September 5–8, 2022, Bologna, Italy

Graph-Based Siamese Network for Authorship Verification

Notebook for PAN at CLEF 2022

Jorge Alfonso Martinez-Galicia[1], Daniel Embarcadero-Ruiz[2], Alejandro Ríos-Orduña[3] and Helena Gómez-Adorno[4]

[1]*Posgrado en Ciencia e Ingeniería de la Computación, Universidad Nacional Autónoma de México, CDMX, México*
[2]*Posgrado en Ciencia e Ingeniería de la Computación, Universidad Nacional Autónoma de México, CDMX, México*
[3]*Facultad de Ciencias, Universidad Nacional Autónoma de México, CDMX, México*
[4]*Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, Universidad Nacional Autónoma de México, Ciudad de México, México*

Abstract

Authorship verification is the task of determining whether or not the same author wrote two texts based on comparing the texts. The PAN@CLEF 2022 [1] Authorship Verification challenge [2] requires solving the task on a cross-discourse type and open-set collection of essays, emails, text messages, and business memos. Our approach is extracting features from the text by modeling it as a graph and using a graph neural network to identify 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, Text graphs, Graph neural networks, Siamese network

1. Introduction

The Authorship analysis research field study the characteristics that help to define an author's writing style. The features can be extracted using text samples of the authors. This reseach area includes different tasks such as authorship attribution, author profiling, author clustering, and plagiarism detection [3]. The authorship verification task aims at determining if the same author wrote two given texts.

To approach the authorship verification task at PAN 2022, we used a Siamese network architecture composed of two graph convolutional neural networks, pooling, and classification layers as introduced in [4]. We also evaluated the three strategies (short, med, and full) for representing texts as graphs based on the relation of the Part of Speech (POS) labels and the

CLEF 2022 – Conference and Labs of the Evaluation Forum, September 5–8, 2022, Bologna, Italy
jorgetonatiuhmg@gmail.com (J. A. Martinez-Galicia); danielembru@gmail.com (D. Embarcadero-Ruiz); aro-6@ciencias.unam.mx (A. Ríos-Orduña); helena.gomez@iimas.unam.mx (H. Gómez-Adorno)
https://helenagomez-adorno.github.io (H. Gómez-Adorno)
0000-0002-6966-9912 (H. Gómez-Adorno)

co-occurrence of words. The graph representation provides structural information that can help us distinguish writing styles independently of the discourse type. The source code of our approach is freely available at out Github repository [1]

This paper is structured as follows: Section 2 presents a brief summary of related works on authorship analysis. Section 3 describes the dataset used for the task. Section 4 presents the the text graph representation used to feed the Siamese network. Section 5 describes the Graph-based Siamese network, while the experiments and the obtained results are presented in Section 5. Section 7 presents the conclusions and future work.

2. Related work

Traditional authorship analysis relies on feature extraction to train a classification algorithm through supervised learning or similarity measures. The extracted feature method can be at any level of language description. The semantic level, i.e., semantic dependencies, synonyms. The syntactic level, i.e., chunks, POS tags, sentence, and phrase structure. The character level, i.e., character types, character n-grams, count of special characters. The lexical level, i.e., misspelled words, sentence length, word length, a bag of words, vocabulary richness [5].

Some of the most commonly used supervised classification algorithms used in authorship verification analysis are discriminant analysis, support vector machines, decision trees, neural networks, and genetic algorithms [6].

Another option that can effectively model relationships and structural information is the mathematical construct graph representation. This representation can be possible using feature terms as vertices and significant relations between the feature terms as edges [7]. The graph-based approach consists of identifying relevant elements in the text, i.e., words, sentences, paragraphs, etc., and modeling them as nodes in the graph. Then meaningful relations between these elements are considered to be edges. Typically, the features used as nodes in the graph are words, sentences, paragraphs, documents, and concepts. To define the edges, syntactic, semantic relations, and statistical counts are usually used [7].

This paper highlights the importance of enriched vs. non-enriched co-occurrence graphs as an alternative to traditional feature representation models such as vector representation [7]. There are many applications where data can be represented as a graph. While deep learning effectively captures hidden patterns of Euclidean data, graph neural networks can help us generalize the deep learning approach to data represented as graphs [8].

Bromley et al. [9] introduce Siamese Neural Networks (SNN) to solve the problem of signature verification. The network architecture consists of two separate sub-networks, each acting on an input pattern to extract features. The two sub-networks share their weights; that means that both sub-networks must receive the features in the same way. This architecture uses the cosine of the angle between the two feature vectors obtained by the sub-networks to assign a distance between the compared instances. The idea is that the siamese network learns how to extract feature vectors from the instances in a way these vectors are close if the instances are similar, and these vectors are far if that is not the case. SNNs are generally computationally expensive but perform better compared to other techniques when learning similarity [10].

[1] https://github.com/PLN-disca-iimas/AuthorshipVerification-GraphBasedSiameseNetwork

The Siamese neural network approach has successfully solved several verification-related tasks. We consider this method a natural and powerful way to take advantage of a graph-based representation of texts.

3. Authorship Verification Dataset at PAN 2022

The training dataset provided by the PAN@CLEF 2022 [1] organization consists of cross-discourse types authorship verification cases using the following discourse types (DT): essays, emails, text messages and business memos. The corpus comprises texts of around 56 authors. All authors have similar ages (18-22) and are native English speakers. The topic of text samples is not restricted. At the same time, the level of formality can vary within a certain DT, the total number of text pairs in the dataset provided by the PAN@CLEF 2022 [1], is 12,264. Each problem is composed of two texts belonging to two different DTs.
According to PAN@CLEF 2022, the test dataset are structured similarly, but its author sets do not overlap with the provided training dataset.

We split the dataset into training, validation, and testing to evaluate our model. We trained our model on the training set and used the validation split to calibrate the hyperparameters. The test set is only to achieve a reference score of the model. We did not use any samples in the testing set to calibrate our model, so the scores obtained when we evaluated the model on this set tells us about the model's generalization ability.

Our splits were done using the pairs provided in the training dataset, we made these splits author-disjoint, which is, no text in one partition has the same author as any text in a different partition. Since we had 12,264 problems and only 56 authors, splitting the dataset in an author-disjoint way yielded unbalanced splits, we got more positive problems than negative ones, since the pairs with authors from different partition were removed. To balance the partitions new instances of the same author (positive) and different authors (negative) were generated. For this, we applied the next methodology: Let sets A and B be the subsets of documents from the partition grouped by author. Positive and negative instances were obtained applying Cartesian product $P = A \times A$ and $N = A \times B$ respectively. Then, we filtered pairs of the same DT, and finally randomly selected positive and negative instances from P and N sets to balance the training, validation, and test partitions.

The new dataset have a balanced proportion of true and false problems. Table 3 shows the total number of problems and the number of problems in the positive class. In addition, the table shows the number of texts and authors on each partition.

Split	Total	Positive	Texts	Authors
Train	15,732	7866	906	47
Validation	754	377	60	4
Test	1070	532	80	5

Table 1
Total number of problems, number of problems in the positive class, number of texts and number of authors on our splits.

4. Modeling Texts as Graphs

Our graph representation attempts to capture the relationship between words and POS labels. Before obtaining our graphic representation, We perform a text pre-processing consisting of the following steps:

- Substitution of non-ASCII characters to their closest ASCII equivalent (we employed unidecode package[2]).
- Tokenize and obtain the POS labels.
- Normalize to Lowercase, since some words are capitalized.

Punctuation of any kind was not removed, and non-ASCII characters were replaced to avoid variability in these punctuation. To get the POS tags, we use a Python package called NLTK[3] package which uses the PENN-Treebank POS labels [11], and then add two additional tags: $PUNCT to mark all punctuation and $OTHER to mark any other words that the NLTK model could not identify. In total, we decided to consider 38 labels. After this process we obtain a list of tuples, each token with its corresponding POS label. To illustrate this process we show the list obtained for the following text:

<nl>I am a Second year <course> student at <university>. I am interested.

```
[('<nl>','NN'), ('i','PRP'), ('am','VBP'), ('a','DT'), ('second','JJ'),
('year','NN'), ('<course>','FW'), ('student','NN'), ('at','IN'),
('<university>','NNP'), ('.','$PUNCT'), ('i','PRP'), ('am','VBP'),
('interested','JJ'), ('.','$PUNCT')]
```

We build a co-occurrence graph considering that two words coexist if they appear next in the text. We define the graph as an ordered pair $G = (V, E)$, where V is a set of vertices composed by (word, pos) tuples and E is a set of weighted edges. The edge set $E \subseteq \{(n_1, n_2, w) | n_1, n_2 \in V, n_1 \neq n_2, w \in \mathbb{R}\}$, where w is the edge weight.

To change the graph's structure and the information abstracted from the text, we create a set of POS labels and denote it as REDUCE_LABELS. To build the graph, let P be the parsed text as a list of tuples, *l*(*P*) the number of elements in the list, and *P*[*i*] the *i*-th element in the list. For each *P*[*i*] = (word, pos) in *P*, we can define:

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

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:

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

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

[3] https://www.nltk.org

Please note that 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:

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

This structure identifies all tuples with a specific label in the REDUCE_LABELS set as a single node. 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, $full\ graph$ to the graph generated using REDUCE_LABELS = $\emptyset$ and we will denominate $med\ graph$ to the graph generated using the following set of REDUCE_LABELS:

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

The med graph and the $short$ graph are showed in (Figure 2) and (Figure 1) respectively. To clarify the construction process, (Figure 3) illustrates the construction with empty REDUCE_LABLES set.

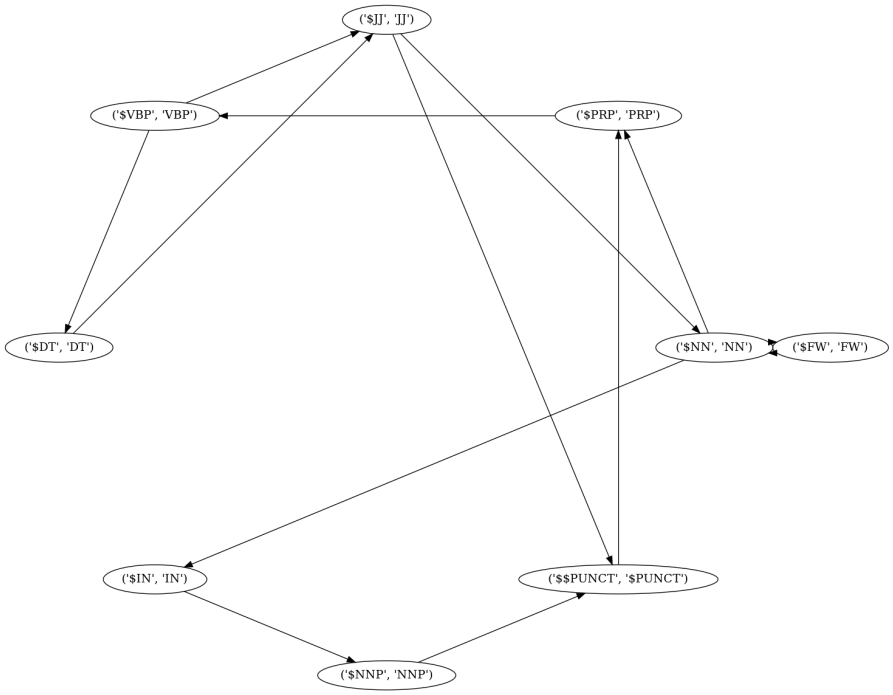

Figure 1: (a) Short graph.

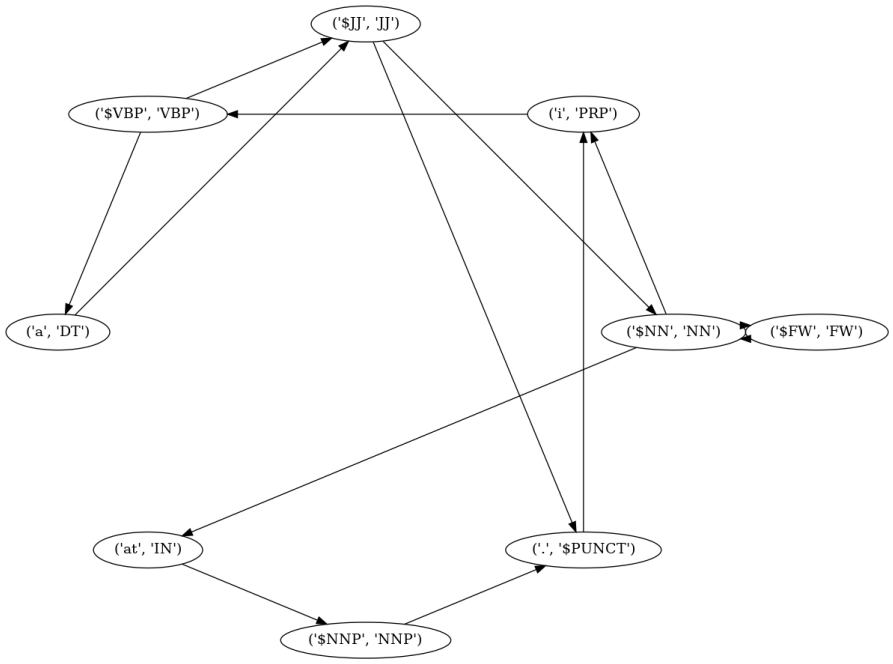

Figure 2: (b) Med graph.

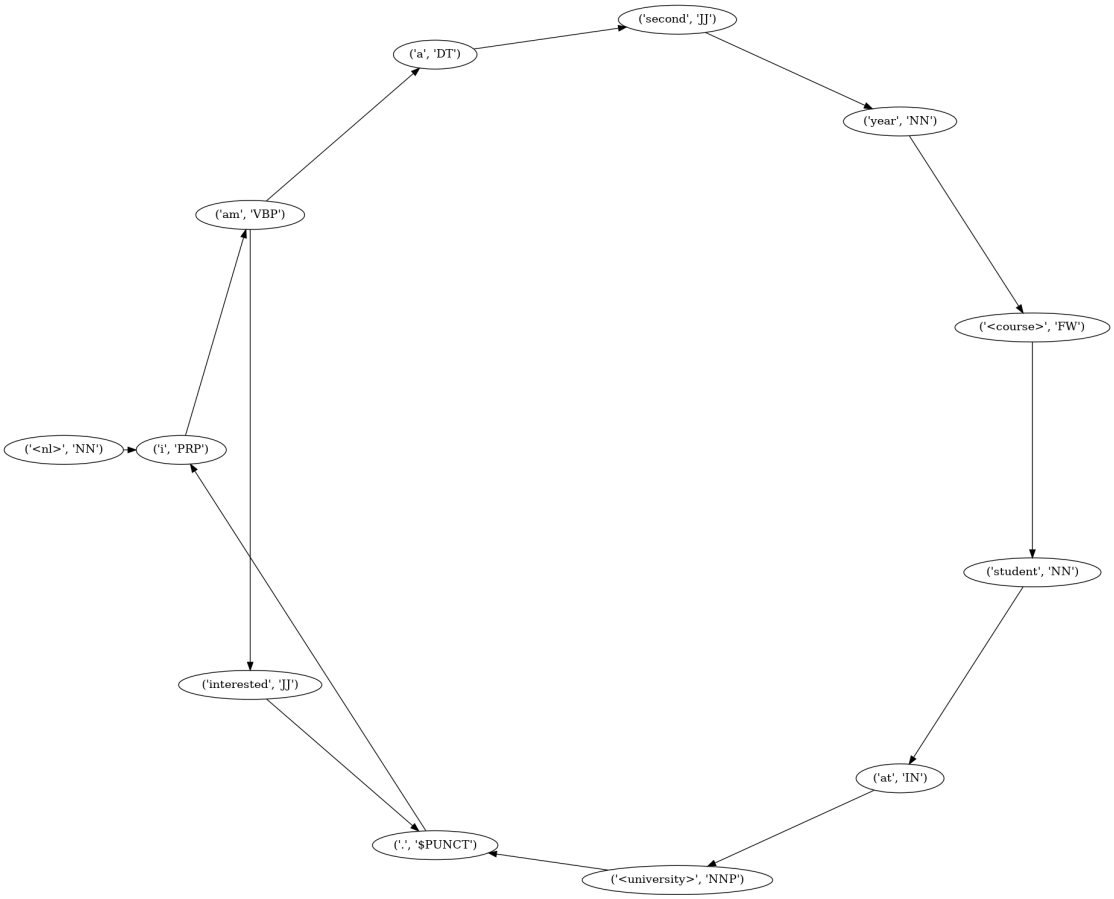

Figure 3: (c) Co-ocurrence graph , equivalent to the graph generated with the empty REDUCE LABELS set.

To input our graph to a neural network, we need to encode each node into a vector. To that end, we use a one-hot encoding representation with respect to the 38 possible POS tags used. There exist references where a deep learning model training over POS embeddings instead of word embeddings obtain better results for an authorship analysis task [12]. Furthermore, the use of low-dimensional POS embeddings to represent nodes allows to reduce the computational cost of the model.

5. Graph-based Siamese Network (GBSN)

To approach the authorship verification task, we use a Siamese network architecture [9] including a component to transform texts as co-occurrence graphs. Our Graph-based Siamese network (Figure 4) comprises two identical feature extraction components with shared weights, a reduction step, and a classification network.

The feature extraction component aims to extract features that represent the author's style

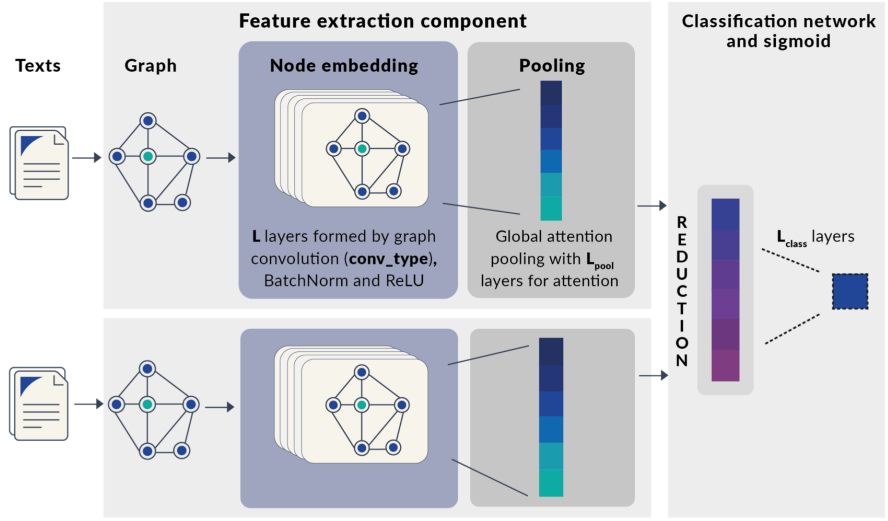

Figure 4: GBSN base architecture

on the graph representation of the texts. Each feature extraction component receives a text, transforms it into a graph, and returns a vector representation of this graph. The feature extraction component has three parts: graph representation, node embedding layers, and global pooling.

In our architecture, a node embedding layer comprises a graph convolutional layer, followed by a batch normalization layer and a ReLU (Rectified linear activation function). We will call *conv_type* to the parameter identifying the graph convolutional layer type and **L** to the parameter specifying the amount of node embedding layers used by the architecture.

The first node embedding layer takes a graph with an initial feature vector in each node as input. Each initial node vector has dimension 38 because this vector is a one-hot representation of the POS label of the node. The output of each node embedding layer is the same graph structure with new feature vectors in each node; the dimension of the vectors obtained can be defined in the same way we define the channels used in a traditional convolutional layer. Our architecture obtains vectors of dimension 64 in each convolutional layer.

For the *conv_type* parameter, we use the proposed convolutional layers to learn features from nodes that also consider the edge weights in their formulation. All the selected layers are implemented with PyTorch-geometric (https://pytorch-geometric.readthedocs.io/en/latest/, accessed on 21 Jun 2022) with the default parameters are explicitly described here:

- GraphConv: A basic implementation of the graph neural network model described by Morris et al. [13]
- LEConv: Originally proposed by Ranjan et al. [14] to select relevant clusters in a graph, the authors prove that it is more expressive than other layers such as the Graph Convolutional Network layer as defined by Kipf and Welling [15] and affirm it can consider both local and global importance of nodes.

- TAGConv: The Topology Adaptive Graph Convolutional network proposed by Du et al. [16]. This layer is defined in a way that in just one layer it can see the information of not just consecutive nodes but nodes at distance K. We performed our experiments with the default K = 3.

We need a pooling layer to obtain a single vector representation for the whole graph.

Our model uses a global attention layer for the pooling [17]. As shown in (Figure 5), this layer takes the final output of the node feature extraction component as its input, i.e., a graph with the vector embedding in each node. The final vector makes a weighted sum of each node vector with a coefficient obtained by doing attention over these same vectors. The formulation is:

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

where V is the set of all nodes in the graphs, and h is a fully connected neural network with a single scalar as output. This fully connected neural network has ReLU (Rectified linear activation function) activation, 32 neurons in each hidden layer, and L_{pool} total layers. The final output of each feature extraction component is a vector with dimension 64.

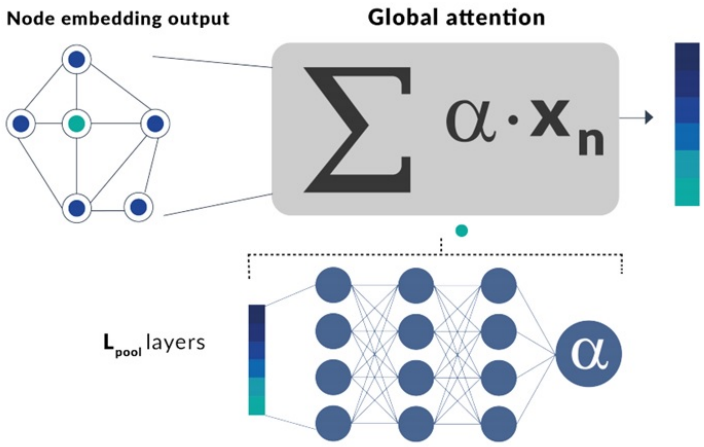

Figure 5: Global Attention Pooling layer.

For the reduction step, we compute the absolute value of the difference between the output of each feature extraction component for each document to be verified. The resulting vector is passed to a final classification network. The classification network is a fully connected network with ReLU (Rectified linear activation function) activation, 64 neurons in each hidden layer, L_{class} total layers, and a final sigmoid function.

Our model returns a single value in the interval [0, 1] that can be interpreted as a measure of how much the two submitted texts are alike, i.e., can be described as the "pseudo-probability" for a same-author text pair. An output close to 1 tells us that the model finds both texts to be from the same author.

6. Results

We trained the network using the binary cross-entropy loss function for all our experiments. The train split pairs are introduced to the model in shuffled order in each epoch. To measure the performance of all models, we use five metrics: Area Under the Receiver Operating Characteristic curve Curve (AUC ROC), F1 score, Brier score [18], F0.5u score [19], and C@1 score [20]. For simplicity, in the results tables, we present the average of these five scores

We trained the neural network with a fixed number of epochs (100 or less) and saved the model that achieved the lowest loss in the validation split as our best model. We report the score of the best model in the test split. The scores reported are the average of three distinct runs over the same architecture. We did all our experiments with a batch size of 256 and a learning rate of 0.001.

In Table 6, we show the average score obtained when using 2, 4, 6, and 8 layers (L columns) of each graph convolutional type (Type column), all these experiments were made with L_{pool} = 2, L_{class} = 2, and a batch size of 256. The experiments were performed independently with the three graph representations (Graph column).

Type	Graph	**L=2**	L=4	L=6	L=8
LEConv	**Short**	**60.01**	58.8	58.34	58.02
	Med	56.12	55.3	55.07	55.32
	Full	59.98	59.23	58.03	58.86
GraphConv	Short	59.04	58.28	57.72	57.89
	Med	56.31	55.76	56.44	56.02
	Full	59.06	59.30	58.94	58.62
TAGConv	Short	59.75	58.01	58.41	58.03
	Med	56.39	55.2	55.76	55.08
	Full	58.45	58.88	58.7	58.12

Table 2
Varying L and type.

We varied the number of layers used for the global pooling (L_{pool}) and the number of layers used in classification (L_{class}). Table 6 shows the scores for the models using 2 and 4 layers of the LEConv graph convolutional layer and for the model using 2 layers of the TAGConv graph convolutional layer. Each row shows the number of pooling layers (L_{pool}), the graph representation used, and the columns that correspond to the number of classification layers (L_{class}).

Table 6 shows the comparative performance of the different configurations. Each row shows the average of the five proposed scores in a model. The first row shows the scores of our submitted model and 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 corresponds to a base architecture model using only the full graph component. Finally, our submission was scored on the test dataset of the PAN 2022 Authorship Verification task [2].

		TAGConv, L = 2		**LEConv, L = 2**		LEConv, L = 4	
		L_{class}		L_{class}		L_{class}	
L_{pool}	Graph	2	4	**2**	4	2	4
2	**Short**	58.09	58.32	**60.09**	58.32	59.01	58.98
	Med	56.02	55.96	56.34	56.07	55.32	55.77
	Full	59.43	59.26	59.79	59.32	59.04	58.30
4	Short	59.32	58.2	59.73	59.02	58.34	58.09
	Med	55.94	55.04	56.89	56.81	55.70	55.72
	Full	58.32	58.98	59.09	59.34	58.91	58.3
6	Short	59.12	58.38	58.34	58.91	58.10	58.07
	Med	55.07	55.10	55.51	55.47	55.72	55.58
	Full	58.22	58.81	58.01	58.89	58.73	58.87

Table 3
Experiments when varying L_{pool}, L_{class}.

	Dataset 1070 problems
Short graph component	**60.09**
Med graph component	56.36
full graph component	59.98

Table 4
Average of performance metrics AUC, F1, c@1, F0.5u and Brier, of the Graph-based Siamese Network (GBSN) with single feature extraction components in our test split dataset

7. Conclusions

This paper presents our Graph-based Siamese Network approach for the authorship verification task at PAN 2022 [2]. In the initial experiments, we tried different convolutional and pooling graph layers configurations, but we observed that the model got overfitted fast in the loss function graph.

After some experiments, we noticed that a superficial convolutional and pooling graph layer delivered better performance. Still, we understand we can modify the dataset or the configuration of the layers in the model to obtain better performance in future work.

We slightly modified the training dataset since we lacked problems to use in the design stages. The model was trained using 15,732 problems which is too limited for learning a classification model with deep learning.

Acknowledgments

This work has been carried out with the support of CONACyT projects CB A1-S-27780, DGAPA-UNAM PAPIIT numbers TA400121 and TA101722. The authors thank CONACYT for the computing resources provided through the Deep Learning Platform for Language Technologies of the INAOE Supercomputing Laboratory. Also thanks to the Engr. Roman Osorio for

supporting the student administration of the project.

References

[1] J. Bevendorff, B. Chulvi, E. Fersini, A. Heini, M. Kestemont, K. Kredens, M. Mayerl, R. Ortega-Bueno, P. Pezik, M. Potthast, F. Rangel, P. Rosso, E. Stamatatos, B. Stein, M. Wiegmann, M. Wolska, E. Zangerle, Overview of PAN 2022: Authorship Verification, Profiling Irony and Stereotype Spreaders, and Style Change Detection, in: A. Barron-Cedeno, G. D. S. Martino, M. D. Esposti, F. Sebastiani, C. Macdonald, G. Pasi, A. Hanbury, M. Potthast, G. Faggioli, N. Ferro (Eds.), Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Thirteenth International Conference of the CLEF Association (CLEF 2022), volume 13390 of *Lecture Notes in Computer Science*, Springer, 2022.

[2] Efstathios Stamatatos and Mike Kestemont and Krzysztof Kredens and Piotr Pezik and Annina Heini and Janek Bevendorff and Martin Potthast and Benno Stein, Overview of the Authorship Verification Task at PAN 2022, in: CLEF 2022 Labs and Workshops, Notebook Papers, CEUR Workshop Proceedings, 2022.

[3] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, R. Shah, Signature verification using a" siamese" time delay neural network, Advances in neural information processing systems 6 (1993).

[4] D. Embarcadero-Ruiz, H. Gómez-Adorno, A. Embarcadero-Ruiz, G. Sierra, Graph-based siamese network for authorship verification, Mathematics 10 (2022) 277.

[5] E. Stamatatos, A survey of modern authorship attribution methods, Journal of the American Society for information Science and Technology 60 (2009) 538–556.

[6] 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, in: CLEF 2014 Evaluation Labs and Workshop Working Notes Papers, Sheffield, UK, 2014, 2014, pp. 1–21.

[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).

[8] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S. Y. Philip, A comprehensive survey on graph neural networks, IEEE transactions on neural networks and learning systems 32 (2020) 4–24.

[9] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, R. Shah, Signature verification using a" siamese" time delay neural network, Advances in neural information processing systems 6 (1993).

[10] A. Nandy, S. Haldar, S. Banerjee, S. Mitra, A survey on applications of siamese neural networks in computer vision, in: 2020 International Conference for Emerging Technology (INCET), IEEE, 2020, pp. 1–5.

[11] M. A. Marcinkiewicz, Building a large annotated corpus of english: The penn treebank, Using Large Corpora 273 (1994).

[12] F. Jafariakinabad, S. Tarnpradab, K. A. Hua, Syntactic recurrent neural network for authorship attribution, arXiv preprint arXiv:1902.09723 (2019).

[13] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, M. Grohe, Weisfeiler

and leman go neural: Higher-order graph neural networks, in: Proceedings of the AAAI conference on artificial intelligence, volume 33, 2019, pp. 4602–4609.
[14] E. Ranjan, S. Sanyal, P. Talukdar, Asap: Adaptive structure aware pooling for learning hierarchical graph representations, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, 2020, pp. 5470–5477.
[15] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907 (2016).
[16] J. Du, S. Zhang, G. Wu, J. M. Moura, S. Kar, Topology adaptive graph convolutional networks, arXiv preprint arXiv:1710.10370 (2017).
[17] Y. Li, D. Tarlow, M. Brockschmidt, R. Zemel, Gated graph sequence neural networks, arXiv preprint arXiv:1511.05493 (2015).
[18] G. W. Brier, et al., Verification of forecasts expressed in terms of probability, Monthly weather review 78 (1950) 1–3.
[19] J. Bevendorff, B. Stein, M. Hagen, M. Potthast, Generalizing unmasking for short texts, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019, pp. 654–659.
[20] A. Peñas, A. Rodrigo, A simple measure to assess non-response, In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (2011).