Style Change Detection
NLP Course Project

Matteo Donati, Ildebrando Simeoni and **Diego Biagini**
Master's Degree in Artificial Intelligence, University of Bologna
{matteo.donati10, ildebrando.simeoni, diego.biagini2}@studio.unibo.it

Abstract

This paper describes two simple but effective approaches for the three Style Change Detection (SCD) tasks for PAN at CLEF 2022. The first approach we propose is based on fine-tuning the BERT transformer and training different classification heads so to solve all three tasks in an end-to-end fashion. The second approach, instead, is based on sentence transformers (i.e. SBERT-based architectures) and cosine similarity to solve the given classification problems by working directly in the embedding space implied by the chosen transformer-based encoder. We show how, using both approaches, we are able to achieve adequate results and also outperform existing state-of-the-art methods in a particular task.

1 Introduction

The SCD problem belongs to the author identification class of problems, and it is the only means to detect plagiarism in a document if no comparison texts are given. Likewise, style change detection can help to uncover gift authorships, to verify a claimed authorship, or to develop new technology for writing support.

The goal of the SCD-2022 tasks is to identify text positions within a given multi-author document at which the author switches (Zangerle et al., 2022). In particular, given a document combined from the StackExchange questions and answers, participants are asked to solve three tasks: 1). Style Change Basic: for a text written by two authors that contains a single style change only, find the position of this change (i.e., cut the text into the two authors' texts on the paragraph-level); 2). Style Change Advanced: for a text written by two or more authors, find all positions of writing style change (i.e., assign all paragraphs of the text uniquely to some author out of the number of authors assumed for the multi-author document); 3). Style Change Real-World: for a text written by two or more authors, find all positions of writing style change, where style changes now not only occur between paragraphs, but at the sentence level. All the possible scenarios are shown in figure 1.

In all the previous editions of the competition, the joining teams tried to tackle the given problems by proposing a mixture of techniques, including ensembles of models, feature-based and similarity-based approaches (Zangerle et al., 2021, 2020, 2019). Based on the results of such approaches, it is possible to notice how end-to-end neural models are the models that best perform at solving the given tasks. This motivates us to explore the use of pre-trained transformer models to address the authorship analysis issue for writing style changes. Indeed, pre-trained models can be used to solve similar problems, using transfer learning techniques to fine-tune the pre-trained model to fit a particular downstream task.

In this paper we propose two similar but different approaches that try to solve all the three SCD-2022 tasks in a unified manner, by relying on neural models that incorporate pre-trained encoders such as the BERT encoder (Devlin et al., 2018) and SBERT-based encoders (Reimers and Gurevych, 2019). In particular, the first approach takes into consideration the BERT encoder so to produce contextual embeddings that can be fed to fully-connected or convolutional layers and produce the final probability distribution that determines whether a couple of paragraphs or sentences of text is written by the same author. The second approach, instead, considers the a sentence transformer encoder so to produce embeddings that can be directly compared using cosine similarity, and thus determining whether two paragraph or sentences of text are written by the same author.

The performances of the two approaches have been measured on the given validation set, using macro F_1-score. In particular, compared to the current state-of-the-art, both of the aforementioned ap-

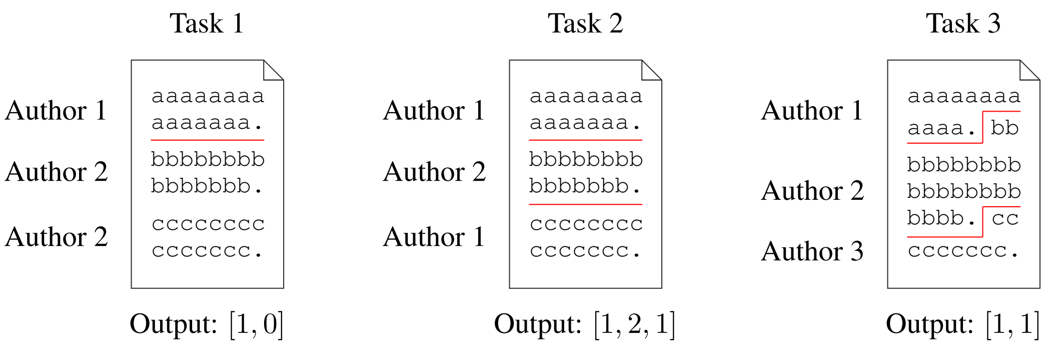


Figure 1: Possible scenarios and expected outputs

proaches worked well, achieving adequate scores. Moreover, the sentence transformer approach outperformed existing state-of-the-art methods at solving the Task 1 problem.

2 Background

The SCD-2018 edition focused on detecting whether a document is single-authored. This task can be treated as a binary classification problem. In that edition, different approaches were used to detect style changes, including ensembles of supervised learning models (Zlatkova et al., 2018) and feature-based approaches (Safin and Ogaltsov, 2018; Khan, 2018).

The SCD-2019 edition added a task for detecting the actual number of authors within a single document. One successful approach (Nath, 2019) was based on clustering algorithms, so to yield a number of clusters corresponding to the number of authors.

The SCD-2020 edition added another task based on detecting style changes between two consecutive paragraphs. In this case, a paragraph representation based on character, lexical and syntactic features was proposed, using a clustering algorithm for style change detection (Castro-Castro et al., 2020). Another group used the BERT model to generate embedding representations and used such embeddings to train a random forest classifier to detect style changes between two paragraphs (Iyer and Vosoughi, 2020).

The SCD-2021 edition also added a task to detect the number of authors and locating specific author changes within the text. Style change and author identification were regarded as binary classifications based on similarity measurements (Zhang et al., 2021).

Lastly, the SCD-2022 edition introduced the Style Change Real-World task, in which style changes can also occur at the sentence level. The work we present with this paper is deeply inspired by last editions' solutions. Indeed, both of our approaches use famous neural language models and use the computed embeddings to train specific classifiers or to compute similarity scores between texts.

The main innovation we tried to bring is the use of metric learning and siamese networks to compute similarity measurements in the vector space implied by the specific encoder. Indeed, the aim of metric learning is to compare low-dimensional embeddings of some input so to find semantically similar inputs in an effective way. One important type of neural models that is capable of finding similar inputs, as stated by the definition of metric learning, is the siamese neural network. This network is virtually composed of two backbones that share weights and work with two different input vectors to compute comparable output vectors. An example of such network is the SBERT model. SBERT is a modification of the BERT model that uses siamese and triplet networks to derive semantically meaningful embeddings that can be compared using similarity measures. The authors of SBERT describe this model as a single BERT model that has been trained using siamese as well as triplet architectures, depending on the structure of the data at hand.

After a careful analysis of the capabilities of such network, we made sure to properly manipulate the given datasets so to fine-tune specific sentence transformers using both the contrastive loss (Hadsell et al., 2006) as well as the triplet objective function described by Reimers and Gurevych, 2019. Indeed, the idea behind metric learning is to use specific losses to guide the model to favor a clustered structure of the embeddings such that: 1). the distance between semantically similar inputs is minimized; 2). the distance between semantically different inputs is maximized.

3 System description

Regarding the first approach, the pipeline we implemented is the following. We started by analysing all the three SCD-2022 datasets and by creating appropriate tables that could fit our needs.

After having computed the aforementioned tables, we noticed how the given datasets, especially the one related to task 1, were not balanced. To solve this problem, we preprocessed the data so to be able to effectively train our models. The techniques we used to preprocess the data are described, in more details, in section 4.

Then, we defined our BERT-based models. All the layers that define these models are shown in table 1. In particular, the BERT version we used is the BERT base cased version[1]. The layers of the classification head of each m_x^1 model were inspired by some successful works presented in the current edition of the SCD challenge (Lao et al., 2022; Lin et al., 2022). Indeed, the classification heads we defined are either composed of fully-connected layers or composed of uni-dimensional convolutional layers. Moreover, in order to be able to detect style changes between two different paragraphs or sentences, we had to manipulate the aforementioned dataframes so to incorporate two different consecutive paragraphs or sentences within the same BERT input. To do this, we used the official BERT tokenizer, that uses the special `[CLS]` and `[SEP]` tokens to build the different inputs. Thus, the inputs of the model are list of tokens of the form `[CLS] P/S-A [SEP] P/S-B [SEP]`, where `P/S-A` corresponds to the encoding of a paragraph/sentence, and `P/S-B` to the encoding of the subsequent paragraph/sentence. Other than the aforementioned list of token, for each possible pair of texts we produced: 1). a boolean mask, having the same length of the input tokens list, that determines at which positions the input list of tokens contains padding; 2). a boolean list, having the same length of the input tokens list, that determines the positions of `P/S-A` and of `P/S-B` within the input list.

The last bit of data manipulation was related to casting the Style Change Advanced problem (i.e. task 2), to be a binary classification problem, as in the case of task 1 and task 3. This was made possible by applying data augmentation to the validation set of task 2, thus considering all possible pairs of paragraphs. By doing this, we were able to tackle all the three tasks by using the same architectures and methodologies. However, by treating task 2 as a binary classification problem, we had to define a way to produce the corresponding list of authors out of the produced binary output of all models. The algorithm that performs the aforementioned conversion is defined as follows. The label of the first paragraph, P_1, is initialized as 1, indicating the first author. The second paragraph, P_2 is compared with P_1. If this pair obtains the class label 1 (i.e. a style change is detected), then the author label for P_2 will be 2 . Otherwise, if the class label is 0, the author of P_2 should be the same as the one for P_1 (i.e. 1). Thus, every paragraph is compared against all subsequent ones (e.g., P_1 with P_3, P_1 with P_4, etc.) to produce an ordered author list (limited to at most five distinct authors).

Compared to the first approach, where a plain transformer with standard softmax is used, the second approach focuses on metric learning. To detect whether two passages are written by the same author, given an encoder model we can proceed as follows: 1). obtain an embedding for the two input passages; 2). compute the cosine similarity between the two; 3). apply a threshold on such similarity, below which we say that the two documents are written by different authors. Indeed, as previously mentioned, the paradigm of metric learning ensures that embeddings of texts from different authors have low similarity, while embeddings from the same author should cluster together.

This general metric training strategy was applied to all the three tasks, however when we actually want to solve them we can exploit some of the properties of the current task to define more effectively how to perform inference to detect where style changes occur. In particular, for task 1 we know that we only have two authors and the change between them happens only once. Thus we can solve the problem of finding the style change by just finding the consecutive pairs of passages that differ the most. In tasks 2 and 3 we don't have this extremely useful bias, and we actually have to take the similarity between two paragraphs or sentences and decide on a threshold below which we can state that the two authors are not the same.

As mentioned above, for task 2 we not only have to detect the style change points, but also assign

[1]bert-base-cased

Model	Encoder	Classifier
m_1^1	BERT	Global Max-Pool, FC-2
m_2^1	BERT	Global Avg-Pool, FC-2
m_3^1	BERT	Global Max-Pool, FC-128, FC-2
m_4^1	BERT	(Conv1D-128, Max-Pool) $\times$ 3, Flatten, FC-2

Table 1: BERT-based architectures

Model	Encoder	Training
m_1^2	Sentence transformer	No fine-tuning
m_2^2	Sentence transformer	Fine-tuned with contrastive loss
m_3^2	Sentence transformer	Fine-tuned with triplet loss
m_4^2	Sentence transformer	Fine-tuned with triplet loss, hard/semi-hard samples only

Table 2: SBERT-based architectures

each paragraph to an author, that is something akin to a clustering problem. Differently to how we handled the same problem in the first approach, here two methods were devised: a threshold based method and an unsupervised clustering method. In particular, the threshold based method works by computing the cosine similarity between each paragraph and all previous ones, and by using a threshold to create clusters of paragraphs. The clustering method, instead, uses the pairwise cosine distances to run the DBSCAN clustering algorithm.

When considering this second approach, task 3 is similar to task 2 in the sense that we cannot work on similarities only, but we need a threshold to decide when to detect a style change.

As previously stated, task 2 and task 3 require the definition of a threshold under which two passages are said to be from different authors. Two methods of threshold selection in the contrastive case were implemented: 1). the threshold is set to be equal to the margin value, α, used by the loss function. Indeed, the contrastive loss is formulated in such a way that:

$$d(e_1, e_2) \begin{cases} \simeq 0 & \text{if } a(x_1) = a(x_2) \\ > \alpha & \text{if } a(x_1) \neq a(x_2) \end{cases} \tag{1}$$

where $d(e_1, e_2)$ is the distance between e_1 and e_2, and $a(e_x)$ defines the author of the paragraph whose embedding is e_x; 2). the threshold is inferred from the training set. After training we can apply our model to the contrastive pairs and measure the average similarity for same-author samples and different-author samples, the threshold should lie between these two averages. In practice it was found to be advantageous to consider the standard deviation of these two groups of similarities as well, in this way if for example the range of similarities for the same-author group is low, the threshold should be put closer to its mean than the mean of the different-authors group.

A similar approach can be used to get the similarity for the triplet examples, by splitting each triplet example in two contrastive ones, thus computing the average of same-author similarities between anchors and positives and the average of different-author similarities between anchors and negatives.

A general overview of all the sentence transformer architectures is shown in table 2.

4 Data

The datasets used for this work are the ones provided by PAN for the competition. In particular, three datasets, one for each task, have been provided; each of which consisted of three sets (i.e. a training, a validation and a test set). Among these sets, only the training and validation sets have been used. We did not use the test set because it was the only one without ground truth targets. A general overview of the given data is presented in table 3.

The datasets are based on user posts from various sites of the StackExchange network, covering different topics. Each input problem (i.e. the document for which to detect style changes) is referred as by an ID. For each document `X`, two files are provided: 1). `problem-X.txt` which contains the actual raw text, where paragraphs are separated by spaces; 2). `truth-problem-X.json` which contains the ground truth and other useful infor-

Task	Split	#Documents	#Authors				
			1	2	3	4	5
1	Train	1400	-	1400	-	-	-
	Validation	300	-	300	-	-	-
2	Train	7000	1400	1400	1400	1400	1400
	Validation	1500	300	300	300	300	300
3	Train	7000	1400	1400	1400	1400	1400
	Validation	1500	300	300	300	300	300

Table 3: Datasets overview

mation about the problem (e.g. number of authors, source of the text), in JSON format.

As a first step in the data management pipeline, the information in both files has been merged into a more convenient dataframe form by means of the `create_dataframe` function. In particular, the dataframe is composed of seven attributes: `authors`, the total number of authors in the generic row (i.e. document); `site`, the source site from which the document was extracted; `multi-author`, a boolean flag indicating whether the generic document has been written by multiple authors; `changes`, a list of boolean varibles determining changes between paragraphs or sentences; `paragraph-authors`, a list of integer variables that label each paragraph or sentence with an author ID; `input_text`, the content of the entire document; `splitted_text`, the list of paragraphs or sentences of the generic document.

After having casted all the tasks to binary classification problems, we noticed how the classes, especially for task 1, were heavily unbalanced toward class 0 (i.e. no style change detected between consecutive passages), being much more frequent the absence of author change in two subsequent sentences/paragraphs, leading to poor performances of our classifiers.

In order to solve the problem, a data augmentation process has been developed. In particular, two types of data augmentation have been defined. In the first case, each passage is not only compared with the following one, but it is compared to all the others paragraphs in the document (total upsampling). In the second case instead, each passage is compared with all the subsequent passages (partial upsampling). By adopting this strategy we were able to introduce synthetic data, that, together with an accurate choice of a total number of samples per classes, resulted in a class distributions rebalancing (visual results of this process can be appreciated in the notebook).

In particular, no data augmentation was performed on the task 1 and 3 validation sets, that have been used as test sets, while a partial upsampling was performed on the task 1 training set, being the most unbalanced one. For task 2 a total upsampling of the validation set was required in order to structure produce the final author IDs list as defined in section 3.

Regarding the second approach, our datasets allow for both a contrastive and triplet approach. To train our m_x^2 models with contrastive loss, a training set of contrastive pairs and their similarity (i.e. 1 if they are from the same author, 0 otherwise) has to be created. As for the previous approach, we made sure to balance the datasets so to be able to effectively train our models. In particular, due to hardware limitations, we set the following maximum values for the number of samples in the computed training sets: 20000 for task 1, 50000 for task 2, 100000 for task 3. For triplet loss, instead, we built a dataset of triplets where the anchor is a passage from an author i, the positive is a different passage in the same conversation by the same author i and the negative is a passage from a different author j. In building such a triplet dataset we can also consider the difficulty of samples, thus we evaluated our models building both general triplet datasets, where no filtering of examples is performed, as well as hard/semi-hard datasets, where only hard and semi hard examples are put into the training set.

5 Experimental setup and results

Regarding the experiments performed for the first approach, the setup was the following one. After having set the needed seeds in order to

Model		Task 1	Task 2			Task 3	
		-	Margin thr.	Mean thr.	DBSCAN	Margin thr.	Mean thr.
MPNet	m_1^2	0.59	-	0.24	0.26	-	0.54
	m_2^2	0.79	0.38	0.44	0.21	0.67	0.69
	m_3^2	0.79	0.42	0.37	0.38	0.60	0.65
	m_4^2	0.78	0.41	0.36	0.39	0.61	0.65
Style-Embedding	m_1^2	0.70	-	0.33	0.31	-	0.58
	m_2^2	0.78	0.32	0.44	0.17	0.64	0.67
	m_3^2	0.78	0.42	0.39	0.38	0.61	0.66
	m_4^2	0.80	0.41	0.36	0.38	0.59	0.64

Table 4: Macro F_1-scores for the m_x^2 architectures

Model	Task 1	Task 2	Task 3
m_1^1	0.72	0.28	0.66
m_2^1	0.71	0.45	0.66
m_3^1	0.71	0.44	0.66
m_4^1	0.68	0.43	0.66
m_1^2	0.70	0.33	0.58
m_2^2	0.79	0.44	0.69
m_3^2	0.79	0.42	0.66
m_4^2	**0.80**	0.41	0.65
Lin et al., 2022	0.74	**0.54**	**0.73**

Table 5: Best macro F_1-scores

ensure reproducibility of our experiments, three different variations of a baseline BERT-based model have been defined. In particular, the baseline was composed of three input layers required by BERT transformer (i.e. the padded `[CLS] P/S-A [SEP] P/S-B [SEP]` list of tokens, the boolean mask that determines at which positions the input list of tokens contains padding, the boolean list that determines the positions of `P/S-A` and of `P/S-B` within the input list), the transformer layer itself and then, after a pooling layer that reduced the dimension of the transformer's output, a dense layer that produced the binary classification output.

Starting from the m_1^1 baseline, some variations have been explored. In particular, we tried to improve the aforementioned baseline by changing the type of pooling operator (m_2^1), by adding more dense or 1D convolutional layers (m_3^1 and m_4^1 respectively), and by also changing the type of transformer. Changing the transformer encoder did not yiled good results, so we preferred to keep BERT as our main and only option.

All the m_x^1 models have been trained for 2 epochs, with Adam as optimizer and cross-entropy as loss objective. Moreover, we used a batch size of 16 and an initial learning rate of $5e{-}5$ for all tasks. A custom learning rate scheduler has been defined so to implement a learning rate schedule that modifies the learning rate within the same epoch (i.e. after a specific number of mini-batches). In particular, both the number of mini-batches to consider and the reduction of the learning rate are hyperparameters that have been carefully tuned after a number of trials. Experiments with both the custom learning rate scheduler and predefined ones led to the conclusion that the former approach was more suitable for our tasks and resulted in better training of our models.

For task 1 and 2 we set the maximum length of a pair of passages to be 256 tokens long, while for task 3 we set this value to 64 tokens, being phrases much more concise than paragraphs. Indeed, considering task 3, this modification led to an increase in both performances and results.

During the training procedure, the BERT layers of each model have been fine-tuned on the particular tasks at hand, so to produce contextual embeddings that are relevant for our specific problems. Once trained, the models have been tested using the validation sets provided by the authors of the SCD-2022 challenge. In particular, the performances of each model were measured using the macro F_1-score, being this the metric used for the evaluation of the challenge.

The metric learning approach, instead, was tested along different dimensions, from the model choice to the training regime, and dataset creation. In particular, for each task we trained the models using only data that was provided for that task. This was needed especially for task 3, where the passages that we had to analyze are sentences, instead of entire paragraphs, and differ considerably from the samples we have for task 1 and 2.

One of the first choices we had to make was which pretrained sentence transformers model to fine-tune. We decided to use two base models, a general purpose one based on the MPNet architecture by Microsoft[2] (Song et al., 2020), and a model which was already trained to discern writing style in text, named Style-Embedding[3], that is based on RoBERTa. In particular, for each task and for each base model, we trained our four architectures as described in table 2.

For all of the tasks we used the AdamW optimizer with a weight decay of $1e{-}2$, a learning rate of $2e{-}5$, with a linear warm-up schedule that peaks at 10% of the training steps, and a batch size of 8; the margin, α, for the contrastive/triplet loss has been set to 0.6. Moreover, the models that follow the m_2^2 and m_3^2 architectures were trained for 2 epochs, while the m_4^2 architectures were trained for 3 epochs, since less examples were available.

For task 1 and 2 we set the maximum length of a passage to be 128 tokens, while for task 3 we set this value to 32 tokens.

Considering task 2, in order to properly produce the final author IDs list, we tested our approach using both the threshold based method as well as the clustering method based on DBSCAN. In the latter case, alongside the already computed similarity matrix, the following hyperparameters we set the minimum number of points needed to form a cluster to 1 and $\varepsilon = 0.5$. Moreover, considering tasks 2 and 3, in order to detect whether two texts are written by the same authors, we tested our approach using as threshold both the margin α of the contrastive loss as well as the aforementioned threshold computed on the training set. The results of all these tests, on each validation set, are shown in table 4.

Lastly, the best macro F_1-scores, for both of the two approaches, are presented in table 5. This tables also presents the current state-of-the-art for all three tasks. Figure 2, instead, shows the performances of each model with respect to the total number of authors of the given documents.

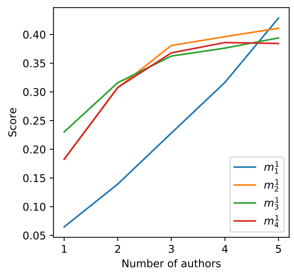
6 Discussion

Table 5 shows all the best F_1-scores we achieved during our experiments. Considering the first approach (i.e. the m_x^1 architectures), it is possible to notice how for task 1 our baseline model is the best performing one, while for task 2 the variations m_2^1, m_3^1 and m_4^1 reached a significant improvement with respect to the baseline. However, no significant and consistent change in the results of a specific model variation enabled us to spot promising directions for the definition of new and more complex models for the problem at hand.

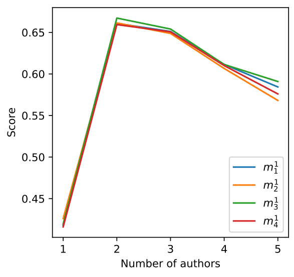
In all cases, with our best performing models, we reached results comparable to the current state-of-the-art for all the tasks, in particular for task 1, while task 2 remains the most challenging one.

Lastly, considering the results on task 2 and task 3 (namely, the two tasks that present a total number of authors within $[1, 5]$), some error analysis, shown in figure 2, was performed. In this last phase, we wanted to study the performances of each model with respect to the total number of authors of the given documents, and to compare the four m_x^1 models based on the trends we obtained from this study. When looking at how the model behaves in tasks 2 and 3 as the number of author changes (figure 2), we can notice that all models have the biggest problems when the conversation is written by a single author. Furthermore, looking at task 2, there seems to exist a positive correlation between the total number of authors and the computed macro F_1-score. Meanwhile, for task 3, the model obtains the best results on conversations between two authors and then gently declines. This last trend could be due to the algorithm used to compute the final author IDs list from the models' output. For future

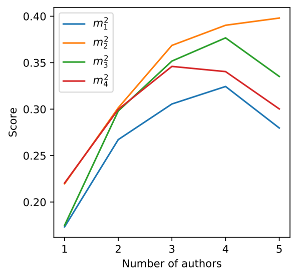
[2]sentence-transformers/all-mpnet-base-v2
[3]AnnaWegmann/Style-Embedding



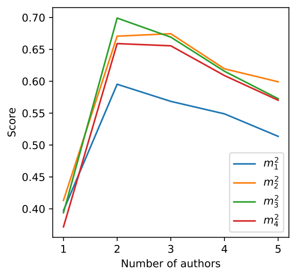
(a) m_x^1 on task 2



(b) m_x^1 on task 3



(c) m_x^2 on task 2



(d) m_x^2 on task 3

Figure 2: Macro F_1-scores with respect to number of authors

developments, it would be worth trying a variation of the proposed labeling algorithm.

Regarding the second approach and considering the baseline sentence transformer models (i.e. the two m_1^2 models), it is possible to notice how the Style-Embedding models always outperform the MPNet models. However, this gap between the two types of architectures is not present when the models are fine-tuned (m_2^2, m_3^2 and m_4^2), with both MPNet and Style-Embedding reaching similar performances on all tasks under all inference methods. In particular, the overall best result on task 1 is obtained with Style-Embedding (m_4^2 with a macro F_1-score of 0.8), while for task 2 the performances are almost the same, and for task 3 the winner is MPNet (m_2^2 with a macro F_1-score of 0.69).

Considering tasks 2 and 3, it seems that computing a threshold based on the training dataset is usually better than fixing it to the margin of the loss function. Surprisingly enough, this does not hold for task 2 when using triplet loss.

Considering only task 2, the results yielded by the DBSCAN-based approach seem to almost always be worse compared to the results yielded by the threshold based method. This is likely due to the decision of not tuning the hyperparameters of DBSCAN.

With regards to the choice of which triplets to choose when using triplet loss, it seems like using only hard and semi-hard examples has overall a very slight negative impact on the results. However this could be due to the lower number of samples obtained and consequently to the lower number of examples seen by the model.

Lastly, it can be noticed how the trends shown in figure 2, for all m_x^2 models, resemble the ones shown for the m_x^1 models.

7 Conclusion

This study describes the model design, system implementation and performances of two different types of architectures for the PAN 2022 style change detection tasks. In particular, we selected pre-trained transformer models as staring point, and fine-tuned the corresponding downstream classifiers, leading to the definition of the m_x^1 architectures. Then, we made use of two successful types of sentence transformer so to exploit metric learning and effectively compute similarity scores between different passages of text, leading to the definition of the m_x^2 architectures.

Based on evaluation results, both of our approaches yielded promising results, although considerably more work is needed for task 2. In particular, we showed how BERT-based models are able to compete with current state-of-the-art approaches, while with the use of sentence transformer and of metric learning we were able to surpass the state-of-the-art in the first task of this year's challenge.

In order to improve the performances on task 2, future works should introduce more advanced labeling algorithms, capable of handling models' output in a more effective manner, thus avoiding the propagation of labeling errors when generating the final author IDs list.

Lastly, from figure 2, we noticed how the computed macro F_1-scores have similar trends to the ones proposed by existing literature (Zangerle et al., 2021).

8 Links to external resources

The following is the link to the dataset for the Style Change Detection task of PAN 2022: `https://zenodo.org/record/6334245#.Y6sRf3bMJPb`. The decision whether to grant/deny access to all datasets is solely under the responsibility of the data owner.

References

Daniel Castro-Castro, Carlos Alberto Rodríguez-Losada, , and Rafael Munoz. 2020. Mixed style feature representation and b_0-maximal clustering for style change detection. *PAN at CLEF 2020*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

R. Hadsell, S. Chopra, and Y. LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742.

Aarish Iyer and Soroush Vosoughi. 2020. Style change detection using bert. *PAN at CLEF 2020*.

Jamal Ahmad Khan. 2018. A model for style change detection at a glance notebook for pan at clef 2018. *PAN at CLEF 2018*.

Qidi Lao, Li Ma, Wenyin Yang, Zexian Yang, Dong Yuan, Zhenlin Tan, and Langzhang Liang. 2022. Style change detection based on bert and conv1d. *PAN at CLEF 2022*.

Tzu-Mi Lin, Chao-Yi Chen, Yu-Wen Tzeng, and Lung-Hao Lee. 2022. Ensemble pre-trained transformer models for writing style change detection. *PAN at CLEF 2022*.

Sukanya Nath. 2019. Style change detection by threshold based and window merge clustering methods. *PAN at CLEF 2019*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084.

Kamil Safin and Aleksandr Ogaltsov. 2018. Detecting a change of style using text statistics. *PAN at CLEF 2018*.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. *CoRR*, abs/2004.09297.

Eva Zangerle, Maximilian Mayerl, Martin Potthast, and Benno Stein. 2020. Overview of the style change detection task at pan 2020. *PAN at CLEF 2020*.

Eva Zangerle, Maximilian Mayerl, Martin Potthast, and Benno Stein. 2021. Overview of the style change detection task at pan 2021. *PAN at CLEF 2021*.

Eva Zangerle, Maximilian Mayerl, Martin Potthast, and Benno Stein. 2022. Style change detection 2022.

Eva Zangerle, Michael Tschuggnall, Gunther Specht, Benno Stein, and Martin Potthast. 2019. Overview of the style change detection task at pan 2019. *PAN at CLEF 2019*.

Zhijie Zhang, Zhongyuan Han, Leilei Kong, Xiaogang Miao, Zeyang Peng, Jieming Zeng, Haojie Cao, Jinxi Zhang, Ziwei Xiao, and Xuemei Peng. 2021. Style change detection based on writing style similarity. *PAN at CLEF 2021*.

Dimitrina Zlatkova, Daniel Kopev, Kristiyan Mitov, and Atanas Atana. 2018. An ensemble-rich multi-aspect approach for robust style change detection. *PAN at CLEF 2018*.