Stylistic Context Clustering for Token-Level Author Diarization

Ivan Grubišić, Milan Pavlović, Mate Šimović

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{ivan.grubisic, milan.pavlovic, mate.simovic}@fer.hr

Abstract

In this paper we present an approach to tackle the PAN 2016 author diarization and intrinsic plagiarism detection problem. We propose a method for unsupervised text diarization based on clustering of tokens using contextual style features obtained with a sliding window. There are two variants of our approach - one that uses different features based on the document being analyzed, and one that uses a fixed feature space and tries to adopt if for clustering with a trainable feature transformation. We have evaluated our approach on the available datasets from PAN 2016.

1. Introduction

In this paper we will focus on the author diarization task proposed on the PAN 2016 competition.[1] The aim of this task is to decompose a document into its authorial parts, i.e., to split a text into segments and assign an author to every segment (Koppel et al., 2011; Aldebei et al., 2015). This is one of the unsupervised variants of a well known authorship attribution problem since text samples of known authorship are not available (Rosso et al., 2016). As we will describe, in two out of three subtasks of this task only a correct number of authors for a given document is known.

The simplest variant of the authorship attribution problem is about finding the most likely author for a given document from a set of candidate authors whose authentic writing examples are available (Stamatatos, 2009b; Stein et al., 2011; Ding et al., 2016). This problem can be tackled with supervised machine learning techniques as a single-label multiclass text classification problem, where one class represents one author (Stamatatos, 2009b).

The authorship attribution problem is also known as authorship identification and it is a part of authorship analysis (Stamatatos, 2009b; Ding et al., 2016). Authorship analysis is a field of stylometry and studies information about the authorship of a document, based on features derived from that document (Layton et al., 2013). Moreover, stylometry analyses literary style with statistical methods (Stein et al., 2011).

Rosso et al. (2016) divided the PAN 2016 author diarization task into three subtasks. The first subtask is traditionally called intrinsic plagiarism detection (IPD). The goal of this task is to find plagiarized parts of a document in which at least 70% of text is written by main author and the rest by one or more other authors. The term *intrinsic* means that a decision whether a part of the document is plagiarized has to be made only by analysing the given document, without any comparisons with external sources. This was our main motivation for solving this and other subtasks. In the rest of the paper we refer to this subtask as Task *a*.

Other two subtasks are more related to the general task of author diarization. In the second subtask we need to segment a given document and group identified segments by author. In the rest of the paper we refer to the second subtask as a Task *b*. The third subtask differs from the second one in the fact that exact number of authors is unknown. In the rest of the paper we refer to the third subtask as a Task *c*.

Table 1: Basic characteristics of training sets. * represents that there is a true author and plagiarism segments which do not have to originate from a single author.

Task	Number of documents	Average length (in tokens)	(min, max) authors
Task *a*	71	1679	(2, 2)*
Task *b*	55	3767	(2, 10)
Task *c*	54	3298	(2, 10)

For each of the three subtasks a training set is publicly available.[1] Rosso et al. (2016) explain that they are collections of various documents which are part of Webis-TRC-12 dataset (Potthast et al., 2013). Every document in that dataset is constructed from texts of various search results (i.e., authors) for one of the 150 topics in total. By varying different parameters such as the number and proportion of the authors, places in a document where an author switch occurs (between words, sentences or paragraphs), three training and test sets were generated (Rosso et al., 2016). Test datasets are currently not publicly available and we could not use them for evaluation of our approach. Some basic characteristics of the training sets are shown in Table 1.

2. Related work

The basic assumption in authorship analysis is that texts of different authors are mutually separable because each author has a more or less unique writing style (Stamatatos, 2009b; Ding et al., 2016). Therefore, the most of related work tries to distinguish writing styles by finding better features and methods which writing style will be quantified and measured with.

[1] http://pan.webis.de/clef16/pan16-web/author-identification.html

Zu Eissen and Stein (2006) used the average sentence length, part of speech tags, the average stop word number and the averaged word frequency class as input features for their linear discriminant analysis and support vector machine (SVM) models. Stamatatos (2009a) introduced a sliding window approach with character tri-grams as input features for a style change function whose peaks indicate positions in the document where style changes occur. This is similar to an outlier detection approach from Stein et al. (2011), but they applied a naive Bayes' algorithm. Rahman (2015) also used a sliding window and an SVM, but introduced new kinds of information theoretical features.

Koppel et al. (2011) used normalized cuts algorithm to obtain initial clusters of segments which were represented only by normalized counts of synonyms from Hebrew synsets. An SVM was then used to classify bag-of-words feature vectors of non-representative cluster samples. Brooke et al. (2013) concluded that a very good initial segmentation of text, at least in poems written by T. S. Elliot, is needed for good performance of their modified k-means algorithm in clustering of voices.

The works by Kuznetsov et al. (2016) and Sittar et al. (2016) were submitted on the PAN 2016 competition for three aforementioned tasks. Their approaches mostly focused on initial segmentation with the help of a style change function and clustering as the final step. To estimate the unknown number of authors in Task *c*, Kuznetsov et al. (2016) defined a cluster discrepancy measure which was than maximized, while Sittar et al. (2016) generated that number randomly.

The most of the described approaches operate on the level of longer text segments or sentences. Since the style change in our tasks can occur even between two tokens in the same sentence, we wanted our model to be able to work on the token level. We were also inspired by Brooke et al. (2013) who said that a more radical approach would not separate the described tasks in segmentation and clustering steps, but rather build authorial segments that would also form good clusters. Instead of clustering tokens directly, we decided to cluster their vectorized stylistic contexts because they obviously contain more valuable stylistic information than tokens alone.

3. Author diarization and intrinsic plagiarism detection

Let Δ be the domain of documents. We define a document $D \in \Delta$ as a finite sequence of tokens $(t_i)_{i=1}^{n}$, where n can differ among documents. Given a document, each of its tokens is unique and defined by its character sequence and position in the document. Therefore, a document can be equivalently represented by its set of tokens $T_D = \{t_i\}_{i=1}^{n}$.

For each document, there is a corresponding mapping to a sequence of labels $(a_i)_{i=1}^{n}$ that are representing groupings of tokens by authors. The labels a_i are indices of authors of the document. Each token $t_i \in T_D$ is assigned a document-level label $a_i \in \{1..c\}$ associating it to one of c authors. The exact value of the label is not important. It is only required that all tokens corresponding to the same author have the same label. Therefore, there are $m!$ equivalent such mappings given a document. In the case of intrinsic plagiarism detection, there are only 2 labels: 0 representing the main author, and 1 representing plagiarized text.

Equivalently, the codomain of the mapping can also be defined as a set of segmentations Σ. A segmentation $S \in \Sigma$ is a minimal set of segments, where each segment s is a set of consecutive tokens $\{t_i\}_{i=i_1}^{i_2}$ where each token is associated with the same author label. For a segmentation to be valid, the segments must cover all terms in the document and not overlap.

The correct mapping of a documents to the corresponding segmentations will be denoted with $\sigma : \Delta \to \Sigma$. Let $\mathcal{D} \subset \Delta \times \Sigma$ be a dataset consisting of a finite set of pairs of documents and corresponding segmentations, i.e., $\mathcal{D} = \{(D_i, \sigma(D_i))\}_{i=1}^{N}$. The goal is to find the model $\hat{\sigma}$ that best approximates the correct mapping σ, i.e., makes good predictions given unseen documents.

3.1. Evaluation measures

For evaluation of intrinsic plagiarism detection, Potthast et al. (2010) define multiple measures for different aspects of a system's performance. The main measures are binary macro-averaged and micro-averaged precision (P), recall (R) and F_1-score. For evaluating author diarization, we use *BCubed* precision, recall and F_1-score described by Amigó et al. (2009), which are specialized for evaluation of clustering results. The same measures were used for evaluation on the PAN 2016 competition (Rosso et al., 2016).

Let l be a function that associates lengths in characters to segments. Specially, $l(\{\}) = 0$. For notational convenience, we also use l to denote the sum of lengths of all segments in a set of segments: $l(S) = \sum_{s \in S} l(s)$, where S is as set of segments. Given a document D, let $S_\mathrm{p} \subseteq \sigma(D)$ be a set of all true plagiarism segments of the document and $\hat{S}_\mathrm{p} \subseteq \hat{\sigma}(D)$ the segments predicted as plagiarism by the model. With $S_\mathrm{tp} = \bigcup_{(s,\hat{s}) \in S_\mathrm{p} \times \hat{S}_\mathrm{p}} l(s \cap \hat{s})$, the micro-averaged evaluation measures for intrinsic plagiarism detection are defined as follows:

$$P_\mu = \frac{l(\hat{S}_\mathrm{tp})}{l(\hat{S}_\mathrm{p})}, \tag{1}$$

$$R_\mu = \frac{l(\hat{S}_\mathrm{tp})}{l(S_\mathrm{p})}, \tag{2}$$

$$F_\mu = \frac{2}{P_\mu^{-1} + R_\mu^{-1}}. \tag{3}$$

The macro-average evaluation measures treat all plagiarism segments as equally important and are not affected by their lengths:

$$P_\mathrm{M} = \frac{1}{|\hat{S}_\mathrm{p}|} \sum_{\hat{s} \in \hat{S}_\mathrm{p}} \frac{\sum_{s \in S_\mathrm{p}} l(s \cap \hat{s})}{l(\hat{s})}, \tag{4}$$

$$R_\mathrm{M} = \frac{1}{|S_\mathrm{p}|} \sum_{\hat{s} \in S_\mathrm{p}} \frac{\sum_{s \in \hat{S}_\mathrm{p}} l(s \cap \hat{s})}{l(s)}, \tag{5}$$

$$F_\mathrm{M} = \frac{2}{P_\mathrm{M}^{-1} + R_\mathrm{M}^{-1}}. \tag{6}$$

In author diarization, document segments have to be clustered into c clusters, where c is the number of authors

that may or may not be known to the system. We divide the segments from the true segmentation S and the predicted segmentation $\hat{S}$ each into sets of segments S_i, $i \in \{1..c\}$, and $\hat{S}_j$, $j \in \{1..\hat{c}\}$, where c is the true number of authors, and $\hat{c}$ the predicted number of authors. We use the following *BCubed* measures for evaluation:

$$P_{\mathrm{B}^3} = \sum_{i=1}^{c} \frac{1}{l(S_i)} \sum_{j=1}^{\hat{c}} \sum_{s \in S_i} \sum_{\hat{s} \in \hat{S}_j} l(s \cap \hat{s})^2, \tag{7}$$

$$R_{\mathrm{B}^3} = \sum_{j=1}^{\hat{c}} \frac{1}{l(\hat{S}_j)} \sum_{i=1}^{c} \sum_{s \in S_i} \sum_{\hat{s} \in \hat{S}_j} l(s \cap \hat{s})^2, \tag{8}$$

$$F_{\mathrm{B}^3} = \frac{2}{P_{\mathrm{B}^3}^{-1} + R_{\mathrm{B}^3}^{-1}}. \tag{9}$$

4. The proposed approach

Our approach can generally be described as a pipeline $P = (f_{\mathrm{b}} : \Delta \to \Phi_{n_{\mathrm{b}}}, f_{\mathrm{t}} : \Phi_{n_{\mathrm{b}}} \to \Phi_{n_{\mathrm{t}}}, f_{\mathrm{c}} : \Phi_{n_{\mathrm{t}}} \to C)$. Here Δ is the tokenized document domain as defined in section 3. $\Phi_{n_{\mathrm{b}}}$ and $\Phi_{n_{\mathrm{t}}}$ are sets of variable-length sequences of feature vectors with dimension n_{b} and n_{t} respectively. n_{b} and n_{t} can either be fixed or depend on the document being processed. C is the set of all sequences of length n (the number of tokens in the document) with elements being indices of authors/clusters.

The basic feature extractor denoted with f_{b} is used to extract stylistic features from the contexts of all tokens. If D is a document with n tokens, the basic feature extractor outputs a sequence of n feature vectors, each vector representing the context of one token. n_{b} denotes the dimension of those vectors. The next step in the pipeline is the feature transformation f_{t} that maps the basic features to a space that they can be better clustered in. The dimension of the new feature space n_{t} generally does not equal n_{b}. The final step in the pipeline is clustering denoted with f_{c}. The clustering algorithm implicitly clusters tokens because it actually clusters their stylistic contexts, each cluster representing an author. Depending on the task, the clustering algorithm can either be given a known number of authors, or try to predict it.

The following steps are done in predicting the segmentation: (1) raw text is tokenized giving a sequence of tokens $D \in \Delta$, (2) for all tokens, features are extracted from their contexts, giving a sequence of feature vectors $\phi_{\mathrm{t}} = (\boldsymbol{t}_i)_{i=1}^{n} = (f_{\mathrm{t}} \circ f_{\mathrm{b}})(D)$, (3) the tokens are clustered based on ϕ_{t}, giving a sequence of author labels $(a_i)_{i=1}^{n}$, where $n = |T_D|$, and (4) a segmentation $\hat{S}$ is generated based on the document D and the sequence of predicted author labels.

We develop two variants of our model which we will refer to as *Model A* and *Model B*. They are mainly differentiated by the fitting of the basic feature extractor, feature transformation learning and in the feature transformation. Model B utilizes a trainable feature transformation that requires a set of basic features with a fixed dimension. Therefore, the basic feature extractor must be fitted on the whole training set. Conversely, Model A's basic feature transformation can be performed specifically allowing for use of less more important features, but it can not use a feature transformation trainable on multiple documents. In both approaches basic features are extracted from a sliding window which moves from token to token and includes the context of each token.

Tokenization. As a preprocessing step, we tokenize each document using NLTK[2] by Bird et al. (2009) to obtain a sequence of tokens D. We also perform part-of-speech tagging with the same tool, to speed up basic feature extraction which we describe below. We did not use other preprocessing techniques such as lemmatization, stemming and stop word removal because they would take away a lot of stylometric data from text (Stamatatos (2009b). The final output from the tokenization step was a finite sequence $\{(t_i, o_i, l(t_i), POS_i)\}_{i=1}^{n}$ where o_i is the offset of token t_i, $l(t_i)$ its length in characters and POS_i its POS tag.

Basic feature extraction. We define the stylistic context of a token t_i as a set of tokens $\{t_k\}_{k=i-cs}^{i+cs} \setminus \{t_i\}$ where cs is a context size. From each context we extracted the most used stylometric features which we found in previous work. The features that we have considered in our models are:

- *Character tri-grams*. Frequencies of n-grams on character level have been very useful in quantifying the writing style (Stamatatos, 2009a). They are able to capture lexical and contextual information, use of punctuation and errors which can be an author's "fingerprint". This feature is also tolerant to noise. Based on work by Stamatatos (2009b) and Rahman (2015), we choose $n = 3$. Maximal dimension of this feature vector was set to 200.

- *Stop words*. According to Stamatatos (2009b), sometimes also called *function words*, these are the most common used topic-independent words in text, such as articles, prepositions, pronouns and others. They are used unconsciously and found to be one of the most discriminative features in authorship attribution since they represent pure stylistic choices of authors' (Burrows, 1987; Argamon and Levitan, 2005). We used frequencies of 156 English stop words available in NLTK.

- *Special characters*. We used counts of all character sequences which satisfied a regular expression defined as `[^\\w]{1,4}`. Although character n-grams can catch the use of those character sequences, we wanted to have a distinct feature for that purpose since Koppel et al. (2009) mentioned that authors can have different punctuation habits.

- *POS tag counts*. This is syntactic feature which Koppel et al. (2009) and Stamatatos (2009b) also identified as a discriminative one in authorship analysis and it was used by Kuznetsov et al. (2016). We used all 12 tags from the universal tag set available in NLTK.

- *Average token length*. Used by Kuznetsov et al. (2016), Sittar et al. (2016), Brooke and Hirst (2012) and Stein et al. (2011). Koppel et al. (2009) characterized this feature as a complexity measure.

[2] `http://www.nltk.org`

- *Bag of Words*. Bag of words text representation more captures content, rather than style (Stamatatos, 2009b). We included this feature because it boosted performance of our initial testing, even without words been previously stemmed or lemmatized (stop words were excluded). Counts of a maximum of 100 unigrams were used.

- *Type-token ratio*. We wanted to use that feature to measure the vocabulary richness in a token's context, but after we finished with performance evaluations we realized that there was a bug in implementation and its impact on final results was unknown to us. The feature should be calculated as the ratio of vocabulary size and total number of tokens of the text (Stamatatos, 2009b).

Feature transformation. As its feature transformation f_{t}, Model A uses feature scaling to zero mean and unit variance for each document separately. Model B uses a trainable transformation which requires a basic feature space of fixed dimension. Let $(\boldsymbol{b}_i)_{i=1}^n = f_{\mathrm{b}}(D)$ be a sequence of basic feature vectors and $(\boldsymbol{b}'_i)_{i=1}^n$ the corresponding sequence of potentially preprocessed basic feature vectors with elements from $\mathbb{R}^{n'_{\mathrm{b}}}$. Let $(a_i)_{i=1}^n$ be the sequence of true author labels with elements from $\{1..c\}$. We want to maximize the *clusterability* of the feature vectors obtained by the feature transformation $T : \mathbb{R}^{n'_{\mathrm{b}}} \to \mathbb{R}^{n_{\mathrm{t}}}$ with trainable parameters $\boldsymbol{\theta}_T$, i.e., we want to maximize segregation and compactness of groups of transformed feature vectors grouped by their target author label. Let $(\boldsymbol{t}_i)_{i=1}^n = (T(\boldsymbol{b}'_i))_{i=1}^n = f_{\mathrm{t}}((\boldsymbol{b}_i)_{i=1}^n)$ be the sequence of transformed feature vectors with elements from $\mathbb{R}^{n_{\mathrm{t}}}$. In order to optimize the transformation T, we define the following loss:

$$L = \alpha L_{\mathrm{c}} + (1 - \alpha) L_{\mathrm{s}} \tag{10}$$

with α chosen to be 0.5. Here L_{c} is the *compactness loss* and L_{s} the *segregation loss*. L_{c} is proportional to the average variance of groups, and L_{s} penalizes centroids being too close to each other. Let N_a represent the number of tokens associated with author a, i.e., $N_a = \sum_{i=1}^n [a_i = a]$, where, for φ being a logical proposition $[\varphi]$ evaluates to either 1 if φ is true or 0 otherwise. Let $\boldsymbol{\mu}_a$ represent the current centroid of the transformed feature vectors associated with the author label a:

$$\boldsymbol{\mu}_a = \frac{1}{N_a} \sum_{i=1}^{n} T(\boldsymbol{b}'_i)[a_i = a]. \tag{11}$$

We define the components of the loss:

$$L_{\mathrm{c}} = \sum_{a=0}^{c} \frac{1}{N_a} \sum_{i=1}^{n} \|T(\boldsymbol{b}'_i) - \boldsymbol{\mu}_a\|^2 [a_i = a], \tag{12}$$

$$L_{\mathrm{s}} = \frac{2}{c} \sum_{a=0}^{c} \sum_{b=a+1}^{c} \|\boldsymbol{\mu}_a - \boldsymbol{\mu}_b\|^{-2}. \tag{13}$$

The *compactness loss* L_{c} is a weighted sum of within-group variances. The *segregation loss* L_{s} is proportional to the sum of magnitudes (not magnitude of the sum) of forces between c equally charged particles each located at one of the centroids. By minimizing the average loss (the error) across all documents in the training set with respect to the parameters of the transformation $\boldsymbol{\theta}_T$ we hope to benefit the clustering algorithm. Instead of squared L^2 distance, some other distance measure may be used. Also, the way of combining the two losses was somewhat arbitrarily chosen as well as the segregation loss. For the transformation T we have tried a nonlinear transformations represented as neural networks with one or more hidden layers. We did not explore nonlinear transformations much because they were slower and did not give as good results as a linear transformation or an elementwise linear transformation:

$$T_1(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x}, \tag{14}$$

$$T_2(\boldsymbol{x}) = \boldsymbol{w} \odot \boldsymbol{x}. \tag{15}$$

Here $\boldsymbol{W} \in \mathbb{R}^{n'_{\mathrm{b}} \times n_{\mathrm{t}}}$ and $\boldsymbol{w} \in \mathbb{R}^{n'_{\mathrm{b}}}$. Before applying the transformations, basic feature vectors are preprocessed by concatanating each with the vector of its squared elements:

$$\boldsymbol{b}'_{\boldsymbol{i}} = (\boldsymbol{b}_{\boldsymbol{i}}^{\mathsf{T}}, (\boldsymbol{b}_{\boldsymbol{i}} \odot \boldsymbol{b}_{\boldsymbol{i}})^{\mathsf{T}})^{\mathsf{T}}, \quad i \in \{1..n\}. \tag{16}$$

The transformation is implemented using TensorFlow.[3] Parameters are randomly initialized with values from a truncated normal distribution with standard deviation 0.1. It is trained on-line (one document per optimization step) using RMSProp (Tieleman and Hinton, 2012) with default parameters and learning rate 5×10^{-4}.

Clustering. The final step in our approach is clustering of obtained feature vectors which represent stylistic contexts of tokens. The number of clusters is equal to the number of authors in tasks *a* and *b*, but in Task *c* it is unknown. We tested several clustering algorithms available in scikit-learn[4] Python toolkit by Pedregosa et al. (2011) in all tasks. In Model A for tasks *a* and *b* we propose hierarchical agglomerative clustering, and for Task *c* DBSCAN algorithm which estimates the number of clusters automatically. In Model B we use k-means for clustering in all tasks, and tried predicting the number of clusters based on the elbow rule. This is described in more detail in section 5.

5. Experimental results

For experiments, we have used the three PAN 2016 training sets. The test sets used in the competition have not been made publicly available. Therefore, our evaluation results might be only moderately comparable to the results of other systems.

In addition to the 2 models from the teams that took part in solving the author diarization problem on the PAN 2016 competition, we defined 3 baselines for evaluation purposes. The first baseline, Single-author dummy baseline, always assigns the whole document a single author label, or, in the case of intrinsic plagiarism detection, it labels the whole document as plagiarism. The second baseline, Stochastic dummy baseline, learns the average total lengths of authorial parts over all documents ranked by the length of their parts. For prediction, by modelling a Markov chain, it generates segmentations with similar label frequencies and predefined expected lengths.

[3] `https://www.tensorflow.org`

[4] `http://scikit-learn.org`

In the third baseline, Simple baseline, we use default k-means clustering of vectorized document's sentences. Average token length, sentence length, universal POS tag counts and bag of at most 100 words (including stop words) serve as features. If the number of authors is unknown, it is randomly generated from a uniform distribution $(0, 0.0003 \times l(D))$ and increased by 1 if the result is 0.

We performed initial testing of usefulness of all features except for the accidentally wrongly implemented type-token ratio which was added afterwards. For testing purposes we used the implementation of the k-means algorithm from scikit-learn with the default settings and all documents from the dataset for Task *b*. As we were adding feature by feature, BCubed precision was becoming higher so we decided to use all features for Model A in all tasks. We do not know the exact impact of the wrongly implemented type-token ratio feature. Instead of the context's vocabulary size, we divided the size of document's vocabulary with the context size so this feature had the same value in all vectors.

5.1. Experimental setup

In every task we used the same tokenization procedure as described in section 4. Model A covers all three tasks and has several hyperparameters for each of them: context size, clustering algorithm and clustering algorithm's hyperparameters. Model B uses a smaller set of features and trainable feature transformations with most of the hyperparameters equal among the tasks. All the source code for our experiments is available in our repository.[5]

Model A. Sliding widow context size was determined manually for tasks *a* and *b*, in a similar way as feature selection. We used a whole dataset and default implementation of k-means from scikit-learn and tested different values. The best result for Task *a* was $cs_{\mathrm{a}} = 16$ and for Task *b* $cs_{\mathrm{b}} = 140$. For Task *c*, we left $cs_{\mathrm{c}} = 140$ as for Task *b*.

For clustering algorithm selection we used manual search over several algorithms. We split the whole dataset into a training and a test set. For each algorithm we were examining, a grid search of its hyperparameters was performed on the training set. We selected an algorithm and hyperparameters that achieved the best performance on the training set (since this is an unsupervised approach), and reported the final performance on the test set. We did not include the k-means algorithm in clustering model selection since we used it for context size estimation.

In Task *a*, we used 50% of the dataset as a training set, and the other half as a test set. Since this task is characterized as an outlier detection task, we tested an isolation forest algorithm, but achieved poor results. Other algorithm which we evaluated was hierarchical agglomerative clustering which achieved the best macro F_1 (described in subsection 3.1.) with cosine affinity and average linkage on the train set. We report its result on the test set.

For Task *b* we split the dataset in two halves as well. Grid search for the best hyperparameters was performed for agglomerative clustering. The best performance on the training set was obtained with Euclidean affinity and average linkage. For Task *c* we should have somehow determined the number of clusters. Due to a lack of time we decided to test algorithms which can determine this number automatically, i.e., this is not a hyperparameter which should be known in advance. Since we noticed that grid searches take a lot of time, we reduced a training set from 50% to 10% and repeated all experiments. We tested DBSCAN and affinity propagation algorithms.

The best results on Task *c* were obtained with DBSCAN. We noticed that the number of clusters estimated with the best hyerparameters (`eps=100, min_samples=71, metric="manhattan"`) was in almost every case greater than a real one, and solid number of examples were not even clustered because they were not close enough to any core sample to be a part of a cluster. In all grid searches we used the BCubed F_1-score and only DBSCAN achieved acceptable performance. Our opinion is that this was a consequence of a larger number of smaller clusters which had a positive influence on BCubed precision and therefore on overall performance.

Since the model with the best achieved Bcubed F_1-score (0.61) is not of practical use in real life applications because the number of clusters is always wrongly estimated, we decided to repeat the grid search in Task *c* for DBSCAN, but this time we used the Calinski-Harabaz index defined by Caliński and Harabasz (1974) as evaluation measure since it does not require the real number of clusters for clustering evaluation. Finally, from all results on training set we chose (`eps=0.23, min_samples=64, metric="cosine"`) for the final hyperparameters. This was not the best combination, but it produced relatively acceptable amount of noisy (non-clustered) elements per document, and performed quite fast but again did not estimate the correct number of clusters in most cases. If some samples were not clustered, we assigned them the majority label from neighborhood (context) of size 10 (5 on each side), or the majority label from all samples if samples in the context were not clustered.

Model B. For Model B, we have decided to use a smaller set of basic features: character tri-gram counts (for 120 trigrams), stop word counts, average token length and bag-of-words (for 120 words). The length of the sliding window is set to 120 tokens. Model B uses k-means clustering. We have chosen two trainable feature transformations: linear and elementwise linear (weighting). For both of the transformations 5×10^{-4} was chosen for the learning rate. All the hyperparameters are selected manually by experimenting on the development dataset, which is approximately 70% of the available dataset. 70% of the development set was used for fitting basic feature extractors and learning the transformation. Considering the results on the validation part of the development set, not very large numbers of iterations seem to be more beneficial. A number of 40 iterations was chosen for the linear transformation, and 20 for the elementwise linear transformation. For the output dimension of the linear transformation we have chosen $n_{\mathrm{t}} = 40$, though not much different results seem to be achievable using smaller numbers like 20, making training and clustering faster.

[5] `https://github.com/Coolcumber/inpladesys/`

Table 2: Results on the intrinsic plagiarism detection task. The standard errors both for F_μ and F_M are for Model A within ± 0.04 and for Model B within ± 0.06. Results of models denoted with * were obtained on the PAN 2016 test set which was not available to us.

Method	P_μ	R_μ	F_μ	P_M	R_M	F_M
Single-author dummy BL	0.13	**1.00**	0.22	0.13	**1.00**	0.22
Stochastic dummy BL	0.11	0.09	0.09	0.08	0.09	0.06
Simple BL	0.13	0.24	0.14	0.13	0.24	0.13
Kuznetsov et al. (2016)*	**0.29**	0.19	0.22	**0.28**	0.15	0.17
Sittar et al. (2016)*	0.14	0.07	0.08	0.14	0.10	0.10
Model A	0.18	0.43	**0.25**	0.18	0.43	**0.24**
Model B (linear)	0.21	0.35	0.23	0.21	0.34	0.22
Model B (weighting)	0.24	0.36	0.23	0.24	0.34	0.23

Table 3: Results on the author diarization task for known (Task *b*) and unknown (Task *c*) numbers of authors. The standard errors for both tasks are for Model A within ± 0.02 and for Model B within ± 0.05. Results of models denoted with * were obtained on the PAN 2016 test set which was not available to us.

Method	Task *b*			Task *c*		
	P_{B^3}	R_{B^3}	F_{B^3}	P_{B^3}	R_{B^3}	F_{B^3}
Single-author dummy BL	0.33	**1.00**	0.47	0.37	**1.00**	0.52
Stochastic dummy BL	0.35	0.59	0.41	0.39	0.59	0.45
Simple BL	0.43	0.50	0.45	0.45	0.54	0.45
Kuznetsov et al. (2016)*	**0.64**	0.46	0.52	0.64	0.42	0.48
Sittar et al. (2016)*	0.28	0.47	0.32	0.31	0.47	0.35
Model A	0.61	0.65	**0.62**	**0.71**	0.50	0.54
Model B (linear)	0.48	0.45	0.46	0.56	0.82	**0.64**
Model B (weighting)	0.51	0.49	0.50	0.57	0.77	**0.64**

For Task c we have implemented a simple wrapper for the k-means algorithm that tries to predict the correct number of authors. It computes the k-means error for each number k in the specified range for the number of clusters. Based on ratios of relative improvements in the score between consecutive k-s, it chooses the k giving the best ratio modulated with a modulation function dependent on k, simply k^α was chosen, where α is selected using linear search. Being limited in time, we did not put much effort into optimizing the selection of the best number of clusters. Moreover, even if knowing the number of clusters in advance, it seems that even the transformed features are not satisfactory for correct clustering. We notice that using the correct numbers of authors compared to using a fixed $k = 2$ results in relative decrease in BCubed precision, recall, and F_1-score by factors of 0.95, 0.64 and 0.79 respectively.

5.2. Evaluation results

We have evaluated our models using micro-averaged and macro-averaged precision, recall and F_1-score for intrinsic plagiarism detection and the analogous BCubed scores for author diarization, as defined in subsection 3.1. The results of our models are compared to the results of our baselines, as well as the results of the two teams that took part in the PAN 2016 competition for this problem. The results of the two teams have been evaluated on test sets that are not publicly available. Only the training sets from the competition have been available to us.

The results of evaluation for the intrinsic plagiarism problem are shown in Table 2. For author diarization with known and unknown numbers of authors, the results are shown in Table 3. It should be noted that the evaluated Model B uses a fixed number of clusters $k = 2$ for task c and that the results are actually not very good. By trying a more honest approach with predicting the number of clusters the BCubed F_1-score drops below 0.5. The results of Stohastic dummy baseline for all tasks are obtained over the 50 runs, as well as for Simple baseline in Task c.

For both Models A and B we calculated standard errors and wanted to obtain 95% t-based confidence intervals. Since the sample sizes from all test sets but a are less than 30, a normality test[6] at $p = 0.05$ was performed first. The Model A data in all tasks did not provide enough evidence to reject the null hypothesis that a sample comes from a normal distribution. In test of Model B's both F_1-scores, the

[6] https://docs.scipy.org/doc/scipy-0.19.0/reference/generated/scipy.stats.normaltest.html

data provided enough evidence to reject the null hypothesis.

Although the Model A achieved the highest micro and macro-averaged F_1-scores on the Task *a*, this task seemed to be very difficult. F_1-scores of Model A are only slightly above the Single-author dummy baseline and lower 95% limit for macro-F_1 is 0.15. Model A also achieved the best, but actually not very good performance on the Task *b* with $(0.57, 0.67)$ 95% confidence interval. Although Model B behaved reasonably well on the Task *c*, a better way of estimating a true number of clusters is needed.

6. Conclusion

Our results, as well as the the results of others, suggest that author diarization and intrinsic plagiarism detection are very challenging problems. Our approach tries to solve the problem by clustering contextual writing style features extracted with a sliding window around each token.

We have presented some ideas that are not completely explored but, based on the data we have available, seem to give comparable, or even better results than the other two approaches that tried to tackle the same problem. There is a lot of room for improvement of our approach, as well as in exploring other new ideas.

For further work, it would be good to search for more useful basic features, make a deeper examination of the feature transformation, try other clustering algorithms and a use more systematic approach for choosing hyperparameters. It would be also interesting to try the trainable feature transformation approach on other authorship analysis problems.

References

Khaled Aldebei, Xiangjian He, and Jie Yang. 2015. Unsupervised Decomposition of a Multi-Author Document Based on Naive-Bayesian Model. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers)*, pages 501–505.

Enrique Amigó, Julio Gonzalo, Javier Artiles, and Felisa Verdejo. 2009. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information retrieval*, 12(4):461–486.

Shlomo Argamon and Shlomo Levitan. 2005. Measuring the usefulness of function words for authorship attribution. In *Proceedings of the 2005 ACH/ALLC Conference*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc.

Thorsten Brants and Alex Franz. 2006. Web 1t 5-gram corpus version 1.1. *Google Inc*.

Julian Brooke and Graeme Hirst. 2012. Paragraph clustering for intrinsic plagiarism detection using a stylistic vector space model with extrinsic features. In *CLEF (Online Working Notes/Labs/Workshop)*.

Julian Brooke, Graeme Hirst, and Adam Hammond. 2013. Clustering voices in the Waste Land. In *Proceedings of the 2nd Workshop on Computational Literature for Literature (CLFL'13), Atlanta*.

John F Burrows. 1987. Word-patterns and story-shapes: The statistical analysis of narrative style. *Literary and linguistic Computing*, 2(2):61–70.

Tadeusz Caliński and Jerzy Harabasz. 1974. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1):1–27.

Steven HH Ding, Benjamin Fung, Farkhund Iqbal, and William K Cheung. 2016. Learning stylometric representations for authorship analysis. *arXiv preprint arXiv:1606.01219*.

Moshe Koppel, Jonathan Schler, and Shlomo Argamon. 2009. Computational methods in authorship attribution. *Journal of the American Society for information Science and Technology*, 60(1):9–26.

Moshe Koppel, Navot Akiva, Idan Dershowitz, and Nachum Dershowitz. 2011. Unsupervised decomposition of a document into authorial components. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1356–1364. Association for Computational Linguistics.

Mikhail Kuznetsov, Anastasia Motrenko, Rita Kuznetsova, and Vadim Strijov. 2016. Methods for intrinsic plagiarism detection and author diarization. *Working Notes Papers of the CLEF*.

Thomas K Landauer and Susan T Dumais. 1997. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211.

Robert Layton, Paul Watters, and Richard Dazeley. 2013. Automated unsupervised authorship analysis using evidence accumulation clustering. *Natural Language Engineering*, 19(01):95–120.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Martin Potthast, Benno Stein, Alberto Barrón-Cedeño, and Paolo Rosso. 2010. An evaluation framework for plagiarism detection. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 997–1005. Association for Computational Linguistics.

Martin Potthast, Matthias Hagen, Michael Völske, and Benno Stein. 2013. Crowdsourcing interaction logs to understand text reuse from the web. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1212–1221. Association for Computational Linguistics.

Rashedur Rahman. 2015. Information Theoretical and Statistical Features for Intrinsic Plagiarism Detection. In *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 144.

Paolo Rosso, Francisco Rangel, Martin Potthast, Efstathios Stamatatos, Michael Tschuggnall, and Benno Stein. 2016. Overview of pan'16. In *International Conference of the Cross-Language Evaluation Forum for European*

Languages, pages 332–350. Springer.

Abdul Sittar, Hafiz Rizwan Iqbal, and A. Nawab. 2016. Author Diarization Using Cluster-Distance Approach. *Working Notes Papers of the CLEF*.

Efstathios Stamatatos. 2009a. Intrinsic plagiarism detection using character n-gram profiles. In *3rd PAN Workshop. Uncovering Plagiarism, Authorship and Social Software Misuse*, pages 38–46.

Efstathios Stamatatos. 2009b. A survey of modern authorship attribution methods. *Journal of the American Society for information Science and Technology*, 60(3):538–556.

Benno Stein, Nedim Lipka, and Peter Prettenhofer. 2011. Intrinsic plagiarism analysis. *Language Resources and Evaluation*, 45(1):63–82.

T. Tieleman and G. Hinton. 2012. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.

Sven Meyer Zu Eissen and Benno Stein. 2006. Intrinsic plagiarism detection. In *European Conference on Information Retrieval*, pages 565–569. Springer.