Style Change Detection by Threshold Based and Window Merge Clustering Methods

Sukanya Nath[1][0000−0003−4210−9090]

University of Neuchâtel, Rue Emile-Argand 11, 2000 Neuchâtel, Switzerland
sukanya.nath@unine.ch

Abstract. The goal of the Style Change Detection task is to detect the stylistic changes in a document and exploit them to determine the number of authors. Our approach is to segment the given document into chunks of text, called windows. Each window is converted into feature vectors of words (around 50 features). The mutual distances among the window feature vectors are measured (using different distance measures like Matusita, Tanimoto etc.) and fed to two clustering algorithms called Threshold Based (TBC) and Window Merge (WMC). The number of clusters yielded correspond to the number of authors. The results of the two algorithms are then compared against each other and a combined minimum model.

Keywords: Authorship · Multiple Authorship · Style Change Detection · Clustering.

1 Introduction

Multiple authorship detection has long been a fascinating subject of research in the literary community. Over time, the applications of multiple authorship have extended into areas like plagiarism detection, forensics and more recently fake news detection. It has been shown by multiple researchers ([2],[15]), how word patterns can discern an exclusive personal style. Presence of multiple consistent personal styles indicate presence of multiple authors.

As part of the CLEF Style Change Detection problem, the model was tasked to identify how many authors had written a given document or a stream of text. More details can be found in [17].The dataset was composed of forum posts from different 'StackExchange' network sites. All documents were written in English. The topics of discussion varied greatly. As the documents were based on forums, each document was independent of others. Therefore, there were no separate training sets for each author. In the previous years, viz. 2018 and 2017, the style change detection problem at CLEF was limited to detecting the presence or absence of style changes. It is to be noted that the Style Change Detection

Table 1. Duplicates in Documents

	Documents	
Authors	with no duplicated sentences	Total
1	1142	1273
2	48	325
3	65	313
4	83	328
5	66	307

is a variation of the Authorship Diarization problem introduced in 2016, which aimed to cluster the contributions of each individual author of a document, when the number of authors is either known or unknown.

The rest of this paper is organized as follows. Section 2 describes the dataset. In Section 3 we describe feature extraction, distance measurement and clustering algorithm. Section 4 shows some of our evaluation results. A conclusion in Section 5 draws the main findings of our experiments.

2 Corpus

In this section, we perform some preliminary statistics on the data to gain a comprehensive understanding of the data. A basic statistical overview of the PAN dataset is provided in Section 2.1 while the effect of duplication is highlighted in Section 2.2.

2.1 Overall Statistics

The training dataset consists of 2546 documents while the validation set consists of 1272 documents. The number of authors ranges from 1 to 5. Each document is composed of responses to a single thread at a forum. The mean number of tokens was 1570 (median: 1452, sd: 810, min: 339, max: 7042). The punctuations were preserved and treated as separate tokens.

2.2 Duplicate Sentences

It was observed that certain documents contain duplicate sentences. In Figure 1, we show a boxplot of duplicated sentences per document, grouped by the number of authors. It can be noted that for documents with a single author, the number of duplicates is rather low.

On the other hand, the documents written by multiple authors have varying number of duplicate sentences. In Table 1, we compare the number of documents containing no duplicate sentences across different author classes. Since, the number of duplicate sentences is low for single authors and relatively high for multiple authors, therefore, the number of duplicate sentence may be used as a feature in identifying single author documents.

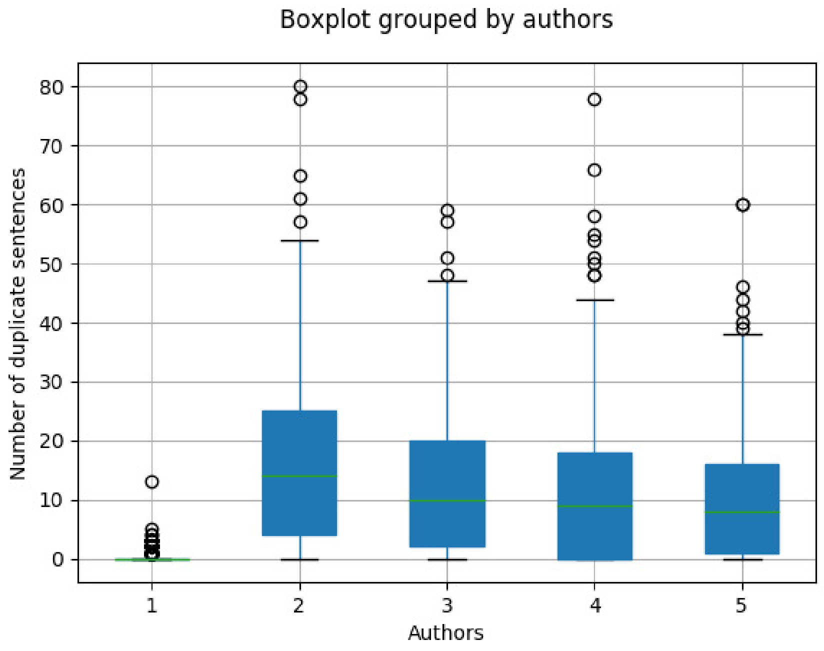

Fig. 1. Number of duplicates grouped by Authors

3 Method

This section details the implementation details of our method. We define our feature selection methods, distance measures and clustering algorithm.

3.1 Feature Selection and Distance Measures

To preserve the logical document structure, we break down the document into paragraphs. A window was initially assumed to be equal to a paragraph of text. However, such a simple approach created highly variable lengths of windows. Thus, the window tokenizer was modified to merge extremely short paragraphs (less than 200 chars) with the previous paragraphs. Similarly, large paragraphs were split into smaller windows to get reasonably length wise balanced windows.

Thereafter, top 50 Most Frequent Tokens (MFT) were selected from each document (and not the whole corpus). Each window was converted to a feature vector based on the normalized frequency of the selected tokens. We used the Matusita [6] distance measure to compare the distances between the window feature vectors. Say, a document has 4 windows, then a symmetric 4x4 matrix will be created representing the distance between the windows, as shown in Table 2.

Table 2. Distance Matrix

	W_0	W_1	W_2	W_3
W_0	0	0.8	0.2	0.55
W_1	0.8	0	0.56	0.42
W_2	0.2	0.56	0	0.77
W_3	0.55	0.42	0.77	0

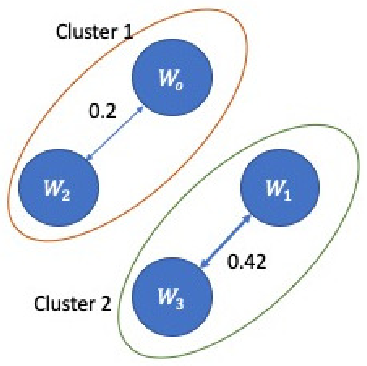

Fig. 2. Clusters representing distance between windows

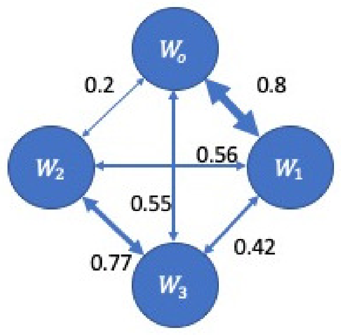

Fig. 3. Distance Representation between windows

Figure 3 represents the windows as the nodes and the distances among them as the edges of a graph. Figure 2 shows the corresponding clusters created from the distance matrix. The two clusters are created based on the smallest distances between windows

3.2 Algorithm

Our solution used a combination of two algorithms. We introduce a few definitions before introducing the algorithms.

Definitions

- **Distance Matrix**: A distance matrix represents the distances among all the windows in a square matrix. The diagonal elements are zero because they represent the distances between a window and itself. Table 2 shows a distance matrix representing the relationships in Figure 3. For the purpose of this paper, we will only deal with symmetric distance matrices.
- **Sorted Distance Array**: Such an array contains non-duplicated entries of the distance matrix, sorted in ascending order. In terms of a graph, an entry in the distance matrix corresponds to the edge distance d between nodes W_i and W_j. The top right half of the distance matrix (cells of Table 2 highlighted in gray) are extracted and converted to an array of the form (W_i,

W_j, d) where W_i, W_j are windows and d is the distance between them. An example of sorted distance array is shown in Table 3.

– **Cluster**: The most similar windows of text are grouped together into clusters. The average distance of a cluster is the sum total of all the distances between members divided by the number of edges. For example, in Figure 2, the average distance of cluster 1 is 0.2/1 or 0.2.
– **Cluster List**: Cluster list contains all the active clusters. Ideally, each cluster corresponds to an author implying that the author has written the text in the windows. Therefore, the number of clusters correspond with the number of authors. However, in some cases, this is not true. For example, it may so happen that an author has written only a few sentences which are contained by a single window. Such a window, if sufficiently distant from other clusters, is likely to be excluded by all clusters. As a result, there is a possibility that the number of authors calculated is lower than the ground truth. On the other hand, if the document is broken down into tiny windows, it is possible that there are too few characters, i.e not sufficient information. As a result, the distance measurement among such windows will be higher, resulting in the formation of a higher number of small clusters. Thus the predicted number of authors is higher than the ground truth.
– **Thresholds** : The following thresholds are required by the Threshold Based Clustering algorithm (TBC) to determine if a new member may be added to an existing cluster or if two clusters may be merged together. Thresholds ensure that existing clusters expand reasonably.
  • **Add Window Threshold**: A new member may be added to a cluster, if upon adding the new member, the revised average distance of the modified cluster is not greater than x% of the average distance of the existing cluster.
  • **Merge Cluster Threshold**: Two clusters may be merged together, if the revised average distance of the new cluster is not greater than x% of the average distance of either of the existing clusters.

  However, fixing a certain threshold as discussed above does not take into account the size of the existing cluster. It may be desirable that, clusters with only a few members or having a small average distance, expand more rapidly as compared to larger clusters. Therefore, size of the cluster and its average distance was used to penalise the respective threshold values of the clusters. Such an adaptation, helped to customize the threshold according to the state of the cluster.

Threshold Based Clustering Algorithm The intuition behind the TBC algorithm is that a good way to start clustering is to select the closest windows in terms of distance because such windows, are likely to belong to the same cluster. As such, the distance matrix is transformed into an array sorted by distance, viz. the sorted distance array or *dist_arr* and fed to the Algorithm 1. The idea is to select the windows with smallest distances iteratively such that, when forming a cluster, the closest members are included first. Thereafter, the members that

Algorithm 1 Threshold Based Clustering Algorithm

```
1: procedure CreateCluster($dist\_arr$)          # Convert sorted distance array to clusters
2:     $cluster\_list \leftarrow \phi$
3:     for all $entry\ e \in dist\_arr$ do          #  where $e$ is of the form ($W_i$, $W_j$, $d$)
4:         $W_i,\ W_j \leftarrow$ GetWindows($e$)
5:         if $W_i \notin c\, and\, W_j \notin c' \,\forall\, c, c' \in cluster\_list$ then          #  where $c$ is a cluster
6:             $c \leftarrow \{W_i, W_j\}$
7:             $cluster\_list \leftarrow cluster\_list \cup c$
8:         end if
9:         if $\exists\, c' : c' \in cluster\_list\, and\, W_i \in c'\ and\, W_j \notin c\, \forall c \in cluster\_list$ then
10:            $c'' \leftarrow c' \cup \{W_j\}$
11:            if $average\_distance(c'') < add\_node\_threshold$ then
12:                $c' \leftarrow c''$          # cluster $c$' is updated by adding a new Window $W_j$
13:            end if
14:        end if
15:        if $\exists\, c' : c' \in cluster\_list\, and\, W_j \in c'\ and\, W_i \notin c\, \forall c \in cluster\_list$ then
16:            $c'' \leftarrow c' \cup \{W_i\}$
17:            if $average\_distance(c'') < add\_window\_threshold$ then
18:                $c' \leftarrow c''$          # cluster $c$' is updated by adding a new Window $W_i$
19:            end if
20:        end if
21:        if $\exists\, c, c' : c, c' \in cluster\_list\, and\, W_i \in c\, and\, W_j \in c'$ then
22:            $c'' \leftarrow c \cup c'$
23:            if $average\_distance(c'') < merge\_cluster\_threshold$ then
24:                $cluster\_list \leftarrow cluster\_list \cup \{c''\} - \{c, c'\}$
25:                # Replace previous smaller clusters with a new merged cluster
26:            end if
27:        end if
28:    end for
29:    return $len(cluster\_list)$          # Number of clusters represent number of authors
30: end procedure
```

Table 3. Sorted Distance Array

W_i	W_j	Distance
W_0	W_2	0.2
W_1	W_3	0.42
W_0	W_3	0.55
W_1	W_2	0.56
W_2	W_3	0.77
W_0	W_1	0.8

are further away, are included. Hence the clusters would be expected to grow from small and dense to large and spread out. Such a growth is controlled by the thresholds shown in Section 3.2. For each entry in the *dist_arr*, the windows may have any one of the following conditions:

– None of the windows are present in any existing cluster of the cluster list. Therefore, a new cluster must be created with these two windows and added to the cluster list.
– One of the windows is already a member of an certain existing cluster, while the other isn't. Thus, the non-member window may be added to an existing cluster, provided that the Add Window Threshold condition is met. While adding a new member, all the corresponding edges between the new member and existing members are extracted from the *dist_arr* and added to the cluster.
– Both windows belong to two different existing clusters which may be merged if the conditions of the Merge Cluster Threshold are satisfied. As in the previous case, all the corresponding edges are added to the cluster

It is to be noted, that the condition that both windows belong to the same cluster is not possible, as otherwise, such an entry would already have been traversed while adding new members or cluster merging.

3.3 Window Merge Clustering

The Window Merge Clustering algorithm (WMC) iteratively combines the most similar windows to generate a new set of windows. From these new windows, the distance matrix is re-calculated for the next iteration. As a result, the clusters formed are hierarchical. The idea is to represent each cluster with a combined representation of all of its members together, rather than individual distances. Figure 4 shows the formation of the clusters by the Window Merge Clustering. The height of the cluster corresponds to the distance between the two clusters. In the first iteration, $Cluster_1$ (W_0, W_1) is formed and W_0 and W_1 are combined. The resulting new set of windows are W_0W_1, W_2 and W_3. In the next iteration $Cluster_2$ (W_2,W_3) is formed. The updated windows are W_0W_1 and W_2W_3. Eventually, these windows are connected in the third iteration resulting in $Cluster_3$ (W_0W_1, W_2W_3).

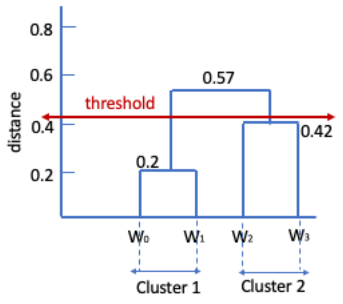

Fig. 4. Hierarchical Clusters created by WMC

To determine the number of clusters, we use a truncation threshold, which is a certain percentage of the height of the largest cluster. In this example shown in Figure 4, the truncation threshold is set to 0.43 or 75% of 0.57 which is the height of $Cluster_3$. With the help of the truncation threshold, we can control the size of the clusters. Also, as in the case of the TBC, the number of clusters relates to the number of authors.

Algorithm 2 Window Merge Algorithm

```
1: procedure CreateHierarchicalCluster($dist\_matrix, text\_windows$)
2:     $window\_merge\_order \leftarrow \phi$
3:     while $size(dist\_matrix) > 1$ do
4:         $W_i, W_j \leftarrow$ FindMinDistanceWindows($dist\_matrix$)
5:         $merge\_order \leftarrow window\_merge\_order \cup \{W_i, W_j\}$
6:         $new\_text\_windows \leftarrow$ MergeWindows($W_i, W_j, text\_windows$)
7:         $dist\_matrix \leftarrow$ CalculateDistMatrix($new\_text\_windows$)
8:     end while
9:     $authors \leftarrow$ Truncate($window\_merge\_order, truncation\_threshold$)
10:    return $authors$
11: end procedure
```

4 Evaluation

The Add Window and Merge Cluster thresholds of the TBC algorithm were varied from 5% to 100%. We observed the optimum performance at 50% for both the thresholds. In our experiments, both thresholds are set to 50%. We show the evaluation of our algorithms in Table 4. In addition to TBC and WMC, we have another model called the combined minimum. The combined minimum model simply selects the smaller predicted value from either of TBC and WMC algorithms. In our evaluation, we find that the the TBC performs better than the WMC and the combined minimum.

In Section 2.2, we proposed the idea of using the number of duplicates sentences to predict the number of authors. We showed how low number of duplicate sentences were indicative of a single author. We modified the algorithms TBC and WMC such that, if no duplicate sentences were present in a document, only a single author was predicted. However, if one or more duplicate sentences were observed, the algorithm was executed normally. In Table 5, the impact of using such a feature is shown. We observe that the performances of all the three models have improved significantly. In Table 5, the results of the test set evaluated on the TIRA for TBC algorithm are shown.

Table 4. Initial Results

	Training set			Validation set		
Algorithm Name	Accuracy	OCI	Rank	Accuracy	OCI	Rank
TBC	0.66	0.83	0.42	0.65	0.82	0.42
WMC	0.62	0.91	0.35	0.63	0.88	0.37
Combined Min	0.65	0.92	0.36	0.66	0.90	0.38

5 Conclusion

In this paper, we have demonstrated that it is possible to identify the number of authors from a relatively short piece of text without any training corpus per author. We have contributed two different algorithms to perform authorship clustering and now we present the chief findings of our work.

We found that, TBC perfoms the best out of the three models under both the scenarios. This is because by prioritizing the selection of the smallest distances, we ensured that the most vital members of a cluster are selected first. Such members are vital because they lay down the cluster structure. Subsequent members can only be included if they expand the cluster within a reasonable limit. We observe that WMC performs only slightly worse, when no information on duplicate sentences is present. This shows that there is scope of improvement in the combined representation approach of WMC.

Also, by including the duplicate sentences feature, the performances of the algorithms were improved by a considerable margin. Since, we have tested using only one test collection, there may be some unknown dataset characteristic

Table 5. Improved Results using Duplicated Sentences

	Training set			Validation set			Test Set (TIRA results)		
Algorithm Name	Accuracy	OCI	Rank	Accuracy	OCI	Rank	Accuracy	OCI	Rank
TBC	0.83	0.87	0.48	0.83	0.85	0.49	0.85	0.87	0.49
WBC	0.72	0.93	0.40	0.74	0.90	0.42	-	-	-
Combined Min	0.70	0.93	0.39	0.72	0.91	0.41	-	-	-

favouring one algorithm over another. As part of our future work, one or two additional datasets would be included for a better evaluation.

Another important part of the window based approaches is to tune the correct size of the window. Ideally the document should be split such that each window contains exactly one style change completely. However, is not possible to know the location of the style changes initially. Hence it is difficult to know the correct size of a window. A possible improvement of our work could be to define a measure for cluster quality achieved to be given as a feedback for further improving the splitting of the windows in an iterative fashion.

The github repository for our approach can be found at [8].

References

1. Burrows, J.: All the way through: testing for authorship in different frequency strata. Literary and Linguistic Computing **22**(1), 27–47 (2006)
2. Craig, H., Kinney, A.F.: Shakespeare, computers, and the mystery of authorship. Cambridge University Press (2009)
3. Crystal, D.: Language and the internet. Cambridge University Press (2009)
4. Daelemans, W., Kestemont, M., Manjavancas, E., Potthast, M., Rangel, F., Rosso, P., Specht, G., Stamatatos, E., Stein, B., Tschuggnall, M., Wiegmann, M., Zangerle, E.: Overview of PAN 2019: Author Profiling, Celebrity Profiling, Cross-domain Authorship Attribution and Style Change Detection. In: Crestani, F., Braschler, M., Savoy, J., Rauber, A., Müller, H., Losada, D., Heinatz, G., Cappellato, L., Ferro, N. (eds.) Proceedings of the Tenth International Conference of the CLEF Association (CLEF 2019). Springer (Sep 2019)
5. Harman, D.: How effective is suffixing? Journal of the american society for information science **42**(1), 7–15 (1991)
6. Jeffreys, H.: Theory of Probability,. OUP Oxford (1948)
7. Kocher, M., Savoy, J.: Distance measures in author profiling. Information Processing & Management **53**(5), 1103–1119 (2017)
8. Nath, S.: SCD_CLEF_ 2019. https://github.com/pan-webis-de/nath19 (2019)
9. Pennebaker, J.W.: The secret life of pronouns: What our words say about us. Bloomsbury Press, New York (2011)
10. Potthast, M., Gollub, T., Wiegmann, M., Stein, B.: TIRA Integrated Research Architecture. In: Ferro, N., Peters, C. (eds.) Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF. Springer (2019)
11. Potthast, M., Rosso, P., Stamatatos, E., Stein, B.: A decade of shared tasks in digital text forensics at pan. In: European Conference on Information Retrieval. pp. 291–300. Springer (2019)
12. Rangel, F., Rosso, P., Montes-y Gómez, M., Potthast, M., Stein, B.: Overview of the 6th author profiling task at pan 2018: multimodal gender identification in twitter. Working Notes Papers of the CLEF (2018)
13. Savoy, J.: Comparative evaluation of term selection functions for authorship attribution. Digital Scholarship in the Humanities **30**(2), 246–261 (2013)
14. Savoy, J.: Analysis of the style and the rhetoric of the 2016 us presidential primaries. Digital Scholarship in the Humanities **33**(1), 143–159 (2017)
15. Savoy, J.: Is starnone really the author behind ferrante? Digital Scholarship in the Humanities **33**(4), 902–918 (2018)

16. Sebastiani, F.: Machine learning in automated text categorization. ACM computing surveys (CSUR) **34**(1), 1–47 (2002)
17. Zangerle, E., Tschuggnall, M., Specht, G., Potthast, M., Stein, B.: Overview of the Style Change Detection Task at PAN 2019. In: Cappellato, L., Ferro, N., Losada, D., Müller, H. (eds.) CLEF 2019 Labs and Workshops, Notebook Papers. CEUR-WS.org (Sep 2019)