An Unorthodox Approach for Style Change Detection

Notebook for PAN at CLEF 2022

Lukas Graner, Paul Ranly

Fraunhofer Institute for Secure Information Technology SIT, Rheinstraße 75, 64295 Darmstadt Germany
E-mail: {firstname}.{lastname}@sit.fraunhofer.de

Both authors have contributed equally

Abstract

The PAN shared tasks include tasks from a variety of disciplines, including authorship analysis, and are held annually. Participants are able to compete with each other by proposing approaches for a task, which are then compared and evaluated on a test dataset with predefined performance metrics. So far, the test datasets have traditionally been withheld, so that participants may only train and optimize approaches on the training and validation sets. In this year's Style Change Detection task, the objective of which is to locate author changes in multi-author text documents, PAN has also published the test set built from publicly available Q&A platform posts, albeit without ground truth labels. In this paper, we show that the ground truth of the test set can be recovered almost entirely by querying search engines with paragraph excerpts from the test set, crawling the query results and parsing author information of corresponding posts. We point out that this allows others to secretly tailor their approaches to the recovered test labels and thus gain an unfair advantage. Furthermore, as part of an in-depth data analysis, we address a variety of issues and finally suggest improvements for future Style Change Detection tasks.

Keywords

Style Change Detection, Multi Author Detection, Half-blind Task, Crawling, Recovering Test Labels

1. Introduction

The PAN Shared Tasks were introduced over a decade ago and span a variety of topics such as authorship analysis, computational ethics and plagiarism detection. One of these tasks is Style Change Detection (SCD) as first introduced in 2017, where the goal is to find changes of writing style inside a given document [1]. This task may help detecting documents or locate sections inside documents written by different authors and possibly provide evidence in those documents, among other use cases.

CLEF 2022 – Conference and Labs of the Evaluation Forum, September 5–9, Bologna, Italy

CEUR Workshop Proceedings (CEUR-WS.org)

In practice, the PAN Shared Tasks provide different datasets, whose content is taken from publicly available websites, such as the Stack Exchange network[1]. The tasks themselves comprise datasets to design and train models on, as well as independent test sets for the purpose of evaluating these models. We want to emphasize, that in the context of a shared task, it is important to withhold the labeled ground-truth, based on which the solutions of all participants are ranked. Otherwise submissions could simply output the correct results or be optimized on them, defeating the very point of a shared task.

In this year's SCD task [2], the test set was published along with the training and validation data. Although the data from the test set was unlabeled, rendering the task "half-blind"[3], this work shows how the labels can be recovered by crawling the websites to find corresponding authors to each given problem. With this information a *found truth* can be extracted, which differs only marginally from the ground truth.

A more thorough explanation of the datasets and their composition will be given in Section 2 along with background on how this year's task setup can be taken advantage of. In Section 3 the methodology and details of our approach are described while its results are presented in Section 4 as well as an in-depth analysis of the provided datasets. Lastly, Section 5 summarizes our findings and suggests possible improvements for future tasks.

2. Background

The shared PAN 2022 SCD task poses three sub-tasks to solve, namely *basic*, *advanced*, and *real-world* representing different scenarios. The specific tasks are as taken from the PAN website[2]:

1. Style Change Basic ($\mathcal{T}_1$): 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 ($\mathcal{T}_2$): 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 ($\mathcal{T}_3$): 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.

For each of the sub-tasks a separate dataset is provided ($\mathcal{D}_1$, $\mathcal{D}_2$ and $\mathcal{D}_3$, respectively), each being split disjointed into 70% labeled training ($\mathcal{D}_*^{\text{train}}$), 15% validation ($\mathcal{D}_*^{\text{valid}}$) and 15% unlabeled test data ($\mathcal{D}_*^{\text{test}}$). Their intended uses are for training a model, optional (hyperparameter) tuning and final evaluation of the model, respectively. The datasets contain a large number of SCD *problems*, where each problem is structured as a list of text *paragraphs* (statistics are shown in Table 2). For simplicity, we will also refer to sentences in $\mathcal{T}_3$ as paragraphs henceforth. Two

[1] Stack Exchange: https://stackexchange.com/
[2] PAN Style Change Detection Website: https://pan.webis.de/clef22/pan22-web/style-change-detection

consecutive paragraphs may be written by different authors, signifying a *style change*. For $\mathcal{T}_1$ and $\mathcal{T}_3$ the task is to predict for each pair of consecutive paragraphs, if a style change is present. For $\mathcal{T}_2$ each paragraph has, furthermore, to be assigned uniquely to an author.

As stated in the task description all problems in the datasets were assembled from questions and answers from the Q&A platform Stack Exchange, which comprises a variety of websites dedicated to diverse topics[3].

3. Methodology

Since all Stack Exchange websites are publicly accessible and indexed by common search engines, a paragraph can be traced back to its originating post. In the following we describe our straightforward approach for finding the source threads in the Stack Exchange network and assigning an author to each paragraph. This approach can be applied to all sub-tasks equally, followed by converting the author names into the output format of the respective sub-task. A corresponding pseudocode is given in Algorithm 1.

3.1. Finding source threads

Upon manual inspection of the data, we discovered that for the majority of the problems in the datasets, all their respective paragraphs originated from a single Stack Exchange thread. Tracing back each paragraph independently would, therefore, lead to unnecessary search queries yielding redundant results. Instead, we decided to cache the post contents of found threads, so that, while iterating over the paragraphs, we only query a search engine when a paragraph is not already present in the cache (cf. Algorithm 1 Line 5).

Furthermore, we discovered that a small portion of the original posts had been edited since the creation of the datasets. An exact phrase search for an entire paragraph, whose originating post has already been edited, would potentially yield no results[4]. To overcome this issue, we only use a substring of the paragraph, specifically the first tokens that add up to 150 (or 100 and further 75 if no results are yielded, cf. Algorithm 1 Line 6) characters.

Among a number of popular search engines we considered, we chose AOL Search[5] and Yahoo Search[6], as we empirically found them best suited for our use case in terms of accuracy and throughput. For each search query (cf. Algorithm 1, function `querySearchEngine`) we randomly sample one engine out of the two and query it with the search string enclosed by quotation marks »"«, indicating an exact phrase search.

The first resulting URL that is part of the Stack Exchange network is then requested, its HTML content parsed, and finally the names of the post authors along with the corresponding text

[3] Total list of Stack Exchange websites: https://data.stackexchange.com/
[4] Whether old versions of edited posts are indexed depends on the given search engine, although we found that this was rarely the case.
[5] AOL Search: https://search.aol.com/
[6] Yahoo Search: https://search.yahoo.com/

```
# Finding source threads
cachedPosts = []
for problem in 𝒫 :
    for paragraph in problem.paragraphs:
        if not any(post.content.contains(paragraph) for post in cachedPosts):
            for subLength in [150, 100, 75]:
                queryResult = querySearchEngine(paragraph[:subLength])
                if queryResult is not None:
                    for post in extractPostsFromStackExchangeURL(queryResult):
                        for editedPost in post.editedVersions:
                            cachedPosts.append(editedPost)
                        cachedPosts.append(post)

# Assigning authors
for problem in 𝒫 :
    paragraphAuthors = []
    for i, paragraph in enumerate(problem.paragraphs):
        matchingPost = None
        for post in cachedPosts:
            if post.contains(paragraph):
                matchingPost = post
        if matchingPost is not None:
            while len(paragraphAuthors) <= i:
                paragraphAuthors.append(matchingPost.author)
            cachedPosts.remove(matchingPost)
        elif len(paragraphAuthors) > 0:
            paragraphAuthors.append(paragraphAuthors[-1])
    outputSolution(problem, paragraphAuthors)
```

Algorithm 1: Python inspired pseudocode to search for paragraphs, fill a cache holding all found posts and finally assigning authors to each paragraph. Hereby, $\mathcal{P}$ denotes the set of problems in a given dataset, `querySearchEngine` is a function which queries a search engine with the given text and returns the first URL of a Stack Exchange result or `None` if none was found. The function `extractPostsFromStackExchangeURL` requests the HTML content, parses it and returns all posts of the given Stack Exchange URL, where a returned post holds information about the content, author name and previous edited versions of the post. `outputSolution` then finally converts the given list of author names to the according output format of the sub-task and outputs the result.

contents extracted, including older versions if edited (cf. Algorithm 1, function `extractPosts-FromStackExchangeURL`).

3.2. Assigning authors

Having populated the post cache, authors can now be assigned to each paragraph. This would be a trivial task if for each paragraph there exists one and only one source post. However, there are three issues we need to address:

1. Some paragraphs can not be found by any means. We suspect that the source post has since been deleted, for example due to being flagged as spam or rudeness. An example is

the last paragraph in problem 12 of $\mathcal{D}_2^{\text{test}}$:

```
"this program holds the left button down - Mouse Emulator [...]"
```

To one such paragraph we assign the found author of the previous paragraph in the corresponding problem, or next paragraph, if said paragraph is the first one in the problem (cf. Algorithm 1 Line 22-27).

2. A few paragraphs were present more than once in the cache but with varying authors. These different authors either copied content from an external source (such as documentation), one author adopted parts of another author's post, or multiple authors authored the same text purely by coincidence. Although our crawling algorithm would skip the search for paragraphs already cached, it may still crawl said posts as a *by-product* in an unrelated thread. An example is the 13th paragraph in problem 987 in $\mathcal{D}_1^{\text{train}}$:

   ```
   "Raspbian has 100MB of swap by default.  You should change it to
   2000MB in the configuration file.  So you will have to find this
   line:"
   ```

   which can be found both in the Raspberry Pi Stack Exchange as well as Stack Overflow in Spanish[7]. In such a case, it is not possible to unambiguously specify which post is the original one, so we simply assign the author of the lastly found matching post (cf. Algorithm 1 Line 19-21 and 24).
3. Although all paragraphs in the datasets represent disjoint posts, there are still paragraphs present multiple times, even within a single problem. This reflects a more intricate situation similar to the previous issue, as these paragraphs could not be considered a by-product of the crawling. Such cases occur, for instance, when an answer responds to a question while quoting the question. An example is problem 879 in $\mathcal{D}_3^{\text{test}}$, whose third and ninth paragraphs:

   ```
   "What filesystem layout will give the best performance?"
   ```

   are equal while their authors are different[8]. We also assign these paragraphs to the authors of the lastly found matching post, although without considering a post multiple times (cf. Algorithm 1 Line 19-21 and 24-25).

Once authors have been assigned to all sections, it is straightforward to output the solutions. For $\mathcal{T}_1$ and $\mathcal{T}_3$, only binary outputs representing the style changes are required, which can be easily generated by checking for inequality of the author names of each consecutive paragraphs. For $\mathcal{T}_2$, each author name will be transformed to an identifier number in order of occurrence starting at 1.

[7] $\mathcal{D}_1^{\text{train}}$ problem 987 paragraph 13 found both in https://raspberrypi.stackexchange.com/questions/44208 and https://es.stackoverflow.com/questions/262860

[8] $\mathcal{D}_3^{\text{test}}$ problem 879 paragraph 3 and 9 found both in https://serverfault.com/questions/243920

4. Experiments and Analysis

In the following, we present our results on the three tasks presented in Section 2. Further, we will go through an analysis of the provided datasets and some challenges a SCD model might face on them.

		$\mathcal{T}_1$	$\mathcal{T}_2$	$\mathcal{T}_3$
$\mathcal{D}_*^{\text{train}}$	Macro-F1	0.970	0.987	0.984
	DER	-	0.006	-
	JER	-	0.008	-
$\mathcal{D}_*^{\text{valid}}$	Macro-F1	0.958	0.990	0.985
	DER	-	0.006	-
	JER	-	0.008	-
$\mathcal{D}_*^{\text{test}}$	Macro-F1	0.993	0.986	0.993
	DER	-	0.004	-
	JER	-	0.004	-

Table 1
Final results of our submission on the respective train, validation and test sets of the three sub-tasks.

4.1. Results

Solutions of all tasks $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$ are evaluated using macro-averaged F1 (Macro-F1) as well as for $\mathcal{T}_2$ additionally with Diarization Error Rate (DER) [4] and Jaccard Error Rate (JER) [5].

Table 1 shows the results of our approach for the test set evaluated by the TIRA research architecture [3] and for the training and validation sets, which we evaluated based on the provided ground truth labels. As can be observed, the results are consistently near-perfect across all sub-tasks and datasets. The issues stated in Section 3.2, such as presumably deleted source posts, prevent the approach to achieve actual perfect results (i.e., values of 1.0 for Macro-F1 and 0.0 for DER and JER).

We would like to emphasize that we do not consider our approach as an appropriate submission for the PAN shared tasks, since it effectively resembles an act of cheating. The purpose of this approach and paper is not to compete against other participants, but to show that it is possible (cf. Table 1) for any participant to uncover the true labels of the test sets with little effort. Participants who do so may then (in addition to training a model on the training set) optimize or manually finetune their model on test labels without disclosing this action, gaining an unfair advantage over other participants, who do not. Moreover, this harms the research community by potentially incorrectly suggesting that some approaches are preferable to others. At this point the final evaluation result is then skewed and, thus, of no use.

4.2. Data

For each shared task, the type of the provided data affects the results, as different source data can vary in complexity, size and significance among other factors. In this section, we will provide a quantitative and qualitative insight into the datasets of the PAN 2022 SCD task.

4.2.1. Quantitative Analysis

We formulated different categories of potential challenges for SCD approaches (and for style analysis in general), representing phrases and character strings that complicate the author inference or add noise to the data. The categories are *URLs*, *file paths*, *citations*, *code* (such as HTML, SQL, shell or LaTeX-commands) and *non-English text*. Furthermore, we inspected individual paragraphs regarding formatting, punctuation quantity as well as the presence of short paragraphs of less than 50 characters and duplicates.

		Dataset							
	Category	$\mathcal{D}_1$		$\mathcal{D}_2$		$\mathcal{D}_3$		Total	
Problems	*Total*	*2,000*	*(100%)*	*10,000*	*(100%)*	*10,000*	*(100%)*	*22,000*	*(100%)*
	URL	336	(16.8%)	1,434	(14.3%)	2,127	(21.3%)	3,897	(17.7%)
	File Path	33	(1.7%)	157	(1.6%)	151	(1.5%)	341	(1.6%)
	Citation	183	(9.1%)	762	(7.6%)	1,847	(18.5%)	2,792	(12.7%)
	Code	27	(1.4%)	57	(0.6%)	126	(1.3%)	210	(1.0%)
	Non-English	49	(2.5%)	54	(0.5%)	401	(4.0%)	504	(2.3%)
Paragraphs	*Total*	*15,862*	*(100%)*	*75,153*	*(100%)*	*159,652*	*(100%)*	*250,667*	*(100%)*
	URL	450	(2.8%)	1,770	(2.4%)	3,016	(1.9%)	5,236	(2.1%)
	File Path	40	(0.3%)	197	(0.3%)	214	(0.1%)	451	(0.2%)
	Citation	315	(2.0%)	963	(1.3%)	2,554	(1.6%)	3,832	(1.5%)
	Code	273	(1.7%)	160	(0.2%)	617	(0.4%)	1,050	(0.4%)
	Non-English	281	(1.8%)	263	(0.3%)	1,667	(1.0%)	2,211	(0.9%)

Table 2
Absolute number of occurrences (and percentage with respect to the total) of different phrases present in entire problems (upper block) and paragraphs (lower block).

As shown in Table 2, the different datasets contain varying amounts of phrases fitting in one of the categories. The absolute counts for $\mathcal{D}_1$ are smaller as this dataset consists of only one fifth of the problems in $\mathcal{D}_2$ or $\mathcal{D}_3$. We used a simple pattern search for URLs[9] and citations[10] and found a surprisingly high amount of each present in the problems. While URLs do not contribute to an authors style at all, citations may in fact represent the style of another entity. With up to 21% of problems and 2% of all paragraphs affected by at least one of these phrases,

[9] Regex pattern to detect URLs:
`https?://(www\.)?[-a-zA-Z0-9@:%._+~#=]+\.[a-zA-Z0-9()]{1,6}\b([-a-zA-Z0-9()@:%_+.~#?&/=]*)`

[10] Regex pattern to detect citations (simplified): `(\x22.*?\x22|(?<=^| )\x27.*?\x27(?=[.,;:\s)]))`
Note, that this is only a naive approach, since not all texts enclosed by quotation marks indicate a citation. Furthermore, we only counted a citation if it covered more than 25% of the paragraph.

all three SCD tasks become more demanding. Moreover, we observed paragraphs that contain multiple URLs and citations or even solely consist of those.

The proportion of code listings present in the data was more difficult to determine. Firstly we also inlcuded HTML, SQL, LaTeX and shell commands. Secondly, we focused on formatted listings, i.e. code snippets that were in the Stack Exchange format of four leading whitespaces followed by the actual code. For this work we took neither code in plain text nor code snippets encapsulated by text phrases into account, even though we occasionally found similar examples in the data files.

In order to detect non-English texts in paragraphs we used the language detection tool *langdetect*[11]. It should be noted here, that a small portion of nonhuman phrases (e.g. code listings and command line outputs) were also considered non-English, as they were assigned to languages such as Swedish. This results in most of the detected non-English phrases being such nonhuman code with only very few phrases originating from other languages. Nevertheless, this proposes another difficulty, as participating models are forced to compare the writing style of texts in different languages, not to mention analyzing nonhuman texts, such as URLs, file paths, logged metadata or error reports. A more in-depth look on those will be given in the qualitative analysis in the next section.

In general, as all datasets are taken from the same network, we assume that the allocation of problems to datasets is somewhat arbitrary. Still it is worth mentioning that $\mathcal{D}_3$ contains more problems with at least one URL, citation, code and non detectable English sentence compared to $\mathcal{D}_1$ and $\mathcal{D}_2$, while the problems in $\mathcal{D}_2$ and $\mathcal{D}_3$ are of similar average size. As the paragraphs in $\mathcal{D}_3$ are single sentences, they are much shorter (on average 114 characters per paragraph, in contrast to 199 for $\mathcal{D}_1$ and 249 for $\mathcal{D}_2$), hence problems in $\mathcal{D}_3$ contain more paragraphs on average. Table 2 shows that the quantity of challenging phrases in paragraphs are similar in all datasets. Considering this and the short length of paragraphs in $\mathcal{D}_3$, $\mathcal{T}_3$ is thus more affected by the phrase induced noise and consequently even more challenging than $\mathcal{T}_1$ and $\mathcal{T}_2$.

4.2.2. Qualitative Analysis

In this section we will discuss specifically selected excerpts of SCD problems, which are listed and categorized in Table 3 along their alleged source URLs. As the datasets provided by the shared task are taken from the publicly available Stack Exchange network, they are subject to their individual authors. This leads to not only different writing styles of the network's members but also different formatting and best practices. The first two examples show duplicates occurring in single problems. While in Example 1 the paragraph is present both in the question and a response by another author, the excerpt found in Example 2 is a repetition of source code from a single author. For both examples, along with direct citations as in Example 3 and automated console outputs as in Example 4, a correct SCD prediction is intractable. In total, we found 92 examples of duplicated texts over all datasets and on average more than one in eight

[11] https://pypi.org/project/langdetect/

#	Problem	Category	Excerpt and Source Stack Exchange URL

1	$\mathcal{D}_3^{\text{test}}$-879	duplicate (different authors)	`What filesystem layout will give the best performance? [...]`
			`What filesystem layout will give the best performance?`
			Source: https://serverfault.com/questions/243920

2	$\mathcal{D}_3^{\text{test}}$-283	duplicate (same author)	`def advance_to_first_non_white_space_on_line(view, pt):` `def advance_to_first_non_white_space_on_line(view, pt):` Source: https://superuser.com/questions/9941833
3	$\mathcal{D}_1^{\text{train}}$-1111	citation	`“Flame was a failure for the antivirus industry. We really should have been able to do better. But we didn’t. We were out of our league, in our own game.”` Source: https://superuser.com/questions/1356507

4	$\mathcal{D}_3^{\text{test}}$-1469	command line output	`Setting up mysql-client-5.1 (5.1.37-1ubuntu5.1) ... [...]`
			`Selecting previously deselected package mysql-server-5.1.`
			Source: https://serverfault.com/questions/114974

5	$\mathcal{D}_1^{\text{test}}$-109	german text	`Die folgenden teilweise installierten Pakete werden konfiguriert:` Source: https://serverfault.com/questions/265372

6	$\mathcal{D}_1^{\text{train}}$-585	source code	`tempArray(2) = sourceSheet.Range("RF_INV_Axial").Value2`
			`tempArray(3) = sourceSheet.Range("RF_INV_Major").Value2 [...]`
			Source: https://serverfault.com/questions/265372

7	$\mathcal{D}_2^{\text{train}}$-1692	URL only	`https://physics.stackexchange.com/questions/80043/how-fast-does-light-travel-through-a-fibre-optic-cable` Source: https://networkengineering.stackexchange.com/questions/16438

8	$\mathcal{D}_3^{\text{train}}$-1207	HTML	`<div class="cont b-1">`
			`<button class="padded" data-type="number">0</button>`
			Source: https://codereview.stackexchange.com/questions/100643

9	$\mathcal{D}_1^{\text{train}}$-1292	LaTeX	`$$ \frac{\Gamma \vdash t : T}{\Gamma, x:A \vdash t : T} \;\mathtt{W}$$` Source: https://cstheory.stackexchange.com/questions/41505

10	$\mathcal{D}_3^{\text{test}}$-1229	repeated sequence	`[Nul][Nul][Nul][Nul][Nul][Nul][Nul][Nul][Nul][Del][Del][Del]`
			`[Blank][Blank][Blank][Blank][Blank][Blank][Blank][Blank][...]`
			Source: https://superuser.com/questions/369231

11	$\mathcal{D}_1^{\text{test}}$-16	table format	`a b ^a ab a+b ^a(a+b) ^(ab) a⊼b [...]`
			`False False True False False False True True [...]`
			Source: https://codereview.stackexchange.com/questions/112446

12	$\mathcal{D}_2^{\text{train}}$-5790	table format	`+----------+------------------------------+------+--------+-[...]` `| 1576128	database_name/#sql-4593_1e9	1	118	[...]` Source: https://dba.stackexchange.com/questions/248723

Table 3
Example excerpts from problems on which challenges might arise for SCD approaches. For each problem, indicated as $\mathcal{D}$-*problem number*, the alleged source URL, as well as a categorization for the underlying issue are given.

problems contained at least one citation[12].

Example 5 shows a rare case of non-English text, namely an error report in German. While this is one of few problems where we found text in non-English (even though it is machine

[12] As we used a pattern search to filter the citations, paragraphs such as Example 8 in Table 3 might also be included in this category. Though this does not represent a citation in the proper sense, we decided to include these kinds of problems as they display further challenges to SCD.

generated text and thus not truly authored by the posts author), it becomes increasingly more difficult as the German paragraphs are shuffled within the English answers, creating a bilingual incoherent problem.

Examples 6 to 12 depict different types of nonhuman texts that could pose challenges for reliable SCD. Represented in Example 6 is source code in plain text. Interestingly, the code listings kept their order while the problems were shuffled and the typical Stack Exchange formatting (multiples of four leading spaces indicating a code listing) remained the same. It could be argued that coding style is another form of writing style and thus should remain in the data. However, as it is written plainly in between paragraphs and even being shuffled along normal texts, a significant gain in information is unlikely. Example 7 shows a whole paragraph consisting of a single URL, which effectively represents pure noise, as it does not contain any style information and can be posted by any author. Among all datasets, we found $1,469$ problems where at least one paragraph exclusively consisted of an URL.

The text excerpt in Example 8 shows a short paragraph of only 38 characters (50 with leading spaces) containing HTML. In Example 9 multiple LaTeX math commands are present.
The task of finding all code listings in the data is less trivial, hence we recognize the difficulty of removing all code from the problem files. Still, with all the examples found as shown in Table 2, some data cleaning would be advisable in order to properly execute SCD. Regarding the short paragraphs, in total we found 235 paragraphs with less than 50 characters excluding leading white spaces. Short paragraphs further increase the difficulty, as they offer less content to detect style on. Moreover, we found that most of the short paragraphs could be omitted from the data as they contained source code, machine outputs or generated logs.

Lastly, in our analysis we searched for repeated sequences and table-style-formatted content. The repetition of words does not pose a challenge for SCD in general, but some examples as the one shown in Example 10 can be effectively classified as noise. Similarly, the table formatted content shown in Example 11 does not provide any information regarding writing style. Without any notion that the underlying concept of the paragraphs is a table formatted with spaces, it could be misinterpreted as a generic sequence of words without any deeper sense to it. Furthermore, we want to note that each table row is a seperate paragraph and shuffled in the problem. Finally, Example 12 shows an SQL table, where table borders are still present and even shuffled between text paragraphs in the problem. Hence, some paragraphs have no individual style, consisting solely of punctuation. In total we discovered $1,180$ paragraphs that consist of more than 20% punctuation. A human author is unlikely to exceed this threshold, especially considering statistics about average usage of punctuation [6]. Most of the examples found can be assigned to one of the previously described categories, as e.g. logs and LaTeX formulas contain a comparatively high amount of punctuation characters. However, other results like the one shown in Example 12 indicate that there are also paragraphs present not covered by any other category.

As shown in the analysis, the datasets proposed for the 2022 PAN SCD shared task show various degrees of noisy data, ranging from minor formatting styles not necessarily attributable to specific authors, over different languages including programming languages up to machine outputs absent of any writing style or even citations that in fact resemble style of other authors.

4.3. Task Definition

We would like to point out, that in this years shared task (contrary to previous competitions) the definition and problem structure of the SCD sub-tasks $\mathcal{T}_1$ and $\mathcal{T}_3$ (and $\mathcal{T}_2$ only partly, since it further requires assigning author identifiers) essentially resemble an Authorship Verification (AV) Task: The set of viable locations where a style change may occur is given, since the problems are structured as separated paragraphs. Furthermore, we found, that these paragraphs have presumably been randomly shuffled, which makes their (semantic) order meaningless. Thus, paragraphs do not depend on preceding ones in any way different than on subsequent ones, as it would be the case in continuous text. Each pair of consecutive paragraphs is, therefore, independent of each other (apart from the fact that they might originate from the same Stack Exchange thread and that for $\mathcal{T}_1$, there can only be one style change inside one problem). The objective of the tasks is to detect if a style change between two paragraphs of such a pair occurs (or not), which then is equal to classifying if the two paragraphs were written by different authors (or not), which again is the objective of AV. Further, all outputs of an SCD approach over all dataset problems are flattened to a list and evaluated against the ground truth, which weighs them all equally. Consequently, without making any intrinsic change to the data, the current $\mathcal{T}_1$ and $\mathcal{T}_3$ can be reformulated to AV tasks, not only from a technical standpoint (regarding the approach or algorithm[13]), but also from a semantic one (regarding the structure and meaning of the used texts). This raises the question of the role of SCD in this year's competition, especially when considering that AV is a dedicated shared task.

5. Conclusion

In this paper we describe a simple method to efficiently recover author labels for publicly available Stack Exchange posts. Based on this and the preliminary availability of the test set, albeit without labels, our submission achieves almost perfect test scores for this year's PAN Style Change Detection task, without performing actual style analysis. Naturally, this is not in the spirit of the competition as such. Nevertheless, this work shows that any participant can recover the test labels and secretly tune their approach based on these, gaining an unfair advantage and skewing research results. Additionally, we present an in-depth analysis of the available data and some issues we see with its compilation, such as individual data instances to be classified that do not contain any stylistic information.

To address the stated issues for the future, we firstly propose to adapt the submission model used in previous years, i.e., no test data on which submissions will be scored should be published before the submission deadline. This prevents all participants from obtaining advantageous information about the data that can potentially be abused. Second, to establish a genuine SCD task that sufficiently separates itself from the other PAN shared task *Authorship Verification*, we advice that text units (i.e. paragraphs or sentences in this years SCD task) are not arbitrarily shuffled inside problems and instead preserve the *semantic flow* of the originating text. Staying

[13] On a technical level, almost all authorship analysis tasks can interchangeably be transformed into each other, whereby authorship verification may arguably be considered as the core branch or a "fundamental problem" [7].

with Stack Exchange: Posts that were partly edited by other authors could, for example, provide an appropriate data foundation, as precise character positions of author information are accessible. Additionally, more flexible style change positions would enhance the focus on the style change itself. In the 2017 PAN SCD task, the task was to find borders in a document where a change in style can be noticed. This emphasis on varied style change possibilities could help shaping more realistic scenarios, towards which $\mathcal{T}_3$ is already pointing.

Addressing issues in the given data, simple preprocessing can improve its quality in respect to a style analysis task. Although textual content of the Stack Exchange network is generally appropriate, one needs to keep in mind that expert communities not only use different writing styles, but also different topic- and domain-related content, such as code listings, documentation or citations. This is particularly critical if text units exclusively consist of such content. A first step to tackle said issues, is to remove content of code cells in Stack Exchange posts and skip any thread comprising clear citations of other authors or websites. For future SCD tasks this will significantly lessen the probability of falsely attributing mixed authorships and the amount of noise that is unrelated to writing style.

References

[1] J. Bevendorff, B. Chulvi, E. Fersini, A. Heini, M. Kestemont, K. Kredens, M. Mayerl, R. Ortega-Bueno, P. Pezik, M. Potthast, F. Rangel, P. Rosso, E. Stamatatos, B. Stein, M. Wiegmann, M. Wolska, E. Zangerle, Overview of PAN 2022: Authorship Verification, Profiling Irony and Stereotype Spreaders, and Style Change Detection, in: A. Barron-Cedeno, G. D. S. Martino, M. D. Esposti, F. Sebastiani, C. Macdonald, G. Pasi, A. Hanbury, M. Potthast, G. Faggioli, N. Ferro (Eds.), Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Thirteenth International Conference of the CLEF Association (CLEF 2022), volume 13390 of *Lecture Notes in Computer Science*, Springer, 2022.

[2] E. Zangerle, M. Mayerl, M. Potthast, B. Stein, Overview of the Style Change Detection Task at PAN 2022, in: CLEF 2022 Labs and Workshops, Notebook Papers, CEUR Workshop Proceedings, 2022.

[3] 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.

[4] The rich transcription spring 2003 (rt-03s) evaluation plan, http://www.itl.nist.gov/iad/mig/tests/rt/2003-spring/docs/rt03-spring-eval-plan-v4.pdf/, 2003.

[5] N. Ryant, K. Church, C. Cieri, A. Cristia, J. Du, S. Ganapathy, M. Liberman, The second dihard diarization challenge: Dataset, task, and baselines, arXiv preprint arXiv:1906.07839 (2019).

[6] V. Cook, Standard punctuation and the punctuation of the street, in: Essential Topics in Applied Linguistics and Multilingualism, Springer, 2014, pp. 267–290.

[7] M. Koppel, J. Schler, S. Argamon, Y. Winter, The “fundamental problem” of authorship attribution, English Studies 93 (2012) 284–291.