Style Change Detection Using BERT

Notebook for PAN at CLEF 2020

Aarish Iyer and Soroush Vosoughi

Department of Computer Science, Dartmouth College, Hanover, NH 03755
aarish.ravikumar.iyer.gr@dartmouth.edu
soroush.vosoughi@dartmouth.edu

Abstract The *Style Change Detection* task is very important in the area of authorship profiling, having one of its main applications in plagiarism detection. Specifically, the goal of the task is to detect where (if any) stylistic changes happen in a document which can be used to estimate the number of authors of a given document. In this paper, we present a method for *Style Change Detection*. We use Google AI's open source BERT pretrained bidirectional models to tokenize and generate embeddings for the sentences in each document in our dataset and use those to train a random forest classifier. We achieved an F1 score of 0.86 for detecting style changes and an F1 score of 0.64 for detecting multi-author documents on the test set, placing us at the top of the competition for both tasks. The code for this project has been made open source so that it can be used for further research: `https://github.com/aarish407/Style-Change-Detection-Using-BERT`

Keywords: Style Change Detection, BERT, Transfomer-based Models

1 Introduction

Detecting the number of authors involved in writing document by analyzing the writing style is an important task that has been a focus of research for centuries. This area of research has traditionally been called *Stylometry* and is defined by the Oxford dictionary as, "the statistical analysis of variations in literary style between one writer or genre and another". It is a centuries-old practice, dating back to the early Renaissance. Its applications include plagiarism detection and forensics (e.g., Vosoughi et al. [12] use computational stylometry techniques to link social media accounts operated by the same user). The main principles of stylometry were compiled and laid out by the philosopher Wincenty Lutosawski in 1890 in his work "Principes de stylomtrie" [6].

Unsurprisingly, style understanding has become one of the core areas of research in natural language understanding, leading to the proposal of various computational methods for understanding and detecting style in written text (e.g., see [7] for a review of the field of authorship attribution). Accordingly, this has become of one of the staple

tasks at PAN. The work presented in this paper was developed as a solution to the *Style Change Detection* task for the competition PAN @ CLEF 2020 [2,9]. The task is described as follows: Given a document, determine whether it has been written by more than one author (task 1). Furthermore, for a multi-author document, identify the positions in the document where the style change occurred (task 2). It is assumed that each paragraph is written by only one author, thus style change can only occur between two paragraphs. All the documents are in English and each document is written by one to three authors and can contain from zero to ten style changes. This is more complicated than the recent editions of the *Style Change Detection* tasks as those were either binary detection of single-/multi-authored documents [10,11] or detecting the actual number of authors in a document [13].

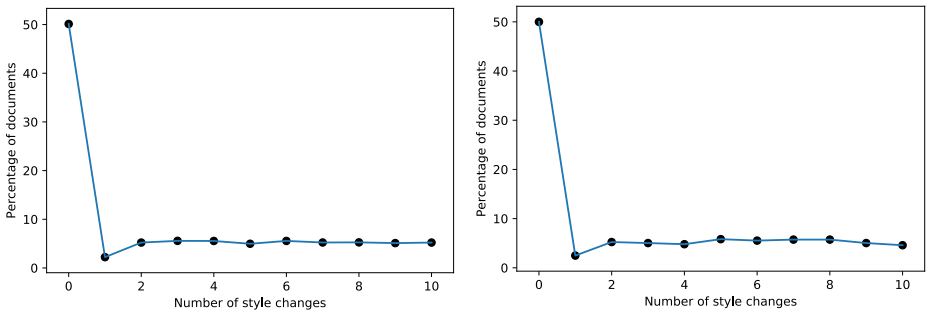
In this paper, we present a solution for this task using a Random Forest classifier in conjunction with embeddings generated by BERT, an open source large-scale pretrained language model developed by Google AI. The remaining of the paper is organized as follows. First, we introduce the dataset, next we describe our approach, including all data cleaning and pre-processing steps. Next, we describe our experiments and results. Finally, we wrap up by discussing future work and summarizing our findings.

2 Dataset

There were two types of datasets [3] that were provided for this task - a narrow dataset and wide dataset. The narrow dataset comprised of documents from similar domains, while the wide dataset did not have any restriction on its contents. For each document in each dataset there was an appropriate truth file given that had a label for task 1 (whether the document was written by more than one author) and a changes array for the task 2 (a list of 1s and 0s indicating if style change occurred between consecutive paragraphs). The F1 metric was used to calculate the score for task 1, and the micro-averaged F1 metric was used to calculate the score for task 2. Both the metrics will be referred to as accuracy hereafter. The results of both datasets were evaluated independently, and were then averaged to produce a final score for each task. The final score was calculated on the test dataset. Both the narrow and wide datasets were mined from the Stack family of websites.

Here is some information about the dataset:

1. Table 1 shows the number of documents in the narrow and wide train and validation sets. For each document, there was an appropriate truth file.
2. Table 2 shows the statistics of number of sentences and number of paragraphs in each document for the train narrow and wide datasets.
3. The truth files had the following data: number of authors, order of authors, source site and the results for task 1 and 2. For our solution, we did not make use of the first three keys.
4. Figures 1a and 1b show the distribution of number of style changes for the train narrow and wide datasets.
5. All datasets were balanced for task 1, i.e., detecting if a document is written by more than one author.



(a) Wide dataset.

(b) Narrow dataset.

Figure 1: Distribution of number of style changes in different datasets.

Table 1: Number of documents in each dataset

Number of Documents	Narrow	Wide
Train	3,442	8,138
Validation	1,722	4,078

Table 2: Statistics of number of sentences and paragraphs in each document for the two train datasets.

	Sentences		Paragraphs	
	Narrow	Wide	Narrow	Wide
Min	18	17	3	2
Max	276	375	82	74
Mean	110.19	106.98	25.28	18.03
Median	108	103	24	16

3 Approach

The general approach to both tasks was to generate embeddings of the words in each document at the sentence level and then use these embeddings for the classification. This is highlighted in Fig. 2.

3.1 Paragraph split

The first step was to split each document into paragraphs, since paragraphs are guaranteed to be atomic (i.e., only a single author has written a paragraph). This is important as the second task involves identifying style change between consecutive paragraphs.

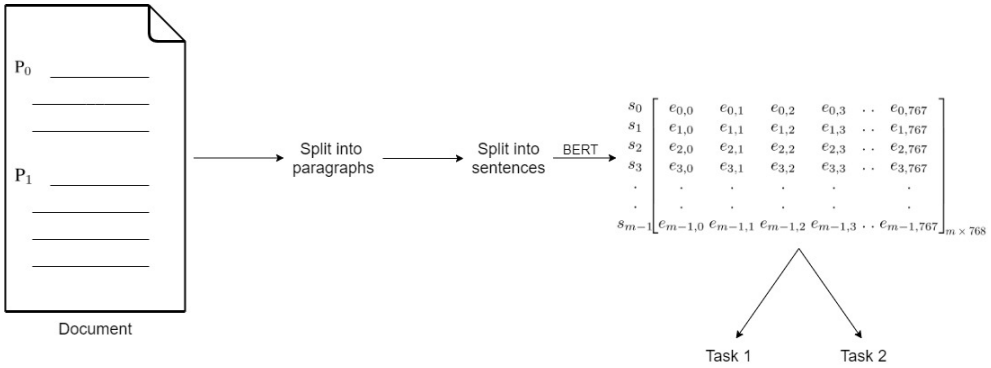


Figure 2: Our approach for generating feature vectors for the two tasks using pretrained BERT.

3.2 Sentence Split

On first thought, the idea of splitting the paragraphs into sentences seems fairly straightforward – split on characters such as '.', '?' and '!'. However, a lot of sentences would be generated that weren't sentences originally. For example, the prefixes 'Dr.' or 'Mr.' would have their own sentences. Thus, it was important to ensure that the sentences were split in a manner that is robust to the variations in usage of the aforementioned punctuation marks. A regular expression approach was used, for which each occurrence of "." which was not meant as a sentence delimiter is identified and replaced with a special token. The following structures were identified and replaced accordingly:

- Prefixes (Mr., Mrs., Dr., Ms., Prof., Capt., Cpt., Lt., Mt.)
- Website domains (.com, .net, .org, .io, .gov, .me, .edu)
- Acronyms (U.S.A., etc.)
- Suffixes (Inc., Ltd., Jr., Sr., Co.)
- Abbreviations (e.g., i.e., ...)
- Any digits separated by a period

The above approach doesn't take into consideration the different usage of '.', ',', '?' and '!' written in code, which is likely to come up in a dataset mined from the Stack family of websites. This can be added in the future to further improve this solution.

3.3 Embeddings

Before generating the embeddings, the sentence had to first be tokenized, which was done by using Google AI's BERT [1] tokenizer (the type of tokenizer depends on the BERT model used, which is described below). Note that BERT can only process sentences of length <= 512 tokens.

In order to generate embeddings for the tokenized sentences, Google AI's BERT [1] pretrained deep bidirectional models were used. BERT offers various models, and

for this task, the BERT Base Cased model was used (layers=12, hidden size=768, self-attention heads=12, total parameters=110M). The authors of BERT recommend that the BERT Base Uncased model should be used for most situations, unless it is certain that having a case-sensitive model would aid the task. We were able to report a 0.94% increase in the accuracy for the first task for the Wide dataset between the Cased and Uncased models, and thus the Cased model was used for the other tasks as well. The BERT Large model (layers=24, hidden size=1024, self-attention heads=16, total parameters=340M) was not explored for this work due to its computationally intensive nature. Furthermore, the BERT Large model in most cases only reported a 1-2% increase in accuracy over the BERT Base model on other NLP benchmarks [1].

Although BERT is used to capture context rather than style, the authors of this work found out that the information captured by these embeddings works well for the style change detection task as well.

3.4 Processing of the Dataset

Although BERT was used to generate the embeddings, they had to be combined in a specific way to fit both the tasks. The following is the method followed:

1. Each individual sentence was processed by the BERT Tokenizer, and truncated to 512 tokens if needed.
2. The tokenized sentence was then processed by BERT, which generated embeddings for each layer. This generated a tensor of dimensions $12 \times l \times 768$, where l is the length of the sentence.
3. The authors of BERT found out that the best results were obtained when the embeddings of the last 4 layers were combined, either by summing them, producing a tensor of dimensions $l \times 768$, or by concatenating them and producing a tensor of dimensions $l \times 3072$. We chose to sum the embeddings of the last 4 layers in order to prevent the dimensions of the tensor from becoming too big. Thus the final dimensions of the tensor at this step are $l \times 768$.
4. The first dimension of the current tensor is the length of the sentence, and can thus change from sentence to sentence. In order to prevent this, the embeddings of the sentence are summed over the first dimension, thus producing a final vector of length 768. Summing the embeddings over the first dimension as opposed to averaging them can lead to large difference in embedding values between sentences that are long and short. However, the length of a sentence is an important factor in detecting style change, and thus it is important to capture that information.

At this stage, we change our approach of combining embeddings for the two tasks.

Detecting style change at the document-level (Task 1) To produce a final tensor for the whole document, all the sentence vectors of the document were averaged. At the document-level, the following approaches were tested:

1. Generate the sentence vectors by summing the embeddings over the length dimension + summing all the sentence vectors of the document to produce a document-level tensor

2. Generate the sentence vectors by summing the embeddings over the length dimension + averaging all the sentence vectors of the document to produce a document-level tensor
3. Generate the sentence vectors by averaging the embeddings over the length dimension + summing all the sentence vectors of the document to produce a document-level tensor
4. Generate the sentence vectors by averaging the embeddings over the length dimension + averaging all the sentence vectors of the document to produce a document-level tensor

The second approach produced the best results for the style change detection task at the document-level. We have described why summing the embeddings over the length dimension as opposed to averaging them works better. While producing the document-level tensor, averaging all the sentence vectors seems to work better. This can be attributed to the fact that the length of the document doesn't really factor into determining whether or not style change occurred in the document, as all style changes occur between paragraphs. Thus, it makes no difference if the document is relatively short or long, as long as it has at least two paragraphs. Thus, there is no need to capture this information. It must be noted that the difference in accuracy for all four approaches was within 2% for the validation wide dataset.

Detecting style change at the paragraph-level (Task 2) Since style change had to be determined between paragraphs, the paragraph-level data points were calculated by averaging the embeddings of two consecutive paragraphs. Thus, the data point was generated by adding the embeddings of all sentences in both paragraphs and then dividing it by the sum of both paragraph lengths (in sentences). The labels were the entries in the changes array of the truth file. It is important to note that the labels of the paragraph-level data points are now imbalanced, as a document with no style change will have all paragraph-level labels as 0, while a document with style-change may still have some consecutive paragraphs that were written by the same author, and thus the labels for those data points would also be 0.

After this step, we essentially have two datasets - one with data points at the document-level and the other with data points at the paragraph-level.

3.5 Classifier

Using Python's off-the-shelf ML library Scikit-learn [8], various supervised models were tested for binary classification, such as Logistic Regression, Decision Trees, Random Forest, Support Vector Machines and Naive Baye's (Multinomial and Gaussian). The Random Forest classifier produced the best results by far for both tasks and on both data sets. Furthermore, once the Random Forest classifier was decided upon, a grid search on the hyperparameters was performed (for both tasks and both datasets) which increased the accuracy by almost 3%. However, the number of estimators for the grid-searched classifier was significantly larger than the default number of estimators, which in turn increased the time the classifier took to generate predictions on the validation set.

Finally, we had 4 classifiers:

1. Document-level classifier for the wide dataset.
2. Document-level classifier for the narrow dataset.
3. Paragraph-level classifier for the wide dataset.
4. Paragraph-level classifier for the narrow dataset.

The final set of hyperparameters for each classifier are given in Table 3

Table 3: Hyperparameters for all four classifiers.

	Document Wide	Document Narrow	Paragraph Wide	Paragraph Narrow
Criterion	entropy	gini	gini	gini
Min Samples Per Leaf	1	1	1	1
Min Samples Per Split	2	2	2	2
Estimators	400	1800	400	250

4 Results

Here we show the performance of our model on the validation and test sets. The validation set was made available during the development of the model, while the test results show the performance of our model in the competition.

Table 4 shows the performance of our model on the validation set and Table 5 shows the performance of our model on the test set. Note that for the test set, we only have cumulative information of the two datasets for the two tasks

Table 4: F1 scores calculated on the validation set for Document-level (task 1) and Paragraph-level (task 2) predictions.

	Narrow	Wide
Document-level	0.7661	0.7575
Paragraph-level	0.8805	0.8306

As can be observed, there is a discrepancy between the results reported on the test set and the validation set. This is because of the difference between the environments in which both the tests were carried out. During the development of this project, the

Table 5: Average F1 scores calculated on the test set for Document-level (task 1) and Paragraph-level (task 2) predictions.

	Average for both datasets
Document-level	0.6401
Paragraph-level	0.8566

BERT model was run using a GPU, which greatly increased the speed of computation. However, since the virtual machine offered by TIRA did not provide a GPU, all computations were significantly slower. The authors of this paper decide to clip the computations after a certain time in order to prevent the session from crashing and not being able to submit our solution.

5 Other Approaches & Future Work

During the course of this project, a number of different approaches were tried. For those approaches, a unique dataset was generated, where each data point was a combination of two sentences from consecutive paragraphs of a document. Thus, if the sentences were from the same paragraph, then the corresponding label would be 0, while two sentences from different paragraphs would have a label of 1 if there was a style change between the two paragraphs. This approach is also susceptible to producing an imbalanced dataset, and hence the dataset was balanced before moving on with the classification task. The dataset produced had nearly 3 million data points by just using the wide dataset. A couple of the approaches have been described below:

Fine-Tuning BERT In this method, the goal was to fine tune BERT using the training data so that it could produce results that were at par or better than the submitted solution. However, it was empirically observed that accuracy plateaued after a point, and was thus not explored further.

Convolutional Neural Network This method is inspired by prior work on sentence classification using convolutional neural networks [5]. In this method, each data point had dimensions $(l1 + l2) \times 768$ where $l1$ and $l2$ are the lengths of the two sentences. Note is that the data points were allowed to have variable lengths (as long as their individual lengths were <= 512). The tensors were then passed through a set of parallel convolutional filters, with Kernel sizes of $(2, 768)$, $(3, 768)$, ..., $(5, 768)$. These were meant to capture n-gram stylistic features (i.e., bigrams, trigrams, etc). The results after applying all convolution filters were globally pooled and then combined to form a vector of length n where n is the number of convolutional filters. At the end, a Fully Connected Layer is used to generate the final label.

Due to a lack of time, this approach could not be explored fully. However, the authors of this paper believe that there is merit to this approach, and intend to study it further in the future.

6 Conclusion

In this paper, we have shown how BERT can be used for the *Style Change Detection* task. Although BERT is used to capture context, this project shows that the information captured by its embeddings can be used for other NLP tasks as well. We intend to work on this project further by expanding on the other methods mentioned in Section 5. The code for the project can be found at [4].

References

1. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
2. Eva Zangerle, Maximilian Mayerl, G.S.M.P.B.S.: Overview of the Style Change Detection Task at PAN 2020. In: Cappellato, L., Eickhoff, C., Ferro, N., Névéol, A. (eds.) CLEF 2020 Labs and Workshops, Notebook Papers. CEUR-WS.org (Sep 2020)
3. Eva Zangerle, Maximilian Mayerl, M.T.G.S.M.P.B.S.: (2020), https://zenodo.org/record/3660984#.XxLhEihKhPY
4. Iyer, A., Vosoughi, S.: (2020), https://github.com/aarish407/Style-Change-Detection-Using-BERT
5. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1746–1751. Association for Computational Linguistics, Doha, Qatar (Oct 2014). https://doi.org/10.3115/v1/D14-1181, https://www.aclweb.org/anthology/D14-1181
6. Lutosławski, W.: Principes de stylométrie (1890)
7. Malyutov, M.B.: Authorship attribution of texts: A review. In: General Theory of Information Transfer and Combinatorics, pp. 362–380. Springer (2006)
8. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in python. the Journal of machine Learning research **12**, 2825–2830 (2011)
9. Potthast, M., Gollub, T., Wiegmann, M., Stein, B.: TIRA Integrated Research Architecture. In: Ferro, N., Peters, C. (eds.) Information Retrieval Evaluation in a Changing World. Springer (Sep 2019)
10. Stamatatos, E., Rangel, F., Tschuggnall, M., Stein, B., Kestemont, M., Rosso, P., Potthast, M.: Overview of PAN 2018: Author Identification, Author Profiling, and Author Obfuscation. In: Bellot, P., Trabelsi, C., Mothe, J., Murtagh, F., Nie, J., Soulier, L., SanJuan, E., Cappellato, L., Ferro, N. (eds.) Experimental IR Meets Multilinguality, Multimodality, and Interaction. 9th International Conference of the CLEF Initiative (CLEF 2018). Springer, Berlin Heidelberg New York (Sep 2017)
11. Tschuggnall, M., Stamatatos, E., Verhoeven, B., Daelemans, W., Specht, G., Stein, B., Potthast, M.: Overview of the Author Identification Task at PAN 2017: Style Breach Detection and Author Clustering. In: Cappellato, L., Ferro, N., Goeuriot, L., Mandl, T. (eds.) Working Notes Papers of the CLEF 2017 Evaluation Labs. CEUR Workshop Proceedings, vol. 1866. CEUR-WS.org (Sep 2017), http://ceur-ws.org/Vol-1866/
12. Vosoughi, S., Zhou, H., Roy, D.: Digital stylometry: Linking profiles across social networks. In: International Conference on Social Informatics. pp. 164–177. Springer (2015)
13. Zangerle, E., Tschuggnall, M., Specht, G., Potthast, M., Stein, B.: Overview of the Style Change Detection Task at PAN 2019. In: Cappellato, L., Ferro, N., Losada, D., Müller, H. (eds.) CLEF 2019 Labs and Workshops, Notebook Papers. CEUR-WS.org (Sep 2019), http://ceur-ws.org/Vol-2380/