Authorship Attribution for YELP Reviews - Project Proposal

Varsha Bahubali Jain
Department of Computer Science,
USC Viterbi School of Engineering,
Los Angeles, CA 90007
varshabj@usc.edu

Ashwin Sharma
Department of Computer Science,
USC Viterbi School of Engineering,
Los Angeles, CA 90007
ashwins@usc.edu

Abstract

In this project we are trying to match unlabeled review text with its author by learning their writing style from the labeled text written by the same author. This can be modelled as a multi-class single-label classification problem in Machine Learning. The final goal of this task is to identify the true author of a review based on their writing style from a learned Deep Learning or Statistical Machine Learning model.

1 Introduction

Identifying the writing style of an author has fascinated the Natural Language Processing community for quite some time now. It has been supported by both statistical and computational methods. During the last couple of years, this field has grown immensely by exploiting advances in machine learning, information retrieval and natural language processing. The goal is to match unlabelled text with its author through a similarity measurement or writing style learned from the labelled text written by the same author. There are several common real-world applications for this task including Plagiarism detection, identification of author of threats, weeding out fake reviews, personalizing ads for people using the same account etc. In Authorship Attribution literature, there are two main sub-tasks. These are Authorship Identification, which follows the closed-world formulation and we attempt to predict which of the closed set of N candidate authors wrote a specific text, and Authorship Verification, in which the attempt is to predict whether or not two different texts are written by the same author. In this project we will be focusing on the Author Identification task using various Machine Learning and Deep Learning techniques. The Authorship Identification task can be modelled as a multi-class single-label text categorization task from a machine learning point-of-view, where each class is represented by a candidate author. A text of unknown authorship is assigned to one candidate author given a set of candidate authors for whom text samples of undisputed authorship are available.

2 General Architecture

The general architecture for the Authorship Attribution task can be designed as shown below. It has been modelled as a multi-class classification problem where the Attribution model is trained on the labeled review texts of multiple authors. The attribution model can be a traditional statistical learning model or a Deep Learning model. When a text of unknown authorship comes along, the Attribution model based on learned writing styles of the authors, predicts the most likely author of the test text.

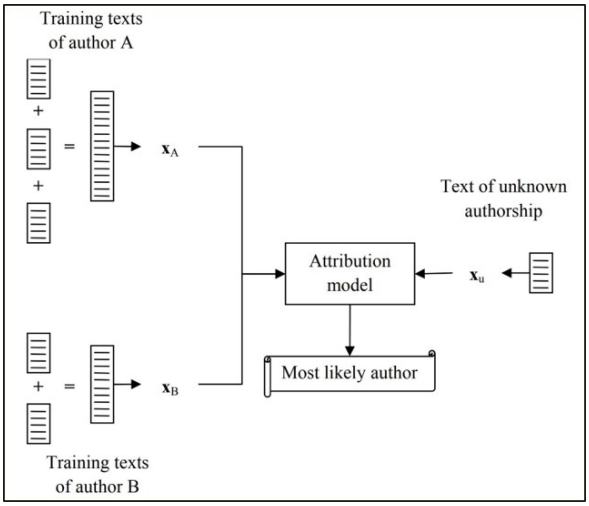

Figure 1: General Architecture of Authorship Attribution System

3 Related Work

The approaches for Author Identification can be broadly categorized into two types of methods: (1) stylometry methods which aim to capture an author's writing style using various statistical features (eg. POS, punctuation marks, word frequencies etc.), (2) content-based methods that intend to identify an author's writing style based on the content of the text (eg. Bag of words, words n-gram, term vectors, TF-IDF etc.). According to PAN competitions, most successful works for Authorship Identification in social media have used combinations of these two kinds of features. Simple machine learning models using TF-IDF vectors have been used in SVM and Ensemble Learning algorithms. Deep learning models for this task have used the approach based on sub word character n-gram embeddings and deep averaging networks (DAN). Bi-RNN with GRU have also been used by combining it with an Attention Mechanism.

4 Dataset

We are using the YELP Reviews dataset for this project. YELP Dataset contains close to 7 Million reviews across 2 million businesses across various domains. Out of these reviews, we take the 50 most-prolific reviewers in the Yelp Reviews Dataset file. Each of these authors have left more than 100000 character of review text in total (across multiple reviews, multiple years, multiple restaurants). For each author, we concatenate all of his or her reviews into a single text. We take the first 50000 characters as training data and the next 50000 characters to create ten test texts for that author, with each test text being 5000 characters in length, resulting in 500 test texts in total.

5 Feature Selection and Extraction

redStylometric Features, or the so called style-markers have been extensively used to identify the writing style in various Authorship Attribution tasks. The current view of extracting these features depend on the computational requirements of measuring them.

5.1 Lexical Features

Lexical features like sentence length counts and word length counts have been used in past implementations. Such features are language independent and only require a tokenizer for feature extraction. The popular approach to represent texts is by converting their frequencies to vectors. Lexical features are the widely used features for representing a writing style of an author. In style-based text classification, the most common words are found to be among the best features to discriminate between writing styles. As a consequence of this, style-based text classification using lexical features need a much lower dimensionality when compared to other Language Processing tasks. Much lesser words can perform the Authorship Attribution task efficiently and can capture pure stylistic choices of the author. Capturing the top 100 words used across all authors and learning their distribution among each author's work has proven to be effective in learning the writing style. To take advantage of contextual information, word n-grams can be used as textual features for the given task.

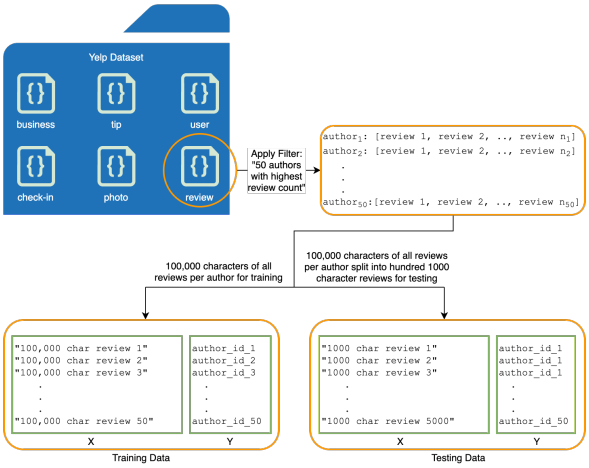

Figure 2: Preparing Dataset from YELP Reviews Dataset for Authorship Attribution

5.2 Character Features

Character features like alphabetic characters count, digit characters count, uppercase and lowercase characters count, letter frequencies, punctuation marks, etc. can represent character level stylistic features for a given author. This type of features has proven to be quite useful to quantify the writing style in previous works. As with lexical features, finding the most frequent n-grams at a character level are the most important features for learning the writing style. However, this increases the dimensionality of features as now we are working on the character-level. Character level features can be easily extracted using available word pro-

cessing techniques. Character features can also be converted into vector space and used as features for the authorship attribution task.

5.3 Syntactic Features

Syntactic Features like POS Tags can be used to enhance the model as authors tend to use similar syntactic patterns unconsciously. But extracting these features requires a special parser which is highly robust and can accurately identify the POS tags for the underlying text. This now becomes extremely language dependent and the availability of a parser which can analyse a particular natural language with high accuracy. These features are not used frequently but if available and used correctly, the model can more accurately learn the writing style of an author based on his/her syntactic language habits.

5.4 Semantic Features

Semantic features require detailed textual analysis for extraction and using them as stylometric features results in less accurate and more noisy models since the underlying techniques for extracting such features are themselves inaccurate and noisy. NLP tools can be applied successfully to low-level tasks, such as sentence splitting, POS tagging, text chunking, partial parsing, so relevant features would be measured accurately and the noise in the corresponding datasets remains low. On the other hand, more complicated tasks such as full syntactic parsing, semantic analysis, or pragmatic analysis cannot yet be handled adequately by current NLP technology for unrestricted text. As a result, very few attempts have been made to exploit high-level features for stylometric purposes.

5.5 Application Specific Features

The previously described lexical, character, syntactic, or semantic features are application independent since they can be extracted from any textual data given the availability of the appropriate NLP tools and resources required for their measurement. Beyond that, one can define application-specific measures in order to better represent the nuances of style in a given text domain. This section reviews the most important of these measures. Structural measures like use of greetings and farewells in reviews, types of signature, use of indentation. paragraph length etc. can be extracted. In order to better capture the properties of an author's style within a particular text domain, content-specific keywords can be used. For YELP reviews, compiling this list of content-specific keywords based on the domain of the business can prove to be useful in enhancing our authorship attribution model.

6 Methodology

6.1 Support Vector Machines using TF- IDF Vectorizer

SVM is a linear classifier and tries to find a separating hyperplane that maximizes the distance between two classes. In case of inseparable data, SVMs employ a slack allowing misclassifications after paying penalty. They have proven to work well in tasks ranging from Named Entity Recognition, Authorship Profiling, such as personality and gender prediction and authorship attribution tasks. For multi-class classification using binary classifiers, we can opt for one-versus-rest (we train n classifiers, one for each class) or a one-versus-one scheme. We employ the former approach, training one SVM per candidate author. We vectorize the extracted features using TF-IDF, a scheme where all terms are represented by their frequency, but lower weights are given to terms that appear in many different documents. This is intended to leave only the frequent and distinctive words as markers. Smoothing can be used to handle terms found in the test text but not found in the training TF-IDF vectors. SVM can then be trained on the 50 known texts and predictions can be generated for the test texts based on the confidence scores assigned by SVM, assigning each test text to its most likely author.

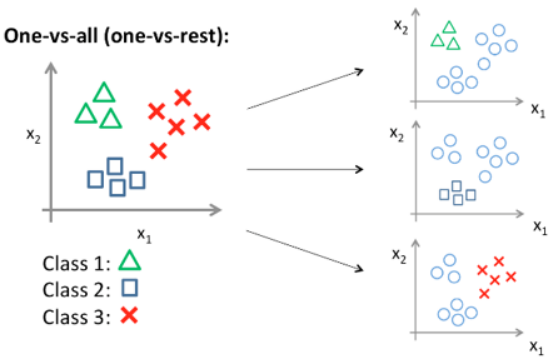

Figure 3: SVM in action for Multi-class classification task with 3 classes

6.2 Ensemble Methods

There are many different types of ensembles; stacking is one of them. It is one of the more general types and can theoretically represent any other ensemble technique. Stacking involves training a learning algorithm to combine the predictions of several other learning algorithms. One of the simplest forms of Stacking, in which we train three different classifiers can be seen in Figure 4. CalibratedClassifierCV can be used from scikit-learn library to compute the likelihood of each candidate in each classifier for a test example. The average outputs of models in the ensemble is used make the final prediction. For a given sample, the candidate with the highest probability is selected as the output.

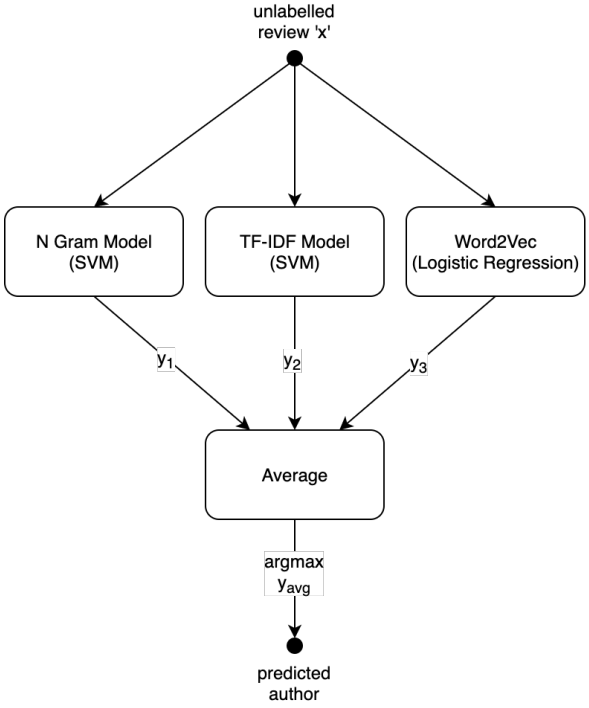

Figure 4: Stacked Ensemble Architecture

6.3 Recurrent Neural Networks with Gated Recurrent Unit (GRU-RNN)

Recurrent Neural Networks with word embeddings have been used successfully in other text classification tasks, such a Sentiment Analysis. Instead of using frequency-based relevance like in SVM, a neural network can read a text sequentially, similarly to how a human might. The network can then learn the most relevant words and relations, and discriminate between classes. Encoder architecture is widely used where the recurrent neural network reads one review at a time. The resulting document vector is fed to a fully connected layer (softmax) for classification. GRU encoders can be used to run through text and concatenate the results into one vector. Pre-trained word embeddings can be used as an input where the vocabulary was built out of the entire training text. Unknown words can be replaced by a special token. To improve training speed, Batch Normalization and Dropout can be use as a form of regularization. Around 10-20 percent of training data can be used as validation data for hyperparameter tuning and generalization. Cross entropy will be used as the loss function.

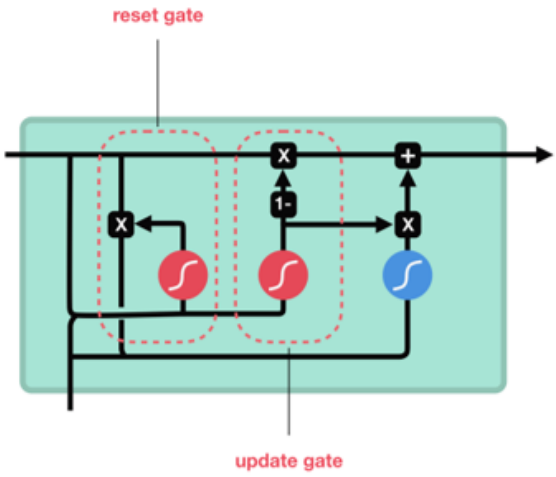

Figure 5: Cross Section of a Gated Recurrent Unit with Update Gate and Reset Gate

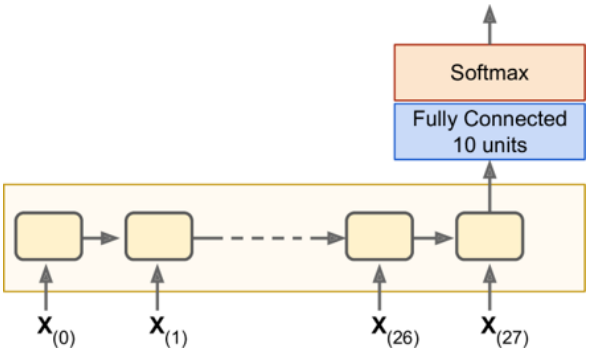

Figure 6: GRU-RNN Architecture for Multi-class classification with a sequence length of 28 and 10 classes

7 Evaluation

There are various open benchmarks available now and the most commonly cited authorship competitions are the Ad-Hoc Authorship Attribution

Competition and PAN (International Workshop on Plagiarism Detection, Author Identification, and Near-Duplicate Detection). Accuracy on the test dataset can be used as an evaluation measure for both SVM and Deep Learning model. We can compare the F1 scores of our SVM model with the official PAN 2019 baseline classifier scores for English language. A confusion matrix can also be generated to evaluate the quality of results of this Supervised Multi-class classification task.

8 Implementation Details

8.1 Primary Usage - Input and Output

Input: Unlabelled textual review, either generated by YELP or by a Sequence model which has learnt various writing styles from the training dataset. Output: Predicted Author from a predefined pool of authors

8.2 Evaluation Metrics

Macro-averaged F1 scores - as used in PAN CLEF Shared Task of Authorship Attribution This calculates F1-score for each label/candidate author, and finds their unweighted mean. This does not take label imbalance into account, although our dataset is perfectly balanced.

8.3 Technologies Used

Python 3.5 Jupyter Notebook Scikit-learn, TensorFlow, Keras, NLTK, Google Colab

8.4 Organization of Code

8.4.1 Dataset Preparation

We use the data from review.json for our project. The data preparation steps are as follows. The reviews are grouped by author and 50 authors with the highest review count are selected. For each of these authors, we concatenate all their reviews into one huge review. From this huge review, we select the first 100,000 characters for training and the next 100,000 characters for testing. The training dataset now has 50 (review, author) tuples, where each review belongs to one author and is 100,000 characters long. For the test set, we split each 100,000 character review into a hundred 1000 character reviews. So the test set will have 5000 (review, author) tuples, where each review is 1000 characters long, and there are 100 such reviews present for each author. We now have the training and testing set which we serialize and store as pickle files. These can be loaded from their files and used later.

8.4.2 TF-IDF Model

SVM has been widely used for tasks like Named Entity Recognition, Authorship Profiling, such as gender and personality and gender prediction among other NLP tasks. For multi-class classification we use a OneVsRest classifier where we train an SVM model for every candidate author. We tried two different Feature preprocessing techniques before feeding it into the SVM: A union of word and character n-grams (unigrams and bigrams for words, bigrams and trigrams for characters). This feature set consisted of 28,206 distinct terms fitted only on the training dataset and the terms in the test texts not found in training was ignored. The texts were first preprocessed using NLTK. Stopwords and punctuations were removed before retaining only the word stems. A custom word and punctuation tokenizer from NLTK was used to tokenize the text. These pre-processed texts were vectorized using TF-IDF where the training vocabulary is used and terms in the test texts not found in the vocabulary are ignored. We trained an SVM model on the 50 training texts available with us using a LinearSVC. Predictions were generated for each of the texts based on the confidence scores assigned by the SVMs. With the initial test texts, the first set of features fed into the SVM model gave an accuracy of 46 percent and the second set of features fed into the SVM model gave an accuracy of 38 percent. The reason for such low accuracy using both feature sets was due to the fact that the average lengths of our test texts was only about 180 words, which was too short for the model to make accurate predictions. After combining 5 test texts and then predicting on this concatenation gave an accuracy of 90 percent for the first set of features and 72 percent for the second set of features.

8.4.3 Character based Language Model (GRU-RNN)

GRU is more generalized than LSTM and takes lesser time for training and predictions. Since we need the generalized writing style of a candidate author to be learnt, GRU seemed like the best option for this task and is supported by existing literature [3]. We use an encoder architecture, which reads each review one word at a time using a recurrent neural network. The resulting document

vector is fed to a fully connected layer for classification. We tried only unidirectional encoding as [4] states that bi-directional encoding showed no improvement. Pre-trained Glove embeddings (glove.6B.300d) of size 300 was used as an input where the vocabulary was built out of the entire training text. We built the vocabulary out of the whole training set. Unknown words were replaced with UNK token. We experimented with several vocabulary sizes by taking the most frequently occurring words. For each author, we split the 50000 character chunks into ten 5000-character samples for training. We tokenized each text chunk into words and transformed them into a matrix of one hot-encoded vectors. We padded the length of each sequence to 900 words. The one-hot-encoded reviews extract the actual word embedding vector from an 128-dimensional word embedding matrix. The resulting sequence of vectors was fed into the Gated Recurrent Unit layer with a dimension of 256. The final output of the GRU layer was then passed to a fully connected output layer with softmax activation to directly identify the author. We employed Batch Normalization in the recurrent layer to improve training speed and Dropout with a keep probability of 0.1 and 0.3 as a form of regularization. During training, we used 10

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_5 (Embedding)      (None, None, 300)         25500
_________________________________________________________________
dropout_8 (Dropout)          (None, None, 300)         0
_________________________________________________________________
batch_normalization_8 (Batch (None, None, 300)         1200
_________________________________________________________________
gru_4 (GRU)                  (None, 256)               427776
_________________________________________________________________
dropout_9 (Dropout)          (None, 256)               0
_________________________________________________________________
batch_normalization_9 (Batch (None, 256)               1024
_________________________________________________________________
dense_4 (Dense)              (None, 85)                21845
=================================================================
Total params: 477,345
Trainable params: 476,233
Non-trainable params: 1,112
_________________________________________________________________
```

Figure 7: Tensorflow Model Summary for each candidate author

We used the Adam optimizer and cross entropy as the loss function. We trained the network for 10 epochs for each candidate model. Each such GRU model was trained for every candidate author to identify distinctive writing style for every author. For predicting, we first pre-processed the text into the expected embedding size and evaluated on every trained candidate model. The model that gave the lowest cross entropy loss was then chosen and the author id associated with this author model was the final prediction. We achieved an accuracy

Hyperparameters	Value
Batch Size	**128**, 256
Epochs	5, **10**, 15
Embedding Size	100, 200, **300**
GRU Hidden Layers	128, **256**
Dropout Probability	**0.1**, **0.3**, 0.5
Dense Layer Activation	**softmax**
Loss Function	**CrossEntropy**, SGD
Optimizer	**Adam**, Adagrad
Optimizer Learning Rate	**0.001**
Validation Split	**0.1**, 0.2

Figure 8: Hyperparameter tuning for our GRU-RNN model, the final selected value is highlighted in bold

of 94.8 percent on our test data. Time taken for training was about 10 hours and the time taken for predicting on 500 test texts was about 16 hours. The huge difference between training and testing times is due to the fact that the model only has to learn the writing styles for 50 authors, whereas it has to predict the writing style of 500 authors. Each test text is evaluated on each of the 50 trained author models and took about 3 minutes per text. Although the generative language modelling predictions took an excessively long time to generate, these are easily parallelizable. Because each language model needs to evaluate each text, it would be trivial to share this workload among multiple cores or machines by deploying one model and all test texts per machine.

8.4.4 Ensemble Methods

We used a stacked ensemble method provided in scikit-learn called CalibratedClassifierCV which models the given data using OneVsRestClassifier , and we choose 3 models to be fed into this ensemble. The individual models included TF-IDF, n-gram as well as Word2vec model with accuracy of 95.8 percent, 99.9 percent and 86.6 percent respectively. Each of these accuracies were attained after lot of hyperparameter tuning using K-fold cross validation in scikit-learn. The final ensemble resulted in an accuracy of 99.2 percent as a result of all the 3 models that were stacked being extremely powerful classifiers for the test dataset.

9 Results

A random baseline approach would have a 1=50 chance of guessing the correct author, so we would expect an accuracy of 0:02. Given that the test texts are relatively short (5 000 characters is approximately 900 words), the task is not easy and it is evident that both the SVM with TF-IDF model and the character-based language (GRU-RNN) model are proficient at discriminating between authorship style. We are pleased with the indication that the less-common GRU neural approach proved suitable for the task.

Model / Parameters	TF-IDF based SVM	Character based GRU-RNN	N-gram based SVM	Word2Vec based SVM	Stacked Ensemble
Accuracy	90.8%	94.8%	99.6%	86.6%	99.6%
Macro averaged F1	90.03%	93.11%	98.1%	85%	99%
Time Taken	A few seconds	10 hours for training, 3 minutes per prediction	10 minutes for training, few seconds for testing	A few minutes	A few minutes

Figure 9: Accuracies for various models tried

References

[1] Joachim Diederich, Jorg Kindermann, Edda Leopold, and Gerhard Paass 2003. *Authorship attribution with support vector machines.* Applied intelligence 19(1):109–123.

[2] Moshe Koppel and Jonathan Schler. 2003. *Exploiting stylistic idiosyncrasies for authorship attribution.* In Proceedings of IJCAI'03 Workshop on Computational Approaches to Style Analysis and Synthesis. volume 69, page 72.

[3] Douglas Bagnall 2015. *Authorship clustering using multi-headed recurrent neural networks* arXiv preprint arXiv:1608.04485

[4] Mart Busger op Vollenbroek, Talvany Carlotto, Tim Kreutz, Maria Medvedeva, Chris Pool, Johannes Bjerva, Hessel Haagsma, and Malvina Nissim. 2016. *GronUP: Groningen User Profiling—Notebook for PAN at CLEF 2016* In Krisztian Balog, Linda Cappellato, Nicola Ferro, and Craig Macdonald, editors, CLEF 2016 Evaluation Labs and Workshop – Working Notes Papers, 5-8 September, Evora, Portugal. CEUR-WS.org. http://ceur-ws.org/Vol-1609/.

[5] Moshe Koppel and YaronWinter. 2014. *Determining if two documents are written by the same author.* Journal of the Association for Information Science and Technology 65(1):178–187.

[6] Kim Luyckx and Walter Daelemans. 2008. *Authorship attribution and verification with many authors and limited data.* In Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1. Association for Computational Linguistics, pages 513–520.

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

[8] Dani Yogatama, Chris Dyer,Wang Ling, and Phil Blunsom. 2017. *Generative and discriminative text classification with recurrent neural networks.* arXiv preprint arXiv:1703.01898 .

[9] M Rahgouy, HB Giglou, T Rahgouy, MK Sheykhlan, E Mohammadzadeh. 2019. *Cross-domain Authorship Attribution: Author Identification using a Multi-Aspect Ensemble Approach - Notebook for PAN at CLEF 2019.* In CLEF 2019 Evaluation Labs and Workshop–Working Notes Papers. CEUR-WS. org.

[10] Rangel Pardo, F., Montes-y-Gómez, M., Potthast, M., Stein, B 2018. *Overview of the 6th Author Profiling Task at PAN 2018: Cross-domain Authorship Attribution and Style Change Detection.* CLEF 2018 Evaluation Labs and Workshop – Working Notes Papers, 10-14 September, Avignon, France. CEUR-WS.org (Sep 2018)

[11] Murauer, B., Tschuggnall, M., Specht, G 2018. *Dynamic Parameter Search for Cross-Domain Authorship Attribution—Notebook for PAN at CLEF 2018.* CLEF 2018 Evaluation Labs and Workshop – Working Notes Papers, 10-14 September, Avignon, France. CEUR-WS.org (Sep 2018)

[12] Joachim Diederich, Jorg Kindermann, Edda Leopold, and Gerhard Paass 2003. *Authorship attribution with support vector machines..* Applied intelligence 19(1):109–123