90	600

Table 17: Classification Report for DistilBERT-base-uncased

Class	Precision	Recall	F1-Score	Support
Books	0.81	0.93	0.87	90
Camera	0.82	0.76	0.79	112
DVD	0.73	0.83	0.78	101
Health	1.00	0.47	0.64	94
Music	0.93	0.94	0.93	114
Software	0.69	0.91	0.78	89
Accuracy		0.81 (600)		
Macro Avg	0.83	0.81	0.80	600
Weighted Avg	0.83	0.81	0.80	600

Table 18: Classification Report for CamemBERT-base

Class	Precision	Recall	F1-Score	Support
Books	0.93	0.91	0.92	90
Camera	0.98	0.78	0.87	112
DVD	0.90	0.92	0.91	101
Health	0.88	0.86	0.87	94
Music	0.91	0.94	0.92	114
Software	0.74	0.91	0.81	89
Accuracy		0.89 (600)		
Macro Avg	0.89	0.89	0.88	600
Weighted Avg	0.89	0.89	0.89	600

Table 19: Classification Report for ALBERT-base-v2

Class	Precision	Recall	F1-Score	Support
Books	0.91	0.93	0.92	90
Camera	0.91	0.88	0.90	112
DVD	0.95	0.87	0.91	101
Health	0.93	0.93	0.93	94
Music	0.92	0.96	0.94	114
Software	0.85	0.90	0.87	89
Accuracy		0.91 (600)		
Macro Avg	0.91	0.91	0.91	600
Weighted Avg	0.91	0.91	0.91	600

Table 20: Classification Report for DeBERTa-base

Class	Precision	Recall	F1-Score	Support
Books	0.86	0.92	0.89	90
Camera	0.96	0.84	0.90	112
DVD	0.92	0.92	0.92	101
Health	0.88	0.89	0.88	94
Music	0.95	0.94	0.94	114
Software	0.83	0.89	0.86	89
Accuracy		0.90 (600)		
Macro Avg	0.90	0.90	0.90	600
Weighted Avg	0.90	0.90	0.90	600

Table 21: Classification Report for BERT-large-uncased

Class	Precision	Recall	F1-Score	Support
Books	0.91	0.94	0.93	90
Camera	0.93	0.84	0.88	112
DVD	0.96	0.93	0.94	101
Health	0.91	0.91	0.91	94
Music	0.94	0.98	0.96	114
Software	0.85	0.90	0.87	89
Accuracy		0.92 (600)		
Macro Avg	0.92	0.92	0.92	600
Weighted Avg	0.92	0.92	0.92	600

Table 22: Classification Report for FinBERT

Class	Precision	Recall	F1-Score	Support
Books	0.91	0.93	0.92	90
Camera	0.97	0.68	0.80	112
DVD	0.90	0.86	0.88	101
Health	0.80	0.91	0.86	94
Music	0.91	0.94	0.92	114
Software	0.77	0.93	0.84	89
Accuracy		0.87 (600)		
Macro Avg	0.88	0.88	0.87	600
Weighted Avg	0.88	0.87	0.87	600

Table 23: Classification Report for TinyBERT