

Siamese BERT for Authorship Verification

Notebook for PAN at CLEF 2021

Jacob Tyo^{1,2}, Bhuwan Dhingra³ and Zachary Lipton¹

¹*Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213*

²*US Army Research Laboratory, 2800 Powder Mill Rd, Adelphi, MD 20783*

³*Duke University, Durham, NC 27708*

Abstract

The PAN 2021 authorship verification (AV) challenge focuses on determining if two texts are written by the same author or not, specifically when faced with new, unseen, authors. In our approach, we construct a Siamese network initialized with pretrained BERT encoders, employing a learning objective that incentivizes the model to map texts written by the same author to nearby embeddings while mapping texts written by different authors to comparatively distant embeddings. Additionally, inspired by related work in computer vision, we attempt to incorporate triplet losses but are unable to realize any benefit. Our method results in a slight performance gain of 0.9% overall score over the baseline and an increase of 8% in F1 score.

1. Introduction

Authorship verification (AV) is the task of determining if two texts were written by the same person or not. While traditionally, this feat has required the expertise of forensic linguists, recent advances in both natural language processing (NLP) and related matching tasks in computer vision, offer several paths for improving automated methods. The traditional machine learning approach to this problem consists of two steps: feature extraction and model fitting. Feature extraction can include the count of specific words/sub-words/punctuation, misspellings, part-of-speech tags, etc. More recent methods have paired these hand-engineered features with modern feature extraction methods such as n-grams [1], pretrained word embeddings [2], and pretrained sentence structure embeddings [3]. The models leveraged for this task have ranged from latent Dirichlet allocation [4] and support-vector machines [5] to convolutional [1, 6] and recurrent neural networks [7, 8]. However, prior work in AV has not yet made extensive use of transformer architectures or pretrained language models.¹

In this work, we apply the pretrained BERT model in a Siamese configuration for the task of AV [12]. We make use of WordPiece [13] for tokenization and do not use engineered features. We set out to determine how well modern methods perform on AV, and the feasibility of removing hand-engineered features in favor of deeper models and prior knowledge in the form of pretraining.

CLEF 2021 – Conference and Labs of the Evaluation Forum, September 21–24, 2021, Bucharest, Romania

✉ jtyo@cs.cmu.edu (J. Tyo); bdhingra@cs.duke.edu (B. Dhingra); zlipton@cmu.edu (Z. Lipton)

🌐 <https://jacobtyo.com/> (J. Tyo)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

¹We note that several works in Authorship Attribution (classifying texts into a fixed list of potential authors) do leverage pretrained language models, including [9, 10, 11].

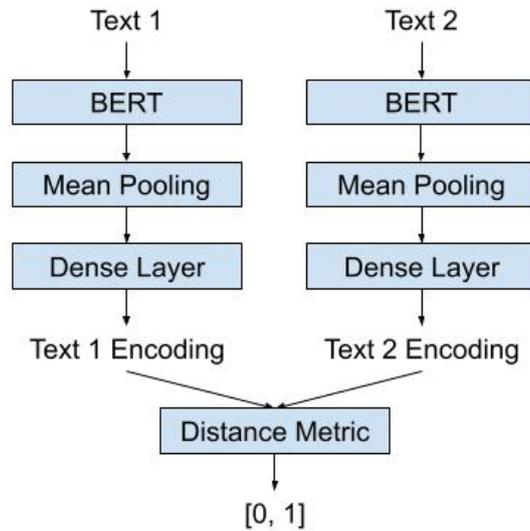


Figure 1: The Siamese BERT for Authorship Verification (SAV) model structure n the givand data flow.

Furthermore, triplet loss has provided benefits in image processing [14], but has not yet been leveraged for AV. We experiment with triplet loss (leveraging multiple sampling strategies), contrastive loss, and a modified version of contrastive loss that has proved beneficial in a previous AV study [15]. The dataset for this task was obtained from fanfiction.net, where each datapoint consists of pairs of text from two different fanfics (an amateur fictional writing based on an existing work of fiction) [12]. More on this dataset in Section 2.1.1.

2. Siamese BERT for Authorship Verification

We introduce Siamese BERT for Authorship Verification (SAV)². Our method uses a pretrained BERT model in a Siamese setup as shown in Figure 1 and originally introduced by [16]. In the AV task, we are given two input texts x_1 and x_2 and the expected output is a score in the interval $[0, 1]$ indicating the likelihood with which they belong to the same author. The maximum input size for the BERT model is 512 tokens, therefore we truncate each text to the first 512 tokens. Separately, for both input texts, they are passed through the BERT model resulting in an output of size $n \times 768$ (where n is the number of tokens in the input and 768 is the dimension of the BERT output for each token). All n representations are then averaged into a 768 dimensional vector (the mean pooling layer), and then passed through a fully connected layer to generate the final text embedding (256 dimensional). This gives the final output representation u and v of input texts x_1 and x_2 , respectively. These representations are then compared using a distance metric, which is then used for loss calculation and model optimization.

During inference, the same procedure is followed. The only difference is that after the distance

²All code for this model can be found here: https://github.com/JacobTyo/PAN21_SAV

between embeddings u and v are calculated, it is compared to a threshold. If the corresponding distance is smaller than the threshold, the texts are predicted to have been written by the same author and vice versa. In Section 4 we discuss more detail on finding the thresholds, as well as an alternative approach to truncating each input to 512 tokens.

2.1. Training

2.1.1. Data Preprocessing

The PAN 2021 AV challenge provided two datasets, both obtained from fanfiction.net. Each datapoint consists of a pair of texts from two different fanfics, as well as a tag representing which fandom (the particular fictional series) each text is from. We leverage only the large dataset in this work, which contains 275,565 text pairs. Roughly 54% of these pairs were written by the same author (i.e. a same-author pair). Approximately 8% of the pairs were texts from the same fandom, but none of the same-author pairs contained texts from the same fandom. In total, the texts were pulled from 1,600 fandoms and over 278,000 authors.

Instead of using these predefined pairs for training, we elected to split all pairs and store all texts individually. However, we don't want to change the data distribution for the test set. Therefore, we sample 10% of the pairs randomly to form the test set. We then ensure that all authors found in this test set have no texts in the training set. If so, the text pair is moved to the test set. We form a secondary test set by splitting all of the test pairs, and then recombining them randomly (based on author, using the same procedure as is used during training). We will refer to this set as the modified test set, as it has the same data distribution as the training set we have created but not as the original data. During training, we randomly sample text pairs (at roughly 50% same-author 50% different author pair rates). Although this changes the data distribution, it allows us to leverage a much larger set of text pairs (≈ 76 billion possible pairs vs ≈ 275 thousand).

2.1.2. Loss Functions

Any Siamese model can be trained using a wide range of loss functions. In this work, we explored training our model with the contrastive, modified contrastive, and triplet loss functions.

The contrastive loss is

$$\mathcal{L}_c(u, v, y) = \frac{1}{2} \left(y d(u, v)^2 + (1 - y) \max\{(m - d(u, v))^2, 0\} \right), \quad (1)$$

where u and v are text embeddings, $y \in \{0, 1\}$ is the label (1 if u and v were written by the same author, 0 otherwise), d is the distance metric, and m is a margin (no loss is incurred for a different-author pair if their representations are further apart than m).

The modified contrastive loss, originally introduced by [15], is

$$\mathcal{L}_{mc}(u, v, y) = \frac{1}{2} \left(y \max\{(d(u, v) - m_s)^2, 0\} + (1 - y) \max\{(m_d - d(u, v))^2, 0\} \right). \quad (2)$$

The modification from the aforementioned contrastive loss is that there are now two margins. m_s refers to a margin for same-author pairs. If the distance between the embeddings of a same-author

pair is smaller than m_s , then no loss is incurred. In normal contrastive loss, loss is incurred unless the pair of texts evaluate to identical representations. This modified contrastive loss allows for some variation among the texts of a single author, which should help account for differences in a single author’s text such as topic differences, and therefore make the resulting model more robust to non-stylistic difference among authors. The second margin m_d refers to a margin for different-author pairs, and performs the same function as m in the original contrastive loss.

The triplet loss function is

$$\mathcal{L}_t(a, p, n) = \max\{d(a, p) - d(a, n) + m, 0\}, \quad (3)$$

where a represents the embedding of an *anchor* text, p represents the embedding of a different text than a but from the same author (*positive* pair), and n represents the embedding of a text from an author different than that of a (*negative* pair). m is the margin to separate the positive and negative pairs by (i.e. the negative sample should be further from the anchor than the positive sample by at least m). Note that the triplet loss does not explicitly push same-author pairs together, but instead only forces different-author pairs to be farther apart than same-author pairs. With the contrastive and modified contrastive loss functions, we sample pairs of texts randomly. With triplet loss, it is common to use different sampling techniques. Hermans et al. [14] describe an efficient way of performing hard negative mining. Given a random batch of samples, the loss is computed (according to the triplet loss function) for all possible, valid triplets. Then the hardest positive and the hardest negative (i.e. the positive that is furthest from the anchor and the negative that is closest to the anchor) are selected, and the loss with respect to these samples is used for updating.

2.1.3. Distance Metrics

We test with the cosine (d_{cos}) and Euclidean (d_{euc}) distance measures:

$$d_{\text{cos}}(u, v) = 1 - \frac{u \cdot v}{\|u\| \|v\|} \quad (4)$$

$$d_{\text{euc}}(u, v) = \|u - v\|_2 \quad (5)$$

2.1.4. Resources

The final model was trained for 3 days on 8 Tesla v100’s. This allowed for 16 samples per GPU, for a total batch size of 128. We used the standard learning rate for the hugging face transformer pretrained models (5×10^{-5}) and anneal it over 4 epochs.

3. Evaluation

We use the evaluation metrics described in [17], as well as the baseline model provided as part of the AV task³ [18]:

³As described in [17], the provided baseline is a simple method that calculates the cosine similarities between TF-IDF-normalized, bag-of-character-tetragrams representations of the texts in a pair. The resulting scores are then shifted using a simple grid search, to arrive at an optimal performance on the calibration data.

- AUC: the conventional area-under-the-curve of the precision-recall curve
- F1-score: the harmonic mean of the precision and recall [19]
- c@1: a variant of the conventional F1-score, which rewards systems that leave difficult problems unanswered (i.e. scores of exactly 0.5) [20]
- F_0.5u: a newly proposed measure that puts more emphasis on deciding same-author cases correctly [21]
- overall: the simple average of all previous metrics

For hyperparameter selection, we predefined 17 models that differ in terms of their loss function, distance metric, and margin(s). Each of these models is trained for 3 days on a single Tesla V100 GPU. Table 2 details the performance of each of these models with respect to the modified testing set. The highest performing model with respect to the overall score is one that leverages the modified contrastive loss along with the Euclidean distance metric and an upper and lower margin of 5 and 0.25 respectively. We choose this hyperparameter combination for our final model, which was then evaluated on a hidden test set via the TIRA environment [22]. Table 1 shows the performance of our model on this hidden test set.

Table 1

The performance of our final model on the hidden test set, evaluated on the TIRA environment for the PAN21 competition [18]

Model	AUC	F1	c@1	F_0.5u	Brier	Overall
Final Model	0.8275	0.7911	0.7594	0.7257	0.8123	0.7832

4. Analysis of the final model

The final model was trained for 3 days on 8 Tesla V100 GPU’s, equating to roughly 4 epochs. All analysis and results are with respect to the test set, not the modified test set as in previous sections. We first examine the score distribution, shown in Figure 2a. The same author pairs are represented by the blue histogram, and different author pairs are represented by the yellow. There is still significant overlap in the scores of the two different groups.

We optimize our thresholds on the test set via grid search, which is visualized in Figure 2b. The z-axis represents the overall performance of the final model as the thresholds are varied (x and y axes). The optimal thresholds are 0.470 and 0.553 respectively, giving an overall performance on the test set of 0.701. The other performance metrics, along with the performance of the baseline on test set is shown in Table 3

4.1. Adding more context

Our model leverages only the first 512 tokens of each text (the maximum tokens that will fit in a single pass through the BERT model). Here, using the final model, we investigate chunking each text longer than 512 tokens into sets, and then combining the final representation of each chunk before passing the text representation to the distance metric. We combine the representations of each individual chunk via averaging, resulting in a fixed-length vector that encodes information

Table 2

The performance of all models during the hyperparameter search. All models were trained for 3 days on a single Tesla V100 GPU, and were evaluated on the modified test set. The “Final Model” entry details the performance of the best performing model on the modified test set after completing the final training phase (3 days on 8 Tesla V100’s - i.e. 8x larger batch size for the same number of training iterations).

Loss Function	Distance Metric	Upper Margin	Lower Margin	AUC	F1	c@1	F_0.5u	Overall
Triplet	Euclidean	1000	-	0.552	0.206	0.522	0.341	0.405
		100	-	0.561	0.361	0.537	0.468	0.482
		10	-	0.529	0.457	0.519	0.495	0.500
Contrastive	Cosine	0.01	-	0.878	0.692	0.556	0.584	0.678
		0.1	-	0.904	0.696	0.563	0.588	0.688
		0.5	-	0.874	0.795	0.773	0.751	0.798
		0.9	-	0.826	0.646	0.712	0.749	0.733
Contrastive	Euclidean	0.1	-	0.839	0.765	0.738	0.72	0.766
		1	-	0.805	0.737	0.722	0.712	0.744
		10	-	0.782	0.663	0.699	0.715	0.715
Modified Contrastive	Euclidean	0.5	2.5	0.813	0.743	0.723	0.712	0.748
		0.25	5	0.871	0.809	0.776	0.757	0.803
		0.5	5	0.789	0.722	0.705	0.698	0.729
		0.25	1	0.881	0.789	0.754	0.688	0.778
		0.1	1	0.858	0.778	0.756	0.713	0.776
		0.75	1	0.815	0.627	0.704	0.698	0.711
Modified Contrastive	Cosine	0.1	0.5	0.811	0.719	0.729	0.735	0.749
Baseline	-	-	-	0.831	0.769	0.78	0.764	0.786
Final Model	-	-	-	0.892	0.811	0.796	0.777	0.819

from the entire input text regardless of length. The score distribution for the final model using chunking is shown in Figure 3. We compare these two score distributions by looking at the percentage of overlap, and find that both the chunking and non-chunking procedures produce roughly 55% overlap. Furthermore, this chunking behavior results in slightly worse overall performance on the test set, 0.701 vs 0.715 of the final model without chunking. Lastly, chunking is computationally expensive. During inference, on an 8-core CPU, the final model takes about 1.5 seconds to process an input pair. On the same machine, the chunking model takes about 9.8 seconds to process that same input pair. Because of this large cost increase along with the indistinguishable performance, we use only the non-chunking model.

Table 3

The performance for the baseline and final model on the test set.

Model	AUC	F1	c@1	F_0.5u	Overall
Baseline	0.779	0.659	0.759	0.628	0.706
Final Model	0.780	0.739	0.731	0.611	0.715

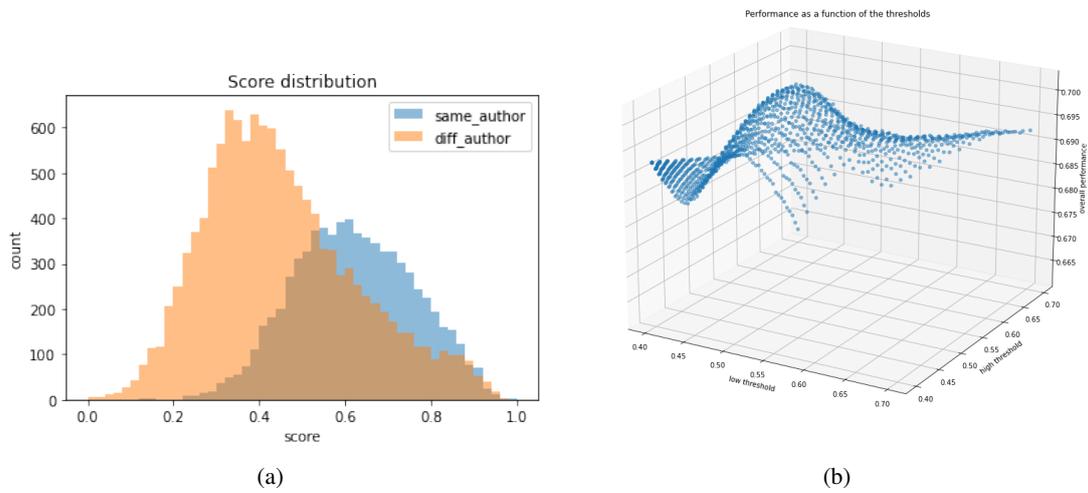


Figure 2: (a) The score distribution of the final model on the test set. (b) The overall performance (z-axis) with respect to the lower and upper thresholds (x-axis and y-axis respectively).

5. Conclusion

In this work, we construct a Siamese network initialized with pretrained BERT encoders, employing a learning objective that incentivizes the model to map texts written by the same author to nearby embeddings while mapping texts written by different authors to comparatively distant embeddings. Our method results in a slight performance gain over a baseline of 0.9% overall score, and an increase of 8% in F1 score. We explore the effectiveness of different loss functions, distance metrics, and margins and our results indicate the need for either hand engineered features or more training time and data. This work represents the first steps in understanding the ability

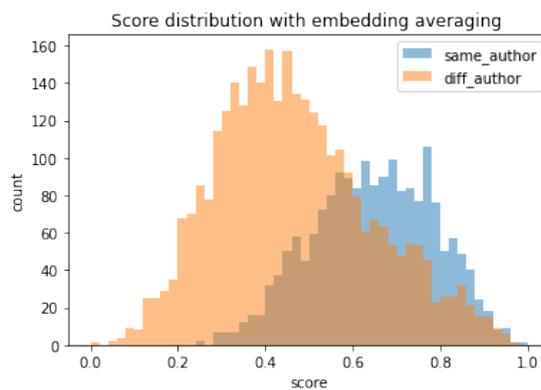


Figure 3: The score distribution of the final model on the test set when using chunking.

of modern language models and tokenizers to perform authorship verification, without any of the common hand engineered features. Some interesting future work includes broadening the training data (incorporating many AV datasets during training) and lengthening the training time, further investigating sampling strategies (we expect approaches such as hard-negative mining to provide improvements vs random sampling), and explore different methods of embedding text longer than the input size of the BERT model.

References

- [1] S. Ruder, P. Ghaffari, J. G. Breslin, Character-level and multi-channel convolutional neural networks for large-scale authorship attribution, arXiv preprint arXiv:1609.06686 (2016).
- [2] B. Boenninghoff, J. Rupp, R. M. Nickel, D. Kolossa, Deep bayes factor scoring for authorship verification, arXiv preprint arXiv:2008.10105 (2020).
- [3] F. Jafariakinabad, K. A. Hua, A self-supervised representation learning of sentence structure for authorship attribution, arXiv preprint arXiv:2010.06786 (2020).
- [4] J. Savoy, Authorship attribution based on a probabilistic topic model, *Information Processing & Management* 49 (2013) 341–354.
- [5] C. Martín del Campo-Rodríguez, H. Gómez-Adorno, G. Sidorov, I. Batyrshin, Cic-gil approach to cross-domain authorship attribution, *Working Notes of CLEF* (2018).
- [6] P. Shrestha, S. Sierra, F. A. González, M. Montes, P. Rosso, T. Solorio, Convolutional neural networks for authorship attribution of short texts, in: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, 2017*, pp. 669–674.
- [7] D. Bagnall, Author identification using multi-headed recurrent neural networks, arXiv preprint arXiv:1506.04891 (2015).
- [8] F. Jafariakinabad, S. Tarnpradab, K. A. Hua, Syntactic recurrent neural network for authorship attribution, arXiv preprint arXiv:1902.09723 (2019).
- [9] G. Barlas, E. Stamatatos, Cross-domain authorship attribution using pre-trained language models, in: *IFIP International Conference on Artificial Intelligence Applications and Innovations*, Springer, 2020, pp. 255–266.
- [10] M. Fabien, E. ú Villatoro-Tello, P. Motlicek, S. Parida, Bertaa: Bert fine-tuning for authorship attribution, in: *Proceedings of the 17th International Conference on Natural Language Processing, CONF, ACL, 2020*.
- [11] O. Fourkioti, S. Symeonidis, A. Arampatzis, Language models and fusion for authorship attribution, *Information Processing & Management* 56 (2019) 102061.
- [12] J. Bevendorff, B. Chulvi, G. L. D. L. P. Sarracén, M. Kestemont, E. Manjavacas, I. Markov, M. Mayerl, M. Potthast, F. Rangel, P. Rosso, E. Stamatatos, B. Stein, M. Wiegmann, M. Wolska, , E. Zangerle, Overview of PAN 2021: Authorship Verification, Profiling Hate Speech Spreaders on Twitter, and Style Change Detection, in: *12th International Conference of the CLEF Association (CLEF 2021)*, Springer, 2021.
- [13] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al., Google’s neural machine translation system: Bridging the gap between human and machine translation, arXiv preprint arXiv:1609.08144 (2016).

- [14] A. Hermans, L. Beyer, B. Leibe, In defense of the triplet loss for person re-identification, arXiv preprint arXiv:1703.07737 (2017).
- [15] B. Boenninghoff, R. M. Nickel, S. Zeiler, D. Kolossa, Similarity learning for authorship verification in social media, in: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2019, pp. 2457–2461.
- [16] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2019. URL: <https://arxiv.org/abs/1908.10084>.
- [17] M. Kestemont, E. Manjavacas, I. Markov, J. Bevendorff, M. Wiegmann, E. Stamatatos, M. Potthast, B. Stein, Overview of the cross-domain authorship verification task at pan 2020, in: CLEF, 2020.
- [18] M. Kestemont, E. Stamatatos, E. Manjavacas, J. Bevendorff, M. Potthast, B. Stein, Overview of the Authorship Verification Task at PAN 2021, in: CLEF 2021 Labs and Workshops, Notebook Papers, CEUR-WS.org, 2021.
- [19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, the Journal of machine Learning research 12 (2011) 2825–2830.
- [20] A. Peñas, A. Rodrigo, A simple measure to assess non-response (2011).
- [21] J. Bevendorff, B. Stein, M. Hagen, M. Potthast, Generalizing unmasking for short texts, 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), 2019, pp. 654–659.
- [22] M. Potthast, T. Gollub, M. Wiegmann, B. Stein, TIRA Integrated Research Architecture, in: N. Ferro, C. Peters (Eds.), Information Retrieval Evaluation in a Changing World, The Information Retrieval Series, Springer, Berlin Heidelberg New York, 2019. doi:10.1007/978-3-030-22948-1_5.