Graph-based Siamese Network for Authorship Verification

Notebook for PAN at CLEF 2021

Daniel Embarcadero-Ruiz[1], Helena Gómez-Adorno[2], Ivan Reyes-Hernández[3], Alexis García[4] and Alberto Embarcadero-Ruiz[1]

[1]*Posgrado en Ciencia e Ingeniería de la Computación, Universidad Nacional Autónoma de México, Ciudad de México, México*

[2]*Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, Universidad Nacional Autónoma de México, Ciudad de México, México*

[3]*Facultad de Ciencias, Universidad Nacional Autónoma de México, Ciudad de México, México*

[4]*Facultad de Estudios Superiores Acatlán, Universidad Nacional Autónoma de México, Ciudad de México, México*

Abstract

Authorship verification is the task of determining whether or not two texts were written by the same author. The PAN@CLEF 2021 Authorship Verification challenge [1] requires to solve the task on a cross-topic and open-set collection of fanfiction texts. We propose a novel approach to extract features from text by first modeling it as a graph and then using a graph neural network to extract relevant features. We use a Siamese Network Architecture because it has shown good generalization on unseen classes in previous work related to verification tasks.

Keywords

Authorship verification, Graph neural networks, Text graphs

1. Introduction

Authorship verification is the task of determining whether or not two texts were written by the same author. To solve this task, we propose to represent the text as a graph structure and let a graph neural network to extract relevant features from this. Our motivation is that the graph structure provides additional information contained in the text that cannot be obtained when it is processed in the traditional sequential manner.

Our model uses a Siamese network architecture [2]. This network has two input layers to compare, in our case texts samples, and one output layer which its state value corresponds to the similarity between the two inputs.

CLEF 2021 – Conference and Labs of the Evaluation Forum, September 21–24, 2021, Bucharest, Romania

danielembru@gmail.com (D. Embarcadero-Ruiz); helena.gomez@iimas.unam.mx (H. Gómez-Adorno); ivanreyes@ciencias.unam.mx (I. Reyes-Hernández); 311115781@pcpuma.acatlan.unam.mx (A. García); albertoemru@gmail.com (A. Embarcadero-Ruiz)

https://helenagomez-adorno.github.io (H. Gómez-Adorno)

0000-0003-1904-1606 (H. Gómez-Adorno)

2. Related work

The problem of authorship verification was first tackled with linguistic approaches and eventually by statistic and computational methods [3]. In general, the most used strategy is to extract features from the text and use them to train a classification algorithm, which can be based either on supervised-learning or similarity. The features extracted from the texts are usually lexical – i.e. bag of words, vocabulary richness, misspelling words, character-level – n-grams, syntactic – ie. POS, chunks, sentence segmentation, and semantic – i.e. dependency trees, synonyms – [4].

Some classifiers used in this problem are support vector machines, decision trees, discriminant analysis, neural networks, and genetic algorithms [5]. The use of a heterogeneous classifier that combines independent classifiers each one with a different approach have achieved good results, usually better than the ones obtained using a single classifier [5, 6].

Before 2015, evaluation was mainly carried out using datasets with training and testing documents that shared the same topic and same genre. It has been observed that, in cross-topic datasets the performance of the traditional approaches decreases [6].

There are several graph-based representations used to model documents. The general approach consists in identifying relevant elements in the text - ie. words, sentences, paragraphs, etc. - and considering them as nodes in the graph. Then meaningful relations between these elements are generated as edges. Traditionally, the elements used as nodes in the graph are: words, sentences, paragraphs, documents, and concepts. To define the edges, usually syntactic, semantic relations and statistical counts are used [7].

In [8], the authors introduced different graph representations for authorship analysis, particularly suggested an enriched co-occurrence graph for authorship verification. Another graph-based approach for document understanding is presented in [9]. This approach considers different levels of linguistic analysis, such as lexical, morphological, syntactical and semantical, in order to build a graph representation of a given document. The authors also introduce a technique for extracting useful text patterns based on shortest paths.

Siamese Neural Networks (SNN) were first presented by Bromley et al. [2] to solve the problem of signature verification. Their formulation defines two separate sub-networks acts on each input pattern to extract features, then they use the cosine of the angle between two feature vectors to assign a distance between the compared instances.

Koch et al. [10] proposed a Siamese convolutional network to solve face verification. Instead of use a contrastive loss as objective loss they use a fully connected layer followed by a sigmoid function on top of the Siamese Network to get a prediction and optimize a cross entropy loss with L_2 normalization.

Last year, two approaches used SNNs to solve the PAN Authorship Verification Task [11]. Boenninghoff et al. [12] used an architecture of LSTMs with attention coefficients to extract first sentence embeddings and after a document embedding. Araujo-Pino et al. [13] used an adapted Residual Network with n-gram vectors as input.

3. Authorship Verification Dataset at PAN 2021

The dataset provided by the PAN@CLEF [14] organization was available in two sizes: the small dataset has 52,601 predefined pairs of texts and the large one has 275,565 pairs of texts. Each problem is composed of two fanfiction texts; a fanfiction is a original fan written history which extends the universe of a fandom topic. All texts are written in English, each one has an average of 21,400 characters, 4950 tokens and 345 sentences.

This year, the Authorship Verification task [1] focused on a cross-topic and open-set scenario. The test dataset has texts of authors and topics never seen in the training dataset. We submitted two models, both with the same architecture but trained using the small or large dataset.

To develop our model, we conducted several experiments, and for these we need to split the dataset in three parts: train, validation, and test. We trained our model on the train split using the validation split to calibrate the hyperparameters, and used the test split to get a reference score of the model. We did not use any of the samples in the test split to calibrate our model, so the score in this set tells us about the generalization ability of the model.

Our splits were done using the same fixed pairs given by the dataset. We made these splits author disjoint, that is, no text in one split has the same author of any text in a different split. At first we tried to do the splits also topic disjoint but it was difficult to achieve because of the natural distribution of topics and authors over pairs of texts. All the splits were defined with a balanced proportion of true and false problems.

After develop and calibrate our models we deployed them on TIRA [15] for testing it.

4. Modeling Texts as Graphs

We represent each text as a graph, our graph attempt to capture the relationships between words and POS labels in the text. To make clear our process we take the next text as an example: `Momo, also known as The Grey Gentlemen or The Men in Grey,`

First, each text is preprocessed in this way:

1. Substitute no ASCII characters to ASCII equivalent (we employed unidecode package[1]).
2. Tokenize and get the POS label.
3. Normalize to lowercase.

We don't remove punctuation of any type, in fact, the non-ASCII character substitution was made hopping it reduce the variability of the punctuation used. We consider the PENN-Treebank POS [2] labels as used in the NLTK package; to get the POS labels we use first the NLTK package and after that we add two additional labels: `$PUNCT` to mark all punctuation and `$OTHER` to mark any other word that the NLTK model failed to identify.

After the preprocessing we obtain a parsed text like this:

```
[('momo', 'NNP'), (',', '$PUNCT'), ('also', 'RB'),
 ('known', 'VBN'), ('as', 'IN'), ('the', 'DT'),
 ('grey', 'NNP'), ('gentlemen', 'NNP'), ('or', 'CC'),
```

[1] https://github.com/avian2/unidecode

[2] https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

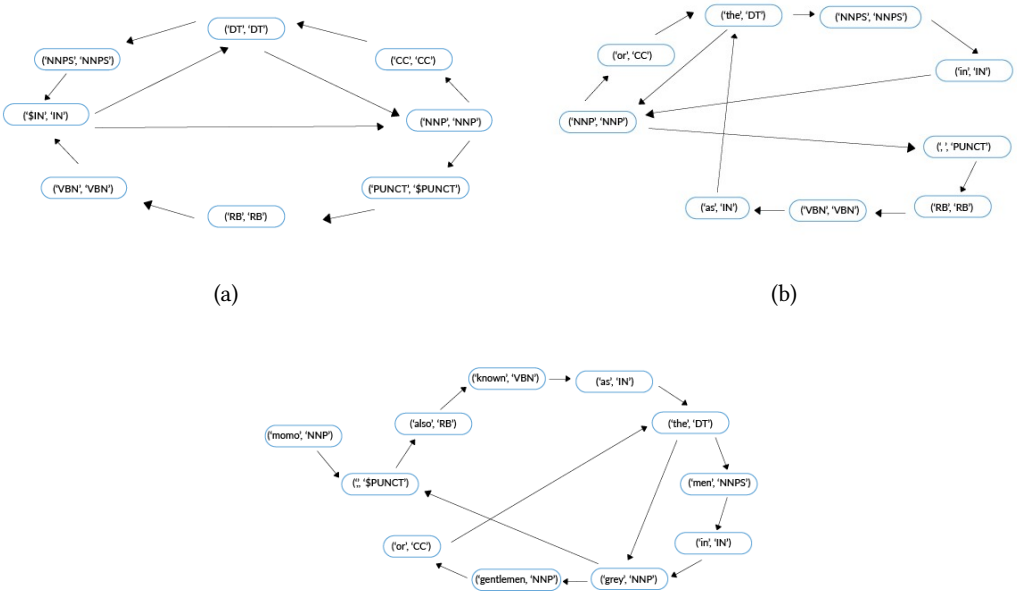

Figure 1: (a) Short graph, (b) Med graph, (c) Co-ocurrence graph , equivalent to the graph generated with empty *REDUCE LABLES* set

```
('the', 'DT'), ('men', 'NNPS'), ('in', 'IN'),
('grey', 'NNP'), (',', '$PUNCT')]
```

We will define a directed graph $G = \{V, E\}$ with tuples *(word, pos)* as nodes and weighted edges. We need to define the node set V and the edge set E, where each element in E is a tuple (n_1, n_2, w) with $n_1, n_2 \in V$ nodes and $w \in \mathbb{R}$ the edge weight. The construction is based in a co-occurrence graph, as explained in [7] with a different edge weighting; we used the Networkx[3] package to construct the graphs.

Let P be the parsed text as a list of tuples, $\ell(P)$ the number of elements in the list and $P[i]$ the i-th element in the list. Our graph construction is made as follow: We define a set of POS labels called REDUCE_LABELS; for each $P[i] = (word, pos)$ in P, we can define:

$$M[i] = \begin{cases} (word, pos) & \text{if pos } \notin \text{REDUCE\_LABELS} \\ (pos, pos) & \text{if pos } \in \text{REDUCE\_LABELS} \end{cases}$$

Where M is the list defined by the tuples masked as explained. For each pair of tuples $T_1, T_2 \in M$ let be $f(T_1, T_2)$ the number of times T_1 is followed by T_2 in M and let be $T = l(P) - 1 = l(M) - 1$; note that T is the total number of times a pair of tuples co-occur in M.

Now we can define the nodes and edges of our graph:

$$V = \{T \mid T \in M\}$$

[3] https://networkx.org/

Note than M is a list with order and V is just the set of all tuples in M. We want to define an edge between any two nodes (tuples) that appear together in the list M:

$$E = \{(T_1, T_2, \frac{f(T_1, T_2)}{T}) \mid T_1, T_2 \in M \wedge f(T_1, T_2) > 0\}$$

With this construction we identify all tuples form parsed text having an specific label in REDUCE_LABELS as a single node, so the graph structure changes and with it the information abstracted from text.

In our experiments we evaluated graphs generated with different REDUCE_LABELS sets. From now we will denominate *short graph* to the graph generated using the set of all possible POS labels as REDUCE_LABELS. We will denominate *med graph* to the graph generated the following set of REDUCE_LABELS:

```
REDUCE_LABELS = ['CD',                              # Cardinal numbers
                 'FW',                              # Foreign words
                 'JJ', 'JJR', 'JJS',                # Adjectives
                 'LS',                              # List item marker
                 'NN', 'NNS', 'NNP', 'NNPS',        # Nouns
                 'RB', 'RBR', 'RBS',                # Adverbs
                 'SYM',                             # Symbols
                 'VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ',  # Verbs
                 '$OTHER'                           # Others]
```

Continuing with our example, the *med graph* and the *short graph* are showed in 1(a) and 1(b) respectively. To make clearer the construction process Figure 1(c) show the construction with empty REDUCE_LABLES set. For simplicity we don't draw the edge weights. To feed our graph to a neural network, we need to transform the node names into vectors. We use a one hot encoder of the POS label as the initial node features.

5. Graph-based Siamese Network (GBSN)

To approach the authorship verification task, we use a Siamese Network architecture. Our Graph-based Siamese Network is composed by two identical feature extraction components with shared weights, a reduction step and a classification network. Each feature extraction component receives a text, transform it to a graph and returns a vector from this graph; the objective is to extract relevant features than can identify the author style from the graph representation of our text.

Figure 2 shows our proposed architecture. We can distinguish three parts in the feature extraction component: first graph representation, second node embedding layers and later a global pooling. The graph representation is constructed as explained in Section 4.

The node embedding layers obtain a relevant vector representation of each node in the graph; each layer is composed of a Local Extreme Convolution (LEConv) as defined in [16], followed by a batch normalization layer and a ReLU activation function, in total we use six of these layers. This layer was originally proposed to compute scores used to select relevant clusters in a graph,

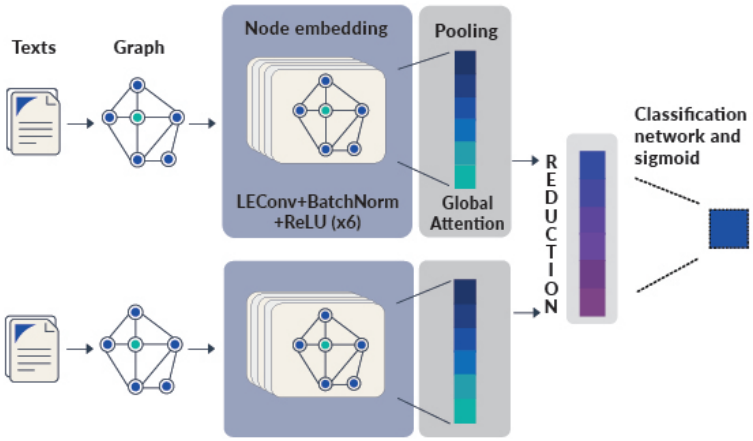

Figure 2: GBSN base architecture

the authors prove that it is more expressive than other layers like the Graph Convolutional Network layer as defined by Kipf and Welling [17] and affirm it has the ability of considering both local and global importance of nodes.

For the pooling layer we choose to use global attention layer, originally proposed by Li et al. [18]. This layer takes the final out of the node feature extraction component as its input, that is, a graph with the vector embedding in each node; to obtain the final vector, it makes a weighted sum of each node vector with a coefficient obtained by doing attention over these same vectors, the formulation is:

$$r = \Sigma_{n \in V}\ softmax(h(x_n)) \cdot x_n$$

Where V is the set of all nodes in the graphs and h is a four layer fully connected neural network with ReLU activation and output a single scalar.

For the reduction step, we simply compute the absolute value of the difference between the out of each feature extraction component, this single vector is passed to a final classification network. The classification network used is a five layer fully connected network with ReLU activation and final sigmoid function.

For training, we use ADAM optimizer and binary cross entropy as loss function, in each epoch all the pairs in the train split are supply to the Siamese network in shuffled order. We calculate the loss and average of the five scores considered for evaluation in the validation split, with these we select the best epoch of the model and choose it as our best model.

Our model was trained to return an output between 0 and 1. Given a threshold th and a margin m we can transform the original output out_o to a new one with the formula:

$$l(out_o) = \begin{cases} \frac{out_o}{2 \cdot (th - m)} & if\ out_o < th - m \\ 0.5 & if\ th - m \leq out_o \leq th + m \\ \frac{out_o - 1}{2 \cdot (1 - (th + m))} + 1 & if\ th + m < out_o \end{cases}$$

This linearly transforms the interval $[0, th - m)$ into $[0, 0.5)$, the interval $[th - m, th + m]$ into 0.5 and the interval $(th + m, 1]$ into $(0.5, 1]$.

Using this transformation, as a final tune, we preformed a grid search over different threshold and margins to optimize the average of the scores proposed in the PAN@CLEF shared task [1].

5.1. Ensemble architecture

We notice that using more than one graph representation and adding traditional stylistic features can improve the performance of our model (See Table 1). We upgrade our base architecture putting side by side different feature extraction components in a new Graph-based Siamese Network Ensemble architecture. Each subnetwork in the Siamese architecture is now formed by several independent feature extraction components, each one getting a vector from the text, as shown in Figure 3. We concatenate the vectors obtained by each component and this one is used in the reduction step.

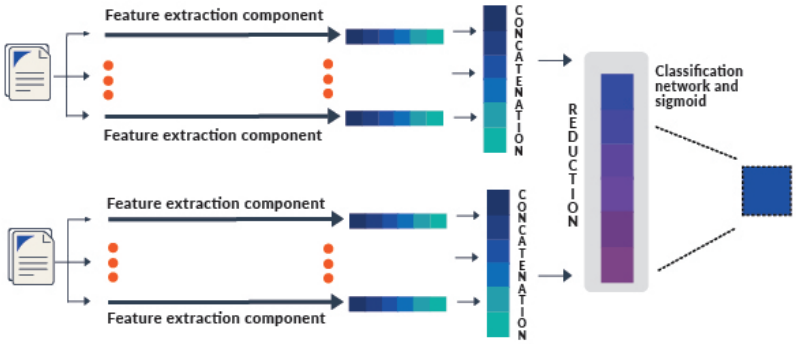

Figure 3: Graph-based Siamese Network Ensemble architecture

For each feature extraction components we use the same components used in the base architecture, so these components will be identified as *small graph* or *med graph* components. Also, we may include a vector of stylistic features of texts as another independent component. The stylistic features were selected from the ones used by Weerasinghe and Greenstadt [19]. For simplicity we use only:

- Frequency of function words (179 function words in the NLTK list)
- Average number of characters per word
- Vocabulary richness
- Frequency of words of *n* characters (from 1 to 10 characters)

We evaluated two training strategies: 1) to train all the network together, and 2) transferring the weights from a base instance of the corresponding GBSN. The later means training first one GBSN with only the short graph component and another one with the med graph component and define the ensemble GBSM using the feature extraction components weights in the corresponding weights. Transferring the weights achieved better results, so we define our model in

this way; this behavior was probably because training both components together did not let each component to stop at the best individual weight configuration due to our experimental framework.

In our experiments, we notice that using an ensemble of two instances of the same graph representation also gives us better results than using a single instance. For our final submissions, we use an ensemble of five feature extraction components in total: Two instances of a short graph feature extraction, two instances of a med graph features extraction and the stylistic features. Figure 4 shows our submitted architecture.

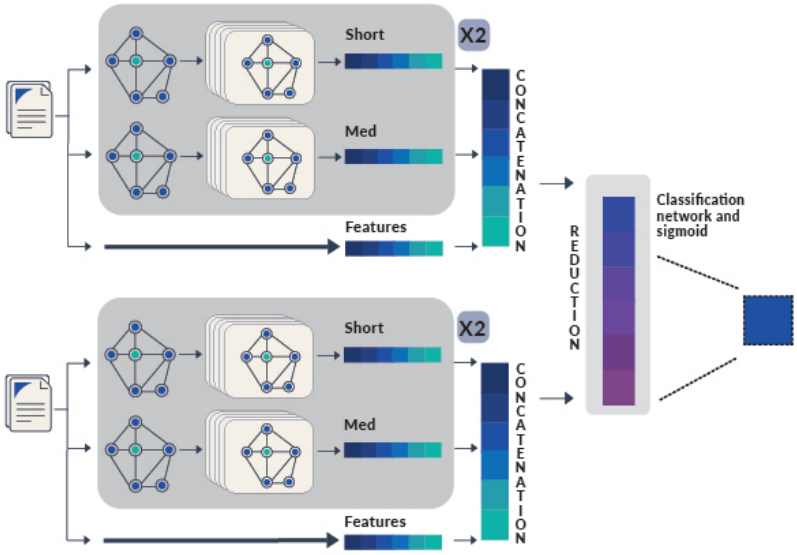

Figure 4: Final model submission: Ensemble with two *small graph*, two *med graph* and stylistic features

6. Results

Table 1 shows the comparative performance of the different configurations, each row shows the average of the five proposed scores in a model, trained in the small and large dataset respectively. The first row corresponds to a base architecture model using only the short graph component for feature extraction. The second row corresponds to a base architecture model using only the med graph component. The third row shows the scores of an ensemble architecture model using both short graph and med graph components but without stylistic features. The fourth row shows the scores of an ensemble architecture model using short graph, med graph and stylistic features components. Finally the fifth row shows the scores of our submitted model, an ensemble architecture with two short graph components, two med graph components and stylistic features.

Finally, our two submissions were scored in the test dataset of the PAN 2021 Authorship Verification task [1]. We obtained the second best score for the model trained in the large dataset

Table 1
Performance of the Graph-based Siamese Network (GBSN) with single and ensemble feature extraction components in our test split dataset

	Small dataset 47,336 problems	Large dataset 247,992 problems
Short graph component	87.81	90.63
Med graph component	88.31	91.50
Short + Med	89.21	92.38
Short + Med + Features	89.8	NA
Short(x2) + Med(x2) + Features (Submitted)	**90.03**	**92.96**

and the third best score for the model trained in the small dataset. Table 2 shows the average of the five metrics proposed.

We want to note that the scores in the test dataset were higher that the ones obtained in our test split. This was probably because we made our splits author disjoint, so in our experiments, we found models capable to solve the task for authors never seen in training.

Table 2
Performance of the Graph-based Siamese Network (GBSN) ensemble in the test dataset of PAN 2021

	Small dataset	Large dataset
Short(x2) + Med(x2) + Features (Submitted)	**90.70**	**93.59**

7. Conclusions

In this paper, we presented our approach for the authorship verification task at PAN 2021 [1], which is based on a novel Graph-based Siamese Network. In our experiments, we observed a good performance and great potential of using a graph representation of the text, graph convolutional neural networks and a siamese architecture.

We want to notice that the convolutional layers used in our model are relatively simple. We experimented by varying the architecture and hyperparameters to improve the model, but there are a lot of different convolutional and pooling graph layers configurations to try in future work.

We do not use any up-sampling technique over the given dataset. In fact, because of our train framework, for each dataset, we train our model using only 90% of the available data; that is, the small model was training using 47,336 problems and the large model using 247,992 problems; this is relevant because our architecture showed to work better with more training pairs.

Acknowledgments

This research was partially funded by CONACyT PNPC scholarship with No. CVU 1004062,

DGAPA-UNAM PAPIIT grant number TA100520. The authors thank CONACYT for the computer resources provided through the INAOE Supercomputing Laboratory's Deep Learning Platform for Language Technologies.

References

[1] M. Kestemont, I. Markov, E. Stamatatos, E. Manjavacas, J. Bevendorff, M. Potthast, B. Stein, Overview of the authorship verification task at PAN 2021, in: [14], 2021.

[2] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, R. Shah, Signature Verification using a "Siamese" Time Delay Neural Network, International Journal of Pattern Recognition and Artificial Intelligence 7 (1993) 669–688. doi:10.1142/S0218001493000339.

[3] S. Mekala, V. V. Bulusu, A Survey On Authorship Attribution Approaches, in: International Journal of Computational Engineering Research (IJCER), volume 8, 2018, p. 8.

[4] E. Stamatatos, A survey of modern authorship attribution methods, Journal of the American Society for Information Science and Technology 60 (2009) 538–556. doi:10.1002/asi.21001.

[5] E. Stamatatos, W. Daelemans, B. Verhoeven, M. Potthast, B. Stein, P. Juola, M. A. Sanchez-Perez, A. Barrón-Cedeño, Overview of the Author Identification Task at PAN 2014, Proceedings of the CLEF PAN Conference (2014) 21.

[6] E. Stamatatos, W. Daelemans, B. Verhoeven, P. Juola, A. López-López, M. Potthast, B. Stein, Overview of the Author Identification Task at PAN 2015, Proceedings of the CLEF PAN Conference (2015) 17.

[7] S. S.Sonawane, P. A. Kulkarni, Graph based Representation and Analysis of Text Document: A Survey of Techniques, International Journal of Computer Applications 96 (2014) 1–8. doi:10.5120/16899-6972.

[8] E. Castillo, O. Cervantes, D. Vilariño, Text Analysis Using Different Graph-Based Representations, Computación y Sistemas 21 (2017) 581–599. doi:10.13053/cys-21-4-2551.

[9] D. Pinto, H. Gomez Adorno, D. Vilariño, V. Singh, A graph-based multi-level linguistic representation for document understanding, Pattern Recognition Letters 41 (2014) 93–102. doi:10.1016/j.patrec.2013.12.004.

[10] G. Koch, R. Zemel, R. Salakhutdinov, Siamese Neural Networks for One-shot Image Recognition, ICML deep learning workshop 2 (2015) 8.

[11] M. Kestemont, E. Manjavacas, I. Markov, J. Bevendorff, M. Wiegmann, E. Stamatatos, M. Potthast, B. Stein, Overview of the Cross-Domain Authorship Verification Task at PAN 2020 (2020) 14.

[12] B. Boenninghoff, J. Rupp, R. M. Nickel, D. Kolossa, Deep Bayes Factor Scoring for Authorship Verification, Notebook for PAN at CLEF 2020 (2020) 12.

[13] E. Araujo-Pino, H. Gómez-Adorno, G. Fuentes-Pineda, Siamese Network applied to Authorship Verification, Notebook for PAN at CLEF 2020 (2020) 8.

[14] J. Bevendorff, B. Chulvi, G. L. D. L. P. Sarracén, M. Kestemont, E. Manjavacas, I. Markov, M. Mayerl, M. Potthast, F. Rangel, P. Rosso, E. Stamatatos, B. Stein, M. Wiegmann, M. Wolska, a. E. Zangerle, Overview of PAN 2021: Authorship Verification,Profiling hate speech

spreaders on Twitter,and style change detection, in: 12th International Conference of the CLEF Association (CLEF 2021), Springer, 2021.

[15] M. Potthast, T. Gollub, M. Wiegmann, B. Stein, TIRA integrated research architecture, in: N. Ferro, C. Peters (Eds.), Information Retrieval Evaluation in a Changing World, The Information Retrieval Series, Springer, Berlin Heidelberg New York, 2019. doi:10.1007/978-3-030-22948-1_5.

[16] E. Ranjan, S. Sanyal, P. P. Talukdar, ASAP: Adaptive Structure Aware Pooling for Learning Hierarchical Graph Representations, Thirty-Fourth AAAI Conference on Artificial Intelligence (2020). arXiv:1911.07979.

[17] T. N. Kipf, M. Welling, Semi-Supervised Classification with Graph Convolutional Networks, Conference paper at ICLR (2017). arXiv:1609.02907.

[18] Y. Li, D. Tarlow, M. Brockschmidt, R. Zemel, Gated Graph Sequence Neural Networks, Proceedings of ICLR (2017). arXiv:1511.05493.

[19] J. Weerasinghe, R. Greenstadt, Feature Vector Difference based Neural Network and Logistic Regression Models for Authorship Verification, Notebook for PAN at CLEF 2020 (2020) 6.