Deep Bayes Factor Scoring for Authorship Verification
Notebook for PAN at CLEF 2020

Benedikt Boenninghoff[1], Julian Rupp[1], Robert M. Nickel[2], and Dorothea Kolossa[1]

[1] Ruhr University Bochum, Germany
{benedikt.boenninghoff, julian.rupp, dorothea.kolossa}@rub.de
[2] Bucknell University, Lewisburg, PA, USA
rmn009@bucknell.edu

Abstract The PAN 2020 authorship verification (AV) challenge focuses on a cross-topic/closed-set AV task over a collection of fanfiction texts. Fanfiction is a fan-written extension of a storyline in which a so-called fandom topic describes the principal subject of the document. The data provided in the PAN 2020 AV task is quite challenging because authors of texts across multiple/different fandom topics are included. In this work, we present a hierarchical fusion of two well-known approaches into a single end-to-end learning procedure: A deep metric learning framework at the bottom aims to learn a pseudo-metric that maps a document of variable length onto a fixed-sized feature vector. At the top, we incorporate a probabilistic layer to perform Bayes factor scoring in the learned metric space. We also provide text preprocessing strategies to deal with the cross-topic issue.

1 Introduction

The task of (pairwise) authorship verification (AV) is to decide if two texts were written by the same person or not. AV is traditionally performed by linguists who aim to uncover the authorship of anonymously written texts by inferring author-specific characteristics from the texts [11]. Such characteristics are represented by so-called **linguistic features**. They are derived from an analysis of errors (e.g. spelling mistakes), textual idiosyncrasies (e.g. grammatical inconsistencies) and stylistic patterns [11].

Automated (machine-learning-based) systems have traditionally relied on so-called **stylometric features** [20]. Stylometric features tend to rely largely on linguistically motivated/inspired metrics. The disadvantage of stylometric features is that their reliability is typically diminished when applied to texts with large topical variations.

Deep learning systems, on the other hand, can be developed to automatically learn **neural features** in an end-to-end manner [5]. While these features can be learned in such a way that they are largely insensitive to the topic, on the negative side, they are generally not linguistically interpretable.

In this work we propose a substantial extension of our published ADHOMINEM approach [4], in which we interpret the neural features produced by ADHOMINEM not just from a metric point of view but, additionally, from a probabilistic point of view.

With our modification of ADHOMINEM we were also cognizant of the proposed future AV shared tasks of the PAN organization [16]. Three broader research questions (cross-topic verification, open-set verification, and "surprise task") are put into the spotlight over the next three years. In light of these challenges we define requirements for automatically extracted neural features as follows:

- **Distinctiveness:** Our extracted neural features should contain all necessary information w.r.t. the *writing style*, such that a verification system is able to distinguish same/different author/s in an open-set scenario. In order to automatically quantify deviations from the standard language, the text sample collection for the training phase must be sufficiently long.
- **Invariance:** Authors tend to shift the characteristics of their writing according to their situational disposition (e.g. their emotional state) and the topic of the text/discourse. Extracted neural features should therefore, ideally, be invariant w.r.t. the topic, the sentiment, the emotional state of the writer, and so forth.
- **Robustness:** The writing style of a text can be influenced, for example, by a desire to imitate another author (e.g. the original author of a fandom topic) or by applying a deliberate obfuscation strategy for other reasons. Our extracted neural features should still lead to reliable verification results, even when obfuscation/imitation strategies are applied by the author.
- **Adaptability:** The writing style is generally also affected by the type of the text, which is called genre. People change their linguistic register depending on the genre that they write in. This, in turn, leads to significant changes in the characteristics of the resulting text. For a technical system, it is thus extremely difficult to establish a common authorship between a WhatsApp message and a formal job application for example. In forensic disciplines, it is therefore important to train classifiers only on one genre at a time. In research, however, it is quite an interesting question how to, e.g., find a joint subspace representation/embedding for text samples across different genres.

We assume that a single text sample has been written by a single person. If necessary, we need to examine a *collaborative authorship* in advance [11]. Dealing with genre-adaption or obfuscation/imitation strategies is not part of the PAN 2020/21 AV task. Another open question is the minimum size of a text sample required to obtain reliable output predictions. This question will also be left for future work.

2 ADHOMINEM: Siamese network for representation learning

Existing AV algorithms can be taxonomically grouped w.r.t. their design and characteristics, e.g. instance- vs. profile-based paradigms, intrinsic vs. extrinsic methods [18], or unary vs. binary classification [12]. We may roughly describe the work flow for a traditional binary AV classifier design as follows: In the feature engineering process, a set of manually defined stylometric features is extracted. Afterwards, a training and/or development set is used to fit a model to the data and to tune possible hyper-parameters of the model. Typically, an additional calibration step is necessary to transform scores provided by the model into appropriate probability estimates. Our modified ADHOMINEM system works differently. We define a deep-learning model architecture with all of its hyper-parameters and thresholds a-priori and let the model learn suitable features for the provided setup on its own. As with most deep-learning approaches, the success of the proposed setup depends heavily on the availability of a large collection of text samples with many examples of representative variations in writing style.

The majority of published papers, using deep neural networks to build an AV framework, have employed a classification loss [1], [17]. However, metric learning objectives present a promising alternative [5], [8]. The discriminative power of our proposed AV method stems from a fusion of two well-known approaches into a single joint end-to-end learning procedure: A precursor of our ADHOMINEM system [4] is used as a *deep metric learning* framework [14] to measure the similarity between two text samples. The features that are implicitly produced by the ADHOMINEM system are then fed into a *probabilistic linear discriminant analysis* (PLDA) layer [9] that functions as a pair-wise discriminator to perform Bayes factor scoring in the learned metric space.

2.1 Neural extraction of linguistic embedding vectors

A text sample can be understood as a hierarchical structure of ordered discrete elements: It consists of a list of ordered sentences. Each sentence consists of an ordered list of tokens. Again, each token consists of an ordered list of characters. The purpose of ADHOMINEM is to map a document to a feature vector. More specifically, its Siamese topology includes a hierarchical neural feature extraction, which encodes the stylistic characteristics of a pair of documents $(\mathcal{D}_1, \mathcal{D}_2)$, each of variable length, into a pair of fixed-length *linguistic embedding vectors* (LEVs) $\boldsymbol{y}_i$:

$$\boldsymbol{y}_i = \mathcal{A}_{\boldsymbol{\theta}}(\mathcal{D}_i) \in \mathbb{R}^{D \times 1},\ i \in \{1, 2\}, \tag{1}$$

where D denotes the dimension of the LEVs and $\boldsymbol{\theta}$ contains all trainable parameters. It is called a *Siamese* network because both documents $\mathcal{D}_1$ and $\mathcal{D}_2$ are mapped through the exact same function $\mathcal{A}_{\boldsymbol{\theta}}(\cdot)$. The internal structure of $\mathcal{A}_{\boldsymbol{\theta}}(\cdot)$ is illustrated in Fig. 1. After preprocessing and tokenization (which will be explained in Section 3), the system passes a fusion of token and character embeddings into a two-tiered bidirectional LSTM [13] network with attentions [2]. We incorporate a characters-to-word encoding layer to take the specific uses of prefixes and suffixes as well as spelling errors into account. An incorporation of attention layers allows us to visualize words and sentences that have been marked as "highly significant" by the system. As shown in Fig. 1, the network produces document embeddings, which are converted into LEVs via a fully-connected dense layer. With this output layer, we can control the output dimension. AV is accomplished by computing the Euclidean distance [14]

$$d(\mathcal{D}_1, \mathcal{D}_2) = \|\mathcal{A}_{\boldsymbol{\theta}}(\mathcal{D}_1) - \mathcal{A}_{\boldsymbol{\theta}}(\mathcal{D}_2)\|_2^2 = \|\boldsymbol{y}_1 - \boldsymbol{y}_2\|_2^2 \tag{2}$$

between both LEVs. If the distance in Eq. (2) is above a given threshold τ, then the system decides on *different-authors*, if the distance is below τ, then the system decides on *same-authors*. Details are comprehensively described in [4].

Pseudo-metric: ADHOMINEM provides a framework to learn a pseuo-metric. Since we are using the Euclidean distance in Eq. (2) we have the following properties:

$$\begin{aligned}
&d(\mathcal{D}_1, \mathcal{D}_2) \geq 0 && \text{(nonnegativity)}\\
&d(\mathcal{D}_1, \mathcal{D}_1) = 0 && \text{(identity)}\\
&d(\mathcal{D}_1, \mathcal{D}_2) = d(\mathcal{D}_2, \mathcal{D}_1) && \text{(symmetry)}\\
&d(\mathcal{D}_1, \mathcal{D}_3) \leq d(\mathcal{D}_1, \mathcal{D}_2) + d(\mathcal{D}_2, \mathcal{D}_3) && \text{(triangle inequality)}
\end{aligned}$$

Note that we may obtain $d(\mathcal{D}_1, \mathcal{D}_2) = 0$ where $\mathcal{D}_1 \neq \mathcal{D}_2$.

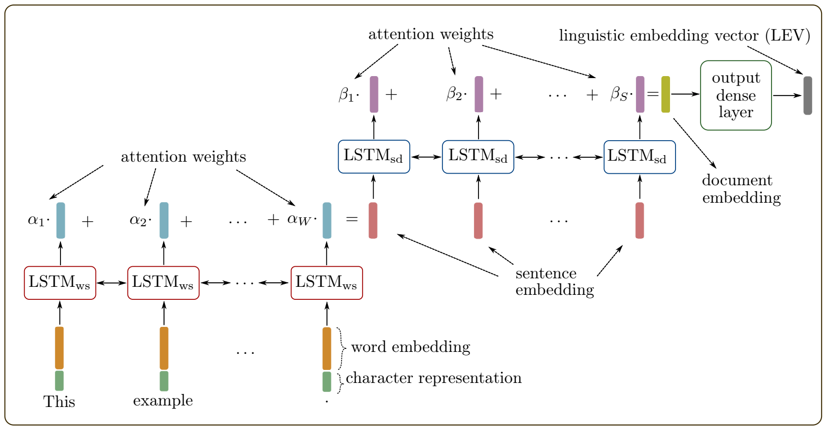


Figure 1. Flowchart of the neural feature extraction as described in [4]. A given text sample is transformed into the learned metric space by the function $\mathcal{A}_{\boldsymbol{\theta}}(\cdot)$.

Loss function: The entire network is trained end-to-end. For the pseudo-metric learning objective, we choose the modified contrastive loss [5]:

$$\mathcal{L}_{\boldsymbol{\theta}} = l \cdot \max\left\{ \|\boldsymbol{y}_1 - \boldsymbol{y}_2\|_2^2 - \tau_s, 0 \right\}^2 + (1-l)\cdot \max\left\{ \tau_d - \|\boldsymbol{y}_1 - \boldsymbol{y}_2\|_2^2, 0 \right\}^2, \quad (3)$$

where $l \in \{0,1\}$, $\tau_s < \tau_d$ and $\tau = \frac{1}{2}(\tau_s + \tau_d)$. During training, all distances between *same-author* pairs are forced to stay below the lower of the two thresholds, τ_s. Conversely, distances between *different-authors* pairs are forced to remain above the higher threshold τ_d. By employing this dual threshold strategy, the system is made more insensitive to topical or intra-author variations between documents [14], [4].

2.2 Two-covariance model for Bayes factor scoring

Text samples are characterized by a high variability. Statistical hypothesis tests can help to quantify the outputs/scores of our algorithm and to decide whether to accept or reject the decision. ADHOMINEM can be extended with a framework for statistical hypothesis testing. More precisely, we are interested in the AV problem where, given the LEVs of two documents, we have to decide for one of two hypotheses:

$\mathcal{H}_s$: The two documents were written by the same person,

$\mathcal{H}_d$: The two documents were written by two different persons.

In the following, we will describe a particular case of the well-known *probabilistic linear discriminant analysis* (PLDA) [15], which is also known as the *two-covariance model* [9]. Let us assume, the author's writing style is represented by a vector $\boldsymbol{x}$. We suppose that our (noisy) observed LEV $\boldsymbol{y} = \mathcal{A}_{\boldsymbol{\theta}}(\mathcal{D})$ stems from a Gaussian generative model that can be decomposed as

$$\underbrace{\boldsymbol{y}}_{\text{linguistic embedding vector}} = \underbrace{\boldsymbol{x}}_{\text{author's writing style}} + \underbrace{\boldsymbol{\epsilon}}_{\text{noise term}}, \quad (4)$$

where $\boldsymbol{\epsilon}$ characterizes residual noise, caused by thematic varitions or by significant changes in the process of text production for instance.

The idea behind this *factor analysis* is that the writing characteristics of the author, measured in the observed LEV $\boldsymbol{y}$, lie in a latent variable $\boldsymbol{x}$. The probability density functions for $\boldsymbol{x}$ and $\boldsymbol{\epsilon}$ in Eq. (4) are defined as in [6]:

$$p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{B}^{-1}), \tag{5}$$

$$p(\boldsymbol{\epsilon}) = \mathcal{N}(\boldsymbol{\epsilon}|\boldsymbol{0}, \boldsymbol{W}^{-1}), \tag{6}$$

where $\boldsymbol{B}^{-1}$ defines the *between-author* covariance matrix and $\boldsymbol{W}^{-1}$ denotes the *within-author* covariance matrix. As mentioned in [7], the idea is to model *inter-author variability* (with the covariance matrix $\boldsymbol{B}^{-1}$) and *intra-author variability* (with the covariance matrix $\boldsymbol{W}^{-1}$). From Eqs. (5)−(6), it can be deduced that the conditional density function is given by [6]:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{W}^{-1}). \tag{7}$$

Assuming we have a set of n LEVs, $\mathcal{Y} = \{\boldsymbol{y}_1, \ldots \boldsymbol{y}_n\}$, verifiably associated to the same author, then we can compute the posterior (see Theorem 1 on page 175 in [10]):

$$p(\boldsymbol{x}|\mathcal{Y}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{L}^{-1}\boldsymbol{\gamma}, \boldsymbol{L}^{-1}), \tag{8}$$

where $\boldsymbol{L} = \boldsymbol{B} + n\boldsymbol{W}$ and $\boldsymbol{\gamma} = \boldsymbol{B}\boldsymbol{\mu} + \boldsymbol{W}\sum_{i=1}^{n}\boldsymbol{y}_i$. Let us now consider the process of generating two *linguistic embedding vector* (LEV) $\boldsymbol{y}_i,\ i \in \{1, 2\}$. We have to distinguish between *same-author* and *different-author* pairs:

Same-author pair: In the case of a same-author pair, a single latent vector $\boldsymbol{x}_0$ representing the author's writing style is generated from the prior $p(\boldsymbol{x})$ in Eq. (5) and both LEVs $\boldsymbol{y}_i,\ i \in \{1, 2\}$ are generated from $p(\boldsymbol{y}|\boldsymbol{x}_0)$ in Eq. (7). The joint probability density function is then given by

$$p(\boldsymbol{y}_1, \boldsymbol{y}_2|\mathcal{H}_s) = \frac{p(\boldsymbol{y}_1, \boldsymbol{y}_2|\ \boldsymbol{x}_0, \mathcal{H}_s)\ p(\boldsymbol{x}_0|\mathcal{H}_s)}{p(\boldsymbol{x}_0|\boldsymbol{y}_1, \boldsymbol{y}_2, \mathcal{H}_s)} = \frac{p(\boldsymbol{y}_1|\boldsymbol{x}_0)\ p(\boldsymbol{y}_2|\boldsymbol{x}_0)\ p(\boldsymbol{x}_0)}{p(\boldsymbol{x}_0|\boldsymbol{y}_1, \boldsymbol{y}_2)}. \tag{9}$$

The term $p(\boldsymbol{x}_0|\boldsymbol{y}_1, \boldsymbol{y}_2)$ can be computed using Eq. (8).

Different-authors pair: For a different-authors pair, two latent vectors, $\boldsymbol{x}_i$ for $i \in \{1, 2\}$, representing two different authors' writing characteristics, are independently generated from $p(\boldsymbol{x})$ in Eq. (5). The corresponding LEVs $\boldsymbol{y}_i$ are generated from $p(\boldsymbol{y}|\boldsymbol{x}_i)$ in Eq. (7). The joint probability density function is then given by

$$p(\boldsymbol{y}_1, \boldsymbol{y}_2|\mathcal{H}_d) = p(\boldsymbol{y}_1|\mathcal{H}_d)\ p(\boldsymbol{y}_2|\mathcal{H}_d) = \frac{p(\boldsymbol{y}_1|\boldsymbol{x}_1)p(\boldsymbol{x}_1)}{p(\boldsymbol{x}_1|\boldsymbol{y}_1)} \cdot \frac{p(\boldsymbol{y}_2|\boldsymbol{x}_2)p(\boldsymbol{x}_2)}{p(\boldsymbol{x}_2|\boldsymbol{y}_2)}. \tag{10}$$

The terms $p(\boldsymbol{x}_1|\boldsymbol{y}_1)$ and $p(\boldsymbol{x}_2|\boldsymbol{y}_2)$ are again obtained from Eq. (8).

Verification score: The described probabilistic model involves two steps: a training phase to learn the parameters of the Gaussian distributions in Eqs. (5)−(6) and a verification phase to infer whether both text samples come from the same author.

For both steps, we need to define the verification score, which can now be calculated as the log-likelihood ratio between the two hypotheses $\mathcal{H}_s$ and $\mathcal{H}_d$:

$$\begin{aligned}\text{score}(\boldsymbol{y}_1,\boldsymbol{y}_2) &= \log p(\boldsymbol{y}_1,\boldsymbol{y}_2|\mathcal{H}_s) - \log p(\boldsymbol{y}_1,\boldsymbol{y}_2|\mathcal{H}_d)\\ &= \log p(\boldsymbol{x}_0) - \log p(\boldsymbol{x}_1) - \log p(\boldsymbol{x}_2)\\ &\quad + \log p(\boldsymbol{y}_1|\boldsymbol{x}_0) + \log p(\boldsymbol{y}_2|\boldsymbol{x}_0) - \log p(\boldsymbol{y}_1|\boldsymbol{x}_1) - \log p(\boldsymbol{y}_2|\boldsymbol{x}_2)\\ &\quad - \log p(\boldsymbol{x}_0|\boldsymbol{y}_1,\boldsymbol{y}_2) + \log p(\boldsymbol{x}_1|\boldsymbol{y}_1) + \log p(\boldsymbol{x}_2|\boldsymbol{y}_2) \end{aligned} \quad (11)$$

Eq. (11) is often called the *Bayes factor*. Since $p(\boldsymbol{y}_1,\boldsymbol{y}_2|\mathcal{H}_s)$ in Eq. (9) and $p(\boldsymbol{y}_1,\boldsymbol{y}_2|\mathcal{H}_d)$ in Eq. (10) are independent of $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, we can choose any values for the latent variables, as long as the denominator is non-zero [6]. Substituting Eqs. (5), (6), (7), (8) in Eq. (11) and selecting $\boldsymbol{x}_0 = \boldsymbol{x}_1 = \boldsymbol{x}_2 = \boldsymbol{0}$, we obtain [9]

$$\begin{aligned}\text{score}(\boldsymbol{y}_1,\boldsymbol{y}_2) = &-\log\mathcal{N}(\boldsymbol{0}|\boldsymbol{\mu},\boldsymbol{B}^{-1}) - \log\mathcal{N}(\boldsymbol{0}|\boldsymbol{L}_{1,2}^{-1}\gamma_{1,2},\boldsymbol{L}_{1,2}^{-1})\\ &+\log\mathcal{N}(\boldsymbol{0}|\boldsymbol{L}_1^{-1}\gamma_1,\boldsymbol{L}_1^{-1}) + \log\mathcal{N}(\boldsymbol{0}|\boldsymbol{L}_2^{-1}\gamma_2,\boldsymbol{L}_2^{-1}),\end{aligned} \quad (12)$$

where $\boldsymbol{L}_{1,2} = \boldsymbol{B}+2\boldsymbol{W}, \gamma_{1,2} = \boldsymbol{B}\boldsymbol{\mu}+\boldsymbol{W}(\boldsymbol{y}_1+\boldsymbol{y}_2)$ and $\boldsymbol{L}_i = \boldsymbol{B}+\boldsymbol{W}, \gamma_i = \boldsymbol{B}\boldsymbol{\mu}+\boldsymbol{W}\boldsymbol{y}_i$ for $i \in \{1,2\}$. As described in [6], the score in Eq. (12) can now be rewritten as

$$\text{score}(\boldsymbol{y}_1,\boldsymbol{y}_2) = \boldsymbol{y}_1\boldsymbol{\Lambda}\boldsymbol{y}_2^T + \boldsymbol{y}_2\boldsymbol{\Lambda}\boldsymbol{y}_1^T + \boldsymbol{y}_1\boldsymbol{\Gamma}\boldsymbol{y}_1^T + \boldsymbol{y}_2\boldsymbol{\Gamma}\boldsymbol{y}_2^T + \big(\boldsymbol{y}_1+\boldsymbol{y}_2\big)^T\boldsymbol{\rho} + \kappa, \quad (13)$$

where the parameters $\boldsymbol{\Lambda}$, $\boldsymbol{\Gamma}$, $\boldsymbol{\rho}$ and κ of the quadratic function in Eq. (13) are given by

$$\begin{aligned}\boldsymbol{\Gamma} &= \frac{1}{2}\boldsymbol{W}^T\big(\widetilde{\boldsymbol{\Lambda}} - \widetilde{\boldsymbol{\Gamma}}\big)\boldsymbol{W}, & \boldsymbol{\Lambda} &= \frac{1}{2}\boldsymbol{W}^T\widetilde{\boldsymbol{\Lambda}}\boldsymbol{W},\\ \boldsymbol{\rho} &= \boldsymbol{W}^T\big(\widetilde{\boldsymbol{\Lambda}} - \widetilde{\boldsymbol{\Gamma}}\big)\boldsymbol{B}\boldsymbol{\mu}, & \kappa &= \widetilde{\kappa} + \frac{1}{2}\bigg(\big(\boldsymbol{B}\boldsymbol{\mu}\big)^T\big(\widetilde{\boldsymbol{\Lambda}} - 2\widetilde{\boldsymbol{\Gamma}}\big)\boldsymbol{B}\boldsymbol{\mu}\bigg)\end{aligned}$$

and the auxiliary variables are

$$\begin{aligned}&\widetilde{\boldsymbol{\Gamma}} = \big(\boldsymbol{B}+\boldsymbol{W}\big)^{-1}, \qquad \widetilde{\boldsymbol{\Lambda}} = \big(\boldsymbol{B}+2\boldsymbol{W}\big)^{-1},\\ &\widetilde{\kappa} = 2\log\det\big(\widetilde{\boldsymbol{\Gamma}}\big) - \log\det\big(\boldsymbol{B}\big) - \log\det\big(\widetilde{\boldsymbol{\Lambda}}\big) + \boldsymbol{\mu}^T\boldsymbol{B}\boldsymbol{\mu}.\end{aligned}$$

Hence, the verification score is a symmetric quadratic function of the LEVs. The probability for a same-author trial can be computed from the log-likelihood ratio score as follows:

$$p(\mathcal{H}_s|\boldsymbol{y}_1,\boldsymbol{y}_2) = \frac{p(\mathcal{H}_s)\,p(\boldsymbol{y}_1,\boldsymbol{y}_2|\mathcal{H}_s)}{p(\mathcal{H}_s)\,p(\boldsymbol{y}_1,\boldsymbol{y}_2|\mathcal{H}_s) + p(\mathcal{H}_d)\,p(\boldsymbol{y}_1,\boldsymbol{y}_2|\mathcal{H}_d)} \quad (14)$$

The AV datasets provided by the PAN organizers are balanced w.r.t. authorship labels. Hence, we can assume $p(\mathcal{H}_s) = p(\mathcal{H}_d) = \frac{1}{2}$. We can rewrite Eq. (14) as follows:

$$p(\mathcal{H}_s|\boldsymbol{y}_1,\boldsymbol{y}_2) = \frac{p(\boldsymbol{y}_1,\boldsymbol{y}_2|\mathcal{H}_s)}{p(\boldsymbol{y}_1,\boldsymbol{y}_2|\mathcal{H}_s) + p(\boldsymbol{y}_1,\boldsymbol{y}_2|\mathcal{H}_d)} = \text{Sigmoid}\big(\text{score}(\boldsymbol{y}_1,\boldsymbol{y}_2)\big) \quad (15)$$

Loss function: To learn the probabilistic layer, we incorporate Eq. (15) into the binary cross entropy:

$$\mathcal{L}_{\boldsymbol{\phi}} = l\cdot\log\{p(\mathcal{H}_s|\boldsymbol{y}_1,\boldsymbol{y}_2)\} + (1-l)\cdot\log\{1 - p(\mathcal{H}_s|\boldsymbol{y}_1,\boldsymbol{y}_2)\}, \quad (16)$$

where $\boldsymbol{\phi} = \big\{\boldsymbol{W},\boldsymbol{B},\boldsymbol{\mu}\big\}$ contains the trainable parameters of the probabilistic layer.

Cholesky decomposition for numerically stable covariance training: We can treat $\phi = \{\boldsymbol{W}, \boldsymbol{B}, \boldsymbol{\mu}\}$ given by Eqs. (5)$-$(6) as trainable paramters in our deep learning framework. For both covariance matrices we need to guarantee positive definiteness. Instead of learning $\boldsymbol{W}$ and $\boldsymbol{B}$ directly, we enforce the positive definiteness of them through Cholesky decomposition by constructing trainable lower-triangular matrices $\boldsymbol{L}_W$ and $\boldsymbol{L}_B$ with exponentiated (positive) diagonal elements. The estimated covariance matrices are constructed via $\widehat{\boldsymbol{W}} = \boldsymbol{L}_W \, \boldsymbol{L}_W^T$ and $\widehat{\boldsymbol{B}} = \boldsymbol{L}_B \, \boldsymbol{L}_B^T$. We computed and updated the gradients of $\boldsymbol{L}_W$ and $\boldsymbol{L}_B$ with respect to the loss function in Eq. (16).

2.3 Ensemble inference

Neural networks are randomly initialized, trained on the same, but shuffled data and affected by regularization techniques like dropout. Hence, they will find a different set of weights/biases each time, which in turn produces different predictions. To reduce the variance, we propose to train an ensemble of models and to combine the predictions of Eq. (15) from these models,

$$\mathbb{E}\big[p(\mathcal{H}_s|\boldsymbol{y}_1, \boldsymbol{y}_2)\big] \approx \frac{1}{m} \sum_{i=1}^{m} p_{\mathcal{M}_i}(\mathcal{H}_s|\boldsymbol{y}_1, \boldsymbol{y}_2), \tag{17}$$

where $\mathcal{M}_i$ indicates the i-th trained model. Finally, we determine the non-answers for predicted probabilities, i.e. $\mathbb{E}\big[p(\mathcal{H}_s|\boldsymbol{y}_1, \boldsymbol{y}_2)\big] = 0.5$, if $0.5 - \delta < \mathbb{E}\big[p(\mathcal{H}_s|\boldsymbol{y}_1, \boldsymbol{y}_2)\big] < 0.5 + \delta$. Parameter δ can be found by applying a simple grid search.

3 Text preprocessing strategies

The 2020 edition of the PAN authorship verification task focuses on *fanfiction* texts, fictional texts written by fans of previous, original literary works that have become popular like "Harry Potter". Usually, authors of fanfiction preserve core elements of the storyline by reusing main characters and settings. Nevertheless, they may also contain changes or alternative interpretations of some parts of the known storyline. The subject area of the original work is called *fandom*. The PAN organizers are providing unique author and fandom (topical) labels for all fanfiction pairs. The dataset has been derived from the corpus compiled in [3]. A detailed description of the dataset is given in [16].

As mentioned in the introduction, automatically extracted neural features should be invariant w.r.t. shifts in topic and/or sentiment. Ideally, LEVs should only contain information regarding the writing style of the authors. What is well-established in automatic AV is that the topic of a text generally matters. What is still not clear, however, is how stylometric or neural features are influenced/affected by the topic (i.e. fandom in this case). To increase the generalization capabilities of our model and to increase the model's resilience towards cross-topic fanfiction pairs we devised the following preprocessing strategies, as outlined in Sections 3.1 through 3.3.

3.1 Topic masking

Experiments show that considering a large set of token types can lead to significant overfitting effects. To overcome this, we reduced the vocabulary size for tokens as well as for characters by mapping all rare token/character types to a special *unknown*

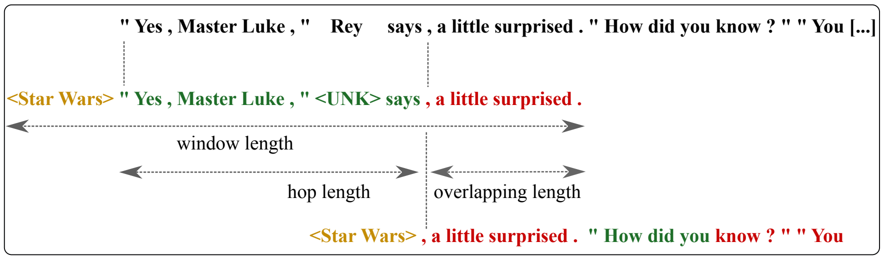

" Yes , Master Luke , " Rey says , a little surprised . " How did you know ? " " You [...]
<Star Wars> " Yes , Master Luke , " <UNK> says , a little surprised .
window length
hop length
overlapping length
<Star Wars> , a little surprised . " How did you know ? " " You

Figure 2. Example of our sliding window approach with contextual prefix.

(<UNK>) token. This is quite similar to the text distortion approach proposed in [21]. However, even when a rare/misspelled token is replaced by the <UNK> token, it can still be encoded by the character representation.

3.2 Sliding window with contextual prefix

Fanfiction frequently contains dialogues and quoted text. Sentence boundary detectors, therefore, tend to be very error prone and steadily fail to segment the data into appropriate sentence units. We decided to perform tokenization without strict sentence boundary detection and generated *sentence-like units* via a sliding window technique instead. An example that illustrates the procedure is shown in Fig. 2. We used overlapping windows to guarantee that semantically and grammatically linked neighboring tokens are located in the same unit. We also added a *contextual prefix* which is provided by the fandom labels. To initialize the prefix embeddings, we removed all non-ASCII characters, tokenized the fandom string and averaged the corresponding word embeddings. The final sliding window length (in tokens) is given by hop_length + overlapping_length + 1.

3.3 Data split and augmentation

To tune our model we split the datasets into a *train* and a *dev* set. Table 1 shows the resulting sizes. The size of the train set can then be increased synthetically by dissembling all predefined document pairs and re-sampling new same-author and different-author pairs in each epoch. We first removed all documents in the *train* set which also appear in the *dev* set. Afterwards, we reorganized the *train* set as described in Alg. 1 - 3. Assuming the i-th author with $i \in \{1, \dots, N\}$ contributes with N_i fanfiction texts, we define a set $\mathcal{A}^{(i)} = \{(a^{(i)}, f_1^{(i)}, d_1^{(i)}), \dots, (a^{(i)}, f_{N_i}^{(i)}, d_{N_i}^{(i)})\}$ containing 3-tuples of the form $(a^{(i)}, d_j^{(i)}, f_j^{(i)})$, where $a^{(i)}$ is the author ID, $d_j^{(i)}$ represents the j-th document and $f_j^{(i)}$ is the corresponding fandom label. The objective is to obtain a new set $\mathcal{D}$ of re-sampled pairs, containing 5-tuples of the form $(d^{(1)}, d^{(2)}, f^{(1)}, f^{(2)}, l)$, where $d^{(1)}$, $d^{(2)}$ defines the sampled fanfiction pair, $f^{(1)}$, $f^{(2)}$ are the corresponding fandom labels and $l \in \{0, 1\}$ indicates whether the texts are written by the same author ($l = 1$) or by different

	train set	**dev set**	**test set**
small dataset	47,340 pairs	5,261 pairs	14,311 pairs
large dataset	261,786 pairs	13,779 pairs	

Table 1. Dataset sizes (including the provided test set) after splitting.

Algorithm 1: MAKETWOGROUPS

```
Input: $\mathcal{A}^{(1)}, \dots \mathcal{A}^{(N)}$
Output: $\mathcal{G}^{(1)}, \mathcal{G}^{(2)}$
Initialize $\mathcal{G}^{(i)} = \{\emptyset\}$ for $i \in \{1,2,3\}$
for $i = 1, \dots, N$ do
    if $|\mathcal{A}^{(i)}| = |\{(a^{(i)}, f^{(i)}, d^{(i)})\}| = 1$ then
        // Authors with single doc
        $\mathcal{G}^{(1)} \longleftarrow \mathcal{G}^{(1)} \cup \{(a^{(i)}, f^{(i)}, d^{(i)})\}$
    else if $|\mathcal{A}^{(i)}| > 1$ and $|\mathcal{A}^{(i)}| \bmod 2 = 0$ then
        // Number of docs = 2,4,..
        $\mathcal{G}^{(2)} \longleftarrow \mathcal{G}^{(2)} \cup \{\mathcal{A}^{(i)}\}$
    else if $|\mathcal{A}^{(i)}| > 1$ and $|\mathcal{A}^{(i)}| \bmod 2 = 1$ then
        // Number of docs = 3,5,..
        $\mathcal{G}^{(3)} \longleftarrow \mathcal{G}^{(3)} \cup \{\mathcal{A}^{(i)}\}$
    end
end
/* Assign one doc of authors in
   group 3 to group 1, assign all
   remaining docs to group 2.    */
forall $\mathcal{A} \in \mathcal{G}^{(3)}$ do
    randomly draw $(a, f, d) \in \mathcal{A}$
    $\mathcal{G}^{(1)} \longleftarrow \mathcal{G}^{(1)} \cup \{(a, f, d)\}$
    $\mathcal{G}^{(2)} \longleftarrow \mathcal{G}^{(2)} \cup \{\mathcal{A} \setminus \{(a, f, d)\}\}$
end
```

Algorithm 2: CLEANAFTERSAMPLING

```
Input: $\mathcal{A}, \mathcal{D}, \mathcal{G}^{(1)}, \mathcal{G}^{(2)}$
Output: $\mathcal{D}, \mathcal{G}^{(1)}, \mathcal{G}^{(2)}$
if $|\mathcal{A}| > 1$ then
    /* Add to group 2, if at least
       two docs remain.            */
    $\mathcal{G}^{(2)} \longleftarrow \mathcal{G}^{(2)} \cup \{\mathcal{A}\}$
else if $|\mathcal{A}| = |\{(a, f, d)\}| = 1$ then
    /* Add to group 1, if only one
       doc remains and no doc was
       assigned to group 1 via
       MakeTwoGroups(). Otherwise,
       make another same-author
       pair.                       */
    if $\forall (a', f', d') \in \mathcal{G}^{(1)} \; \exists \, a' \; : \; a = a'$ then
        $\mathcal{D} \longleftarrow \mathcal{D} \cup \{(d, d', f, f', 1)\}$
        $\mathcal{G}^{(1)} \longleftarrow \mathcal{G}^{(1)} \setminus \{(a', f', d')\}$
    else
        $\mathcal{G}^{(1)} \longleftarrow \mathcal{G}^{(1)} \cup \{(a, f, d)\}$
    end
end
```

authors ($l = 0$). We obtain re-sampled pairs via $\mathcal{D} = \text{SAMPLEPAIRS}(\mathcal{A}^{(1)}, \dots, \mathcal{A}^{(N)})$ in Alg. 3. The epoch-wise sampling of new pairs can be accomplished beforehand to speed up the training phase.

4 Evaluation

Table 2 reports the evaluation results[3] for our proposed system over the *dev* set and the *test* set[4]. Rows 1-3 show the performance on the dev set and rows 6-8 show the corresponding results on the test set. We used the *early-bird* feature of the challenge to get a first impression of how our model behaves on the test data. The comparatively good results of our early-bird submission on the dev data (see row 1) suggest that our train and dev sets must be approximately stratified. Comparing these results with the significantly lower performance of the early-bird system on the test set (see row 6), however, indicates that there must be some type of intentional mismatch between the train set and the test set of the challenge. We suspect a shift in the relation between authors and fandom topics. For our early-bird submission we did not yet use the provided fandom labels. After the early-bird deadline, however, we incorporated the contextual prefixes. Comparing row 1 (without prefix) with row 4 (prefix included) we observe a noticeable improvement. One possible explanation for this improvement could be that the model is now better able to recognize stylistic variations between authors who are writing in the same fandom-based domain. If we compare rows 4 & 5 with rows 2 & 3, we see the

[3] The source code will be publicly available to interested readers after the peer review notification, including the set of hyper-parameters.

[4] The *test* set was not accessible to the authors. Results on the *test* set were generated by the organizers of the PAN challenge via the submitted program code.

Algorithm 3: SAMPLEPAIRS

```
Input: $\mathcal{A}^{(i)} = \{(a^{(i)}, f_1^{(i)}, d_1^{(i)}), \ldots, (a^{(i)}, f_{N_i}^{(i)}, d_{N_i}^{(i)})\} \; \forall i \in \{1, \ldots N\}$
Output: $\mathcal{D}$
Initialize $\mathcal{D} = \{\emptyset\}$
$\{\mathcal{G}^{(1)}, \mathcal{G}^{(2)}\} = \text{MAKETWOGROUPS}(\mathcal{A}^{(1)}, \ldots \mathcal{A}^{(N)})$
while $|\mathcal{G}^{(2)}| > 0$ or $|\mathcal{G}^{(1)}| > 1$ do
    // Sample same-author pair
    if $|\mathcal{G}^{(2)}| > 0$ then
        randomly draw $\mathcal{A} \in \mathcal{G}^{(2)}$
        $\mathcal{G}^{(2)} \longleftarrow \mathcal{G}^{(2)} \setminus \{\mathcal{A}\}$
        randomly draw $(a, f_1, d_1), (a, f_2, d_2) \in \mathcal{A}$
        $\mathcal{A} \longleftarrow \mathcal{A} \setminus \{(a, f_1, d_1), (a, f_2, d_2)\}$
        $\mathcal{D} \longleftarrow \mathcal{D} \cup \{(d_1, d_2, f_1, f_2, 1)\}$
        $\{\mathcal{D}, \mathcal{G}^{(1)}, \mathcal{G}^{(2)}\} \longleftarrow \text{CLEANAFTERSAMPLING}(\mathcal{A}, \mathcal{D}, \mathcal{G}^{(1)}, \mathcal{G}^{(2)})$
    end
    // Sample different-authors pair
    if $|\mathcal{G}^{(1)}| > 1$ then
        randomly draw $(a^{(1)}, f^{(1)}, d^{(1)}) \in \mathcal{G}^{(1)}$ and $(a^{(2)}, f^{(2)}, d^{(2)}) \in \mathcal{G}^{(1)}$
        $\mathcal{G}^{(1)} \longleftarrow \mathcal{G}^{(1)} \setminus \{(a^{(1)}, f^{(1)}, d^{(1)}), (a^{(2)}, f^{(2)}, d^{(2)})\}$
        $\mathcal{D} \longleftarrow \mathcal{D} \cup \{(d^{(1)}, d^{(2)}, f^{(1)}, f^{(2)}, 0)\}$
    else if $|\mathcal{G}^{(2)}| > 1$ then
        randomly draw $\mathcal{A}^{(1)}, \mathcal{A}^{(2)} \in \mathcal{G}^{(2)}$
        $\mathcal{G}^{(2)} \longleftarrow \mathcal{G}^{(2)} \setminus \{\mathcal{A}^{(1)}\mathcal{A}^{(2)}\}$
        randomly draw $(a^{(1)}, f^{(1)}, d^{(1)}) \in \mathcal{A}^{(1)}$ and $(a^{(2)}, f^{(2)}, d^{(2)}) \in \mathcal{A}^{(2)}$
        $\mathcal{A}^{(1)} \longleftarrow \mathcal{A}^{(1)} \setminus \{(a^{(1)}, f^{(1)}, d^{(1)})\}$ and $\mathcal{A}^{(2)} \longleftarrow \mathcal{A}^{(2)} \setminus \{(a^{(2)}, f^{(2)}, d^{(2)})\}$
        $\mathcal{D} \longleftarrow \mathcal{D} \cup \{(d^{(1)}, d^{(2)}, f^{(1)}, f^{(2)}, 0)\}$
        for $\mathcal{A} \in \{\mathcal{A}^{(1)}, \mathcal{A}^{(2)}\}$ do
            $\{\mathcal{D}, \mathcal{G}^{(1)}, \mathcal{G}^{(2)}\} \longleftarrow \text{CLEANAFTERSAMPLING}(\mathcal{A}, \mathcal{D}, \mathcal{G}^{(1)}, \mathcal{G}^{(2)})$
        end
    end
end
```

benefits of the proposed ensemble inference strategy. Combining a set of trained models leads to higher scores. Comparing rows 2 & 3 and rows 7 & 8 we find, unsurprisingly, that the training on the large dataset improves the performance results as well.

Besides the losses in Eqs. (3) and (16), we can also take into account the between-author and within-author variations to validate the training progress of our model. Both, between-author and within-author variations can be characterized by determining the entropy w.r.t. the estimated covariance matrices $\widehat{\boldsymbol{B}}^{-1}$ and $\widehat{\boldsymbol{W}}^{-1}$. It is well-known that entropy can function as a measure of uncertainty. For multivariate Gaussian densities, the analytic solution of the entropy is proportional to the determinant of the covariance matrix. From Eq. (5) and (6), we have

$$H\big(\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{B}^{-1})\big) \propto \log\det \widehat{\boldsymbol{B}}^{-1} \quad \text{and} \quad H\big(\mathcal{N}(\boldsymbol{\epsilon}|\boldsymbol{0}, \boldsymbol{W}^{-1}\big) \propto \log\det \widehat{\boldsymbol{W}}^{-1}. \quad (18)$$

Fig. 3 presents the entropy curves. As expected, during the training, the within-author variability decreased while the between-author variability increased.

Fig. 4 shows the attention-heatmaps of two fanfiction excerpts. From a visual inspection of many of such heatmaps we made the following observations: In contrast to the Amazon reviews used in [4], fanfiction texts do not contain a lot of "easy-to-visualize" linguistic features such as spelling errors for example. The model focuses on different aspects. Similar to [4], the model rarely marked function words (e.g. articles,

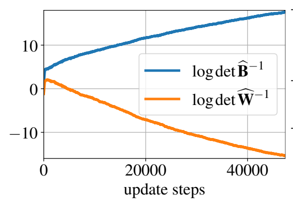
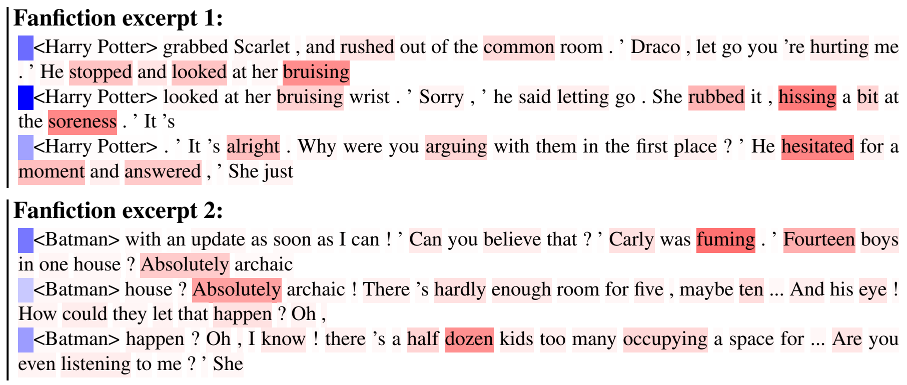


Figure 3. Entropy curves.

AdHominem	train set	evaluation	AUC	c@1	f_05_u	F1	overall
1 early-bird	small	dev set	0.964	0.919	0.916	0.932	0.933
2 ensemble	small	dev set	0.977	0.942	0.938	0.946	0.951
3 ensemble	large	dev set	0.985	0.955	0.940	0.959	0.960
4 single	small	dev set	0.975	0.943	0.921	0.951	0.948
5 single	large	dev set	0.983	0.950	0.944	0.954	0.958
6 early-bird	small	test set	0.923	0.861	0.857	0.891	0.883
7 ensemble	small	test set	0.940	0.889	0.853	0.906	0.897
8 ensemble	large	test set	0.969	0.928	0.907	0.936	0.935

Table 2. Results w.r.t the provided metrics on the dev and test sets.



Fanfiction excerpt 1:

<Harry Potter> grabbed Scarlet , and rushed out of the common room . ' Draco , let go you 're hurting me . ' He stopped and looked at her bruising

<Harry Potter> looked at her bruising wrist . ' Sorry , ' he said letting go . She rubbed it , hissing a bit at the soreness . ' It 's

<Harry Potter> . ' It 's alright . Why were you arguing with them in the first place ? ' He hesitated for a moment and answered , ' She just

Fanfiction excerpt 2:

<Batman> with an update as soon as I can ! ' Can you believe that ? ' Carly was fuming . ' Fourteen boys in one house ? Absolutely archaic

<Batman> house ? Absolutely archaic ! There 's hardly enough room for five , maybe ten ... And his eye ! How could they let that happen ? Oh ,

<Batman> happen ? Oh , I know ! there 's a half dozen kids too many occupying a space for ... Are you even listening to me ? ' She

Figure 4. Attention-heatmaps. Blue hues encode the sentence-based attention weights and red hues denote the relative word importance. All tokens are delimited by whitespaces.

pronouns, conjunctions). Surprisingly, punctuation marks like "`...`" seem to be less important than observed in [4]. In the first sentence of excerpt 1, the phrase "`stopped and looked`" is marked. In the second sentence, the word "`look`" of this phrase is repeated in the overlapping part but not marked anymore. Contrarily, repeated single words like "`Absolutely`" in excerpt 2 remain marked. It seems that our model is able to analyze how an author is using a word in a particular context.

Lastly, to keep the CPU memory requirements as low as possible on Tira [19], we fed every single test document separately and sequentially into the ensemble of trained models, resulting in a runtime of approximately 6 hours. This can, of course, be done batch-wise and in parallel for all models in the ensemble to reduce training time.

5 Conclusion and future work

We presented a new type of authorship verification (AV) system that combines neural feature extraction with statistical modeling. By recombining document-pairs after each training epoch, we significantly increased the heterogeneity of the train data. The proposed method achieved excellent overall performance scores, outperforming all other systems that participated in the PAN 2020 Authorship Verification Task, in both the small dataset challenge as well as the large dataset challenge. In AV there are many variabilities (such as topic, genre, text length, etc.) that negatively affect the system performance. Great opportunities for further gains can, thus, be expected by incorporating *compensation techniques* that deal with these aspects in future challenges.

Acknowledgment

This work was in significant parts performed on a HPC cluster at Bucknell University through the support of the National Science Foundation, Grant Number 1659397. Project funding was provided by the state of North Rhine-Westphalia within the Research Training Group "SecHuman - Security for Humans in Cyberspace."

References

1. Bagnall, D.: Author Identification using multi-headed Recurrent Neural Networks. In: CLEF Evaluation Labs and Workshop – Working Notes Papers (2015)
2. Bahdanau, D., Cho, K., Bengio, Y.: Neural Machine Translation by Jointly Learning to Align and Translate. In: Proc. ICLR (2015)
3. Bischoff, S., Deckers, N., Schliebs, M., Thies, B., Hagen, M., Stamatatos, E., Stein, B., Potthast, M.: The Importance of Suppressing Domain Style in Authorship Analysis. CoRR abs/2005.14714 (2020)
4. Boenninghoff, B., Hessler, S., Kolossa, D., Nickel, R.M.: Explainable Authorship Verification in Social Media via Attention-based Similarity Learning. In: Proc. IEEE BigData (2019)
5. Boenninghoff, B., Nickel, R.M., Zeiler, S., Kolossa, D.: Similarity Learning for Authorship Verification in Social Media. In: Proc. ICASSP (2019)
6. Niko Brümmer, Edward de Villiers: The speaker partitioning problem. In: Proc. Odyssey. ISCA (2010)
7. Brümmer, N.: A farewell to SVM: Bayes factor speaker detection in supervector space. Tech. rep. (2006)
8. Chung, J.S., Huh, J., Mun, S., Lee, M., Heo, H.S., Choe, S., Ham, C., Jung, S., Lee, B., Han, I.: In defence of metric learning for speaker recognition. CoRR abs/2003.11982 (2020)
9. Cumani, S., Brümmer, N., Burget, L., Laface, P., Plchot, O., Vasilakakis, V.: Pairwise Discriminative Speaker Verification in the I-Vector Space. IEEE Trans. Audio, Speech, Lang. Process. (2013)
10. DeGroot, M.: Optimal statistical decisions. McGraw-Hill (1970)
11. Ehrhardt, S.: Authorship attribution analysis. In: Visconti, J. (ed.) Handbook of Communication in the Legal Sphere. pp. 169–200. de Gruyter, Berlin/Boston (2018)
12. Halvani, O., Winter, C., Graner, L.: Assessing the Applicability of Authorship Verification Methods. In: Proc. ARES (2019)
13. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Neural Comp. (1997)
14. Hu, J., Lu, J., Tan, Y.P.: Discriminative Deep Metric Learning for Face Verification in the Wild. In: Proc. CVPR (2014)
15. Ioffe, S.: Probabilistic Linear Discriminant Analysis. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) Proc. ECCV (2006)
16. Kestemont, M., Manjavacas, E., Markov, I., Bevendorff, J., Wiegmann, M., Stamatatos, E., Potthast, M., Stein, B.: Overview of the Cross-Domain Authorship Verification Task at PAN 2020. In: Cappellato, L., Eickhoff, C., Ferro, N., Névéol, A. (eds.) CLEF 2020 Labs and Workshops, Notebook Papers. CEUR-WS.org (2020)
17. Litvak, M.: Deep Dive into Authorship Verification of Email Messages with Convolutional Neural Network. In: Proc. SIMBig (2018)
18. Potha, N.: Authorship Verification. Ph.D. thesis, University of the Aegean (2019)
19. Potthast, M., Gollub, T., Wiegmann, M., Stein, B.: TIRA Integrated Research Architecture. In: Ferro, N., Peters, C. (eds.) IR Evaluation in a Changing World. Springer (2019)
20. Stamatatos, E.: A Survey of Modern Authorship Attribution Methods. J. Assoc. Inf. Sci. Technol. (2009)
21. Stamatatos, E.: Authorship Attribution Using Text Distortion. In: Proc. EACL (2017)