CS230

Recognizing Literary Merit with Deep Learning

David K. Wendt
Stanford University
dwendt@stanford.edu

Abstract

This project trains a deep learning model to recognize literary merit in English literature. Specifically, a pre-trained distilBERT model is fine-tuned on a corpus of 30 English novels, achieving a final test set accuracy of 92%. It is then retrained on an extended corpus of 108 English novels, achieving a final test set accuracy of 75%. Results demonstrate that end-to-end literary analysis with deep learning can accurately determine whether a text excerpt is of high literary merit.

1 Introduction

With the recent advancements in artificial intelligence, it is commonplace to compare such technology with human intellect, especially with the "uniquely" human endeavor of art. The distinction slowly blurs as machine learning technologies become capable of producing original visual or musical works. An interesting question arises: does artificial intelligence have the capacity not only to produce art but also to recognize artistic merit? This project, therefore, seeks to train a neural network to recognize literary merit in a sample of English novels.

2 Related work

Recently developed deep learning models for natural language processing, in particular transformer models (1) such as BERT (2), have made great progress in the field. In particular, fine-tuning such pretrained models for text classification has proven highly effective, even when simply training a logisitc regression classifier on the model's classification output (3). Applying such advancements for end-to-end analysis of literary texts, though, seems to be lacking. Specifically focusing on English literature, text classification studies have been performed in which machine learning techniques are applied to certain extracted features from texts (4), but not directly to the texts themselves. Studies have even been performed on the task of literary merit detection (5), but again these techniques train on features, such as sentence length or lexical diversity, extracted from texts rather than directly on the words of a text. This work seeks to contribute to this legacy of computational literary analysis and address the task of literary merit detection in an end-to-end manner by applying a pretrained transformer model, similar to those mentioned above, directly to texts.

3 Dataset and Features

3 .1 Data Source

For a repository of free, public domain (in the U.S.) works of English fiction, this project uses Project Gutenberg (PG) (6). In particular, to control for literary subgenre, the novels are restricted to the Library of Congress Classification class P: Language and Literature, subclass PR: English literature

CS230: Deep Learning, Spring 2021, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

Good books	Bad books
Ulysses by James Joyce	Black Sheep by Edmund Yates
Wuthering Heights by Emily Bronte	Blood Royal by Grant Allen
Pride and Prejudice by Jane Austen	Plashers Mead by Compton MacKenzie
Great Expectations by Charles Dickens	The Crimson Azaleas by H. De Vere Stacpoole
⋮	⋮

Table 3.2.1: Data Sample

(7)(8) (henceforth "LoCC-PR"). This classification includes other forms of literature including drama and poetry, but for consistency this project is restricted to using novels.

3 .2 Labelling

To label the data as 1 or 0 according to the presence or lack of literary merit (henceforth "good books" and "bad books"), this project uses a list of the "greatest books of all time" (9), a list of over 2500 books, itself compiled from over 100 other similar "best of" books lists. Novels appearing on both this list and the PG LoCC-PR list are included in the set of good books, while novels included in PG LoCC-PR but whose authors have no books on this "greatest books" list are included in the set of bad books. A sample of some of the books used from each category, along with their authors, are listed in Table 3.2.1 above.

3 .3 Preprocessing

Since this project uses distilBERT (described in §4) for classification, and this model can classify sequences of up to 512 tokens, the text files for each book are preprocessed as follows: after tokenizing the entire text file, each book is split into sequences of 512 tokens, each of which is later independently run through distilBERT. Thus, the number of independent data points is the total number of 512-token sequences, which, as described below, is much higher than the number of books.

3 .4 First Phase

This project occurred in two phases. In the first phase, 15 good books and 15 bad books were used, with the good books extending to rank $\approx$100 on the "greatest books" list. Furthermore, the dataset was restricted to one novel per author to avoid the confounding effect of a particular author's style. In this phase, there were a total of 5,972 sequences worth of good books and 4,996 sequences worth of bad books, for a total of 10,968 sequences.

The train/test split for this phase was performed by randomly distributing all of the books into the train and test sets, and subsequently placing all sequences from a given book into the book's set, rather than performing the train/test split on the sequences themselves. This was chosen to avoid the confounding effect of having sequences from a given book appear both in the training and test sets, which would allow the model to learn the style of specific books rather than learning overall literary merit. The random train/test split for this phase was 11/4 books from each of the good and bad categories. This resulted in 4,918 sequences worth of good books and 4,027 sequences worth of bad books in the training set, and 1,054 sequences worth of good books and 969 sequences worth of bad books in the test set, for an overall train/test split with 18.44% of sequences in the test set.

3 .5 Second Phase

In the second phase, the good books were expanded to include those up to rank 300 on the "greatest books" list, and were allowed to include multiple books from a single author (with measures taken to account for this described below). A total of 43 good books were selected, which contained a total of 19,256 sequences. The set of bad books was also expanded, this time chosen to approximately match the number of sequences worth of good books rather than simply the total number of good books. In this phase there were 65 bad books containing a total of 19,128 sequences. The total number of sequences in the second phase was therefore 38,384.

To account for the potential confounding effect of having multiple books from a single author, the books were divided into train and test sets by author rather than by book, so that all of the sequences from all of an author's books are placed into the same set. This prevents the model from increasing its test set accuracy by learning styles of particular authors from the training set. Upon performing this split, there were 14,477 sequences of good books and 14,299 sequences of bad books in the training set, and 4,779 sequences of good books and 4,829 sequences of bad books in the test set, for an overall train/test split with 25.03% of sequences in the test set.

Finally, to quantify the two confounding effects mentioned above (books by the same author in both the train and test set, or sequences from the same book in both the train and test set), additional (independent) training runs were performed in the second phase without the aforementioned measures in place to counteract these effects. Specifically, one round of training was performed with books split into train/test sets without regard to author, and additionally one round of training was performed with sequences split into train/test sets without regard to the book they came from. Henceforth, these two methods will be referred to as "splitting by book" and "splitting by sequence," respectively, in contrast to the original method in this phase of "splitting my author."

4 Methods

This project uses the transformer model distilBERT (10), as a less computationally intensive but still highly effective alternative to BERT. In contrast to (3), which trains a logistic regression classifier on the classification outputs from pretrained distilBERT, this method involves fine-tuning distilBERT by training it directly on the text data. No layers were frozen; in other words, all of distilBERT was trained.

The pretrained distilBERT used was downloaded using the Hugging Face Transformers library (11), with "DistilBertForSequenceClassification.from_pretrained('distilbert-base-uncased')" being the specific model used from this library. Correspondingly, the tokenizer used on the novels is "DistilBertTokenizer.from_pretrained('distilbert-base-uncased')", again from the Hugging Face Transformers library. The backend used for training was PyTorch (12), and Scikit-learn (13) was used for the evaluation metrics involving the ROC as well as for the train-test split when it was performed directly on sequences rather than books. (The latter was used only once, for the training run in the second phase where the data was split by sequence.) Throughout training, the AdamW optimizer was used. Baseline hyperparameters are described in Table 4.1, and variations on these parameters described in §5.

Hyperparameter	**Baseline value**
Number of Epochs	5
Minibatch Size	16
Learning Rate (α)	1e$-$5
Betas	(0.9, 0.999)
Weight Decay	0.01

Table 4.1: Baseline Hyperparameters

5 Experiments/Results/Discussion

5 .1 First Phase

During the first phase of the project, the learning rate in Table 4.1 was found to be too high. A brief hyperparameter search was performed, carrying out training runs with learning rates a factor of 10 higher and lower, with the results being that $\alpha = 1\text{e}{-6}$ was a sufficiently low learning rate for this project (see Figures A.1.1 - A.1.3). The final model training plot from this first phase, with learning rate $\alpha = 1\text{e}{-6}$ (which was adopted for the remainder of the project) is Figure 5.1.1. The corresponding evaluation metrics are displayed in Table 5.1.1.

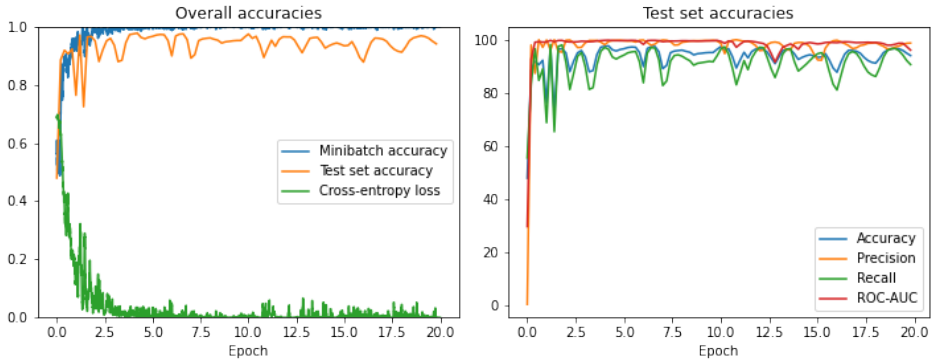

Figure 5.1.1: First Phase Training Plot

Data Set	Accuracy	Precision	Recall	ROC-AUC
Training set	99.92%	100.0%	99.86 %	0.9998
Test set	92.88%	98.86%	88.76 %	0.9432

Table 5.1.1: First Phase Evaluation Metrics

With a test set accuracy of over 92%, this can certainly be considered a successful model, and constitues proof-of-concept that deep learning NLP models can, in an end-to-end fashion, have high success in determining literary merit.

5 .2 Second Phase

After the success of the first phase, this project sought to determine whether the model would be quite so accurate on an expanded data set. The original training run for this phase used the “splitting by author" method described in §3.5. The results are plotted in Fig. 5.2.1, with evaluation metrics displayed in Table 5.2.1

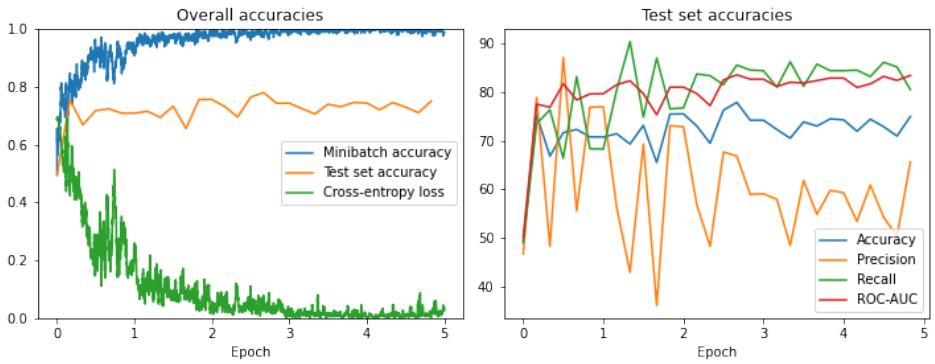

Figure 5.2.1: Second Phase Training Plot (Split by Author)

The plot from this phase shows clear overfitting of the training set. The initial weight decay value, as noted in Table 4.1, was 0.01; to reduce overfitting, increased regularization was attempted, with weight decay values of 0.1 and 1.0, but this failed to noticeably reduce the overfitting (see §A.2).

Finally, as discussed in §3.5, data splitting by book and by sequence was performed in addition to the aforementioned splitting by author, and the model was retrained on the resulting training sets. The plots are included in §A.3, with the evaluation metrics included in Table 5.2.1

As expected, splitting by sequence achieves a much higher test set accuracy, even higher than the first phase test accuracy, likely due to the confounding effect of sequences from the same book being

Split By	Data Set	Accuracy	Precision	Recall	ROC-AUC
Author	Training set	99.36%	99.98%	98.76 %	0.9999
	Test set	74.99%	65.54%	80.56 %	0.8346
Book	Training set	99.38%	99.99%	98.81 %	0.9994
	Test set	75.29%	79.29%	73.25 %	0.7673
Sequence	Training set	99.52%	99.96%	99.10 %	0.9999
	Test set	98.31%	99.22%	97.50 %	0.9980

Table 5.2.1: Second Phase Evaluation Metrics

included in both training and test sets, allowing the model to learn the style of specific books. (A secondary result of this effect is that overfitting has not occurred, as can be seen clearly in Fig. A.3.2.) Surprisingly, though, the results of splitting by book are very similar to those of splitting by author, demonstrating that the confounding effect of having multiple books by a given author split between train and test sets are not too significant; this could also, however, partly have been a chance result of the randomized splitting performed.

6 Conclusion/Future Work

In summary, the first phase model training was highly successful, achieving a test set accuracy of over 92% and an ROC-AUC score of over 0.94. With an expanded dataset, the model training was less successful, with test set accuracy of only 75% and ROC-AUC score of 0.83, and it was additionally prone to overfitting. The comparison of final ROC curves from the two phases is displayed in Fig. 6.1 (and details of the first phase ROC curve are in §A.1). These test set accuracies, even for the second phase, are significantly higher than a random-guessing score of 50%, demonstrating proof-of-concept that deep learning NLP models can, in an end-to-end fashion, have high success in determining literary merit.

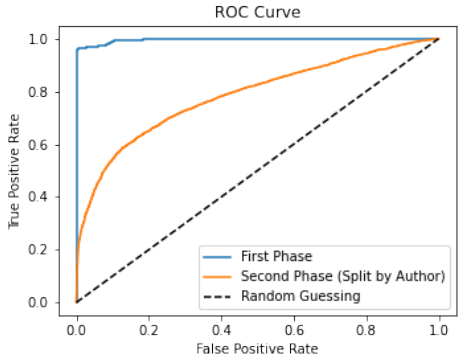

Figure 6.1: Final First Phase Plot

With these results in consideration, future work would primarily focus on regularization to reduce the overfitting in the second phase results, potentially trying to implement dropout regularization. Additionally, one could repeat this project with a larger NLP model, specifically choosing something like BERT over distilBERT. Finally, one could try a similar experiment but with an expanded corpus of novels, including American literature or English translations of originally non-English works, i.e. expanding outside the LoCCS-PR category.

7 Contributions

I would like to acknowledge the guidance of my Project Mentor TA, Dor Arad, throughout the project, and discussions with my friend, Viswesh Krishna, about general deep learning techniques and how they apply to this project.

References

[1] J. Alammar, "The illustrated transformer [blog post]." [Online]. Available: https://jalammar.github.io/illustrated-transformer/

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[3] J. Alammar, "A visual guide to using bert for the first time [blog post]." [Online]. Available: https://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

[4] A. M. Jacobs and A. Kinder, "Quasi error-free text classification and authorship recognition in a large corpus of english literature based on a novel feature set," 2020.

[5] T. Crosbie, "A computer assisted analysis of literary text: from feature analysis to judgements of literary merit," 2016.

[6] "Project gutenberg." [Online]. Available: https://www.gutenberg.org/

[7] "Project gutenberg: Browse by library of congress class: Language and literatures: English literature." [Online]. Available: https://www.gutenberg.org/browse/loccs/pr

[8] "Wikipedia: Library of congress classification:class p – language and literature: Pr - english literature." [Online]. Available: https://en.wikipedia.org/wiki/Library_of_Congress_Classification:Class_P_--_Language_and_Literature#PR_-_English_literature

[9] "The greatest books of all time." [Online]. Available: https://thegreatestbooks.org/

[10] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," 2020.

[11] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-demos.6

[12] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

A Appendix

A .1 First Phase and Learning Rate Search

The plot of training progress from the first phase of the project while using the original (i.e. baseline) hyperparameters from Table 4.1, specifically with a learning rate of $\alpha = 1\text{e}{-5}$, is shown below.

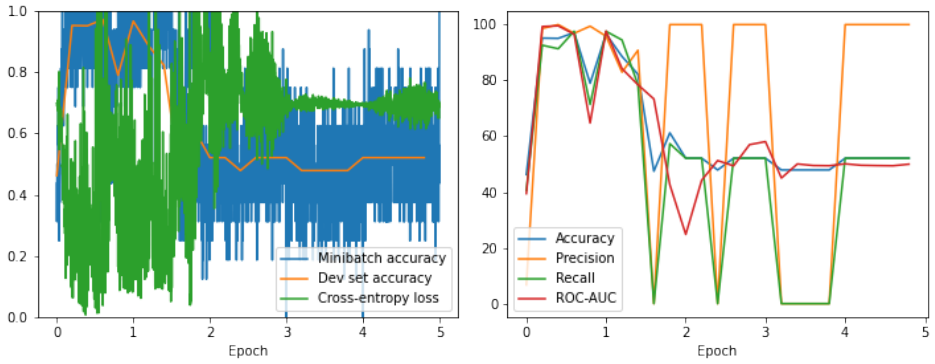

Figure A.1.1: Learning rate $\alpha = 1\text{e}{-5}$

This is clearly unstable, with all three metrics (minibatch accuracy, test set accuracy, and loss) decaying after just about one training epoch. To correct this, a brief hyperparameter search for the learning rate was conducted, trying out values $\alpha = 1\text{e}{-4}$ and $\alpha = 1\text{e}{-6}$. These are shown below.

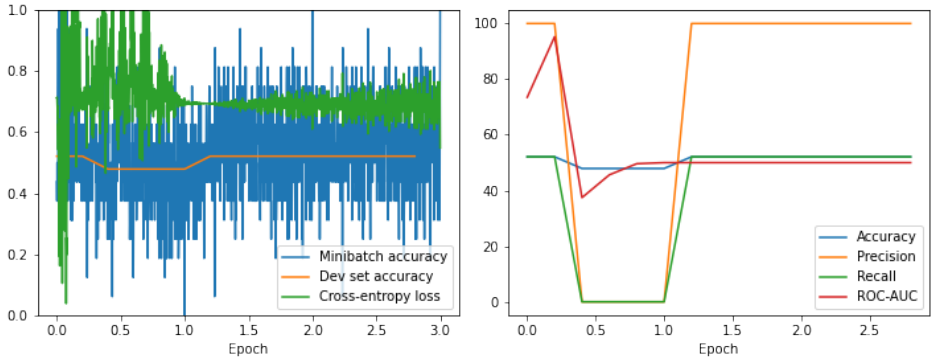

Figure A.1.2: Learning rate $\alpha = 1\text{e}{-4}$

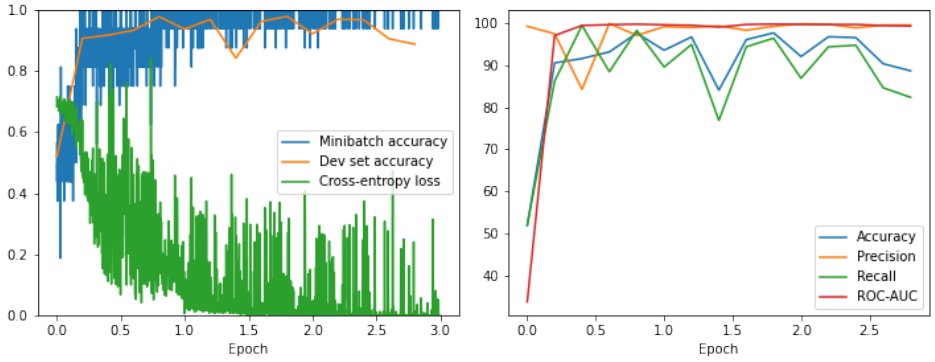

Figure A.1.3: Learning rate $\alpha = 1\text{e}{-6}$

Thus, a learning rate $\alpha = 1\text{e}{-6}$ was found to be by far the most stable, and was adopted for the remainder of the project.

One final correction was made in the first phase; specifically, in the model training function, shuffling of the training data each epoch before splitting into minibatches was implemented. The result of this

change, which was run for an extended 20 epochs, is displayed above in Fig. 5.1.1 and Table 5.1.1, and reproduced below in Fig. A.1.4 and Table A.1.1

Note that the first phase ROC curve in Fig. 6.1 corresponds to the result of the above Fig. A.1.3, with test accuracy 95.06% and ROC-AUC score 0.9957, rather than the training from Fig. A.1.4 below with the evaluation metrics in Table A.1.1. This choice was made to reflect the fact that the red ROC-AUC curve in Fig. A.1.4 happens to end in a trough, and the final ROC-AUC score is thus lower than what would be truly reflective of the model's performance.

Additional note: for all training plots from this point forth in the Appendix, and for all training plots in the main sections before the Appendix, the minibatch accuracy and cross-entropy loss are plotted as moving averages over the past 20 minibatches. (In the first phase, there are 560 minibatches per epoch. In the second phase, there are 1800 minibatches per epoch.)

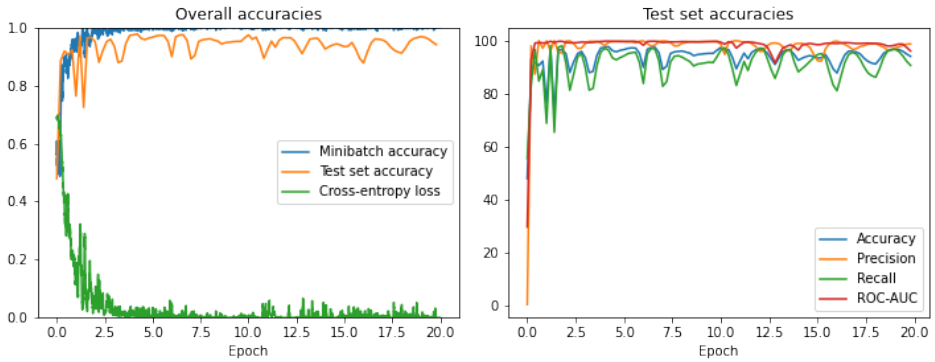

Figure A.1.4: Final First Phase Plot

Data Set	Accuracy	Precision	Recall	ROC-AUC
Training set	99.92%	100.0%	99.86 %	0.9998
Test set	92.88%	98.86%	88.76 %	0.9432

Table A.1.1: Final First Phase Evaluation Metrics

A .2 Second Phase: Regularization

As discussed in §5.2, the second phase training run with data split by author displayed clear overfitting. To reduce this, increased regularization was attempted by increasing the weight decay from 0.01 to 0.1, and then again from 0.1 to 1.0. The results of these increased regularization training runs are shown below.

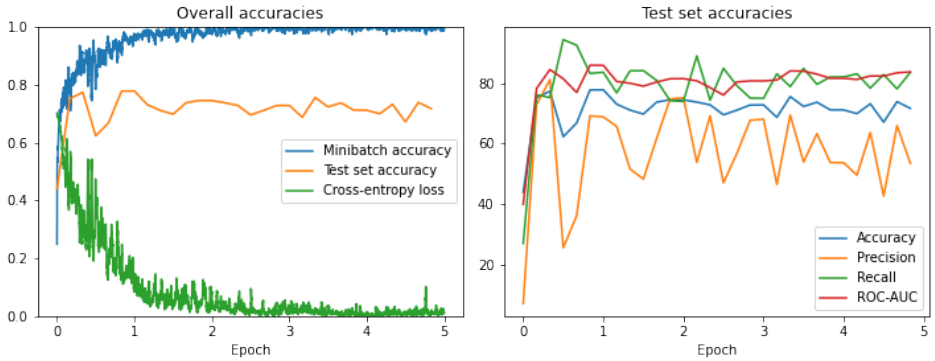

Figure A.2.1: Training Plot for Increased Regularization: Weight decay = 0.1

Data Set	Accuracy	Precision	Recall	ROC-AUC
Training set	99.48%	99.55%	99.42 %	0.9999
Test set	71.74%	53.94%	83.38 %	0.8363

Table A.2.1: Evaluation Metrics for Increased Regularization: Weight decay = 0.1

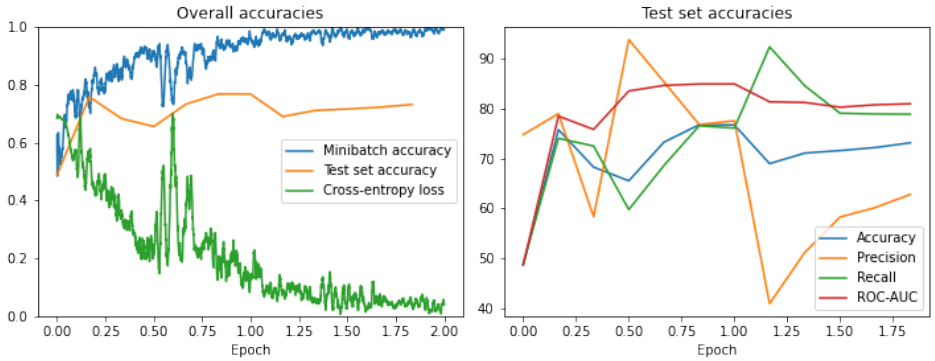

Figure A.2.2: Training Plot for Increased Regularization: Weight decay = 1.0

Data Set	Accuracy	Precision	Recall	ROC-AUC
Training set	99.11%	99.34%	98.91 %	0.9997
Test set	73.15%	62.96%	78.79 %	0.8091

Table A.2.2: Evaluation Metrics for Increased Regularization: Weight decay = 1.0

Clearly, this regularization fails to address the overfitting. Further regularization, potentially dropout, is suggested for future work, as mentioned in §6.

A .3 Second Phase: Data Set Splitting

As discussed in §3.5 and §5.2, different data splitting methods were performed in order to quantify certain potential confounding effects. The evaluation metrics for the three resulting different training runs are included in Table 5.2.1 (reproduced below as Table A.3.1), and the three training plots are shown below (with Fig. 5.2.1 reproduced below as Fig. A.3.1).

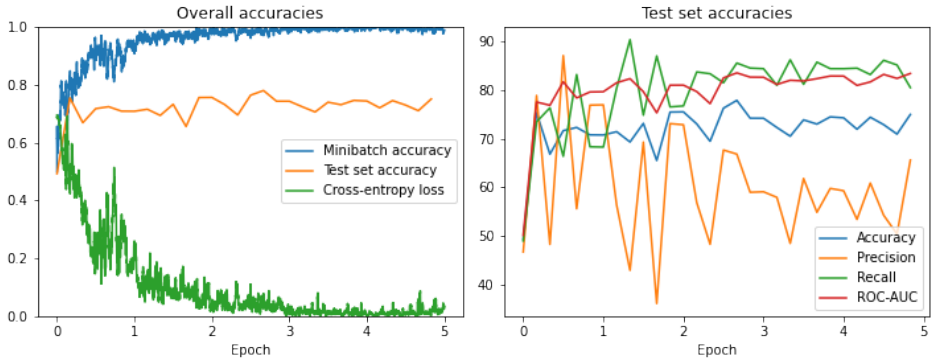

Figure A.3.1: Training Plot for Splitting by Author

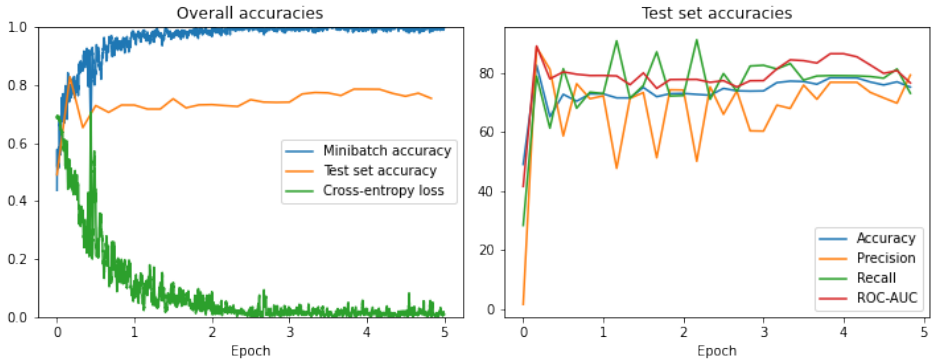

Figure A.3.2: Training Plot for Splitting by Book

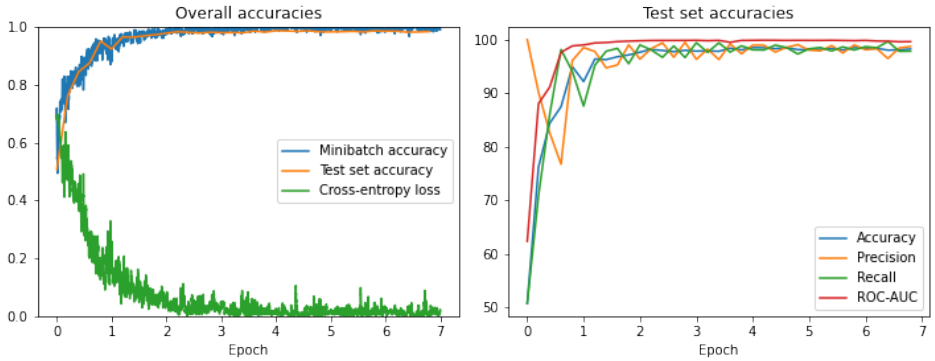

Figure A.3.3: Training Plot for Splitting by Sequence

Split By	Data Set	Accuracy	Precision	Recall	ROC-AUC
Author	Training set	99.36%	99.98%	98.76 %	0.9999
	Test set	74.99%	65.54%	80.56 %	0.8346
Book	Training set	99.38%	99.99%	98.81 %	0.9994
	Test set	75.29%	79.29%	73.25 %	0.7673
Sequence	Training set	99.52%	99.96%	99.10 %	0.9999
	Test set	98.31%	99.22%	97.50 %	0.9980

Table A.3.1: Second Phase Evaluation Metrics