CEUR-WS.org/Vol-2936/paper-191.pdf

Multi-label Style Change Detection by Solving a Binary Classification Problem

Notebook for PAN at CLEF 2021

Eivind Strøm[1]

[1]*Norwegian University of Science and Technology, Høgskoleringen 1, 7491 Trondheim, Norway*

Abstract

Style change detection is an important area of research with many applications, including plagiarism detection, digital forensics, and authorship attribution. This paper proposes a solution to the three sub-tasks for the PAN 2021 shared task on style change detection, featuring a challenging multi-label multi-output classification problem. We devise a pragmatic approach to the problem, solving it by binary classification before obtaining final predictions. A custom stacking ensemble is developed and trained separately on previously successful text embeddings and features for increased performance. Our solution achieves a macro-averaged F1-score of 0.7954 for single- and multi-author classification and is the best performing solution submitted for task 1. On task 2, we obtain a score of 0.7069 when detecting author change between paragraphs. For multi-label author attribution, our solution achieves a score of 0.4240 and performs significantly better than the random baseline. Being a pragmatic solution to a novel problem in the series of PAN tasks on style change detection, our approach offers several opportunities for further research.

Keywords

Style change, Multi-authorship, Binary classification, Stacking ensemble, BERT, NLP

1. Introduction

The task of quantifying writing style has been a challenging task since the 19th century, beginning with the pioneering study by Mendenhall [1] on Shakespeare plays. This area of research, *stylometry*, has several modern-day applications, including forensics, plagiarism detection and the linking of social media accounts [2, 3]. In this paper, we examine a fundamental and challenging question within the field of stylometry: Given a document, can we find evidence for the document being written by multiple authors, and are we able to attribute paragraphs to respective authors based on their writing style?

Style change detection was introduced as a shared task for the PAN 2017 event, with the goal of identifying the border positions where authorship changes [4]. The results, however, proved the task to be extremely challenging. As a result, the task was significantly relaxed for the following years. First simplified as a binary classification task, the goal was detecting whether a document is single- or multi-authored [5]. As participants obtained encouraging results and accuracies of up to 0.8993, the task was further expanded to detecting the exact

CLEF 2021 – Conference and Labs of the Evaluation Forum, September 21–24, 2021, Bucharest, Romania
stromeivind@gmail.com (E. Strøm)
© 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)

number of authors within a document [6] and identifying style changes between two consecutive paragraphs [7].

This paper proposes a solution to the PAN 2021 shared task on style change detection, which has been steered back towards its original goal: detecting the position of authorship change and assigning each paragraph to its respective author [8]. We propose a pragmatic solution by which the multi-label multi-output classification problem is first solved by binary classification before recursively converted to its original formulation. For classification, we employ several custom-built stacking ensembles that are trained on textual features and embeddings that have been proven effective in previous tasks. Our solution is the best performing submitted solution to task 1 and the second best on task 2. On task 3, our model performs significantly better than random guesses. Lastly, we release our project code for reproducibility and aid in further research: https://github.com/eivistr/pan21-style-change-detection-stacking-ensemble

The rest of the paper is structured as follows. Section 2 provides an overview of the sub-tasks, dataset and evaluation. The methodology and details of our approach are presented in section 3. Section 4 and section 5 present the experimental setting and final results. Lastly, section 6 summarizes our findings and proposes directions for future work.

2. Background

The shared PAN 2021 style change detection task is split into three sub-tasks [8]:

1. Given a text, find out whether it is written by a single or multiple authors.
2. Given a multi-authored text that contains a number of style changes between paragraphs, find the position of the changes.
3. Given a multi-authored text, assign all paragraphs of the text uniquely to some author out of the number of authors you assume for the document.

Of these sub-tasks, task 1 and 2 are the same as the previous year's edition of the style change task [7]. However, this year the dataset provided is slightly different and does not contain a separation of wide and narrow topic collections. The overall structure of the task is illustrated by Figure 1.

The dataset consists of a single collection of documents, where each document is based on a user post (or concatenation of several user posts) from the StackExchange[1] network, covering a wide range of topics. The dataset is split into a training and validation set comprising $11,200$ and $2,400$ documents, respectively. The documents in the training and validation set contain anywhere between 1 and 4 unique authors and the dataset is balanced in terms of number of authors per document.

Evaluation of the solution is performed on a test set consisting of $2,400$ documents and scored by the macro averaged F1-score. As we do not have the test set and test labels available during development, our solution is evaluated on the available validation set. Final test results are obtained through submission to the TIRA evaluation platform [9].

[1]https://stackexchange.com/

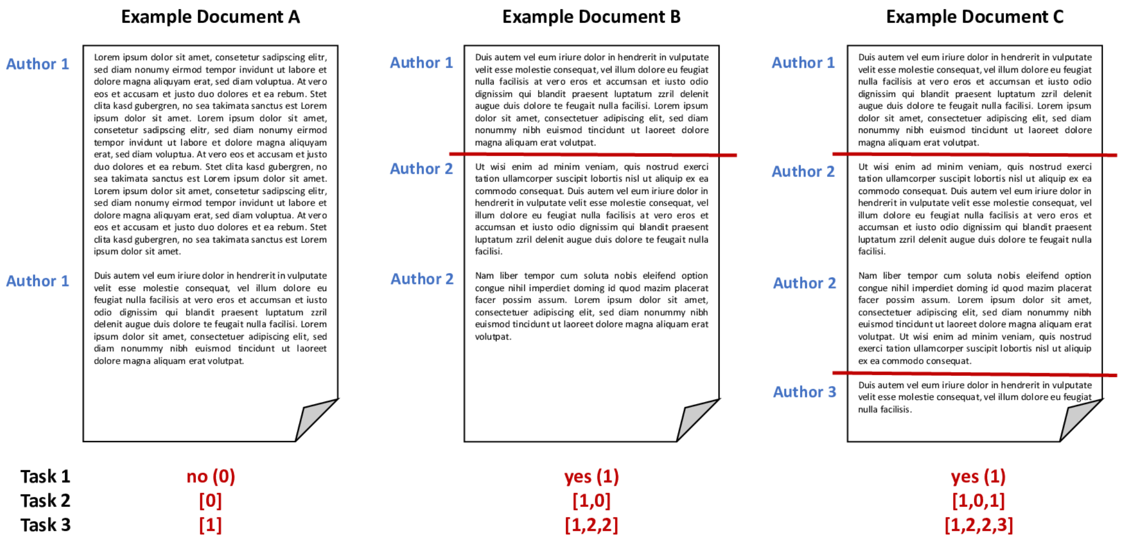

Figure 1: Illustration of PAN 2021 style change detection sub-tasks [8]

3. Methodology

This section describes the methodology of our proposed solution. First, we present the feature extraction process fundamental to our approach. Secondly, we describe the stacking ensemble classifier and our reasoning behind this choice of architecture. Lastly, we present our approach to solving the tasks as binary classification problems.

3.1. Feature extraction

For feature extraction we rely on two methods that have been successfully applied in previous tasks: generating textual embeddings using Google AI's BERT transformer [10] and extracting textual features and statistics based on the winning submission of the PAN 2018 task [11]. Features are extracted in two rounds, firstly at the document level (for use in task 1) and secondly on the paragraph level, i.e., for each individual paragraph (task 2 and 3). In the following, we summarize the important points of the feature extraction approach, for further details we refer to the work of Iyer and Vosoughi [12] and Zuo et al. [13].

Text embeddings: We follow the approach of Iyer and Vosoughi [12] for obtaining embedded text features. Firstly, tokenized sentences appropriate for generating embeddings using the BERT model are generated by splitting each paragraph into sentences before tokenizing. Prior to splitting, paragraphs are pre-processed by the removal of noisy '.', '?' and '!' characters to ensure that prefixes, suffixes, website domains, acronyms, abbreviations, and digits do not introduce incomplete sentences. After processing, sentences are tokenized and embedded using the BERT Base Cased model,[2] generating a $12 \times l \times 768$ tensor for each sentence, where l is the sentence

[2]The BERT Base Cased model is case sensitive and configured with default parameters: layers=12, hidden size=768, self-attention heads=12, total parameters=110M. We use the transformers library for PyTorch provided

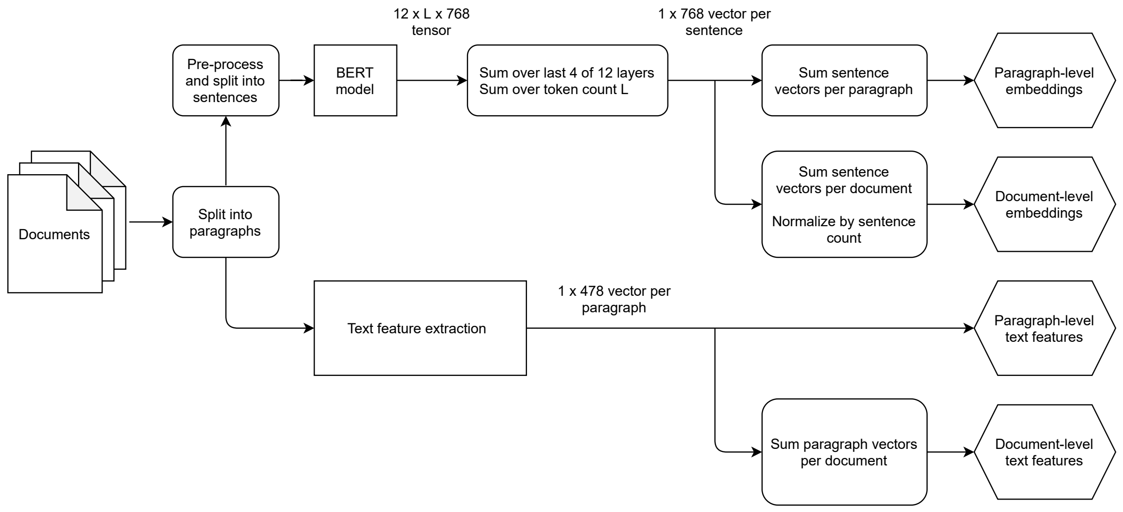
12 x L x 768
tensor
1 x 768 vector per
sentence
Pre-process and split into sentences
BERT model
Sum over last 4 of 12 layers
Sum over token count L
Sum sentence vectors per paragraph
Paragraph-level embeddings
Documents
Split into paragraphs
Sum sentence vectors per document
Normalize by sentence count
Document-level embeddings
Text feature extraction
1 x 478 vector per paragraph
Paragraph-level text features
Sum paragraph vectors per document
Document-level text features

Figure 2: Overview of the feature extraction process

length. We follow the recommended approach and sum the embeddings of the last four layers to obtain an $l \times 768$ tensor, and further sum over the first dimension to obtain sentence-level embeddings. While summing rather than averaging can lead to large discrepancies between sentences of different lengths, sentence length can be an important factor for style detection and is likely important to capture [12].

Converting sentence embeddings to paragraphs is achieved by adding each sentence embedding to produce a final paragraph feature vector of size 1×768. Although, Iyer and Vosoughi [12] achieves best performance by normalizing paragraph embeddings by sentence count, we achieve slightly higher performance by simply summing each vector. Document embeddings are obtained by summation of the containing paragraph embeddings and normalization by the document sentence count.

Text features: To extract text features that characterize writing style, we extract the same features as Zuo et al. [13] and Zlatkova et al. [11] on the paragraph level:

1. *Character-based lexical features:* The number of each distinct special character, spaces, punctuation, parentheses and quotation marks as separate features.
2. *Sentence- and word-based features:* Distribution of POS-tags, token length, number of sentences, sentence length, average word length, words in all-caps and counts of words above and below 2-3 and 6 characters as separate features.
3. *Contracted word forms:* Count of preference towards one type of contraction, e.g. "I'm" versus "I am". The total number of occurrences of contractions and fully written forms are used as separate features.
4. *Function words:* The frequency of each function word is counted and used as a separate feature. We use the same list as Zuo et al. [13] which is a combination of previously

by Hugging Face: https://huggingface.co/transformers/model_doc/bert.html defined words and the function word list from the NLTK[3] library.

5. *Readability indexes obtained using the Python Textstat[4] library:* Flesch reading ease score, Dale-Chall readability score, SMOG grade, Flesch-Kincaid grade, Coleman-Liau index, Gunning-Fog index, automated readability index and the Linsear Write readability metric. Additionally, we count the number of difficult words and keep all indexes as separate features.

Features are both extracted and saved per paragraph, before summed per document to obtain both paragraph- and document-level features. A total of 478 features per document and per paragraph.[5]

3.2. Stacking ensemble classifier

The idea behind using a stacking ensemble for classification is to combine the learning capabilities of multiple classifiers and reduce overall bias. Preliminary experiments showed that LightGBM, a state-of-the-art gradient boosting tree for classification [14], trained on the extracted text features outperformed models trained on BERT-embeddings. Furthermore, training on both BERT-embeddings and text features did not improve upon solely using text features. We hypothesized that training two sets of classifiers separately on each feature vector would allow the classifiers to obtain more discriminatory power on each vector. Thus, we train classifiers on text features and BERT-embeddings separately and combine their predictions by a meta-learner, i.e., a stacking ensemble architecture. Figure 3 presents an overview of the classification process for each task, and how the ensemble for each task consists of two sets of classifiers. More details on how each task is structured as binary classification is presented in the next subsection.

Our architecture is that of stacking with k-fold cross validation to avoid target leakage from base level classifiers to the meta-learner [15]. The meta-learner trains on k out-of-fold predictions produced by the base level classifiers and is evaluated on the validation set. We use LightGBM as our state-of-the-art classifier in all ensembles and select the best three Scikit-learn classifiers for each feature vector on each task. The exact configuration of the ensemble architecture is dependent on the task and we describe it in further detail in section 4.

Stacking ensembles have frequently been the winner of Kaggle competitions, in which avoiding overfitting to the training set is crucial for winning chances. Our results on the test set show a marginal decrease in performance compared to the validation set on task 2 and 3, and even *improved* performance on task 1. This indicates that we have been successful in avoiding overfitting to the training data.

3.3. Classification approach

With a pragmatic mindset we solve each task as binary classification. This requires some combination and processing of extracted text features and embeddings. An overview of the

[3]https://www.nltk.org/

[4]https://pypi.org/project/textstat/

[5]We kindly thank a reviewer for pointing out that addition of readability indexes does not make intuitive sense. Thus, performance could potentially be further improved by more carefully combining the paragraph-level feature vectors.

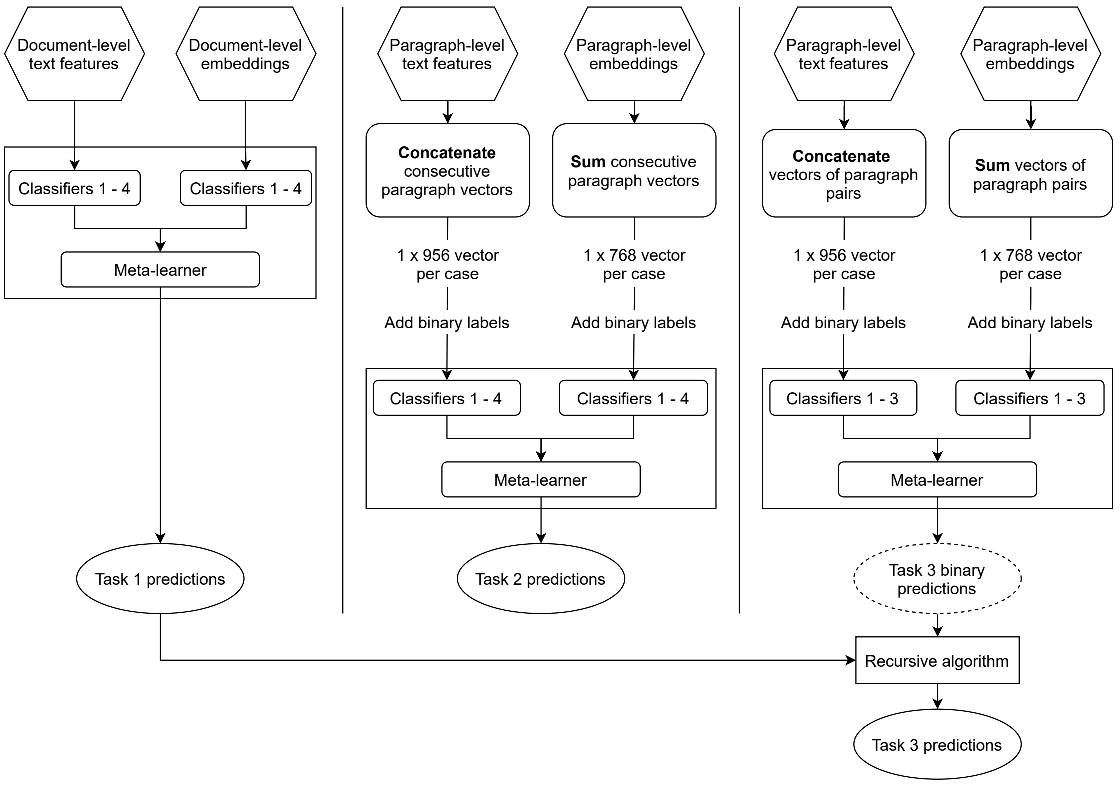

Figure 3: Overview of the classification process using stacking ensembles

approach is provided by Figure 3.

Task 1: Classifying single- or multi-author documents is achieved by classification on the document-level features already obtained in the feature extraction process. In this case, we have one feature vector per document and label and classify each document as being either single- or multi-authored. Previous research finds that classification on document-level features generally works very well when identifying the presence of style changes in documents [11] and multi-authored documents [12]. This is further supported by our high performance (macro F1-score of 0.7954) on task 1.

Task 2: To classify author change between two consecutive paragraphs, we combine the features of the two considered paragraphs to obtain the feature vector and classify an author change. This approach is similar to that of Iyer and Vosoughi [12]. Experimentation shows that, when considering text embeddings, addition of the two feature vectors yields the best result. For text features, we find that concatenating the arrays produce significantly better results than addition. Reasons for this could be that addition of the feature vectors simply average out the differences in the paragraphs, while concatenation keeps the features for each paragraph separate and, thus, provide more discriminative power. Keep in mind that this doubles the text feature vector to a size of 1×956. This formulation results in a total of $n - 1$ cases for each document where

```
Input: Binary predictions from task 1 and task 3
Output: Multi-label predictions for task 3
for $doc_k$ in documents do
    Predict single- or multi-author document;
    if single author then
        Assign all paragraphs to author 1;
        Continue to next document;
    Initialize empty array of paragraph labels, assign first label 1;
    for each $par_i$ in $doc_k$ starting from the second paragraph do
        Initialize empty array of similarity scores;
        Compare each prior paragraph with $par_i$ and predict same author probability;
        Add probability prediction to similarity scores;
        if $max(similarityscores) > 0.5$ then
            Assign $par_i$ same author as $\arg\max(similarityscores)$;
        else if number of assigned authors in $doc_k$ < 4 then
            Assign new author to $par_i$;
        else
            Assign $par_i$ same author as $\arg\max(similarityscores)$;
        Assign the author of $par_i$;
    Save array of author predictions for $doc_k$;
```

Figure 4: Recursive algorithm for obtaining task 3 multi-label predictions

n is the number of paragraphs.

As a reviewer points out in retrospect, we could have used task 1 predictions to first predict whether a document is single- or multi authored before obtaining task 2 predictions, as we do on task 3. We find this would have resulted in a performance increase of roughly 0.02 on task 2. Unfortunately, we did not implement this before the submission deadline as our focus was on solving task 3, being the novel task on style change detection.

Task 3 (binary): To assign paragraphs uniquely to all authors, we realize that we can pose this as a binary classification problem by asking whether any pair of paragraphs in the same document are written by the same author. In other words, we need to compare each paragraph to every other paragraph in a document and convert the provided multi-author label into a binary label. This implies $\binom{N}{2} = \frac{1}{2}(n-1)n$ cases for a document with n paragraphs. To classify two paragraphs, we combine their features in the same way as for task 2, i.e., addition of embeddings and concatenation of text features. The approach of classifying whether two paragraphs have the same author turns out to be very effective and achieves similar performance to that of task 2. However, the challenge becomes assigning each paragraph to its unique author in the multi-author multi-label setting.

Task 3 (multi-label): To obtain final multi-label predictions and assign each paragraph a unique author per document we devise a recursive strategy, using our task 1 and task 3 binary predictions. The process is detailed as an algorithm in Figure 4. Firstly, we use our task 1 classifier to determine whether we believe the document is single- or multi-authored. Documents classified as single-authored are assigned author 1 for all paragraphs. Secondly, in any given multi-authored document, the first paragraph is always assigned to author 1. Continuing, we compare the second paragraph with the first and determine the probability that it is authored by the same author, i.e., using our task 3 binary classification. If the probability is greater than 0.5, paragraph two is assigned to author 1, otherwise we assign it to author 2. Paragraph 3 is compared to *both* paragraph 1 and 2, and we assign its author to the *most similar* author that we previously determined. If it is unsimilar to both, we assign paragraph 3 to a new author. At some point we might have assigned paragraphs to all 4 authors, at which point we continue by assigning paragraphs by the most similar paragraph even though all previous paragraphs could have similarity scores below 0.5. This is a pragmatic approach, and there is certainly some information loss in this process. Especially for cases where there are errors on the first paragraphs as these will propagate and cause additional errors for later paragraphs. We did experiment with tuning the probability thresholds, however, this resulted in very minor performance improvements and we elected to keep similarity thresholds standard at 0.5. The improvement of this method is a suggested direction for future work.

4. Experimental Setting

In this section we present details on the experimental setting. For training the ensemble we used 4-fold stratified cross validation. Processing the training dataset for binary classification yields a total of 8,400 positive (2,800 negative) training cases for task one, 35,416 positive (30,636 negative) cases for task 2 and 125,679 positive (154,082 negative) cases for task 3.

The stacking ensemble was trained with half of the base level classifiers on text embeddings and the other half on text features. The selection of classifiers for each task and feature vector was based on selecting the best performing classifiers with different underlying methods (not only tree and boosting methods). We relied on the scikit-learn library [16] for all classifiers except LightGBM and evaluated the following classifiers as candidates: Support Vector Machines, AdaBoost, Decision Trees, Random Forest, Extra Trees, Multi-layer Perceptron, k-Nearest Neighbors, Logistic Regression, Linear Discriminant Analysis and Bernoulli- and Gaussian Naive Bayes. Table 1 shows the final ensemble configuration and which classifiers are trained on each feature vector and each task. Note that we only used 6 classifiers for task 3 due to the larger dataset. Furthermore, due to initial size and performance restrictions using the TIRA submission platform [9], we excluded the random forest classifier on task 3 which reduced validation performance by roughly 0.01.[6]

Hyperparameters was optimized for the LightGBM classifiers only, being the most important classifier in the ensemble. All other classifiers were trained with default parameters in the scikit-learn library, version 0.24.2. Given the large dataset, the tuning of 6-8 individual classifiers for each task was intractable given the scope and available resources. Furthermore, one could argue that optimization of each base level classifier is unnecessary as differences in bias is optimized by the meta-learner. LightGBM was optimized using the Optuna hyperparameter optimization framework [17]. We find that correcting for unbalanced datasets using LightGBMs

[6]Using random forest on task 3 resulted in model files upwards of 12GB and unreasonable memory usage due to the high dimensional data.

Table 1
The stacking ensemble configuration for each task. Each stacking ensemble consists of 3-4 classifiers trained on embeddings and 3-4 classifiers trained on text features, i.e., 6-8 classifiers in each ensemble.

	LightGBM	Random Forest	MLP	KNN	BernoulliNB
Task 1 embeddings	✓	✓	✓		✓
Task 1 features	✓	✓	✓		✓
Task 2 embeddings	✓	✓	✓		✓
Task 2 features	✓	✓	✓	✓	
Task 3 embeddings	✓		✓		✓
Task 3 features	✓		✓		✓

Note: MLP: Multi-layer perceptron. KNN: k-nearest neighbors. NB: Naïve Bayes.

"is_unbalance" parameter improves performance only on task 1 and 2. Preliminary experiments showed that reduction of the number of features based on LightGBMs feature importance reduced performance. Therefore, feature vectors were kept at their original size. Although common practice, we also found data scaling to reduce overall performance and was, thus, omitted.

5. Results

In the following, we present the results obtained by our approach. In addition to the macro F1-score we provide the accuracy score to allow comparison of our results with the PAN 2020 style change detection sub-tasks.[7]

Table 2 shows the validation results for the best LightGBM and ensemble model on each task. The best validation scores are marked in bold, achieving a macro F1-score of 0.7828, 0.7107 and 0.4261 on task 1, 2, and 3, respectively. Our results indicate that the ensemble model is superior to the best LightGBM model for any combination of task and performance measure. On task 3, we compare the results to that of using a uniform random guesser instead of our binary classifier and find that the ensemble outperforms the random baseline by 0.1375 points. However, as expected there is a significant reduction in performance going from binary to multi-label predictions, especially in the macro F1-score. As for binary classification, the custom ensemble outperforms LightGBM. Note that the random guesser has slightly higher than 0.5 accuracy, as task 1 predictions are still used to classify single-authored documents and assign labels to these cases to obtain comparable results.

Table 3 shows our final test results after submitting and evaluating our model on the TIRA platform. We achieve final scores of 0.7954, 0.7069 and 0.4240 for tasks 1, 2, and 3, respectively. Our solution is the best performing solution among submitted solutions for task 1, and among top 3 for tasks 2 and 3. The test scores are similar to validation scores, and we observe a performance *increase* on task 1, indicating that our models have generalized well to the test set.

[7]The PAN 2020 style detection tasks was evaluated by micro-averaged F1, which reduces to accuracy for binary classification problems.

Table 2
Results on validation set. Task 3 binary results are preliminary to the final multi-label results.

	Task 1		Task 2		*Task 3 (binary)*		Task 3 (multi-label)		
	LGBM	Ensemble	LGBM	Ensemble	*LGBM*	*Ensemble*	Random	LGBM	Ensemble
Macro F1	0.7761	**0.7828**	0.6783	**0.7107**	*0.6968*	*0.7175*	0.2886	0.4115	**0.4261**
Accuracy	0.8446	0.8475	0.6829	0.7141	*0.7020*	*0.7209*	0.5232	0.6232	0.6400

Table 3
Final results on the test set using the stacking ensembles on all tasks

	Task 1	Task 2	Task 3
Macro F1-score	0.7954	0.7069	0.4240

6. Conclusion and Future Work

In this paper, we propose a solution to the PAN 2021 shared task on style change detection, attempting to answer the question: Given a document, can we find evidence for the document being written by multiple authors, and are we able to attribute paragraphs to respective authors based on their writing style?

Our solution provides encouraging results. We apply previously tested feature extraction methods and develop a custom stacking ensemble framework trained separately on different feature vectors. Furthermore, we propose a pragmatic solution to the difficult problem of multi-label multi-output classification by first solving a relaxed problem by binary classification.

We achieve macro F1-scores on the validation set of 0.7761 when classifying single- and multi-authored documents (task 1), and 0.7107 when detecting an author change between consecutive paragraphs (task 2). For multi-label author attribution (task 3), our solution achieves a score of 0.4261, significantly outperforming random guesses (0.2886). The relaxed formulation of task 3 achieves a macro F1-score of 0.7175. On the test set, our final submission scores are 0.7954, 0.7069, and 0.4240 for tasks 1, 2 and 3 respectively and our solution is the best performing solution on task 1 among other submitted solutions. Lastly, our results indicate that the use of stacking ensembles improves performance across reported metrics when compared to the optimized LightGBM model. We suggest interesting directions for future work:

1. The use of stacking ensembles yields small performance improvements considering the effort spent on building the models. Thus, we hypothesize that identifying additional key features is paramount for further improvements.
2. Given the reduced multi-label performance on task 3, there are likely opportunities to increase performance going from binary to multi-label predictions. Exploring the use of hierarchical classification or comparison of document paragraphs in unison (as opposed to recursive comparisons) would be interesting for further research and might significantly increase performance on task 3.

Acknowledgments

This work was motivated by academic interests and undertaken as a project in the course *TDT4310 - Intelligent Text Analytics and Language Understanding* at the Norwegian University of Science and Technology in Trondheim. I would like to thank professor Björn Gambäck for organizing a complete and well rounded course on natural language processing.

References

[1] T. C. Mendenhall, The characteristic curves of composition, Science (1887) 237–246. doi:10.1126/science.ns-9.214S.237.

[2] F. Iqbal, H. Binsalleeh, B. C. Fung, M. Debbabi, Mining writeprints from anonymous e-mails for forensic investigation, Digital Investigation 7 (2010) 56–64. doi:10.1016/j.diin.2010.03.003.

[3] S. Vosoughi, H. Zhou, D. Roy, Digital stylometry: Linking profiles across social networks, in: Social Informatics, Springer International Publishing, 2015, pp. 164–177. doi:10.1007/978-3-319-27433-1_12.

[4] M. Tschuggnall, E. Stamatatos, B. Verhoeven, W. Daelemans, G. Specht, B. Stein, M. Potthast, Overview of the Author Identification Task at PAN 2017: Style Breach Detection and Author Clustering, in: L. Cappellato, N. Ferro, L. Goeuriot, T. Mandl (Eds.), Working Notes Papers of the CLEF 2017 Evaluation Labs, volume 1866 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2017. URL: http://ceur-ws.org/Vol-1866/.

[5] M. Kestemont, M. Tschuggnall, E. Stamatatos, W. Daelemans, G. Specht, B. Stein, M. Potthast, Overview of the Author Identification Task at PAN-2018: Cross-domain Authorship Attribution and Style Change Detection, in: L. Cappellato, N. Ferro, J.-Y. Nie, L. Soulier (Eds.), Working Notes Papers of the CLEF 2018 Evaluation Labs, volume 2125 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2018. URL: http://ceur-ws.org/Vol-2125/.

[6] E. Zangerle, M. Tschuggnall, G. Specht, M. Potthast, B. Stein, Overview of the Style Change Detection Task at PAN 2019, in: L. Cappellato, N. Ferro, D. Losada, H. Müller (Eds.), CLEF 2019 Labs and Workshops, Notebook Papers, CEUR-WS.org, 2019. URL: http://ceur-ws.org/Vol-2380/.

[7] E. Zangerle, M. Mayerl, G. Specht, M. Potthast, B. Stein, Overview of the Style Change Detection Task at PAN 2020, in: L. Cappellato, C. Eickhoff, N. Ferro, A. Névéol (Eds.), CLEF 2020 Labs and Workshops, Notebook Papers, CEUR-WS.org, 2020. URL: http://ceur-ws.org/Vol-2696/.

[8] E. Zangerle, M. Mayerl, M. Potthast, B. Stein, Overview of the Style Change Detection Task at PAN 2021, in: CLEF 2021 Labs and Workshops, Notebook Papers, CEUR-WS.org, 2021.

[9] 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.

[10] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, 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, Association for Computational Linguistics, 2019, pp. 4171–4186. doi:10.18653/v1/N19-1423.

[11] D. Zlatkova, D. Kopev, K. Mitov, A. Atanasov, M. Hardalov, I. Koychev, P. Nakov, An Ensemble-Rich Multi-Aspect Approach for Robust Style Change Detection—Notebook for PAN at CLEF 2018, in: L. Cappellato, N. Ferro, J.-Y. Nie, L. Soulier (Eds.), CLEF 2018 Evaluation Labs and Workshop – Working Notes Papers, 10-14 September, Avignon, France, CEUR-WS.org, 2018. URL: http://ceur-ws.org/Vol-2125/.

[12] A. Iyer, S. Vosoughi, Style Change Detection Using BERT—Notebook for PAN at CLEF 2020, in: L. Cappellato, C. Eickhoff, N. Ferro, A. Névéol (Eds.), CLEF 2020 Labs and Workshops, Notebook Papers, CEUR-WS.org, 2020. URL: http://ceur-ws.org/Vol-2696/.

[13] C. Zuo, Y. Zhao, R. Banerjee, Style Change Detection with Feed-forward Neural Networks, in: L. Cappellato, N. Ferro, D. Losada, H. Müller (Eds.), CLEF 2019 Labs and Workshops, Notebook Papers, CEUR-WS.org, 2019. URL: http://ceur-ws.org/Vol-2380/.

[14] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, Lightgbm: A highly efficient gradient boosting decision tree, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, volume 30, Curran Associates, Inc., 2017.

[15] C. C. Aggarwal, Data Classification: Algorithms and Applications, 1st ed., Chapman & Hall/CRC, 2014.

[16] 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, Édouard Duchesnay, Scikit-learn: Machine learning in python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[17] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: A next-generation hyperparameter optimization framework, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 2623–2631. doi:10.1145/3292500.3330701.