# Linear Classifier: An Often-Forgotten Baseline for Text Classification

Yu-Chen Lin[1,2,3], Si-An Chen[1], Jie-Jyun Liu[1], and Chih-Jen Lin[1,3]

[1]National Taiwan University
[2]ASUS Intelligent Cloud Services
[3]Mohamed bin Zayed University of Artificial Intelligence
{b06504025,d09922007,d11922012,cjlin}@csie.ntu.edu.tw

## Abstract

Large-scale pre-trained language models such as BERT are popular solutions for text classification. Due to the superior performance of these advanced methods, nowadays, people often directly train them for a few epochs and deploy the obtained model. In this opinion paper, we point out that this way may only sometimes get satisfactory results. We argue the importance of running a simple baseline like linear classifiers on bag-of-words features along with advanced methods. First, for many text data, linear methods show competitive performance, high efficiency, and robustness. Second, advanced models such as BERT may only achieve the best results if properly applied. Simple baselines help to confirm whether the results of advanced models are acceptable. Our experimental results fully support these points.

## 1 Introduction

Text classification is an essential topic in natural language processing (NLP). Like the situations in most NLP tasks, nowadays, large-scale pre-trained language models (PLMs) such as BERT (Devlin et al., 2019) have become popular solutions for text classification. Therefore, we have seen that many practitioners directly run pre-trained language models with a fixed number of epochs on their text data. Unfortunately, this way may only sometimes lead to satisfactory results. In this opinion paper, through an intriguing illustration, we argue that for text classification, a simple baseline like linear classifiers on bag-of-words features should be used along with the advanced models for the following reasons.

- Training linear classifiers such as linear SVM (Boser et al., 1992) or logistic regression on bag-of-words features is simple and efficient. This approach may give competitive performance to advanced models for some problems. While various settings of bag-of-words features such as bi-gram or tri-gram can be considered, we advocate that simple uni-gram TF-IDF features trained by linear classifiers can be a useful baseline to start with for text classification.
- Advanced architectures such as BERT may only achieve the best results if properly used. Linear methods can help us check if advanced methods' results are reasonable.

In the deep-learning era, the younger generation often thinks that linear classifiers should never be considered. Further, they may be unaware of some variants of linear methods that are particularly useful for text classification (see Section 3.1). Therefore, the paper serves as a reminder of this often-forgotten technique.

For our illustration, we re-investigate an existing work (Chalkidis et al., 2022) that evaluates both linear SVM and pre-trained language models, but the authors pay more attention to the latter. The linear method is somewhat ignored even though the performance is competitive on some problems. We carefully design experiments to compare the two types of methods. Our results fully demonstrate the usefulness of applying linear methods as simple baselines.

Some recent works (e.g., Yu et al., 2022; Gomes et al., 2021) have shown the usefulness of linear classifiers in the deep-learning era. However, they either consider sophisticated applications or investigate advanced settings in which linear methods are only one component. In contrast, in this paper, we consider the basic scenario of text classification. A more related work (Wahba et al., 2023) has demonstrated the effectiveness of linear classifiers over PLMs on some problems. However, our investigation on linear methods is more comprehensive.

The discussion also reminds us the trade-off between performance gain and the cost including running time, model size, etc. Simple methods are useful to benchmark and justify the usage of advanced methods.

*Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*
*Volume 2: Short Papers*, pages 1876–1888
July 9-14, 2023 ©2023 Association for Computational Linguistics

| Method | ECtHR (A) | | ECtHR (B) | | SCOTUS | | EUR-LEX | | LEDGAR | | UNFAIR-ToS | | # |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | μ-$F_1$ | $T$ | μ-$F_1$ | $T$ | μ-$F_1$ | $T$ | μ-$F_1$ | $T$ | μ-$F_1$ | $T$ | μ-$F_1$ | $T$ | params |
| TF-IDF+SVM | 64.5 | N/A | 74.6 | N/A | **78.2** | N/A | 71.3 | N/A | 87.2 | N/A | 95.4 | N/A | N/A |
| BERT | **71.2** | 3h 42m | 79.7 | 3h 9m | 68.3 | 1h 24m | 71.4 | 3h 36m | 87.6 | 6h 9m | 95.6 | N/A | 110M |
| RoBERTa | 69.2 | 4h 11m | 77.3 | 3h 43m | 71.6 | 2h 46m | 71.9 | 3h 36m | 87.9 | 6h 22m | 95.2 | N/A | 125M |
| DeBERTa | 70.0 | 7h 43m | 78.8 | 6h 48m | 71.1 | 3h 42m | **72.1** | 5h 34m | 88.2 | 9h 29m | 95.5 | N/A | 139M |
| Longformer | 69.9 | 6h 47m | 79.4 | 7h 31m | 72.9 | 6h 27m | 71.6 | 11h 10m | 88.2 | 15h 47m | 95.5 | N/A | 149M |
| BigBird | 70.0 | 8h 41m | 78.8 | 8h 17m | 72.8 | 5h 51m | 71.5 | 3h 57m | 87.8 | 8h 13m | 95.7 | N/A | 127M |
| Legal-BERT | 70.0 | 3h 52m | **80.4** | 3h 2m | 76.4 | 2h 2m | **72.1** | 3h 22m | 88.2 | 5h 23m | **96.0** | N/A | 110M |
| CaseLaw-BERT | 69.8 | 3h 2m | 78.8 | 2h 57m | 76.6 | 2h 34m | 70.7 | 3h 40m | **88.3** | 6h 8m | **96.0** | N/A | 110M |

Table 1: Micro-F1 scores (μ-$F_1$), training time ($T$) and number of parameters presented in Chalkidis et al. (2022). In each Micro-F1 column, the best result is bold-faced. "N/A" means not available in their work. For example, the authors did not report the training time and the number of parameters of linear SVMs.

This paper is organized as follows. In Section 2 we take a case study to point out the needs of considering linear methods as a baseline for text classification. We describe the linear and BERT-based methods used for investigation in Section 3. The experimental results and main findings are in Section 4, while Section 5 provides some discussion. Additional details are in Appendix. Programs used for experiments are available at `https://github.com/JamesLYC88/text_classification_baseline_code`.

## 2 Text Classification These Days: Some Issues in Applying Training Methods

Large PLMs have shown dramatic progress on various NLP tasks. In the practical use, people often directly fine-tune PLMs such as BERT on their data for a few epochs. However, for text classification, we show that this way may not always get satisfactory results. Some simple baselines should be considered to know if the obtained PLM model is satisfactory. We illustrate this point by considering the work on legal document classification by Chalkidis et al. (2022), which evaluates the following sets.

- Multi-class classification: SCOTUS, LEDGAR; for this type of sets, each text is associated with a single label.
- Multi-label classification: ECtHR (A), ECtHR (B), EUR-LEX, UNFAIR-ToS; for this type of sets, each text is associated with multiple (or zero) labels.
- Multiple choice QA: CaseHOLD.

We focus on text classification in this work, so CaseHOLD is not considered. For each problem, training and test sets are available.[1]

The study in Chalkidis et al. (2022) comprehensively evaluates both BERT-based PLMs and linear SVMs. They use Micro-F1 and Macro-F1 to measure the test performance.[2] In Table 1, we present their Micro-F1 results and running time of each model.

### 2.1 Linear Models Worth More Investigation

The investigation in Chalkidis et al. (2022) focuses on BERT and its variants, even though from Table 1, the performance of BERT-based methods may not differ much. While they did not pay much attention to linear SVM, by a closer look at the results, we get intriguing observations:

- Linear SVM is competitive to BERT-based PLMs on four of the six data sets. For SCOTUS, linear SVM even outperforms others with a clear gap.
- Surprisingly, given linear SVM's decent performance, its training time was not shown in Chalkidis et al. (2022), nor was the number of parameters; see the "N/A" entries in Table 1.

With the observations, we argue that the results of linear models are worth more investigation.

## 3 Settings for Investigation

To better understand the performance of linear models and BERT-based PLMs, we simulate how people work on a new data set by training these methods. We consider a text classification package LibMultiLabel[3] because it supports both types of train-

[1] Indeed, training, validation, and test sets are available. See details in Appendix H about how these sets are used.

[2] Some data instances are not associated with any labels; see Appendix A about how Chalkidis et al. (2022) handle such situations.

[3] `https://github.com/ASUS-AICS/LibMultiLabel`

ing methods.

### 3.1 Linear Methods for Text Classification

To use a linear method, LibMultiLabel first generates uni-gram TF-IDF features (Luhn, 1958; Jones, 1972) according to texts in the training set, and the obtained factors are used to get TF-IDF for the test set. It then provides three classic methods that adopt binary linear SVM and logistic regression for multi-class and multi-label scenarios.[4] Here we consider linear SVM as the binary classifier behind these methods.

- One-vs-rest: This method learns a binary linear SVM for each label, so data with/without this label are positive/negative, respectively. Let $f_\ell(\boldsymbol{x})$ be the decision value of the $\ell$-th label, where $\boldsymbol{x}$ is the feature vector. For multi-class classification, $\hat{y} = \operatorname{argmax}_\ell f_\ell(\boldsymbol{x})$ is predicted as the single associated label of $\boldsymbol{x}$. For multi-label classification, all labels $\ell$ with positive $f_\ell(\boldsymbol{x})$ are considered to be associated with $\boldsymbol{x}$. This method is also what "TF-IDF+SVM" in Chalkidis et al. (2022) did, though our TF-IDF feature generation is simpler than theirs by considering only uni-gram.[5]
- Thresholding (Yang, 2001; Lewis et al., 2004; Fan and Lin, 2007): This method extends one-vs-rest by modifying the decision value for optimizing Macro-F1. That is, we change the decision value to $f_\ell(\boldsymbol{x}) + \Delta_\ell$, where $\Delta_\ell$ is a threshold decided by cross validation.
- Cost-sensitive (Parambath et al., 2014): For each binary problem, this method re-weights the losses on positive data. We decide the re-weighting factor by cross validation to optimize Micro-F1 or Macro-F1.

These methods basically need no further hyper-parameter tuning, so we can directly run them. The last two methods are extensions of one-vs-rest to address the imbalance of each binary problem (i.e., few positives and many negatives). The design relies on the fact that the binary problems are independent, so such approaches cannot be easily applied to deep learning, which considers all labels together in a single network.

### 3.2 BERT-based Methods for Text Classification

LibMultiLabel also provides BERT-based methods, which involve several hyper-parameters, such as the learning rate. While practitioners may directly choose hyper-parameters, to seriously compare with linear methods, we run BERT by conducting hyper-parameter selection. More details are in Appendix F.

## 4 Experimental Results and Analysis

In Table 2, we follow Chalkidis et al. (2022) to report Micro-F1 and Macro-F1 on the test set. The training time is in Table 3.

### 4.1 Linear Methods are Good Baselines

In Table 2, our one-vs-rest results are slightly worse than the linear SVM results in Chalkidis et al. (2022), which also applies the one-vs-rest strategy. As mentioned in Section 3.1, the difference is mainly due to our use of simple uni-gram TF-IDF features. Anyway, our one-vs-rest is still competitive to BERT results in Chalkidis et al. (2022) on the last four problems.

More importantly, the two extensions of one-vs-rest (i.e., thresholding and cost-sensitive) improve almost all situations. For data sets ECtHR (A) and ECtHR (B), where originally one-vs-rest is significantly lower than BERT results in Chalkidis et al. (2022), the gap reduced considerably.

For the training time in Table 3, though the two extensions take more time than the basic one-vs-rest strategy, all the linear methods are still hundreds of times faster than BERT. Further, linear methods were run on a CPU (Intel Xeon E5-2690), while for BERT we need a GPU (Nvidia V100). The model sizes listed in Table 4 also show that linear SVM requires a much smaller model than BERT, where details of our calculation are in Appendix D.

The results demonstrate that linear methods are useful baselines. They are extremely simple and efficient, but may yield competitive test performance.

### 4.2 Linear Methods can Help to See if Advanced Methods Are Properly Used

Surprisingly, our running of LibMultiLabel's BERT leads to worse test performance than linear methods on almost all data sets. More surprisingly, a comparison between the BERT results by LibMultiLabel and those in Chalkidis et al. (2022) shows

[4] Besides descriptions in this section, some additional details are in Appendix B.

[5] See details in Appendix C.

| Method | ECtHR (A) | | ECtHR (B) | | SCOTUS | | EUR-LEX | | LEDGAR | | UNFAIR-ToS | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | μ-F$_1$ | m-F$_1$ | μ-F$_1$ | m-F$_1$ | μ-F$_1$ | m-F$_1$ | μ-F$_1$ | m-F$_1$ | μ-F$_1$ | m-F$_1$ | μ-F$_1$ | m-F$_1$ |
| Linear | | | | | | | | | | | | |
| one-vs-rest | 64.0 | 53.1 | 72.8 | 63.9 | 78.1 | 68.9 | 72.0 | 55.4 | 86.4 | 80.0 | 94.9 | 75.1 |
| thresholding | 68.6 | **64.9** | 76.1 | 68.7 | **78.9** | **71.5** | **74.7** | **62.7** | 86.2 | 79.9 | 95.1 | 79.9 |
| cost-sensitive | 67.4 | 60.5 | 75.5 | 67.3 | 78.3 | **71.5** | 73.4 | 60.5 | 86.2 | 80.1 | 95.3 | 77.9 |
| Chalkidis et al. (2022) | 64.5 | 51.7 | 74.6 | 65.1 | 78.2 | 69.5 | 71.3 | 51.4 | 87.2 | **82.4** | 95.4 | 78.8 |
| BERT | | | | | | | | | | | | |
| Ours | 61.9 | 55.6 | 69.8 | 60.5 | 67.1 | 55.9 | 70.8 | 55.3 | 87.0 | 80.7 | 95.4 | 80.3 |
| Chalkidis et al. (2022) | **71.2** | 63.6 | **79.7** | **73.4** | 68.3 | 58.3 | 71.4 | 57.2 | **87.6** | 81.8 | **95.6** | **81.3** |

Table 2: Micro-F1 (μ-F$_1$) and Macro-F1 scores (m-F$_1$) for our investigation on two types of approaches: linear SVM and BERT. For each type, we show results achieved by LibMultiLabel and scores reported in Chalkidis et al. (2022). In each column, the best result is bold-faced.

| Method | ECtHR (A) | ECtHR (B) | SCOTUS | EUR-LEX | LEDGAR | UNFAIR-ToS |
|---|---|---|---|---|---|---|
| Linear | | | | | | |
| one-vs-rest | 28s | 29s | 1m 11s | 4m 2s | 28s | 2s |
| thresholding | 59s | 1m 0s | 2m 11s | 28m 8s | 3m 26s | 3s |
| cost-sensitive | 1m 38s | 1m 43s | 3m 28s | 50m 36s | 4m 45s | 4s |
| Chalkidis et al. (2022) | N/A | N/A | N/A | N/A | N/A | N/A |
| BERT | | | | | | |
| Ours | 5h 8m | 5h 51m | 3h 21m | 38h 14m | 43h 48m | 4h 5m |
| Chalkidis et al. (2022) | 3h 42m | 3h 9m | 1h 24m | 3h 36m | 6h 9m | N/A |

Table 3: Training time for our multiple settings on linear SVM and BERT. We show results from running LibMultiLabel and values reported in Chalkidis et al. (2022). Note that Chalkidis et al. (2022) use fixed parameters for BERT, while for our BERT, we use 4 GPUs to conduct the hyper-parameter search and report the total time used.

| Method | ECtHR (A) | ECtHR (B) | SCOTUS | EUR-LEX | LEDGAR | UNFAIR-ToS |
|---|---|---|---|---|---|---|
| Linear | 924K | 924K | 2M | 15M | 2M | 50K |
| BERT variants | 110M $\sim$ 149M | | | | | |

Table 4: A comparison between the model size of linear methods and BERT variants. Note that all three linear methods in LibMultiLabel have the same model size. For BERT variants, we borrow the calculation in Table 1 by Chalkidis et al. (2022). More details are in Appendix D.

that the former is much worse on data sets ECtHR (A) and ECtHR (B). Interestingly, from Section 4.1, only for these two sets the BERT results in Chalkidis et al. (2022) are much better than linear methods. Thus, our direct run of BERT in LibMultiLabel is a total failure. The training time is much longer than linear methods, but the resulting model is worse.

It is essential to check the discrepancy between the two BERT results. We find that Chalkidis et al. (2022) use some sophisticated settings to run BERT for the first three sets (i.e., ECtHR (A), ECtHR (B), and SCOTUS). They split every document into 64 segments, each of which has no more than 128 tokens, and apply BERT on each segment. Then, they collect the intermediate results as inputs to an upper-level transformer. After repeating the same process via LibMultiLabel, we can reproduce the results in Chalkidis et al. (2022); see details in Appendices E, F, and G.

We learned that they considered the more sophisticated setting of running BERT because by default, BERT considers only the first 512 tokens. Thus, for long documents, the training process may miss some important information. However, in practice, users may forget to check the document length and are not aware of the need to apply suitable settings. The above experiments demonstrate that BERT can achieve superior results if properly used, but sometimes, a direct run lead to poor outcomes. Linear methods can serve as efficient and robust baselines to confirm the proper use of an advanced approach.

## 5 Discussion and Conclusions

In our experiments, we encounter an issue of whether to incorporate the validation set for training the final model, which is used for predicting the test set. For linear methods, we follow the common practice to include the validation set for obtaining the final model. However, for BERT or some other deep learning models, the validation set is often used only for selecting the best epoch and/or the best hyper-parameters. To fully use the available data, we have investigated how to incorporate the validation set for BERT. Experimental results and more details are in Appendix H.

For some text sets evaluated in this work, we have seen that simple linear methods give competitive performance. The reason might be that each document in these sets is not short.[6] Then TF-IDF features are sufficiently informative so that linear methods work well. Across all NLP areas, an important issue now is when to use PLMs and when not. We demonstrate that when PLMs may not perform significantly better, traditional methods are much simpler and require fewer resources. However, having a simple quantitative measurement to pre-determine when to use which remains a challenging future research problem. In summary, the study reminds us of the importance of employing simple baselines in NLP applications.

[6] See details in Appendix Table 6.

## Limitations

In this work, we do not propose any new methods because, as an opinion paper, we focus on raising the problems and making vivid demonstrations to readers. The experiments are limited to linear SVM and BERT on data sets in the benchmark LexGLUE. We hope that, within the page limit, our experiments sufficiently convey the points to readers.

## Ethics Statement

We ensure that our work complies with the ACL Ethics Policy.

## Acknowledgements

This work was supported by NSTC of Taiwan grant 110-2221-E-002-115-MY3 and ASUS Intelligent Cloud Services. The authors thank Ming-Wei Chang and reviewers for constructive comments.

## References

Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. 1992. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152. ACM Press.

Ilias Chalkidis, Abhik Jana, Dirk Hartung, Michael Bommarito, Ion Androutsopoulos, Daniel Katz, and Nikolaos Aletras. 2022. LexGLUE: A benchmark dataset for legal language understanding in English. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 4310–4330.

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, NAACL-HLT*, pages 4171–4186.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: a library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Rong-En Fan and Chih-Jen Lin. 2007. A study on threshold selection for multi-label classification. Technical report, Department of Computer Science, National Taiwan University.

Christian Gomes, Marcos André Gonçalves, Leonardo Rocha, and Sérgio D. Canuto. 2021. On the cost-effectiveness of stacking of neural and non-neural methods for text classification: Scenarios and performance prediction. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP*, pages 4003–4014.

Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. The MIT Press.

Karen S. Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–20.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397.

Li-Chung Lin, Cheng-Hung Liu, Chih-Ming Chen, Kai-Chin Hsu, I-Feng Wu, Ming-Feng Tsai, and Chih-Jen Lin. 2022. On the use of unrealistic predictions in hundreds of papers evaluating graph representations. In *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI)*.

Hans Peter Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165.

Shameem A. Puthiya Parambath, Nicolas Usunier, and Yves Grandvalet. 2014. Optimizing f-measures by cost-sensitive classification. In *Advances in Neural Information Processing Systems*, volume 27.

Yasmen Wahba, Nazim Madhavji, and John Steinbacher. 2023. A comparison of svm against pre-trained language models (plms) for text classification tasks. In *Machine Learning, Optimization, and Data Science*, pages 304–313. Springer Nature Switzerland.

Yiming Yang. 2001. A study on thresholding strategies for text categorization. In *Proceedings of the 24th ACM International Conference on Research and Development in Information Retrieval*, pages 137–145, New Orleans, US. ACM Press, New York, US.

Hsiang-Fu Yu, Kai Zhong, Jiong Zhang, Wei-Cheng Chang, and Inderjit S. Dhillon. 2022. PECOS: Prediction for enormous and correlated output spaces. *Journal of Machine Learning Research*, 23(98):1–32.

## A Issue about Data without Labels

For multi-label problems considered in Chalkidis et al. (2022), instances that are not associated with any labels, called unlabeled instances as follows, account for a considerable portion in some data sets: ECtHR (A) (11.3%), ECtHR (B) (1.6%) and UNFAIR-ToS (89.0%). In the training process, Chalkidis et al. (2022) keep the unlabeled

| Parameter | LibMultiLabel | Chalkidis et al. (2022) |
|---|---|---|
| Data pre-processing (TF-IDF feature generation) | | |
| stop_words | None | english |
| ngram_range | (1, 1) | (1, 3) |
| min_df | 1 | 5 |
| max_features | None | [10000, 20000, 40000] |
| Model | | |
| loss | squared_hinge | ['hinge', 'squared_hinge'] |
| solving primal/dual | primal | dual |
| $C$ | 1.0 | [0.1, 1.0, 10.0] |

Table 5: Key differences in the one-vs-rest linear method between the default setting of LibMultiLabel and the implementation in Chalkidis et al. (2022). Any values covered by [] mean the hyper-parameter search space. See Appendix C for details of hyper-parameters.

training instances without any modification. Thus, in, for example, the one-vs-rest setting described in Section 3.1, an unlabeled instance is on the negative side in every binary problem. However, in evaluating the validation and test sets, they introduce an additional class to indicate the unlabeled data. Specifically, an unlabeled instance is associated with this "unlabeled" class, but not others. Chalkidis et al. (2022) consider this way to more seriously evaluate the model predictability on unlabeled instances. However, this setting is not a standard practice in multi-label classification, nor is it supported by LibMultiLabel. Thus we modify the scripts in LibMultiLabel to have the same evaluation setting as Chalkidis et al. (2022).

## B Additional Details of Linear Methods

The binary linear SVM is in the following form.

$$\min_{\boldsymbol{w}} \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + C \sum_i \xi(y_i \boldsymbol{w}^\top \boldsymbol{x_i}), \qquad (1)$$

where $(\boldsymbol{x_i}, y_i)$ are data-label pairs in the data set, $y_i = \pm 1$, $\boldsymbol{w}$ is the parameters of the linear model, and $\xi(\cdot)$ is the loss function. The decision value function is $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x}$.

For one-vs-rest, please see descriptions in Section 3.1. We follow the default setting in LibMultiLabel by using $C = 1$. For more details about thresholding and cost-sensitive, please refer to the explanations in Lin et al. (2022).

## C Differences Between Our Implementation of Linear Methods and Chalkidis et al. (2022)

We summarize the implementation differences between LibMultiLabel and Chalkidis et al. (2022) in Table 5.

For the data-preprocessing part, both use scikit-learn for TF-IDF feature generations. The meanings of each parameter are listed as follows.

**stop_words:** Specify the list of stop words to be removed. For example, Chalkidis et al. (2022) set stop_words to "english," so tokens that include in the "english" list are filtered.

**ngram_range:** Specify the range of n-grams to be extracted. For example, LibMultiLabel only uses uni-gram, while Chalkidis et al. (2022) set ngram_range to (1, 3), so uni-gram, bi-gram, and tri-gram are extracted into the vocabulary list for a richer representation of the document.

**min_df:** The parameter is used for removing infrequent tokens. Chalkidis et al. (2022) remove tokens that appear in less than five documents, while LibMultiLabel does not remove any tokens.

**max_features:** The parameter decides the number of features to use by term frequency. For example, Chalkidis et al. (2022) consider the top 10,000, 20,000, and 40,000 frequent terms as the search space of the parameter.

For more detailed explanations, please refer to the TfidfVectorizer function in scikit-learn.

The binary classification problem in (1) is referred to as the primal form. The optimization problem can be transferred to the dual form and the optimal solutions of the two forms lead to the same decision function. Thus we can choose to solve the primal or the dual problem; see Table 5. For the model training, they both use the solver provided by LIBLINEAR (Fan et al., 2008).

| Property | ECtHR (A) | ECtHR (B) | SCOTUS | EUR-LEX | LEDGAR | UNFAIR-ToS |
|---|---|---|---|---|---|---|
| # labels | 10 | 10 | 13 | 100 | 100 | 8 |
| $\overline{W}$ | 1,662.08 | 1,662.08 | 6,859.87 | 1,203.92 | 112.98 | 32.70 |
| # features | 92,402 | 92,402 | 126,406 | 147,465 | 19,997 | 6,291 |

Table 6: Data statistics for LexGLUE, the benchmark considered in Chalkidis et al. (2022). $\overline{W}$ means the average # words per instance of the whole set. The # features indicates the # TF-IDF features used by linear methods.

| Parameter | LibMultiLabel | | | | Chalkidis et al. (2022) |
|---|---|---|---|---|---|
| | | | reproduced | | |
| | default | tuned | SCOTUS<br>LEDGAR | other<br>problems | |
| maximum #epochs | 15 | 15 | 20 | 15 | 20 |
| weight_decay | 0.001 | 0 | 0 | 0 | 0 |
| patience | 5 | 5 | 5 | 5 | 3 |
| val_metric | Micro-F1 | Micro-F1 | Micro-F1 | Micro-F1 | Micro-F1 |
| early_stopping_metric | Micro-F1 | Micro-F1 | loss | Micro-F1 | loss |
| learning_rate | 5e-5 | See Table 8 | 3e-5 | 3e-5 | 3e-5 |
| dropout | 0.1 | | 0.1 | 0.1 | 0.1 |

Table 7: Parameter differences of BERT between LibMultiLabel and Chalkidis et al. (2022). For the meaning of each parameter, please refer to the software LibMultiLabel.

| Parameter | | ECtHR (A) | ECtHR (B) | SCOTUS | EUR-LEX | LEDGAR | UNFAIR-ToS |
|---|---|---|---|---|---|---|---|
| max_seq_length | space | [128, 512] | | | | | |
| | selected | 512 | 512 | 512 | 512 | 512 | 512 |
| learning_rate | space | [2e-5, 3e-5, 5e-5] | | | | | |
| | selected | 2e-5 | 3e-5 | 2e-5 | 5e-5 | 2e-5 | 3e-5 |
| dropout | space | [0.1, 0.2] | | | | | |
| | selected | 0.1 | 0.2 | 0.1 | 0.1 | 0.2 | 0.1 |

Table 8: Hyper-parameter search space and the selected values of LibMultiLabel's tuned setting.

# D Additional Details about Model Size

We calculate the model size of linear SVM by multiplying the number of TF-IDF features by the number of labels; see details in Table 6. For BERT, we directly copy the number of parameters from Chalkidis et al. (2022).

# E Additional Details about BERT Design in Chalkidis et al. (2022)

## E.1 Standard BERT for Classification

The setting considers the original implementation in Devlin et al. (2019). They truncate the documents to have at most 512 tokens. We then take a pre-trained BERT appended with an additional linear layer for fine-tuning.

## E.2 Document Lengths

In Table 6, we present the document length for each data set in LexGLUE, the benchmark considered in Chalkidis et al. (2022). For ECtHR (A), ECtHR (B), SCOTUS, and EUR-LEX, the document lengths all exceed 512, the length limitation of BERT. Note that the numbers are underestimated because BERT uses a sub-word tokenizer that further tokenizes some words into sub-words.

## E.3 Hierarchical BERT

Chalkidis et al. (2022) design a variant of the standard BERT for ECtHR (A), ECtHR (B), and SCOTUS to deal with long document lengths. The detailed steps are as follows.

- Each document is split into 64 segments, where each segment contains at most 128 tokens.
- Each segment is then fed into BERT.
- The [CLS] tokens generated from each segment

| Method | ECtHR (A) | | ECtHR (B) | | SCOTUS | | EUR-LEX | | LEDGAR | | UNFAIR-ToS | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | µ-F$_1$ | m-F$_1$ | µ-F$_1$ | m-F$_1$ | µ-F$_1$ | m-F$_1$ | µ-F$_1$ | m-F$_1$ | µ-F$_1$ | m-F$_1$ | µ-F$_1$ | m-F$_1$ |
| BERT in LibMultiLabel | | | | | | | | | | | | |
| default | 60.5 | 53.4 | 68.9 | 60.8 | 66.3 | 54.8 | 70.8 | 55.3 | 85.2 | 77.9 | 95.2 | 78.2 |
| tuned | 61.9 | 55.6 | 69.8 | 60.5 | 67.1 | 55.9 | 70.8 | 55.3 | 87.0 | 80.7 | 95.4 | 80.3 |
| **reproduced** | 70.2 | 63.7 | 78.8 | 73.1 | 70.8 | 62.6 | 71.6 | 56.1 | 88.1 | 82.6 | 95.3 | 80.6 |
| BERT in Chalkidis et al. (2022) | | | | | | | | | | | | |
| paper | 71.2 | 63.6 | 79.7 | 73.4 | 68.3 | 58.3 | 71.4 | 57.2 | 87.6 | 81.8 | 95.6 | 81.3 |
| **reproduced** | 70.8 | 64.8 | 78.7 | 72.5 | 70.9 | 61.9 | 71.7 | 57.9 | 87.7 | 82.1 | 95.6 | 80.3 |

Table 9: Micro-F1 (µ-F$_1$) and Macro-F1 scores (m-F$_1$) for our investigation on BERT.

| Method | ECtHR (A) | ECtHR (B) | SCOTUS | EUR-LEX | LEDGAR | UNFAIR-ToS |
|---|---|---|---|---|---|---|
| BERT in LibMultiLabel | | | | | | |
| default | 59m 48s | 1h 2m | 39m 49s | 6h 38m | 8h 44m | 47m 48s |
| tuned | 5h 8m | 5h 51m | 3h 21m | 38h 14m | 43h 48m | 4h 5m |
| **reproduced** | 10h 27m | 9h 41m | 9h 26m | 6h 37m | 5h 49m | 15m 9s |
| BERT in Chalkidis et al. (2022) | | | | | | |
| paper | 3h 42m | 3h 9m | 1h 24m | 3h 36m | 6h 9m | N/A |
| **reproduced** | 7h 56m | 6h 59m | 7h 5m | 4h 30m | 5h 11m | 7m 3s |

Table 10: Training time for our multiple settings on BERT. The average time of running five seeds is reported.

are collected and fed into an upper-level transformer encoder.

- Max pooling is applied to the output of the transformer encoder.
- The pooled results are then fed into a linear layer for the final prediction.

## F Differences between the Two BERT Implementations

We summarize the implementation differences of BERT between LibMultiLabel and Chalkidis et al. (2022) in Table 7. Here we also try to reproduce results in Chalkidis et al. (2022) by using LibMultiLabel.

For LibMultiLabel, we explain our choices of hyper-parameters as follows.

**default:** This method references the parameters chosen in an example configuration[7] from LibMultiLabel.

**tuned:** This method performs a parameter search and is marked as "our BERT" in the main paper; see Table 8 for the search space and the chosen values.

**reproduced:** This method aims to reproduce the BERT results from Chalkidis et al. (2022) using LibMultiLabel. We begin with imposing the same weight_decay, learning_rate, and dropout values as Chalkidis et al. (2022) and also the same validation metric. However, for other parameters, which may less affect the results, we use the same values as **default** and **tuned**; see Table 7. Except SCOTUS and LEDGAR, we were able to generate similar results to those in Chalkidis et al. (2022). To fully reproduce the results on SCOTUS and LEDGAR, we try to follow every setting did in Chalkidis et al. (2022). Specifically, we replace the PyTorch trainer originally used in LibMultiLabel with the Hugging Face trainer adopted in Chalkidis et al. (2022) and align some of the parameters with the ones used in Chalkidis et al. (2022); see a column in Table 7 for these two sets.

LibMultiLabel supports standard BERT discussed in Appendix E.1. For the "default" and "tuned" settings, we directly run standard BERT. For the "reproduced" method, we follow Chalkidis et al. (2022) to use hierarchical BERT explained in Appendix E.3 for ECtHR (A), ECtHR (B), and SCOTUS and use standard BERT for other data sets.

## G Detailed BERT Results

In Tables 9 and 10, we respectively present the test performance and the training time. For settings of running LibMultiLabel, see Appendix F. For BERT

[7] https://github.com/ASUS-AICS/LibMultiLabel/blob/master/example_config/EUR-Lex-57k/bert.yml

| Method | ECtHR (A) | | ECtHR (B) | | SCOTUS | | EUR-LEX | | LEDGAR | | UNFAIR-ToS | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\mu$-F$_1$ | m-F$_1$ | $\mu$-F$_1$ | m-F$_1$ | $\mu$-F$_1$ | m-F$_1$ | $\mu$-F$_1$ | m-F$_1$ | $\mu$-F$_1$ | m-F$_1$ | $\mu$-F$_1$ | m-F$_1$ |
| BERT in LibMultiLabel | | | | | | | | | | | | |
| default | 60.5 | 53.4 | 68.9 | 60.8 | 66.3 | 54.8 | 70.8 | 55.3 | 85.2 | 77.9 | 95.2 | 78.2 |
| tuned | 61.9 | 55.6 | 69.8 | 60.5 | 67.1 | 55.9 | 70.8 | 55.3 | 87.0 | 80.7 | 95.4 | 80.3 |
| BERT in LibMultiLabel (re-trained) | | | | | | | | | | | | |
| default | 63.0 | 56.1 | 69.6 | 62.8 | 69.5 | 58.8 | 75.6 | 59.2 | 85.3 | 78.4 | 94.0 | 65.4 |
| tuned | 62.4 | 55.9 | 70.3 | 62.3 | 71.4 | 61.9 | 75.6 | 59.2 | 87.2 | 81.5 | 95.2 | 79.8 |

Table 11: A performance comparison between the setting without and with re-training.

in Chalkidis et al. (2022), we present the following two results.

**paper:** Results in the paper by Chalkidis et al. (2022) are directly copied.

**reproduced:** Results from our running of their scripts.[8]

For ECtHR (A), ECtHR (B), and SCOTUS, because there exist some issues when running the fp16 setting in our environment, we run the code of Chalkidis et al. (2022) by using fp32 instead. This change causes the time difference between the "paper" and "reproduced" settings in Table 10.

Except numbers borrowed from Chalkidis et al. (2022), we run five seeds for all BERT experiments and report the mean test performance over all seeds. Chalkidis et al. (2022) also run five seeds, but their test scores are based on the top three seeds with the best Macro-F1 on validation data.

For the "tuned" setting, because the version of LibMultiLabel that we used does not store the checkpoint after hyper-parameter search, we must conduct the training again using the best hyper-parameters. Thus, the total time includes hyper-parameter search and the additional training.[9]

In Appendix I, we give an additional case study to assess the performance of the hierarchical BERT when documents are long.

## H Issue of Using Training, Validation, and Test Sets

For each problem in LexGLUE, training, validation, and test sets are available. In our experiments, of course the test set is independent from the training process. However, some issues occur in the use of the training and validation sets.

For linear methods, in contrast to deep learning methods, they do not need a validation set for the termination of the optimization process or for selecting the iteration that yields the best model. Further, they may internally conduct cross-validation to select hyper-parameters (e.g., thresholds in the thresholding method). Therefore, we combine training and validation subsets as the new training set used by the linear methods. This is the standard setting in traditional supervised learning.

For BERT training, the validation set is used for selecting the best epoch and/or the best hyper-parameters. We follow the common practice to deploy the model achieving the best validation performance for prediction. However, in linear methods, the model used for prediction, regardless of whether internal cross-validation is needed, is always obtained by training on all available data (i.e., the combination of training and validation sets). Therefore, for BERT we may also want to incorporate the validation set for the final model training. We refer to such a setting as the re-training process. Unfortunately, an obstacle is that the optimization process cannot rely on a validation set for terminating the process or selecting the best model in all iterations. Following Goodfellow et al. (2016), we consider the following setting to train the combined set.

1. Record the number of training steps that leads to the best validation Micro-F1 as $e^*$.
2. Re-train the final model using the combination of training and validation sets for $e^*$ epochs.

BERT results without/with re-training are shown in Table 11. In general, the re-training process improves the performance, especially for the data sets SCOTUS and EUR-LEX. However, results are slightly worse in both the default and tuned settings for the data set UNFAIR-ToS. Thus the outcome of re-training may be data-dependent.

A comparison between linear methods and BERT with re-training shows that conclusions made earlier remain the same. Because re-training

[8] https://github.com/coastalcph/lex-glue

[9] We run five seeds in the part of additional training. Thus, we obtain five values of the total time and report the average.

| Property | Value |
|---|---|
| # training instances | 10,182 |
| # validation instances | 1,132 |
| # test instances | 7,532 |
| # classes | 20 |
| $\overline{W}$ | 283.66 |
| $W_{\max}$ | 11,821 |
| $\overline{T}$ | 552.82 |
| $T_{\max}$ | 138,679 |
| # documents exceeding 512 tokens | 4,927 (26.14%) |

Table 12: Data statistics for 20 Newsgroups. We conduct a 90/10 split to obtain the validation data. $\overline{W}$/$\overline{T}$ means the average # words/tokens per instance of the whole set, and $W_{\max}$/$T_{\max}$ means the maximum # words/tokens of the whole set.

| Method | μ-F$_1$ | m-F$_1$ |
|---|---|---|
| Linear | | |
| one-vs-rest | 85.3 | 84.6 |
| thresholding | 85.3 | 84.6 |
| cost-sensitive | 85.2 | 84.5 |
| BERT | | |
| default | 84.0 | 83.3 |
| tuned | **85.6** | **84.9** |
| hierarchical | 84.9 | 84.2 |

Table 13: Experimental results of 20 Newsgroups by linear methods and BERT. For the default setting, we follow the default parameters in Table 7. For the tuned and hierarchical setting, we use the same parameter search range as the one in Table 8. Further, to process the set for the hierarchical setting, each document is split into 40 segments based on the presence of consecutive newline characters, where each segment contains at most 128 tokens.

is not conducted in Chalkidis et al. (2022), in the main paper we report the results without re-training.

## I A Case Study of BERT on 20 Newsgroups

Wahba et al. (2023) applied BERT for training the data set 20 Newsgroups (Lang, 1995) but did not check the document length. To assess the importance of the document length, we downloaded the 20 Newsgroups set from scikit-learn[10] with default parameters. Further, we checked the document length from the word and token levels where the tokens are obtained by the "bert-base-uncased" tokenizer. The data statistics are presented in Table 12. We found that the 20 Newsgroups data set includes a considerable number of documents that exceed 512 tokens. This may be an issue because BERT can only process up to 512 tokens without further design; see Appendix E for more details. To investigate this problem, we conducted experiments using both linear classifiers and BERT. Results are in Table 13. The observations are summarized as follows.

- The results of linear classifiers do not improve by using thresholding and cost-sensitive techniques to handle class imbalance. The reason is that the data set has a small number of labels and a more balanced class distribution. In addition, linear methods are still competitive with BERT.
- The tuned setting of BERT has the best Micro-F1 among all the methods. Thus, for running BERT on this set, parameter selection seems to be important. Interestingly, when we considered the document length using the hierarchical methods in Appendix E.3, the performance was not better than the tuned setting.

In conclusion, linear methods are still a simple and efficient solution to this problem. For BERT, we showed that using the hierarchical setting to handle long document length may not always lead to the best performance. The result of applying hierarchical BERT may be data-dependent. Thus a general setting for handling long documents still need to be investigated.

[10] See https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_20newsgroups.html for more details. We have checked that the set used in scikit-learn is the same as the "20 Newsgroups sorted by date" set from the original source at http://qwone.com/~jason/20Newsgroups/.

## ACL 2023 Responsible NLP Checklist

### A For every submission:

☑ A1. Did you describe the limitations of your work?
*Section Limitations.*

☒ A2. Did you discuss any potential risks of your work?
*Left blank.*

☑ A3. Do the abstract and introduction summarize the paper's main claims?
*Sections Abstract and 1.*

☒ A4. Have you used AI writing assistants when working on this paper?
*Left blank.*

### B ☑ Did you use or create scientific artifacts?

*Section 3.*

☑ B1. Did you cite the creators of artifacts you used?
*Section 3.*

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
*Left blank.*

☒ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
*Left blank.*

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
*Left blank.*

☒ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
*Left blank.*

☒ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
*Left blank.*

### C ☑ Did you run computational experiments?

*Section 4.*

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
*Section 4.*

*The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.*

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
*Appendix F.*

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
*Appendix G.*

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
*Appendix C.*

**D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?**

*Left blank.*

□ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
*No response.*

□ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
*No response.*

□ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
*No response.*

□ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
*No response.*

□ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
*No response.*