SIMILARITY LEARNING FOR AUTHORSHIP VERIFICATION IN SOCIAL MEDIA

Benedikt Boenninghoff[1], *Robert M. Nickel*[2], *Steffen Zeiler*[1], *Dorothea Kolossa*[1]

[1]Faculty of Electrical Engineering and Information Technology, Ruhr University Bochum, Germany
[2]Department of Electrical and Computer Engineering, Bucknell University, Lewisburg, PA, USA

ABSTRACT

Authorship verification tries to answer the question if two documents with unknown authors were written by the same author or not. A range of successful technical approaches has been proposed for this task, many of which are based on traditional linguistic features such as n-grams. These algorithms achieve good results for certain types of written documents like books and novels. Forensic authorship verification for social media, however, is a much more challenging task since messages tend to be relatively short, with a large variety of different genres and topics. At this point, traditional methods based on features like n-grams have had limited success. In this work, we propose a new neural network topology for similarity learning that significantly improves the performance on the author verification task with such challenging data sets.

Index Terms— Authorship verification, forensic document analysis, Siamese network, similarity learning

1. INTRODUCTION

Social media platforms and text messaging have become a pervasive way of communication in the modern world. A hallmark of such systems is that the true identity of anyone accessing the systems is typically not verified. As a result, users can fall victim to false identity claims. These claims may be made for criminal purposes and/or the distribution of fake news and hate speech for example. An analysis of the authorship of a piece of text can help to reduce the success rate of potentially criminal perpetrators.

Forensic linguistics is the scientific discipline that performs *authorship analysis,* i.e. the analysis of text documents with respect to authorship, origination, the author's biographical background, and so forth [1]. The work is usually executed by highly-trained experts. The term *author profiling* tends to encompass both, the analysis of authorship of documents as well as the author's biographical background. The term *authorship attribution* describes a traditional closed-set classification task, where, given a set of candidate authors and documents, the objective is to determine which of the authors has written a set of anonymous or disputed documents [2, 3]. Lastly, in *authorship verification*, one merely attempts to determine whether two separate documents were written by the same author [4, 5].

Forensic linguistics has developed powerful tools for *authorship analysis, author profiling, authorship attribution,* and *authorship verification.* The sheer amount of data produced by social media networks and messaging services, however, makes it infeasible to solely rely on trained experts. Engineers have thus begun to develop technical means to process the data by machine or to at least preprocess the data to identify suspicious texts for a later review by experts. In the last few decades, different computational approaches have been published, especially for authorship attribution. They achieve good results for certain types of written documents such as novels, blog entries or news texts [1, 2, 3, 6, 7, 8, 9].

An important subtask for many of the proposed algorithms, e.g. [10, 11, 12], is the automatic extraction of linguistic features as a preprocessing step. In [1], for example, stylometric features are categorized into five different groups: lexical, character, syntactic, and semantic features, as well as application-specific features. Some of these, e.g. character or word n-grams, are easy to extract but more sensitive to a document's content than to its author. Syntactic features, on the other hand, are less sensitive to content but require the use of a robust part-of-speech tagger. In addition, non-linguistic features such as compression-based models were shown to be successful as well [13, 14]. The employed compression model, i.e. the *prediction by partial matching* approach, requires n-gram counting, however, and is therefore also sensitive to context.

Despite successful approaches in other domains, forensic authorship analysis for *social media* still remains a great technical challenge, primarily because small sample sizes, i.e. short texts, are quite common and a high variability of genre and topic choice is prevalent [3]. In addition, the writing styles from various social media types, such as email, blog entries, articles, and tweets differ significantly from those of news texts, which are typically used to train pre-processing tools like a part-of-speech tagger [15].

We are exclusively considering the authorship verification task for social media data in this work, i.e. we investigate similarities in the writing styles for two different social media texts with unknown authors [4, 5]. The technical core of our approach is implemented by a hierarchical recurrent Siamese neural network (HRSN). The recurrent neural network (RNN) topology allows us to automate the extraction of sensible and context-independent features, even if they may not be linguistically interpretable [16, 17].

The Siamese network topology was originally proposed in [18] to verify hand-written signatures. Applications in other domains, such as face verification for example, demonstrate the efficiency of the approach [19, 20, 21, 22]. In [23, 24] a recurrent Siamese network with long-short term memory (LSTM) cells was proposed to learn semantic similarities between sentences. Recurrent models based on LSTM cells have shown a remarkable ability to encode sentences into meaningful, yet compact vector representations [25]. In addition, RNNs are naturally suited for variable-length sequences, which affords them a structural advantage over convolutional neural network types (CNNs) [8, 9, 26, 27]. Furthermore, hierarchical structures with RNNs at multiple levels were proposed to handle not only sentences but also paragraphs and/or entire documents [16, 17].

The discriminative power of our proposed authorship verification method stems from a fusion of two disparate mechanisms into a single algorithm. Firstly, we are using the Siamese network concept with a specific contrastive loss function to entice maximal separation between same-author/different-author scores during training. Secondly, we are fusing a 2-level hierarchical RNN topology into the Siamese network, which produces fixed-length document representation vectors for variable length texts. The document representation

vectors encode those stylistic characteristics of a document that are relevant for authorship verification. Borrowing from the term *word embeddings* for a collection of vectors that characterize semantic categories for words, we chose to call our representation vectors *document embeddings* since they encode the stylistic characteristics of entire documents.

2. SIAMESE NETWORK TOPOLOGY

Siamese networks consist of two identical neural networks which share the same weights as illustrated in Figure 1. Traditional neural networks may learn to classify a given input. Siamese neural networks, however, learn to analyze the similarity of two inputs. In our case, the input to each RNN is a sequence of word embedding vectors $\boldsymbol{x}_i^{(\mathrm{w})}$ for each document i with $i \in \{1, 2\}$. Let $\boldsymbol{x}_i^{(\mathrm{d})}$ for $i \in \{1, 2\}$ be the pair of data points at the output of the RNN for each document. We refer to the $\boldsymbol{x}_i^{(\mathrm{d})}$ as *document embeddings*. The output of each sister network is not an easily interpretable object but rather an abstract nonlinear mapping into a high dimensional space. A distance measure, here, the Euclidean distance, is applied to the document embeddings, yielding a similarity measure. A final threshold decision indicates, whether the inputs are similar or not. In the following, we will present our hierarchical neural network topology.

2.1. Hierarchical Recurrent Siamese Network (HRSN)

The proposed encoder architecture is illustrated in Figure 2. The main objective is to feed word embeddings into the system in a sentence-by-sentence fashion until each document has been fully encoded. According to [16] we use the LSTM cell, a type of RNN designed to better learn long-term dependencies.

Let $\boldsymbol{x}_{t,n}^{(\mathrm{w})} \in \mathbb{R}^{D_\mathrm{w} \times 1}$ be the D_w-dimensional pretrained word embedding of the t-th word in the n-th sentence. The word-to-sentence encoding scheme can be written as

$$(\boldsymbol{h}_{t,n}^{(\mathrm{w})}, \boldsymbol{c}_{t,n}^{(\mathrm{w})}) = \mathrm{LSTM}_{\mathrm{w}\to\mathrm{s}}\big(\boldsymbol{x}_{t,n}^{(\mathrm{w})}, \boldsymbol{h}_{t-1,n}^{(\mathrm{w})}, \boldsymbol{c}_{t-1,n}^{(\mathrm{w})}\big), \tag{1}$$

where the hidden state and the memory state are denoted by $\boldsymbol{h}_{t,n}^{(\mathrm{w})} \in \mathbb{R}^{D_\mathrm{s} \times 1}$ and $\boldsymbol{c}_{t,n}^{(\mathrm{w})} \in \mathbb{R}^{D_\mathrm{s} \times 1}$, respectively. D_s defines the dimension of the sentence embeddings. The final states are given by $(\boldsymbol{h}_{T^{(\mathrm{w})},n}^{(\mathrm{w})}, \boldsymbol{c}_{T^{(\mathrm{w})},n}^{(\mathrm{w})})$, where the parameter $T^{(\mathrm{w})}$ denotes the fixed maximum number of words per sentence. The first-level update equations are given by

$$\begin{aligned}
\boldsymbol{f}_{t,n}^{(\mathrm{w})} &= \sigma(\boldsymbol{W}_f^{(\mathrm{w})}\, \boldsymbol{h}_{t-1,n}^{(\mathrm{w})} + \boldsymbol{U}_f^{(\mathrm{w})}\, \boldsymbol{x}_{t,n}^{(\mathrm{w})} + \boldsymbol{b}_f^{(\mathrm{w})}),\\
\boldsymbol{i}_{t,n}^{(\mathrm{w})} &= \sigma(\boldsymbol{W}_i^{(\mathrm{w})}\, \boldsymbol{h}_{t-1,n}^{(\mathrm{w})} + \boldsymbol{U}_i^{(\mathrm{w})}\, \boldsymbol{x}_{t,n}^{(\mathrm{w})} + \boldsymbol{b}_i^{(\mathrm{w})}),\\
\boldsymbol{o}_{t,n}^{(\mathrm{w})} &= \sigma(\boldsymbol{W}_o^{(\mathrm{w})}\, \boldsymbol{h}_{t-1,n}^{(\mathrm{w})} + \boldsymbol{U}_o^{(\mathrm{w})}\, \boldsymbol{x}_{t,n}^{(\mathrm{w})} + \boldsymbol{b}_o^{(\mathrm{w})}),\\
\tilde{\boldsymbol{c}}_{t,n}^{(\mathrm{w})} &= \tanh(\boldsymbol{W}_c^{(\mathrm{w})}\, \boldsymbol{h}_{t-1,n}^{(\mathrm{w})} + \boldsymbol{U}_c^{(\mathrm{w})}\, \boldsymbol{x}_{t,n}^{(\mathrm{w})} + \boldsymbol{b}_c^{(\mathrm{w})}),\\
\boldsymbol{c}_{t,n}^{(\mathrm{w})} &= \boldsymbol{f}_{t,n}^{(\mathrm{w})} \odot \boldsymbol{c}_{t-1,n}^{(\mathrm{w})} + \boldsymbol{i}_{t,n}^{(\mathrm{w})} \odot \tilde{\boldsymbol{c}}_{t,n}^{(\mathrm{w})},\\
\boldsymbol{h}_{t,n}^{(\mathrm{w})} &= \boldsymbol{o}_{t,n}^{(\mathrm{w})} \odot \tanh(\boldsymbol{c}_{t,n}^{(\mathrm{w})}),
\end{aligned} \tag{2}$$

where $\boldsymbol{W}_f^{(\mathrm{w})}, \boldsymbol{W}_i^{(\mathrm{w})}, \boldsymbol{W}_o^{(\mathrm{w})}, \boldsymbol{W}_c^{(\mathrm{w})} \in \mathbb{R}^{D_\mathrm{s} \times D_\mathrm{s}}$, $\boldsymbol{U}_f^{(\mathrm{w})}, \boldsymbol{U}_i^{(\mathrm{w})}, \boldsymbol{U}_o^{(\mathrm{w})}, \boldsymbol{U}_c^{(\mathrm{w})} \in \mathbb{R}^{D_\mathrm{s} \times D_\mathrm{w}}$ and $\boldsymbol{b}_f^{(\mathrm{w})}, \boldsymbol{b}_i^{(\mathrm{w})}, \boldsymbol{b}_o^{(\mathrm{w})}, \boldsymbol{b}_c^{(\mathrm{w})} \in \mathbb{R}^{D_\mathrm{s} \times 1}$. If the n-th sentence is shorter, i.e. for $t_n^{(\mathrm{w})} < T^{(\mathrm{w})}$, then the hidden state as well as the memory state are kept fixed for the remaining iterations, i.e.

$$\boldsymbol{h}_{t,n}^{(\mathrm{w})} = \boldsymbol{h}_{t_n^{(\mathrm{w})},n}^{(\mathrm{w})}, \tag{3}$$

$$\boldsymbol{c}_{t,n}^{(\mathrm{w})} = \boldsymbol{c}_{t_n^{(\mathrm{w})},n}^{(\mathrm{w})} \quad \forall t \in \{t_n^{(\mathrm{w})}, t_n^{(\mathrm{w})} + 1, \ldots, T^{(\mathrm{w})}\}, \tag{4}$$

which is readily incorporated into the backpropagation-through-time algorithm during training.

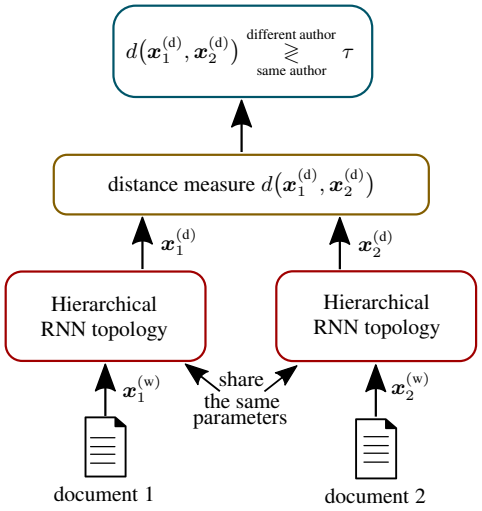

Fig. 1: General Siamese network topology.

The hidden and memory states are initialized with zero vectors, $\boldsymbol{h}_{0,n}^{(\mathrm{w})} = \mathbf{0}_{D_\mathrm{s} \times 1}$, $\boldsymbol{c}_{0,n}^{(\mathrm{w})} = \mathbf{0}_{D_\mathrm{s} \times 1}$. The hidden states at index $T^{(\mathrm{w})}$ are then used as the compressed sentence embeddings,

$$\boldsymbol{x}_n^{(\mathrm{s})} = \boldsymbol{h}_{T^{(\mathrm{w})},n}^{(\mathrm{w})} \quad \forall n \in \{1, \ldots, T^{(\mathrm{s})}\}. \tag{5}$$

On the next level, we have the sentence-to-document encoder represented by another LSTM cell,

$$(\boldsymbol{h}_n^{(\mathrm{s})}, \boldsymbol{c}_n^{(\mathrm{s})}) = \mathrm{LSTM}_{\mathrm{s}\to\mathrm{d}}\big(\boldsymbol{x}_n^{(\mathrm{s})}, \boldsymbol{h}_{n-1}^{(\mathrm{s})}, \boldsymbol{c}_{n-1}^{(\mathrm{s})}\big), \tag{6}$$

where sentence-based hidden and memory states are given by $\boldsymbol{h}_n^{(\mathrm{s})} \in \mathbb{R}^{D_\mathrm{d} \times 1}$ and $\boldsymbol{c}_n^{(\mathrm{s})} \in \mathbb{R}^{D_\mathrm{d} \times 1}$, respectively. D_d finally defines the dimension of the document embeddings and the final states are given by $(\boldsymbol{h}_{T^{(\mathrm{s})}}^{(\mathrm{s})}, \boldsymbol{c}_{T^{(\mathrm{s})}}^{(\mathrm{s})})$, where the parameter $T^{(\mathrm{s})}$ denotes the fixed maximum number of sentences per document. Analogously to the first level encoder, the second-level update equations are given by

$$\begin{aligned}
\boldsymbol{f}_n^{(\mathrm{s})} &= \sigma(\boldsymbol{W}_f^{(\mathrm{s})}\, \boldsymbol{h}_{n-1}^{(\mathrm{s})} + \boldsymbol{U}_f^{(\mathrm{s})}\, \boldsymbol{x}_n^{(\mathrm{s})} + \boldsymbol{b}_f^{(\mathrm{s})}),\\
\boldsymbol{i}_n^{(\mathrm{s})} &= \sigma(\boldsymbol{W}_i^{(\mathrm{s})}\, \boldsymbol{h}_{n-1}^{(\mathrm{s})} + \boldsymbol{U}_i^{(\mathrm{s})}\, \boldsymbol{x}_n^{(\mathrm{s})} + \boldsymbol{b}_i^{(\mathrm{s})}),\\
\boldsymbol{o}_n^{(\mathrm{s})} &= \sigma(\boldsymbol{W}_o^{(\mathrm{s})}\, \boldsymbol{h}_{n-1}^{(\mathrm{s})} + \boldsymbol{U}_o^{(\mathrm{s})}\, \boldsymbol{x}_n^{(\mathrm{s})} + \boldsymbol{b}_o^{(\mathrm{s})}),\\
\tilde{\boldsymbol{c}}_n^{(\mathrm{s})} &= \tanh(\boldsymbol{W}_c^{(\mathrm{s})}\, \boldsymbol{h}_{n-1}^{(\mathrm{s})} + \boldsymbol{U}_c^{(\mathrm{s})}\, \boldsymbol{x}_n^{(\mathrm{s})} + \boldsymbol{b}_c^{(\mathrm{s})}),\\
\boldsymbol{c}_n^{(\mathrm{s})} &= \boldsymbol{f}_n^{(\mathrm{s})} \odot \boldsymbol{c}_{n-1}^{(\mathrm{s})} + \boldsymbol{i}_n^{(\mathrm{s})} \odot \tilde{\boldsymbol{c}}_n^{(\mathrm{s})},\\
\boldsymbol{h}_n^{(\mathrm{s})} &= \boldsymbol{o}_n^{(\mathrm{s})} \odot \tanh(\boldsymbol{c}_n^{(\mathrm{s})}),
\end{aligned} \tag{7}$$

where $\boldsymbol{W}_f^{(\mathrm{s})}, \boldsymbol{W}_i^{(\mathrm{s})}, \boldsymbol{W}_o^{(\mathrm{s})}, \boldsymbol{W}_c^{(\mathrm{s})} \in \mathbb{R}^{D_\mathrm{d} \times D_\mathrm{d}}$, $\boldsymbol{U}_f^{(\mathrm{s})}, \boldsymbol{U}_i^{(\mathrm{s})}, \boldsymbol{U}_o^{(\mathrm{s})}, \boldsymbol{U}_c^{(\mathrm{s})} \in \mathbb{R}^{D_\mathrm{d} \times D_\mathrm{s}}$ and $\boldsymbol{b}_f^{(\mathrm{s})}, \boldsymbol{b}_i^{(\mathrm{s})}, \boldsymbol{b}_o^{(\mathrm{s})}, \boldsymbol{b}_c^{(\mathrm{s})} \in \mathbb{R}^{D_\mathrm{d} \times 1}$. If the document has fewer than the maximum number of sentences, i.e. for $n^{(\mathrm{s})} < T^{(\mathrm{s})}$, then the hidden and memory state are, again, kept fixed for the remaining iterations, i.e.

$$\boldsymbol{h}_n^{(\mathrm{s})} = \boldsymbol{h}_{n^{(\mathrm{s})}}^{(\mathrm{s})}, \tag{8}$$

$$\boldsymbol{c}_n^{(\mathrm{s})} = \boldsymbol{c}_{n^{(\mathrm{s})}}^{(\mathrm{s})} \quad \forall n \in \{n^{(\mathrm{s})}, n^{(\mathrm{s})} + 1, \ldots, T^{(\mathrm{s})}\}. \tag{9}$$

The fixed sentence length allows for unrolling the recurrent network and hence, for backpropagation-through-time training. The hidden and memory states are, again, initialized with zero vectors, $\boldsymbol{h}_0^{(\mathrm{w})} = \mathbf{0}_{D_\mathrm{d} \times 1}$, $\boldsymbol{c}_0^{(\mathrm{s})} = \mathbf{0}_{D_\mathrm{d} \times 1}$. The final hidden states are used as

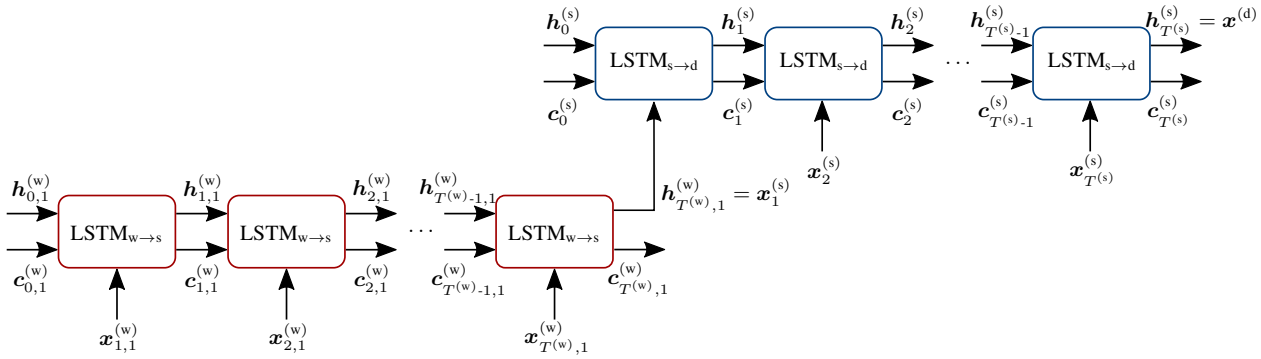

Fig. 2: LSTM-based words-to-document encoding scheme.

the compressed document embeddings,

$$\boldsymbol{x}^{(\mathrm{d})} = \boldsymbol{h}^{(\mathrm{w})}_{T^{(\mathrm{s})}}. \tag{10}$$

2.2. Distance measure and loss function

Given a pair of document embeddings, $\boldsymbol{x}^{(\mathrm{d})}_i$ for $i \in \{1, 2\}$ via Equation (10), we can measure the similarity of both documents by determining the *Eulidean* distance,

$$d\big(\boldsymbol{x}^{(\mathrm{d})}_1, \boldsymbol{x}^{(\mathrm{d})}_2\big) = \Big\|\boldsymbol{x}^{(\mathrm{d})}_1 - \boldsymbol{x}^{(\mathrm{d})}_2\Big\|_2 = \sqrt{\sum_{i=1}^{D^{(\mathrm{d})}} \Big(x^{(\mathrm{d})}_{1,i} - x^{(\mathrm{d})}_{2,i}\Big)^2}. \tag{11}$$

Hence, documents written by the same author result in small values for Equation (11), while the measure returns large values for documents of different authors.

The loss function should be chosen carefully. In real-world applications, it happens that two documents of the same author may treat diverse topics while two documents written by different authors may consider the same topic. Such *cross-topic* variations in the dataset leads to missclassifications when using context-sensitive features. The left part in Figure 3 illustrates this behavior. The distance between the cross-topic documents written by author A is larger than the distance between the same-topic documents of author A and B.

Thus, it is desirable to capture context-independent discriminative information through the obtained document embeddings. We can accomplish this by judiciously choosing our loss function during the training phase of the Siamese network. For cross-topic/same-author instances, the output of Equation (11) shall be smaller than a predefined positive threshold τ_1, i.e. $d\big(\boldsymbol{x}^{(\mathrm{d})}_1, \boldsymbol{x}^{(\mathrm{d})}_2\big) < \tau_1$. On the other hand, for same-topic/different-author instances the output of Equation (11) shall be larger than a second predefined threshold τ_2, i.e. $d\big(\boldsymbol{x}^{(\mathrm{d})}_1, \boldsymbol{x}^{(\mathrm{d})}_2\big) > \tau_2$ with $\tau_1 < \tau_2$. Both constraints can be incorporated into a modified version of the *contrastive* loss function [22],

$$\begin{aligned}\mathcal{L}\big(\boldsymbol{x}^{(\mathrm{d})}_1, \boldsymbol{x}^{(\mathrm{d})}_2\big) = \frac{l}{2} \cdot \max\Big\{d\big(\boldsymbol{x}^{(\mathrm{d})}_1, \boldsymbol{x}^{(\mathrm{d})}_2\big) - \tau_1, 0\Big\}^2 \\ + \frac{1-l}{2} \cdot \max\Big\{\tau_2 - d\big(\boldsymbol{x}^{(\mathrm{d})}_1, \boldsymbol{x}^{(\mathrm{d})}_2\big), 0\Big\}^2,\end{aligned} \tag{12}$$

which is applied to all training samples. The labels are defined as $l \in \{0, 1\}$, where $l = 1$ indicates that both texts are written by the same author and $l = 0$ means both documents are written by different authors. After training, the decision threshold may be chosen as

$$d\big(\boldsymbol{x}^{(\mathrm{d})}_1, \boldsymbol{x}^{(\mathrm{d})}_2\big) \underset{\text{same author}}{\overset{\text{different authors}}{\gtrless}} \tau = \frac{\tau_1 + \tau_2}{2}. \tag{13}$$

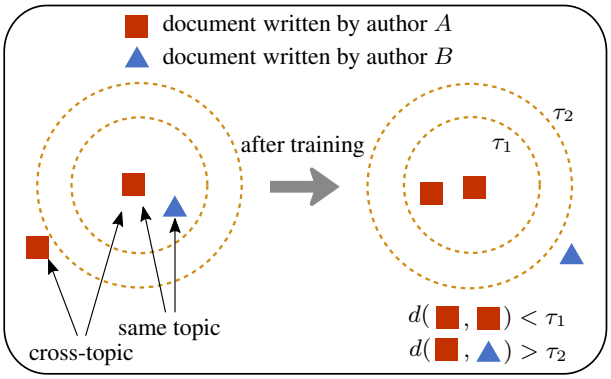

Fig. 3: Intuitive illustration of the loss function after [20].

3. EXPERIMENTAL RESULTS

In this section, we present experimental results for our proposed authorship verification model with real-world social media data.

3.1. Dataset

Currently, there are, unfortunately, not many standardized social media corpora publicly available to reproduce and compare authorship verification results. The dataset employed in this work is based on the one employed by Halvani *et al.* in [11], which also incorporates data provided by the PAN evaluation lab [28, 29, 30]. It is comprised of a collection of publicly available texts, including samples from different genres like novels, essays, tweets, news articles, social news, product reviews, blog entries, forum posts and emails. The data is specifically designed for authorship verification tasks. It is considered to be particularly challenging, because of its comparatively small size for training and because it contains not only *cross-topic* variations but also handles *cross-genre* conditions, since different text genres are available.

The accessable dataset (see [11, 28, 29, 30]) contains about 9300 instances of the form $\big(\mathcal{D}_{\text{known}}, d_{\text{unknown}}, l\big)$, where $\mathcal{D}_{\text{known}}$ is a set of documents from a known author, d_{unknown} defines a document in question and $l \in \{0, 1\}$ denotes the class label. The average number of tokens per sentence 16.8 with a standard deviation of ± 13.2 and the average number of sentences per document is 63.6 ± 30.2. A token can refer to a word, abbreviation or punctuation symbol. As mentioned in [11], the number of known documents in each instance varies from one to ten. In [11] all documents in $\mathcal{D}_{\text{known}}$ are concatenated into one single document, d_{known}. For the purpose of comparability we decided to do the same. After preprocessing, the average number of sentences per document is finally given by 160.4 ± 126.3.

3.2. Baseline Method

As our baseline reference, we implemented the authorship verification method published by Halvani *et al.* [11] in 2016. In [11] the authors reported that their approach outperforms two other state-of-the-art methods over the dataset described in Section 3.1.

The baseline method works as follows: a set of character-level features is constructed for each document according to [12]. Features are grouped into an odd number of different categories. A similarity function (see [11] for details) is applied to each feature pair from the known/unknown documents. A threshold for each category is used to accept or reject the hypothesis that the document in question belongs to the known author. Finally, a majority vote is applied with respect to the decisions of all feature categories. A training set is needed to find the optimum model parameters, e.g. the n-gram sizes as well as the equal-error-rate thresholds for all categories.

3.3. Training Details

Both algorithms, i.e. our proposed method and the baseline method, were implemented in Python. We utilized the library `textacy` for preprocessing to reduce the amount of noise in the data. For instance, we removed URLs, email addresses, and phone numbers and replaced them with universal tokens. Specific URLs, email addresses, and phone numbers are typically not part of an author's writing style. After text preprocessing we used `spaCy` for sentence boundary detection and tokenization. The training of the neural networks was accomplished with `Tensorflow`. The pretrained GloVe [31] word embeddings were taken from `spaCy`. Here, only a small number of $\approx 2\%$ of the segmented tokens were out of vocabulary. The following training details are mostly inspired by [16, 32, 33]:

- All LSTM parameters were initialized from a uniform distribution between -0.05 and 0.05.
- The size of the pretrained word embedding was given by $D_{\text{w}} = 300$. We then halved the dimensions for each next layer, i.e. $D_{\text{s}} = 150$ and $D_{\text{d}} = 75$.
- The maximum lengths were set to $T^{(\text{w})} = 33$ and $T^{(\text{s})} = 123$ to cover $> 90\%$ of the tokens of a single document in average.
- We noticed that using the Adadelta optimizer with a fixed initial learning rate of 1.0 yields the best results [34].
- Variational dropout on both LSTM levels was set with a rate of 0.3 [35, 33].
- Gradients were normalized by scaling them when the norm exceeded a threshold of 5.
- A batch size of 32 was chosen.
- 10-fold cross-validation was applied to test the proposed model. We constructed a development set from the training set such that 80% of the data was assigned to the training set, 10% to the development set, and another 10% to the test set. Since the baseline method does not require a development set, its training is performed on the entire training set (including development set).
- Data augmentation: We tried to artificially increase the number of training samples. After each epoch, we randomized the order in which documents $\mathcal{D}_{\text{known}}$ were concatenated into a single document d_{known}.
- Semi-supervised pretraining: Instead of randomly initializing the trainable LSTM parameters we also tried to pretrain the model. In a first step, a sentence-based Siamese network based on [23] was trained. For this purpose we constructed a labeled dataset from the SNLI corpus [36]. In a second step, a hierarchical neural autoencoder was used for unsupervised pretraining of $\text{LSTM}_{\text{s}\to\text{d}}$ [16]. The fixed parameters of the $\text{LSTM}_{\text{w}\to\text{s}}$ were then given by a previously trained sentence-based Siamese network. According to [37], the loss function during the unsupervised pretraining was the Euclidean distance. The unsupervised training was performed using the contents of the original Wikipedia paragraphs from [16].

3.4. Results

Table 1 summarizes the average verification accuracies over a 10-fold cross-validation for the proposed Siamese network and the baseline method. For the baseline method, we obtained an accuracy of $70.9\% \pm 1.7$, which is comparable to the average results reported in [11], where Table 7 shows $\approx 71.1\%$ accuracy for a fixed test set.

Comparing the results of both methods, it can readily be seen that the proposed approach significantly outperforms the baseline system. We were able to increase the average accuracy from around 71% to $83.2\% \pm 0.8$. An additional improvement was possible with the proposed data augmentation, described in Section 3.3. The resulting increase in the size of our training data set led to a further gain of around 2% in performance. Interestingly, no improvement was accomplished with the proposed pretraining, also described in Section 3.3. The reason for this is likely found in the substantial differences in the nature of the datasets used for pretraining in comparison to the data used for testing.

Table 1: Average precision, recall, F1-scores and verification accuracies for the test set over a 10-fold cross-validation.

	precision	recall	F1-score	accuracy
baseline	68.8 ± 2.5	67.8 ± 1.8	68.2 ± 1.9	70.9 ± 1.7
HRSN	81.7 ± 2.4	81.9 ± 2.7	81.7 ± 1.2	83.2 ± 0.8
HRSN (data augmentation)	84.3 ± 2.1	83.7 ± 2.1	83.9 ± 1.1	85.3 ± 0.9
HRSN (data augmentation + pretraining)	84.4 ± 1.6	81.8 ± 1.7	83.0 ± 0.8	84.7 ± 0.8

4. CONCLUSION

We introduced a hierarchical recurrent Siamese network topology for the task of authorship verification. A modified contrastive loss function was chosen during system training to effectively reduce the cross-topic sensitivities of the employed word embeddings. The proposed overall system was shown to be better adjusted than earlier methods [11] to the challenges posed by authorship verification with cross-topic and cross-genre social media texts. Improvements of about 15 percentage points in accuracy were observed.

Further work is still necessary for the development of an effective pretrainig strategy. Here, a better homogeneity between the pretraining data and the training data may be critical.

In addition, we may go beyond the proof-of-concept presented in this paper and study in greater detail the effects of the loss function with respect to cross-topic and same-topic instances. For this purpose, we intend to compile a larger dataset based on Amazon reviews [38] which is labeled w.r.t. authorship and review categories. Having more data, we are motivated to extend the existing framework: Inspired by [39] we may integrate a three-level attention mechanism for characters, words and sentences. Including character-based representations similar to the ones proposed in [40] may be beneficial to take the author-specific use of prefixes and suffixes into account. An advantage of the proposed method is, after all, that it is readily modified to incorporate additional text features separately on the character, word, sentence, and document level.

5. REFERENCES

[1] Efstathios Stamatatos, "A survey of modern authorship attribution methods," *J. Assoc. Inf. Sci. Technol.*, vol. 60, no. 3, pp. 538–556, 2009.

[2] Patrick Juola, "Authorship attribution," *Foundations and Trends in Information Retrieval*, vol. 1, no. 3, pp. 233–334, 2006.

[3] A. Rocha, W. J. Scheirer, C. W. Forstall, T. Cavalcante, A. Theophilo, B. Shen, A. R. B. Carvalho, and E. Stamatatos, "Authorship attribution for social media forensics," *IEEE Trans. Inf. Forensic Secur.*, vol. 12, no. 1, pp. 5–33, 2017.

[4] Moshe Koppel, Jonathan Schler, Shlomo Argamon, and Yaron Winter, "The "fundamental problem" of authorship attribution," *English Studies*, vol. 93, pp. 284–291, 2012.

[5] Moshe Koppel and Jonathan Schler, "Authorship verification as a one-class classification problem," in *Proc. ICML*. 2004, pp. 62–69, ACM.

[6] S. Segarra, M. Eisen, and A. Ribeiro, "Authorship attribution using function words adjacency networks," in *Proc. ICASSP*, 2013, pp. 5563–5567.

[7] Hugo Jair Escalante, Thamar Solorio, and Manuel Montes-y Gómez, "Local histograms of character n-grams for authorship attribution," in *Proc. ACL*. 2011, pp. 288–298, AC.

[8] Julian Hitschler, Esther van den Berg, and Ines Rehbein, "Authorship attribution with convolutional neural networks and POS-Eliding," in *Proc. StyleVar*. 2017, pp. 53–58, ACL.

[9] Sebastian Ruder, Parsa Ghaffari, and John G. Breslin, "Character-level and multi-channel convolutional neural networks for large-scale authorship attribution," *CoRR*, vol. abs/1609.06686, 2016.

[10] Moshe Koppel and Yaron Winter, "Determining if two documents are written by the same author," *Journal of the Association for Information Science and Technology*, vol. 65, no. 1, pp. 178–187.

[11] Oren Halvani, Christian Winter, and Anika Pflug, "Authorship verification for different languages, genres and topics," *Digital Investigation*, vol. 16, no. S, pp. S33–S43, 2016.

[12] Upendra Sapkota, Steven Bethard, Manuel Montes y Gomez, and Thamar Solorio, "Not all character n-grams are created equal: A study in authorship attribution," in *Proc. NAACL*. ACL, 2015, pp. 93–102.

[13] E. Frank, Chang Chui, and I. H. Witten, "Text categorization using compression models," in *Proceedings DCC 2000. Data Compression Conference*, 2000, pp. 555–.

[14] Oren Halvani, Christian Winter, and Lukas Graner, "On the usefulness of compression models for authorship verification," in *Proc. ARES*. 2017, ACM.

[15] Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith, "Part-of-speech tagging for twitter: Annotation, features, and experiments," in *Proc. ACL*. 2011, pp. 42–47, ACL.

[16] Jiwei Li, Minh-Thang Luong, and Dan Jurafsky, "A hierarchical neural autoencoder for paragraphs and documents," *CoRR*, vol. abs/1506.01057, 2015.

[17] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy, "Hierarchical attention networks for document classification," in *Proc. NAACL*, 2016, pp. 1480–1489.

[18] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah, "Signature verification using a "Siamese" time delay neural network," in *Proc. NIPS*. 1993, pp. 737–744, Morgan Kaufmann Publishers Inc.

[19] Sumit Chopra, Raia Hadsell, and Yann LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Proc. CVPR*. 2005, pp. 539–546, IEEE Computer Society.

[20] J. Hu, J. Lu, and Y. P. Tan, "Discriminative deep metric learning for face verification in the wild," in *Proc. CVPR*, 2014, pp. 1875–1882.

[21] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov, "Siamese neural networks for one-shot image recognition," 2015.

[22] J. Lu, J. Hu, and J. Zhou, "Deep metric learning for visual understanding: An overview of recent advances," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 76–84, 2017.

[23] Jonas Mueller and Aditya Thyagarajan, "Siamese recurrent architectures for learning sentence similarity," in *Proc. AAAI*. 2016, pp. 2786–2792, AAAI Press.

[24] Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru, "Learning text similarity with Siamese recurrent networks," in *Proc. RepL4NLP*. 2016, pp. 148–157, ACL.

[25] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le, "Sequence to sequence learning with neural networks," in *Proc. NIPS*. 2014, pp. 3104–3112, MIT Press.

[26] Yoon Kim, "Convolutional neural networks for sentence classification," *CoRR*, vol. abs/1408.5882, 2014.

[27] Prasha Shrestha, Sebastian Sierra, Fabio Gonzalez, Manuel Montes, Paolo Rosso, and Thamar Solorio, "Convolutional neural networks for authorship attribution of short texts," in *Proc. EACL*. 2017, pp. 669–674, ACL.

[28] Patrick Juola and Efstathios Stamatatos, "Overview of the author identification task at PAN 2013," in *CLEF (Working Notes)*, 2013, vol. 1179 of *CEUR Workshop Proceedings*.

[29] Efstathios Stamatatos, Walter Daelemans, Ben Verhoeven, Benno Stein, Martin Potthast, Patrick Juola, Miguel A. Sánchez-Pérez, and Alberto Barrón-Cedeño, "Overview of the author identification task at PAN 2014," in *CLEF (Working Notes)*, 2014, vol. 1180 of *CEUR Workshop Proceedings*, pp. 877–897.

[30] Efstathios Stamatatos, Walter Daelemans, Ben Verhoeven, Patrick Juola, Aurelio López-López, Martin Potthast, and Benno Stein, "Overview of the author identification task at PAN 2015," in *CLEF (Working Notes)*, 2015, vol. 1391 of *CEUR Workshop Proceedings*.

[31] Jeffrey Pennington, Richard Socher, and Christopher D Manning, "Glove: Global vectors for word representation.," in *EMNLP*. 2014, pp. 1532–1543, ACM.

[32] Ilya Sutskever, Oriol Vinyals, and Quoc V Le, "Sequence to sequence learning with neural networks," in *NIPS*, pp. 3104–3112. Curran Associates, Inc., 2014.

[33] Yarin Gal and Zoubin Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *NIPS*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds., pp. 1019–1027. Curran Associates, Inc., 2016.

[34] Matthew D. Zeiler, "ADADELTA: an adaptive learning rate method," *CoRR*, vol. abs/1212.5701, 2012.

[35] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *JMLR*, vol. 15, pp. 1929–1958, 2014.

[36] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning, "A large annotated corpus for learning natural language inference," in *Proc. EMNLP*. 2015, ACL.

[37] Andrew M Dai and Quoc V Le, "Semi-supervised sequence learning," in *Proc. NIPS*. 2015, pp. 3079–3087, Curran Associates, Inc.

[38] Ruining He and Julian McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *Proc. of the 25th International Conference on World Wide Web*, 2016.

[39] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy, "Hierarchical attention networks for document classification," in *Proc. NAACL*, 2016, pp. 1480–1489.

[40] Xuezhe Ma and Eduard H. Hovy, "End-to-end sequence labeling via Bi-directional LSTM-CNNs-CRF," *CoRR*, vol. abs/1603.01354, 2016.