Cross-domain Authorship Attribution: Author Identification using Char Sequences, Word Uni-grams, and POS-tags Features

Notebook for PAN at CLEF 2018

Yaakov HaCohen-Kerner, Daniel Miller, Yair Yigal, and Elyashiv Shayovitz

Department of Computer Science, Jerusalem College of Technology, Lev Academic Center

21 Havaad Haleumi St., P.O.B. 16031, 9116001 Jerusalem, Israel

kerner@jct.ac.il,3danmi@gmail.com,
yigalyairn@gmail.com,elyashiv12@gmail.com

Abstract. Authorship Attribution deals with identifying the author of an anonymous text, i.e., to attribute each test text of unknown authorship to one of a set of known authors, whose training texts are given. In this paper, we describe the participation of our teams (miller18 and yigal18, both teams contain the same people, but in another order) in the PAN 2018 shared task on cross-domain Author Identification. Given a set of documents authored by known authors, there is a need to identify the authors of documents from another set of documents. All documents are in the same language that may be one of the five following languages: English, French, Italian, Polish, or Spanish. In this paper, we describe our pre-processing, feature sets, the applied machine learning methods and the average F1 scores of three submitted models. For the evaluation corpus, we sent the top three models according to their results on the development corpus using PCA and Linear SVC. The first model scored an average of 0.582. Its features consist of the frequencies of all char 6-gram sequences, POS-tags sequences frequencies, Orthographic features, Quantitative features, and lexical richness features. The second model scored an average of 0.598. Its features consist of all the char sequences of length between 3 to 8, all word Uni-grams, POS-tags features, and all stylistic features from the first model. The third model scored an average of 0.611. Its features consist of the content-based features mentioned in the second model and POS-tags features.

Keywords: Authorship Attribution, Author Identification, Content-based Features, Style-based Features, Supervised Machine Learning, Text Classification.

1 Introduction

Authorship Attribution (AA) is a sub-task of the text classification (TC) paradigm, which deals with the identification of the author of a text [32]. The principle task in AA is to attribute test texts of unknown authorship to one of a set of known authors, whose training texts are given. This task is usually performed as a text classification process using supervised machine learning (ML) method(s). "The main idea behind statistically or computationally supported authorship attribution is that by measuring some textual features, we can distinguish between texts written by different authors" [35].

In this paper, we describe the participation of our teams (miller18 and yigal18, both teams contain the same people, but in another order) in the PAN 2018 shared task on cross-domain authorship attribution. More specifically, the shared task has been set up as follows. Given a set of documents (known fanfics) by a small number (up to 20) of candidate authors, identify the authors of another set of documents (unknown fanfics). Each candidate author has contributed at least one of the unknown fanfics, which all belong to the same target fandom. The known fanfics belong to several fandoms (excluding the target fandom), although not necessarily the same for all candidate authors. An equal number of fanfics per candidate author is provided. In contrast, the unknown fanfics are not equally distributed over the authors. The text-length of fanfics varies from 500 to 1,000 tokens. All documents are in the same language that may be English, French, Italian, Polish, or Spanish.

The rest of the paper is organized as follows. Section 2 provides background and presents some related work on text classification in general and authorship attribution in particular. Section 3 introduces the feature sets that we have implemented and used in our extensive experiments. Section 4 presents the experimental setup, the results and their analysis. Finally, Section 5 summarizes and suggests ideas for future research.

2 Related Work

2.1 Text classification

TC is the supervised learning task of assigning natural language text documents to one or more predefined categories [26]. There are two main types of TC: topic-based classification and style-based classification. An example of a topic-based classification application is classifying news articles to various categories such as Business-Finance, Lifestyle-Leisure, Science-Technology, and Sports, which were downloaded from three well-known news web-sites (BBC, Reuters, and TheGuardian) [24]. An example of a style-based classification application is classification based on different literary genres, e.g., action, comedy, crime, fantasy, historical, political, saga, and science fiction [21, 9].

These two classification types often require different types of feature sets to achieve the best performance. Topic-based classification is typically performed using word uni-grams and/or n-grams (n > 2) [1, 12]. Style-based classification is typically performed using linguistic features such as quantitative features, orthographic features, part of speech (POS) tags, function words, and vocabulary richness features [21, 13, 14, 9].

2.2 Authorship attribution

AA can be viewed as a sub-task of TC. At the beginning of the history of AA, it had only limited application, mainly to literary works of unknown or disputed authorship [27]. However, during the last two decades, AA grew and developed impressively due to its many applications in a widespread variety of domains, e.g., criminal law, computer forensics, humanities research, and military intelligence [35].

Current research in AA focuses mainly on the extraction of the best features (both style-based and content-based) for identification of document authors and suitable ML methods.

3 Feature Sets

In this section, we present the content-based and style-based features that we applied for the authorship attribution task. Each combination of those features is defined by the following template: *"number k-skip-n-type [style]"*. All the features were scaled by the Sklearn's MaxAbsScaler[1].

3.1 Content-based features

Our content-based features include various n-gram feature sets according to the following pattern "*number k-skip-n-type*", where *'number'* is the number of wanted n-grams in the set (the size of our vocabulary, e.g., 1000 or 5000, or 'all', which means that we used all the tokens in the dataset), '*k'* is the size of the wanted skip (0 – no skip, 1 – skip of one unit, 2 – skip of 2 units, …), '*n'* is code of the gram type (1 for Uni-grams 2 for bigrams, 3 for trigrams, …), and '*type'* is 'word' for words or 'char' for characters. All values are represented by TF-IDF values. The specific various n-gram feature sets that were applied will be presented later in the framework of the experiments.

3.2 Style-based features

Our style-based features include the following feature sets: POS-Tags, Quantitative features, Orthographic features, and Lexical-Richness features. The *'[style]'* pattern in the template mentioned above is a list of style features separated by *'_'* where 'pos' stands for POS-tags, 'quan' for quantitative features, 'orth' for orthographic features and 'rich' for lexical richness features. The POS-tags features are the frequencies of POS-tags sequences of length 1 to 10 (we tried various lengths of sequences and by using the value of 10 we obtained the best results. Maybe some authors tend to use specific POS sequences of this length, maybe over more than one sentence). The POS-tags features for English and Spanish were created using the Stanford POS-Tagger[2] and for French, Italian and Polish using the RDRPOSTagger[3]. While most of the POS-

[1] http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MaxAbsScaler.html
[2] https://nlp.stanford.edu/software/tagger.shtml
[3] http://rdrpostagger.sourceforge.net/

taggers we used have fine grained tags, the Polish POS-tagger has only UniPOS-tags. The Quantitative set includes 12 features: Average and median number of characters of a word (with and without punctuation) or a token, Average and median number of words or tokens in a sentence and Average and median number of character in a sentence. The Orthographic set includes 32 features. Each one represents the frequency of a specific character (normalized by the number of characters in the tested document) from the following list of characters: !"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~. The Lexical-Richness set includes two features: the percentage of unique words in the text, and the percentage of hapaxes in the text.

The tokenization was performed by NLTK's word tokenizer[4]. Sentence separation was performed by NLTK's sentence tokenizer[5]. Word separation was performed using the single space character.

4 Experimental Setup and Results

The PAN CLEF 2018 [36] launched an evaluation campaign. Ten teams have participated in this campaign. Each team has proposed its algorithm, which has been evaluated using the TIRA platform [24]. The algorithms and the results of the participated teams have been overviewed in [22].

General approach: Our approach to authorship Attribution is to apply supervised ML methods to TC as was suggested by Sebastiani [32]. The process is as follows. First, given a corpus of training documents, where each document is labeled by its author, we processed each document to produce values for various combinations of features from different types of features sets: content-based features and style-based features. Second, we apply several popular ML methods on the generated combinations of features. Third, we try additional combinations of features and parameter tuning. Finally, the best model(s) are tested on out-of-training data (i.e., evaluation data).

Preprocessing: There is a widespread variety of text preprocessing types such as: conversion of uppercase letters into lowercase letters, HTML object removal, stopword removal, punctuation mark removal, reduction of different sets of emoticon labels to a reduced set of wildcard characters, replacement of HTTP links to wildcard characters, word stemming, word lemmatization, correction of common misspelled words, and reduction of replicated characters. Not all the preprocessing types are considered as effective by all TC researchers. Many systems use only a small number of simple preprocessing types (e.g., conversion of all the uppercase letters into lowercase letters and or stopwords removal).

In our classification experiments, we found that most of the preprocessing types are irrelevant, because the data we have is already processed and it is mostly free of errors and mistakes. We mainly used characters for the content-based features, so stopwords removal did not improve the TC results. We also tried stemming, but the stems hurt the

[4] http://www.nltk.org/api/nltk.tokenize.html#nltk.tokenize.punkt.PunktLanguageVars.word_tokenize

[5] http://www.nltk.org/api/nltk.tokenize.html#nltk.tokenize.sent_tokenize

results. Therefore, the only preprocessing type we used was converting uppercase letters into lowercase letters.

ML methods: We applied five popular ML methods: MLP – Multilayer Perceptron[6], SVC[7], LinearSVC[8], LR - Logistic regression[9], and RF - Random Forest[10].
A brief description of these ML methods follows: MLP is a feedforward neural network ML method [17] where artificial neural network (ANN) can be viewed as a weighted directed graph in which nodes are artificial neurons and directed edges (with weights) are connections from the outputs of neurons to the inputs of neurons. Support vector machine (SVM, also called support vector network) [6] is a model that assigns examples to one of two categories, making it a non-probabilistic binary linear classifier. SVC is a type of SVM with an RBF kernel implemented using LibSVM [5]. LinearSVC is a type of SVM with a linear kernel implemented using LibLinear [7], which is recommended for TC because most of TC problems are linearly separable [18] and training an SVM with a linear kernel is faster compared to other kernels. Logistic regression (LR) is a variant of a statistical model that tries to predict the outcome of a categorical dependent variable (i.e., a class label) [4, 16]. Random Forest (RF) is an ensemble learning method for classification and regression [3]. RF operates by constructing a multitude of decision trees at training time and outputting classification for the case at hand. RF combines the "bagging" idea presented by Breiman [2] and random selection of features introduced by Ho [15] to construct a forest of decision trees.

Tools and information sources: We used the three following tools and information sources:

1. Scikit-Learn[11] - a python library for ML methods and algorithms [25]
2. NLTK [12]- a suite of libraries and programs for symbolic and statistical natural language processing [28]
3. Numpy[13] - a library that performs fast mathematical processing on arrays and metrices [37].

Development corpus: Development corpora in five languages (English, French, Italian, Polish and Spanish). For each language, we have two corpora, which each of them has both train and test sub-corpora.

4.1 Experimental results using content-based feature sets

The first TC experiments were performed for the development corpus using five ML methods without any parameter tuning. The results of the average F1 score for all the

[6] http://scikit-learn.org/stable/modules/neural_networks_supervised.html
[7] The default SVC is with RBF kernel.
http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[8] http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html
[9] http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
[10] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[11] http://scikit-learn.org/stable/index.html
[12] https://www.nltk.org/
[13] http://www.numpy.org/

examined feature sets for the development corpus are shown in Table 1. The best result for each tested ML method (a column) is shown in bold.

Table 1. Average $F1\ score$ results using sets of word multi-grams.

Features	MLP	SVC	LinearSVC	LR	RF
1000 Word Uni-grams	0.4435	**0.4858**	0.5316	0.5504	0.2178
1000 Word Bi-grams	0.3584	0.3425	0.3432	0.3636	0.1574
1000 Word 3-grams	0.2543	0.2431	0.2456	0.2687	0.1384
2000 Word Uni-grams	0.4149	0.4779	0.5359	0.5378	0.2043
2000 Word Bi-grams	0.282	0.3638	0.3888	0.3977	0.1901
2000 Word 3-grams	0.2566	0.2162	0.2541	0.2565	0.0865
3000 Word Uni-grams	**0.4518**	0.469	0.5352	0.5497	0.1967
3000 Word Bi-grams	0.3255	0.3459	0.4056	0.4054	0.1784
3000 Word 3-grams	0.2594	0.2329	0.2649	0.2713	0.0848
4000 Word Uni-grams	0.365	0.4686	0.5538	0.5474	0.1882
4000 Word Bi-grams	0.37	0.3754	0.4194	0.4099	0.1593
4000 Word 3-grams	0.2626	0.259	0.2689	0.2751	0.1121
5000 Word Uni-grams	0.3977	0.4585	0.5479	**0.5607**	0.2004
5000 Word Bi-grams	0.3217	0.4001	0.4185	0.4126	0.1685
5000 Word 3-grams	0.2897	0.2567	0.2675	0.2742	0.1209
7500 Word Uni-grams	0.3357	0.4395	**0.5582**	0.5599	**0.2361**
7500 Word Bi-grams	0.3256	0.4001	0.4421	0.4451	0.1361
7500 Word 3-grams	0.2425	0.3004	0.2839	0.2806	0.0981
10000 Word Uni-grams	0.3347	0.4401	0.5217	0.5551	0.1979
10000 Word Bi-grams	0.304	0.3894	0.4353	0.4234	0.1404
10000 Word 3-grams	0.2558	0.2879	0.3153	0.3088	0.1314

The results shown in Table 1 are quite bad, and the vocabulary size does not seem to have any noticeable effect on the results. We also noticed that the best results were achieved by the word uni-gram feature sets, which is not a surprise according to many previous TC studies. It also interesting to note that although the LinearSVC and SVC methods are both based on the SVM method the differences in the kernel and in the implementation have a big effect on their scores.

The next experiment we conducted was similar, but with characters instead of words. The results of the Average $F1\ score$ are shown in Table 2. The best result for each tested ML method (a column) is shown in bold.

Table 2. Average $F1\ score$ results using sets of char 1-4-grams.

Features	MLP	SVC	LinearSVC	LR	RF
1000 Char Uni-grams	0.4625	0.3275	0.4528	0.4523	0.2879
1000 Char Bi-grams	0.474	0.3841	0.5199	0.5247	0.2514
1000 Char 3-grams	0.5008	0.4429	0.5252	0.519	0.1764
2000 Char Uni-grams	0.4682	0.3275	0.4528	0.4523	0.3079
2000 Char Bi-grams	0.4894	0.3739	0.5357	0.5164	0.242
2000 Char 3-grams	0.4958	0.5042	0.6193	0.6005	0.2073
3000 Char Uni-grams	0.4383	0.3275	0.4528	0.4523	**0.3388**
3000 Char Bi-grams	0.4944	0.3739	0.5357	0.5164	0.2501
3000 Char 3-grams	0.4934	0.5163	**0.6239**	0.5918	0.2
4000 Char Uni-grams	0.4259	0.3275	0.4528	0.4523	0.3132
4000 Char Bi-grams	0.4724	0.3739	0.5357	0.5164	0.2257
4000 Char 3-grams	0.5027	0.4922	**0.6239**	0.5753	0.2439
5000 Char Uni-grams	0.4363	0.3275	0.4528	0.4523	0.2761
5000 Char Bi-grams	**0.5188**	0.3739	0.5357	0.5164	0.2341
5000 Word 3-grams	0.5134	**0.5371**	0.6193	0.5905	0.2283
7500 Char Uni-grams	0.4486	0.3275	0.4528	0.4523	0.3027
7500 Char Bi-grams	0.4815	0.3739	0.5357	0.5164	0.2203
7500 Char 3-grams	0.478	0.5282	**0.6239**	0.6078	0.2197
10000 Char Uni-grams	0.4567	0.3275	0.4528	0.4523	0.256
10000 Char Bi-grams	0.4881	0.3739	0.5357	0.5164	0.1866
10000 Char 3-grams	0.4752	0.5338	0.6193	**0.6175**	0.1841
10000 Char 4-grams	0.4112	0.5385	0.593	0.6026	0.3158

We have also tried many combinations that include various skip character/word n-gram sets. However, their results were quite low.

Next, we tried to eliminate some overfitting by removing hapaxes (i.e., words occurring only once in the text) or removing any words that appear less than four times in the corpus. The results of these experiments were, again, quite low.

Since the best results we have seen so far were obtained by using char 3-gram and 4-gram features using Linear SVC and Logistic Regression, we decided to try n-gram features with higher numbers of characters in the next experiments. In addition, we noticed that the results were significantly higher when we used a big vocabulary size (i.e., more features), therefore we also tried to use all the char-grams in the corpus. The results of this experiment are shown in Table 3. The best result for each tested ML method (a column) is shown in bold.

Table 3. Average $F1\ score$ results using sets of characters 5 to 8 -grams.

Features	L-SVC	LR	Features	L-SVC	LR
8000 Char 5-grams	0.6122	0.5991	12000 Char 7-grams	0.634	0.6272
8000 Char 6- grams	0.6038	0.6078	12000 Char 8-grams	0.6113	0.621
8000 Char 7-grams	0.6011	0.6107	13000 Char 5-grams	0.6224	0.607
8000 Char 8-grams	0.5686	0.5604	13000 Char 6-grams	0.6452	0.6562
9000 Char 5-grams	0.6126	0.6052	13000 Char 7-grams	0.6368	0.636
9000 Char 6-grams	0.6247	0.6189	13000 Char 8-grams	0.6265	0.6288
9000 Char 7-grams	0.615	0.6239	14000 Char 5-grams	0.6256	0.6148
9000 Char 8-grams	0.5759	0.5768	14000 Char 6-grams	0.6468	0.6466
10000 Char 5-grams	0.6159	0.6081	14000 Char 7-grams	0.6293	0.6382
10000 Char 6-grams	0.6321	0.6314	14000 Char 8-grams	0.6235	0.6219
10000 Char 7-grams	0.6277	0.6298	15000 Char 5-grams	0.622	0.6134
10000 Char 8-grams	0.5684	0.585	15000 Char 6-grams	0.6544	0.652
11000 Char 5-grams	0.6392	0.6142	15000 Char 7-grams	0.6185	0.6344
11000 Char 6-grams	0.6582	0.6434	15000 Char 8-grams	0.6168	0.6246
11000 Char 7-grams	0.6348	0.6328	all Char 5-grams	0.6622	**0.6693**
11000 Char 8-grams	0.5957	0.5889	all Char 6-grams	**0.6752**	0.6551
12000 Char 5-grams	0.6351	0.6122	all Char 7-grams	0.6378	0.6417
12000 Char 6-grams	0.6338	0.6434	all Char 8-grams	0.6219	0.6306

4.2 Experimental results using style-based feature sets

We also tried to add various style-based feature sets (POS-tags, Quantitative features, Orthographic features and lexical richness features) in our feature combinations. However, most of these feature sets did not improve the average f1-score, except POS-tags. We used the frequencies of POS-tag sequences as features in addition to Char 6-gram features, which produced the best result so far. By using POS-tags we increased our average f1-score, but after a closer look we noticed that the results of the datasets in Spanish and Polish got worse. We guess that has been caused by an inaccurate tagging of the text (the Polish POS-tagger was just a UniPOS-tagger and not a fine-grained one). Therefore, we ran our tests again, using POS tagger only for the datasets in English, French, and Italian. The results are shown in Table 4.

We also applied the Principal component analysis (PCA) [20] algorithm using the Scikit-Learn's PCA module[14]. According to this model, we filtered out the least significant features. The results improved, but again, in closer look we noticed that the result got worse in Spanish and Polish, maybe due to the lack of the POS-tags features. Therefore, we decided not to use PCA for those languages. The results are shown in Table 5.

[14] http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

Table 4. Average $f1 - score$ for style features combined with 6-chars.

Features	Linear SVC
all_6-char_linear_svc	0.6752
all_6-char_orth_linear_svc	0.681
all_6-char_orth_quan_linear_svc	0.6792
all_6-char_orth_quan_rich_linear_svc	0.6792
all_6-char_orth_rich_linear_svc	0.6791
all_6-char_pos_linear_svc	**0.6917**
all_6-char_pos_orth_linear_svc	0.69
all_6-char_pos_orth_quan_linear_svc	0.6903
all_6-char_pos_orth_quan_rich_linear_svc	0.689
all_6-char_pos_orth_rich_linear_svc	0.6872
all_6-char_pos_quan_linear_svc	0.6903
all_6-char_pos_quan_rich_linear_svc	0.6903
all_6-char_pos_rich_linear_svc	0.6902
all_6-char_quan_linear_svc	0.6741
all_6-char_quan_rich_linear_svc	0.6745
all_6-char_rich_linear_svc	0.6737

Table 5. Average $f1 - score$ for 6-chars with style-features and PCA applied.

Features	Linear SVC
all_6-char_pca_linear_svc	0.6775
all_6-char_orth_pca_linear_svc	0.6813
all_6-char_orth_quan_pca_linear_svc	0.6818
all_6-char_orth_quan_rich_pca_linear_svc	0.6818
all_6-char_orth_rich_pca_linear_svc	0.6798
all_6-char_pos_orth_pca_linear_svc	**0.7234**
all_6-char_pos_orth_quan_pca_linear_svc	0.722
all_6-char_pos_orth_quan_rich_pca_linear_svc	0.723
all_6-char_pos_orth_rich_pca_linear_svc	0.7219
all_6-char_pos_pca_linear_svc	0.7224
all_6-char_pos_quan_pca_linear_svc	0.721
all_6-char_pos_quan_rich_pca_linear_svc	0.721
all_6-char_pos_rich_pca_linear_svc	0.7209
all_6-char_quan_pca_linear_svc	0.6778
all_6-char_quan_rich_pca_linear_svc	0.6794
all_6-char_rich_pca_linear_svc	0.676

4.3 Experimental results for the final three models

For the final competition we sent three models, two of which were sent by miller18 and another one by yigal18.

The first model, sent by miller18, was based on the results we achieved on the development corpus. Its features consist of the frequencies of all char 6-gram sequences, POS-tags sequences frequencies, Orthographic features, Quantitative features, and Lexical richness features. Then, we applied PCA and Linear SVC, which was the best ML method on the development corpus. We tried several parameter combinations for the classifier using GridSearchCV[15], including ‘C’ (Penalty parameter of the error term) values between 0.001 to 10 and ‘tol’ (Tolerance for stopping criteria) values between 1e-7 to 1 but the results did not improve, so we sent the model with the default parameters. This model scored an average of 0.582 on the evaluation corpus.

For the second model, sent by yigal18, we tried to use all the char sequences of length between 3 to 8. We also tried to add all the unigram frequencies to the feature sets in addition to the stylistic features from the first model. Again, we applied PCA and Linear SVC as our classifier with the default parameters. This model scored an average 0.598 on the evaluation corpus.

For the third model, which is the second model sent by miller18, we used the same content features as the second model, but this time we did not used any style-based features at all except for POS-tags. The PCA and the classifier were the same as the other models. This model scored an average of 0.611 on the evaluation corpus. It is important to note that the organizers of this PAN’s task did not accept this model as a formal one because the results of tournament were already published. However, they allowed us to write about this model and its results in our notebook paper. In table 6, we present a comparison of these three models including the average $F1\ score$ results for the development and evaluation corpora.

Table 6. A comparison of the three models sent to the competition.

Model	Features	Development Corpus Score	Evaluation Corpus Score
Model 1 (miller18)	all_6-char_pos_orth_quan_rich	0.723	0.582
Model 2 (yigal18)	all_3-8-char_uni-grams_pos_orth_quan_rich	0.718	0.598
Model 3 (“new” miller18)	all_3-8-char_Uni-grams_pos	0.724	0.611

The most striking and surprising finding in Table 6 is that there is no direct connection between the results of the three models on the development corpus to their results on the evaluation corpus. Furthermore, the results on the evaluation corpus are significantly lower, which implies that the evaluation corpus is quite different in its nature from the development corpus. Another possible explanation is that the three models have overfitted and are not optimal for general corpora from this type.

5 Summary and Future Work

In this paper, we describe our participation in the PAN 2018 shared task on author identification. We tried various pre-processing types, a widespread variety of feature sets, and five ML methods.

For the evaluation corpus, we sent the top three models according to their results on the development corpus. The first model was sent by miller18. Its features consist of

[15] http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

the frequencies of all char 6-gram sequences, POS-tags sequences frequencies, Orthographic features, Quantitative features, Lexical richness features. Then we applied PCA. The best ML method on the development corpus was the Linear SVC. This model scored an average of 0.582 on the evaluation corpus.

The second model was sent by yigal18. Its features consist of all the char sequences of length between 3 to 8. We also tried to add all the unigram frequencies to the feature sets in addition to the stylistic features from the first model. Again, we applied PCA and Linear SVC as our classifier with the default parameters. This model scored an average 0.598 on the evaluation corpus (an improvement of 0.016 comparing to the first model).

The third model, which is the second model sent by miller18, was composed of the same content-based features as the second model, but this time we did not use any style-based features at all except for POS-tags. The PCA and the classifier were the same as the other models. This model scored an average result of 0.611 (an improvement of 0.029 comparing to the first model).

The main findings of our experiments are: (1) there is no direct connection between the results of the three models on the development corpus to their results on the evaluation corpus and (2) the results on the evaluation corpus are significantly lower, which implies that the evaluation corpus is quite different in its nature from the development corpus. A possible explanation is that the three models are too overfitting and are not optimal for general corpora from this type.

Future research proposals include: (1) applying additional combinations of feature sets in order to find a general enough model; (2) tuning the models' parameters; (3) applying various deep neural models; (4) defining and applying skip-POS sequences; and (5) building model that will perform authorship attribution using key phrases [10, 11] that distinguish each of the text authors.

Acknowledgments. This work was partially funded by the Jerusalem College of Technology (Lev Academic Center) and we gratefully acknowledge its support.

References

1. Argamon, S., Whitelaw, C., Chase, P., Hota, S. R., Garg, N., Levitan, S.: Stylistic text classification using functional lexical features: Research articles. Journal of the Amer. Society for Info. Science and Technology. 58, 6, 802–822 (2007).
2. Breiman, L.: Bagging predictors. Univ. California Technical Report No. 421. (1994).
3. Breiman, L.: Random forests. Machine learning, 45(1), 5-32 (2001).
4. Cessie, S. Le, Van Houwelingen, J. C.: Ridge estimators in logistic regression, Applied statistics, pp. 191-201 (1992).
5. Chang, C. C., Lin, C. J.: LIBSVM: a library for support vector machines. ACM transactions on intelligent systems and technology (TIST), 2(3), 27 (2011).
6. Cortes, C., Vapnik, V.: Support-vector networks. Machine learning, 20(3), 273-297 (1995).

7. Fan, R. E., Chang, K. W., Hsieh, C. J., Wang, X. R., Lin, C. J.: LIBLINEAR: A library for large linear classification. Journal of machine learning research, 9(Aug), 1871-1874 (2008).
8. Fox, C.: A stop list for general text. In Acm sigir forum (Vol. 24, No. 1-2, pp. 19-21). ACM (1989).
9. Gianfortoni, P., Adamson, D., Rosé, C. P.: Modeling of stylistic variation in social media with stretchy patterns. In Proceedings of the First Workshop on Algorithms and Resources for Modelling of Dialects and Language Varieties (pp. 49-59). Association for Computational Linguistics (2011).
10. HaCohen-Kerner, Y., Gross, Z., Masa, A.: Automatic extraction and learning of keyphrases from scientific articles. In Int. Conf. on Intelligent Text Processing and Computational Linguistics, Springer, Berlin, Heidelberg, pp. 657-669 (2005).
11. HaCohen-Kerner, Y., Stern, I., Korkus, D., Fredj, E.: Automatic machine learning of keyphrase extraction from short html documents written in Hebrew. Cybernetics and Systems: An International Journal, 38(1), 1-21 (2007).
12. HaCohen-Kerner, Y., Mughaz, D., Beck, H., Yehudai, E.: Words as classifiers of documents according to their historical period and the ethnic origin of their authors. Cybernetics and Systems: An International Journal, 39(3), 213-228 (2008).
13. HaCohen-Kerner, Y., Beck, H., Yehudai, E., Rosenstein, M., Mughaz, D.: Cuisine: Classification using stylistic feature sets &/or name-based feature sets. Journal of the American Society for Information Science and Technology 61 (8), 1644–57 (2010A).
14. HaCohen-Kerner, Y., Beck, H., Yehudai, E., Mughaz, D.: Stylistic feature sets as classifiers of documents according to their historical period and ethnic origin. Applied Artificial Intelligence 24 (9), 847–62 (2010B).
15. Ho, T. K.: Random Decision Forests. Proc. of the 3rd Int. Conf. on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. 278–282 (1995).
16. Hosmer Jr, D. W., Lemeshow, S., Sturdivant, R. X.: Applied logistic regression (Vol. 398). John Wiley & Sons (2013).
17. Jain, A. K., Mao, J., Mohiuddin, K. M.: Artificial neural networks: A tutorial. Computer, 29(3), 31-44 (1996).
18. Joachims, T.: Text categorization with support vector machines: Learning with many relevant features. In European conference on machine learning (pp. 137-142). Springer, Berlin, Heidelberg (1998).
19. Jockers, M. L., Witten, D. M.: A comparative study of machine learning methods for authorship attribution. Literary and Linguistic Computing, 25(2), 215-223 (2010).
20. Jolliffe, I. T.: Principal component analysis and factor analysis. In Principal component analysis (pp. 115-128). Springer, New York, NY (1986).
21. Kessler, Brett, Nunberg, Geoffrey, Hinrich Schutze.: Automatic detection of text genre. In P. R. Cohen and W. Wahlster (Eds.), In Proc. of the 35th annual meeting of the ACL and 8th conf. of the Eur. chapter of the As. for Comp. Linguistics. 32-38, Somerset, New Jersey: Association for Computational Linguistics (1997).
22. Kestemont, M., Tschugnall, M., Stamatatos, E., Daelemans, W., Specht, G., Stein, B., Potthast, M.: Overview of the Author Identification Task at PAN-2018: Cross-

domain Authorship Attribution and Style Change Detection. In: Cappellato, L., Ferro, N., Nie, J.Y., Soulier, L. (eds.) Working Notes Papers of the CLEF 2018 Evaluation Labs. CEUR Workshop Proceedings, CLEF and CEUR-WS.org (2018)

23. Koppel, M., Argamon, S., Shimoni, A. R.: Automatically categorizing written texts by author gender. Literary and linguistic computing, 17(4), 401-412 (2002).
24. Liparas, D., HaCohen-Kerner, Y., Moumtzidou, A., Vrochidis, S., Kompatsiaris, I. News articles classification using Random Forests and weighted multimodal features. In Information Retrieval Facility Conf. (pp. 63-75). Springer, Cham. (2014).
25. Loper, E., Bird, S.: NLTK: The natural language toolkit. In Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics-Volume 1 (pp. 63-70). Association for Computational Linguistics (2002).
26. Meretakis, D., Wuthrich, B.: Extending naive Bayes classifiers using long itemsets, Proc. of the 5th ACM-SIGKDD Int. Conf. Knowledge Discovery, Data Mining (KDD'99), San Diego, USA, 165-174 (1999).
27. Mosteller, F., and Wallace, D. L.: Applied Bayesian and classical inference: the case of the Federalist papers. Addison-Wesley: Reading, MA (1984).
28. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Vanderplas, J.: Scikit-learn: Machine learning in Python. Journal of machine learning research, 12(Oct), 2825-2830 (2011).
29. Potthast, M., Gollub, T., Rangel, F., Rosso, P., Stamatatos, E., Stein, B.: Improving the Reproducibility of PAN's Shared Tasks: Plagiarism Detection, Author Identification, and Author Profiling. In: Kanoulas, E., Lupu, M., Clough, P., Sanderson, M., Hall, M., Hanbury, A., Toms, E. (eds.) Information Access Evaluation meets Multilinguality, Multimodality, and Visualization. 5th International Conference of the CLEF Initiative (CLEF 14). pp. 268–299. Springer, Berlin Heidelberg New York (2014).
30. Rangel, F., Rosso, P., Potthast, M., Stein, B.: Overview of the 5th author profiling task at pan 2017: Gender and language variety identification in twitter. Working Notes Papers of the CLEF (2017).
31. Schler, J., Koppel, M., Argamon, S., Pennebaker, J. W.: Effects of age and gender on blogging. In AAAI spring symposium: Computational approaches to analyzing weblogs(Vol. 6, pp. 199-205) (2006).
32. Sebastiani, F.: Machine learning in automated text categorization. ACM computing surveys (CSUR), 34(1), 1-47 (2002).
33. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014).
34. Stamatatos, E.: Authorship attribution based on feature set subspacing ensembles. International Journal on Artificial Intelligence Tools, 15(05), 823-838 (2006).
35. Stamatatos, E.: A survey of modern authorship attribution methods. Journal of the Association for Information Science and Technology, 60(3), 538-556 (2009).
36. Stamatatos, E., Rangel, F., Tschuggnall, M., Kestemont, M., Rosso, P., Stein, B., Potthast, M.: Overview of PAN-2018: Author Identification, Author Profiling, and Author Obfuscation. In: Bellot, P., Trabelsi, C., Mothe, J., Murtagh, F., Nie, J.,

Soulier, L., Sanjuan, E., Cappellato, L., Ferro, N. (eds.) Experimental IR Meets Multilinguality, Multimodality, and Interaction. 9th Int. Conference of the CLEF Initiative (CLEF 18). Springer, Berlin Heidelberg New York (2018).

37. Walt, S. V. D., Colbert, S. C., Varoquaux, G.: The NumPy array: a structure for efficient numerical computation. Computing in Science & Engineering, 13(2), 22-30 (2011).