An Ensemble-Rich Multi-Aspect Approach for Robust Style Change Detection

Notebook for PAN at CLEF-2018

Dimitrina Zlatkova*[1], Daniel Kopev*[1], Kristiyan Mitov*[1], Atanas Atanasov*[1], Momchil Hardalov[1], Ivan Koychev[1], and Preslav Nakov[2]

[1] FMI, Sofia University "St. Kliment Ohridski", Sofia, Bulgaria
{dvzlatkova, dkopev, kmmitov, amitkov}@uni-sofia.bg
{hardalov, koychev}@fmi.uni-sofia.bg
https://www.uni-sofia.bg/
[2] Qatar Computing Research Institute, HBKU, Doha, Qatar
pnakov@qf.org.qa

Abstract. We describe the winning system for the PAN@CLEF 2018 task on Style Change Detection. Given a document, the goal is to determine whether it contains style change. We present our supervised approach, which combines a TF.IDF representation of the documents with features specifically engineered for the task and which makes predictions using an ensemble of diverse models including SVM, Random Forest, AdaBoost, MLP and LightGBM. We further perform comparative analysis on the performance of the models on three different datasets, two of which we have developed for the task. Moreover, we release our code in order to enable further research.

Keywords: Multi-authorship · Stylometry · Style change · Stacking ensemble · Natural Language Processing· Gradient boosting machines· Deep Learning

1 Introduction

While there has been a lot of research on authorship attribution for full-size documents, the problem of identifying multi-authorship is much less explored. Previous editions of the PAN competition have shown that tasks such as finding clusters of authors inside a document or spotting the exact locations of style change are too challenging [13, 15, 16]. Thus, the 2018 edition of PAN offered a simplified version of these problems: given a text document, identify whether there is style change in it or not. This formulation is related to intrinsic plagiarism detection, but an important difference is that multi-authorship assumes uniform distribution of text segments by different authors.

Most previous work used features based on stylometry [6, 8, 14] and term frequencies [4, 8], and so did we. In particular, we borrowed ideas for features from [2, 12], but we also crafted some new ones as described in Section 3.3 below.

*Equal contribution

2 Data

This section describes the data from the competition as well as two additional datasets that we created and used.

2.1 PAN Dataset

The data provided by the PAN organizers was based on user posts from StackExchange covering different topics with 300–1,000 tokens per document. It included a training set of 3,000 documents and a validation set of 1,500 documents. The submissions by the participants were tested on a testing dataset of 1,500 documents. All three datasets were balanced between the two classes: style change vs. no style change.

We found the training data too small in size, and thus we compiled and used two additional datasets, which we describe below.

2.2 External Movie Reviews Dataset

We created an additional training dataset based on a pre-existing Amazon movie reviews dataset [11], which consists of eight million reviews. We merged pairs of reviews for the same movie written by different authors, making sure that the combined length of the merged documents would be 2,800–5,500 characters. In particular, we aimed for the character length of the merged documents to be as close as possible to 4,150. For comparison, the training dataset provided by the PAN organizers had an average character length of 4,300. We ended up with 216,250 new examples with balanced classes. However, training on this new dataset yielded poor results, possibly due to our naïve concatenation of complete single-author statements, which often yielded locally incoherent documents.

2.3 External StackExchange Dataset

In another attempt to extend our training dataset, we collected posts from StackExchange. We used the StackExchange Data Explorer[3] to execute an SQL query against the databases of 35 sites from the StackExchange network. The query aimed at collecting all posts with at least three answers with a character length of 300 or more and at least one answer with a character length of at least 1,000. We then used a script to combine some of the answers (taking length into account) for each post while maintaining the original distribution of the number of splits from the PAN dataset. We ended up with 50,000 new examples covering a wide range of topics. This dataset also yielded poor results, possibly for the same reasons as above.

[3] `https://data.stackexchange.com/`

3 Method

In this section, we first describe how we pre-process the data, then we discuss the importance of text segmentation, and we present the features we extract from text. Finally, we describe the supervised models we use for classification.

3.1 Data Preprocessing

We pre-process the input documents in two phases. The first phase is applied before any feature extraction takes place and replaces URLs and long numbers with specific tokens. The second phase is applied during feature computation. It filters the stream of words and replaces file paths, long character sequences and very long words with special tokens. Additionally, an attempt is made to split long hyphenated words (with three or more parts) by checking whether most of the sub-words are present in a dictionary of common words (from the NLTK words corpus [10]). The goal of all these steps is to reduce the impact of long words, which could adversely affect features that take word length into account. Such features are those from the lexical group and this pre-processing is applied to them only since it might have unexpected effect on the rest of the features.

3.2 Text Segmentation

Style change in text documents entails that parts of the text will differ in some respect. In an attempt to spot such differences automatically, we split the document into three equal segments of words, we calculate the feature vectors for each of them, and we find the maximum difference per feature between any pair of segments. We choose the number of segments based on the prior distribution of the number of style changes across the entire dataset. In order to obtain more data points, we apply a sliding window for each document. We use this segmentation procedure for four of the feature groups, three of which use a sliding window. See Section 3.3 for more detail.

3.3 Text Representation

Below we describe the features we engineered specifically for the task of discovering style changes.

3.3.1 Repetition Repetition can be an important feature for the task. We account for it by looking at the average number of occurrences of unigrams, bigrams, ..., 5-grams in the document, and we create a vector of five features with the respective averages.

3.3.2 Contracted Wordforms Another feature we use is based on the discrepancies in spelling for words that allow contracted forms, e.g., *I will (I'll)*, *are not (aren't)*, *they are (they're)*. People typically favor one of the alternatives, and thus we use forms based on contracted apostrophes as discriminative features for identifying whether a piece of text is single- or multi-authored.

3.3.3 Frequent Words Frequent words are known stylometric indicators for authorship attribution, and are thus useful for the present task as well. We include stop words (from the NLTK stopwords list [10]) and function words (compiled from three separate lists[4,5,6]). We use each such frequent word as a feature, and its frequency of occurrence in the text segment as the feature value.

3.3.4 Lexical We use as features the proportion of various types of lexical elements per text segment, computed using a sliding window:

Character-based These include spaces, digits, commas, colons, semicolons, apostrophes, quotes, parentheses, number of paragraphs and punctuation in general.

Word-based We POS-tag each segment using NLTK, and we extract pronouns, prepositions, coordinating conjunctions, adjectives, adverbs, determiners, interjections, modals, nouns, personal pronouns, and verbs. Other word-based features include words of length 2 or 3, words with over 6 characters, as well as average word length, number of all-caps words and of capitalized words.

Sentence-based This includes question marks, periods, exclamation sentences, short and long sentences, and average sentence length.

3.3.5 Quotation Marks Some authors may prefer either single or double quotation marks. We use the difference between the number of single and double quotes in a given segment as a single feature.

3.3.6 Vocabulary Richness Similarly to [2], we represent vocabulary richness as an averaged word frequency class. We use the Google Books common words list[7] to compute the frequency class of a word x as $\log_2 \frac{f(X)}{f(x)}$, where f is the frequency function and X is the most frequent word in the corpus (in our case, this is the word *the*). We then extract two features per segment: the average frequency class of all words in it and the proportion of unknown word tokens (words not present in the common words list).

3.3.7 Readability We compute the following readability features per text segment, using the *Textstat*[8] Python package: Flesch reading ease, SMOG grade, Flesch-Kincaid grade, Coleman-Liau index, automated readability index, Dale-Chall readability score, difficult words, Linsear write formula, and Gunning fog.

[4] `https://semanticsimilarity.files.wordpress.com/2013/08/jim-oshea-fwlist-277.pdf`

[5] `http://www.sequencepublishing.com/1/academic.html`

[6] `https://www.edu.uwo.ca/faculty-profiles/docs/other/webb/essential-word-list.pdf`

[7] `http://norvig.com/google-books-common-words.txt`

[8] `https://github.com/shivam5992/textstat`

Word	Normalized Frequency
however	1.00
one	0.96
note	0.92
edit	0.87
first	0.69
yes	0.63
also	0.60
another	0.50
finally	0.40
since	0.37
update	0.35
let	0.31

(a) beginning

Word	Normalized Frequency
etc	1.00
time	0.95
well	0.76
question	0.68
way	0.61
god	0.58
p	0.45
example	0.45
work	0.39
war	0.39
though	0.37
answer	0.37

(b) end

Table 1: The most frequent beginning and ending words, excluding stopwords.

3.3.8 Beginning and Ending of Author Statements As can be seen in Table 1, author statements begin and end with quite different words. This can be used to locate points in documents, where word clusters of small size contain high concentration of beginning/ending terms, which could indicate a change of author. We experimented with word n-grams of length 1, 2 and 3, but we eventually found that single words, after stopword removal, yielded the best results. We experimented with two approaches. Our first approach assigns a score to each word type based on the number of times it appears at the beginning or at the end of a segment by the same author (these counts are min-max normalized). The second approach scores words based on how close they are to such a position. It transforms the frequency using a steep half-sigmoid function (see Equation 1, where k denoting the steepness), taking the relative position and rewarding words that are extremely close to the beginning or to the end of a segment by the same author. Then, each word list of position scores is averaged across all documents. Finally, we extract a vector of features for each document by looking at local document clusters of three words that contain multiple high-scored words, which could indicate that there is an end of a statement by one author immediately followed by a beginning of a statement by a different author.

$$x = \frac{|\frac{statementLength}{2} - (position + 1)|}{\frac{statementLength}{2}}$$
$$Score(position_{\text{statement}}) = \frac{(0 + k) * x}{(1 + k) - x} \tag{1}$$

We did not use this document representation as part of our stacking classifier (Section 3.6); yet, it works very well in isolation, yielding 65% accuracy.

3.3.9 Named Entity Spelling Next, we model personal preference for different spelling variants for the same named entity. In particular, we use the Damerau-Levenshtein string edit distance [1, 9] to find inconsistencies in the wording of identical named entities within an edit distance of 1. The feature vector consists of the minimum counts between the different spellings for each found named entity.

3.3.10 Other Features In the process of feature engineering, we explored many other ideas, some of which performed poorly and thus were not included in the final system. Yet, we feel some of them are worth mentioning:

Preprocessing tokens We used the normalized frequency for each token that was used to substitute URLs and other long sequences of characters (see Section 3.1).

British/American English spelling We looked for American vs. British spelling of 170 words,[9] which could signal change of author/style.

ASCII characters Extensive non-ASCII characters use could in theory find region-based regularities, but the short document length and the resulting infrequency of such cases made this useless as a feature.

3.4 Deep Learning

We experimented with deep neural networks. Note that the performance of such networks depends on the nature of the task, the architecture of the network, and the selection of hyper-parameters. For tasks where feature detection in text is important, CNNs work well and can outperform RNNs [7].

We use a CNN, which consists of a layer computed over an 11-word sentence, with 300-dimensional word embeddings, a Squeeze-and-Excitation block [3], and two additional dense layers. We trained the embeddings from scratch. The vocabulary size is the combined vocabulary size of the training and of the testing sets: about 70,000 words.

For regularization, we used dropout on each layer with no constraint on the L2-norms of the weight vectors. This consistently added 6–8 percentage points in terms of accuracy. We used the ReLU activation function, 64 filters of size 11, a dropout rate of 0.1, and a mini-batch optimization of size 256.

The Squeeze-and-Excitation convolutional block improves channel interdependencies at almost no additional computational cost. It allows the network to adaptively adjust the weighting of each feature map. We add a max pooling layer on top of this block.

Although we did not tune the hyper-parameters extensively, this relatively simple CNN performed remarkably well, achieving accuracy of about 86%, even on the comparatively small training dataset.

Unfortunately, we explored and developed this CNN architecture very close to the competition deadline, and thus we did not use it in the final submission.

[9] `https://en.wikipedia.org/wiki/Wikipedia:List_of_spelling_variants`

Classifier	Hyper-parameter	Value
Support Vector Machine	kernel	Radial basis function
	penalty C	1.0
	tolerance	0.001
Random Forest	estimators	300
	with replacement	Yes
AdaBoost Trees	base estimator	Decision tree
	estimators	300
Multi-layer Perceptron	layers	1
	layer size	100
	activation	ReLU
	optimization	Adam
	regularization	L2
	regularization term	0.0001
	learning rate	0.001
	mini-batch size	200
	maximum iterations	10000
LightGBM	learning rate	0.1
	number of leaves	31
	metric'	AUC
	bagging fraction	0.8
	feature selection	0.6
	l1 regularization term	1.0
	l2 regularization term	1.0
	min data in leaf	100
Logistic Regression (meta-classifier)	optimization	liblinear
	regularization	L2
	penalty C	1.0
	tolerance	0.0001
	maximum iterations	100

Table 2: Stacking meta and zero-level classifier hyper-parameters.

3.5 LightGBM

Our Gradient Boosting Approach is based on combining LightGBM [5] with Logistic Regression and TF.IDF, which we fit on both the testing and the training data.[10] Fitting TF.IDF on the test data gives us a way to capture some properties and insights about the word distribution. Then, we used the SelectFromModel[11] meta-transformer with a Logistic Regression estimator to select the best TF.IDF features before passing them to the LightGBM model. We tuned the parameters of the Logistic Regression model using cross-validation, and we selected the sag solver and $C = 2$. Moreover, we only used features with weight of more than 0.3.

A simple LightGBM baseline achieved 73% accuracy on the PAN validation set. Tuning its parameters increased the accuracy to 86%, which was supported by a cross-validation score of 85%. The final parameters can be seen in Table 2. Our main goal during tuning was to prevent overfitting. With that in mind, we will look at the parameters in detail. Adjusting the *learning_rate* was crucial for increasing the accuracy. Throughout all our experiments, 0.1 was the best choice for the learning rate. Robust regularization also affected accuracy: setting both L1 and L2 regularization to 1.0 boosted it by 3–4 points absolute. Another strategy that we used to handle overfitting was setting $min\_data\_in\_leaf$ to 100, which improved accuracy by another 3 points. Setting $feature\_selection$ to 0.6 means LightGBM will select 60% of the parameters randomly in each iteration for building trees, which increased the training time, but decreased overfitting. The fraction of data to be used for each iteration, generally controlling the generalization of our model, was set to 0.8, boosting it by 2–3 points.

The model was trained using bagging with five folds and we used a custom callback to measure the relative accuracy of each bagged fold. For better monitoring and another perspective to look at, we added AUC as a metric in our LightGBM model.

3.6 Stacking

The basic idea behind our Stacking Ensemble classifier is to take into account different independent points of view in the context of distinguishing multi-authored documents and to learn dependencies between them. At the bottom level, we train four different non-linear classifiers (described in Table 2) for each feature vector derived from the representations in Sections 3.3.1 to 3.3.7 on 75% of the training data. Then, each classifier makes a prediction for the remaining 25% of the data and assigns a weight, based on its confidence. These groups form a single vector each with prediction class probabilities, based upon the weights and the outputs of the classifiers. These vectors, together with the predictions of the LightGBM classifier (Section 3.5), serve as an input to a simple linear Logistic Regression meta-learner. The process of training is visualized in Figure 1.

Before predicting, each classifier is trained again on the whole dataset (except the LightGBM model, which is trained once and not weighted across a group).

[10] This is not cheating, as we have no access to the gold labels for the testing dataset.
[11] http://scikit-learn.org/

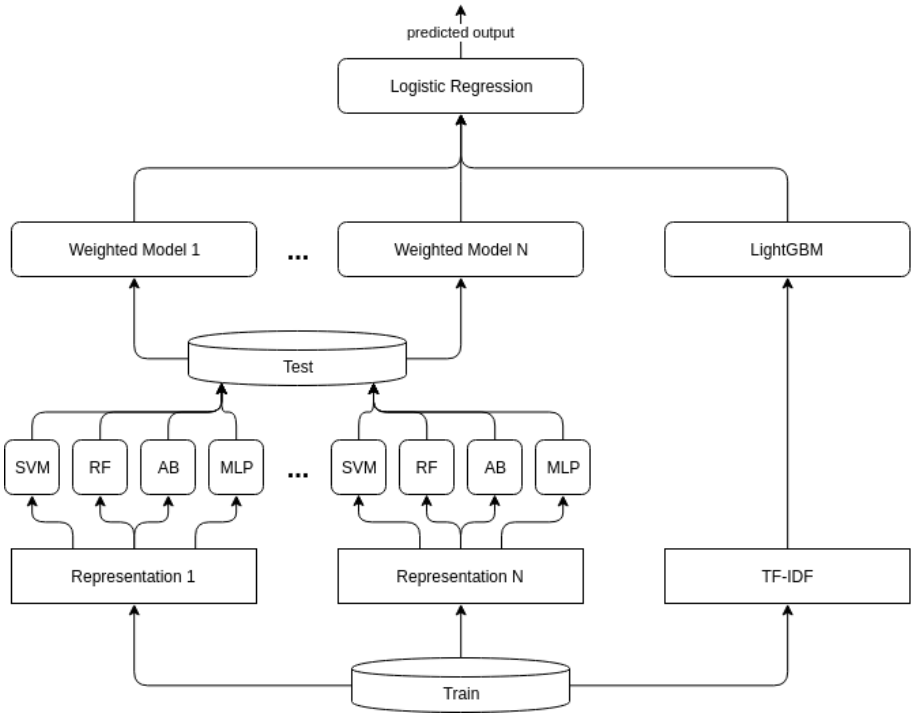

Fig. 1: The architecture of the stacking classifier.

4 Experiments and Evaluation

Below, we discuss the setup for our experiments and the achieved results.

4.1 Setup

During the experiments and the tuning of the models, we only used the training dataset from PAN using 5-fold cross-validation. We consulted the validation dataset only at the end, i.e., for scoring. To make use of all the available data, for the final submission, we trained the stacking classifier on the combination of the training and of the validation datasets.

4.2 Results

Table 4 shows some notable results from our experiments. As mentioned in Sections 2.2 and 2.3, training on a combination of documents from the PAN dataset and from an external source and then testing on the PAN validation dataset was not successful, and thus we do not report these results here.

Instead, we experimented with training and testing on the same external dataset to validate the performance of our approaches in other domains and on data that was collected differently. We performed experiments on subsets of our two external datasets that match closely the setup on the original PAN dataset: training on 3k documents and testing on 1.5k from the respective source. The evaluation results show promising accuracy, which goes up to 95% for the movie reviews dataset.

The confusion matrix of the Stacking with LightGBM model (Table 5) shows that the model misses a style change more often than misclassifying a document written by one author.

5 Conclusion

Our experiments have shown that it is possible for machine learning algorithms to achieve good performance for this problem. We present our supervised approach, which combines a TF.IDF representation of the documents with features specifically engineered for the task and which makes predictions using an ensemble of diverse models including SVM, Random Forest, AdaBoost, MLP and LightGBM. We further performed comparative analysis on the performance of the models on three different datasets. Our LightGBM model was able to tap this potential by utilizing feature selection and gradient boosting to learn which combinations of such features are most indicative of multi-authorship. Furthermore, the features we engineered to detect style breaches, while not discriminative enough on their own, yielded good performance when combined in our stacking model. Finally, adding the LightGBM classifier to the stacking scheme yielded the highest accuracy and outperformed the submissions by the other teams who participated in the Style Change Detection task. The strength of the resulting ensemble lies in the diversity of the combined models and it performed well on both the PAN datasets and on the external datasets we tested it on.

6 Future Work

There are several observations we intend to follow up on. It is worth pointing out that for the PAN dataset in particular, achieving top-notch performance required vectorizing the documents using TF.IDF on both the training and the testing datasets, which seems to provide additional insight and always yielded higher accuracy. However, the improvement we see for the PAN dataset is much larger (around 20% absolute) compared to the one achieved when testing on the external data (<1%), which is something we intend to investigate further. Our CNN approach is promising as well, but it is much more computationally intensive and we were unable to run enough experiments with it. It would be interesting to evaluate its performance more thoroughly in the future. Moreover, we noticed that while our top model performed well when trained on part of one of our datasets and tested on another part of the same dataset, training on one dataset and testing on another one yielded much lower results. Indeed, we can see in Table 3 that the top TF.IDF features learned on the datasets have very little overlap. Another aspect not fully explored are features established on the possibly contrasting views of text projection by the authors. Last but not least, we have made our code available in order to facilitate further research.[12]

Acknowledgements. This work was supported by Project UNITe BG05M2OP001-1.001-0004 funded by the Operational Program "Science and Education for Smart Growth", co-funded by the EU through the ESI Funds.

Feature	Score
'considered'	0.5708
'e'	0.5571
'answers'	0.5168
'?'	0.5089
'questions'	0.6428
'i have'	0.4836
'individuals'	0.4816
'class'	0.4669
'at all'	0.4655
'it for'	0.4535
'e "'	0.4344
'i e'	0.4336

(a) PAN

Feature	Score
'this movie'	0.8485
'begins'	0.8441
'version'	0.8368
'find'	0.7964
'reviews'	0.7313
'movie'	0.7035
'become'	0.6797
'young'	0.6675
'i was'	0.6613
'first'	0.65
'two'	0.6474
'saw'	0.6458

(b) Ext. Movie Reviews

Feature	Score
'http'	1.2028
'com'	1.0426
'i would'	0.8522
'http www'	0.8413
'www'	0.8257
'would be'	0.7778
'etc'	0.7603
'would'	0.7598
'hand'	0.7589
'use'	0.7557
'most'	0.7393
'current'	0.7063

(c) Ext. StackExchange

Table 3: Top 12 TF.IDF features per dataset.

[12] `https://github.com/machinelearning-su/style-change-detection`

Classifier	Dataset	Fit on validation	Accuracy
MLP with TF-IDF (Baseline)	PAN	No	67.09
	PAN	Yes	70.64
	External StackExchange	No	61.36
	External StackExchange	Yes	63.49
	External Movie Reviews	No	64.62
	External Movie Reviews	Yes	66.08
LightGBM	PAN	No	67.16
	PAN	Yes	86.53
	External StackExchange	No	80.67
	External StackExchange	Yes	81.47
Stacking with LightGBM	PAN	No	67.43
	PAN	Yes	87.00
	External StackExchange	No	81.33
	External StackExchange	Yes	81.73
	External Movie Reviews	No	95.00
	External Movie Reviews	Yes	95.13
CNN	PAN	Yes	85.92
Stacking	PAN	No	71.11
	External StackExchange	No	80.47
	External Movie Reviews	No	94.6
Stacking with CNN	PAN	Yes	78.02
Stacking with CNN and LightGBM	PAN	Yes	86.93

Table 4: Accuracy scores of different classifiers on the validation dataset.

		Predicted Value	
		False	**True**
Actual Value	**False**	666	80
	True	114	632

Table 5: Confusion matrix for the final model on the validation dataset.

References

1. Damerau, F.J.: A technique for computer detection and correction of spelling errors. Commun. ACM **7**(3), 171–176 (1964)
2. Meyer zu Eissen, S., Stein, B., Kulig, M.: Plagiarism detection without reference collections. In: Decker, R., Lenz, H.J. (eds.) Advances in Data Analysis. pp. 359–366. Springer (2007)
3. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. CoRR **abs/1709.01507** (2017)
4. Karaś, D., Śpiewak, M., Sobecki, P.: OPI-JSA at CLEF 2017: Author Clustering and Style Breach Detection—Notebook for PAN at CLEF 2017. In: Cappellato, L., Ferro, N., Goeuriot, L., Mandl, T. (eds.) CLEF 2017 Evaluation Labs and Workshop – Working Notes Papers. CEUR-WS.org, Dublin, Ireland (Sep 2017)
5. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y.: LightGBM: A highly efficient gradient boosting decision tree. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30, pp. 3146–3154. Long Beach, California, USA (2017)
6. Khan, J.: Style Breach Detection: An Unsupervised Detection Model—Notebook for PAN at CLEF 2017. In: Cappellato, L., Ferro, N., Goeuriot, L., Mandl, T. (eds.) CLEF 2017 Evaluation Labs and Workshop – Working Notes Papers. CEUR-WS.org, Dublin, Ireland (2017)
7. Kim, Y.: Convolutional neural networks for sentence classification. CoRR **abs/1408.5882** (2014)
8. Kuznetsov, M., Motrenko, A., Kuznetsova, R., Strijov, V.: Methods for Intrinsic Plagiarism Detection and Author Diarization—Notebook for PAN at CLEF 2016. In: Balog, K., Cappellato, L., Ferro, N., Macdonald, C. (eds.) CLEF 2016 Evaluation Labs and Workshop – Working Notes Papers. pp. 912–919. CEUR-WS.org, Évora, Portugal (2016)
9. Levenshtein, V.I.: Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Soviet Physics Doklady **10**, 707 (1966)
10. Loper, E., Bird, S.: NLTK: The natural language toolkit. In: Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. pp. 63–70. ETMTNLP '02, Philadelphia, Pennsylvania, USA (2002)
11. McAuley, J.J., Leskovec, J.: From amateurs to connoisseurs: Modeling the evolution of user expertise through online reviews. In: Proceedings of the 22nd International Conference on World Wide Web. pp. 897–908. WWW '13, Rio de Janeiro, Brazil (2013)
12. Pervaz, I., Ameer, I., Sittar, A., Nawab, R.: Identification of Author Personality Traits using Stylistic Features—Notebook for PAN at CLEF 2015. In: Cappellato, L., Ferro, N., Jones, G., San Juan, E. (eds.) CLEF 2015 Evaluation Labs and Workshop – Working Notes Papers. CEUR-WS.org, Toulouse, France (2015)
13. Potthast, M., Rangel, F., Tschuggnall, M., Stamatatos, E., Rosso, P., Stein, B.: Overview of PAN'17: Author Identification, Author Profiling, and Author Obfuscation. In: Jones, G., Lawless, S., Gonzalo, J., Kelly, L., Goeuriot, L., Mandl, T., Cappellato, L., Ferro, N. (eds.) Experimental IR Meets Multilinguality, Multimodality, and Interaction. 7th International Conference of the CLEF Initiative (CLEF 17). Évora, Portugal (2017)

14. Sittar, A., Iqbal, H., Nawab, R.: Author Diarization Using Cluster-Distance Approach—Notebook for PAN at CLEF 2016. In: Balog, K., Cappellato, L., Ferro, N., Macdonald, C. (eds.) CLEF 2016 Evaluation Labs and Workshop – Working Notes Papers. pp. 1000–1007. CEUR-WS.org, Évora, Portugal (2016)
15. Stamatatos, E., Tschuggnall, M., Verhoeven, B., Daelemans, W., Specht, G., Stein, B., Potthast, M.: Clustering by authorship within and across documents. In: Working Notes Papers of the CLEF 2016 Evaluation Labs. CEUR Workshop Proceedings, vol. 1609, pp. 691–715. CLEF and CEUR-WS.org (2016)
16. Zmiycharov, V., Alexandrov, D., Georgiev, H., Kiprov, Y., Georgiev, G., Koychev, I., Nakov, P.: Experiments in authorship-link ranking and complete author clustering. In: Balog, K., Cappellato, L., Ferro, N., Macdonald, C. (eds.) Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum. CEUR Workshop Proceedings, vol. 1609, pp. 1018–1023. CEUR-WS.org, Évora, Portugal (2016)