Siamese Network applied to Authorship Verification
Notebook for PAN at CLEF 2020

Emir Araujo-Pino[1], Helena Gómez-Adorno[2], and Gibran Fuentes-Pineda[2]

[1] Posgrado en Ciencia e Ingeniería de la Computación
Universidad Nacional Autónoma de México,
Mexico City, Mexico
emiraraujoing@gmail.com
[2] Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas
Universidad Nacional Autónoma de México,
Mexico City, Mexico
helena.gomez@iimas.unam.mx, gibranfp@unam.mx

Abstract In this paper, we present our approach to the Authorship task at PAN 2020. The task consists in comparing two documents and automatically determine if they are written by the same author. To solve this task, we introduce a Siamese network architecture that is trained on character n-grams of the document pairs to be compared. We experimented with different hyperparameters when training the model on a large and a small dataset. Our best model achieved an overall evaluation of 0.804, which is the average of AUC, c@1, f_05_u, and F1 scores.

1 Introduction

PAN [3] is a CLEF [4] Lab on uncovering plagiarism, authorship, and social software misuse. PAN [1] has a series of shared tasks of text forensics and stylometry, this year's campaign focuses on authorship verification, celebrity profiling, profiling fake news Spreaders on Twitter, and Style Change Detection. In this paper, we describe our approach to the authorship verification task, which aims to identify if both texts belong to the same author or not. From the machine learning perspective, this task can be seen as a binary classification problem.

Nowadays, there are many unknown authorship letters such as email Fraud [24], suicide [8], and terrorism [3] for which it is necessary to verify the authorship. Currently, to solve authorship verification problems there are different approaches such as distance based [23], machine learning based [18], and impostors [13] which have shown great results in previous PAN tasks. Besides, there are four different approaches to solve

[3] https://pan.webis.de/
[4] https://https://clef2020.clef-initiative.eu/

these problems and can be separated by instance or profile based and intrinsic or extrinsic based [20]. This year's shared task consists of closed set problems, which we solved by an instance-intrinsic based approach.

As far as we know, there are few implementations of deep learning based methods to solve this problem. *Lopez-Velazco 2016* [17] introduced a siamese architecture [4] with character embeddings as input, to the authorship verification problem. Exploring architectures and methods we finally train a deep learning siamese network on character 3-grams to solve the PAN 2020 authorship verification shared task.

2 Previous Work

Authorship identification related problems still have challenges to overcome, there are many methods to solve specific problems [9]. Last year, PAN 2019 shared the cross domain authorship attribution task [11] which consisted of identifying the author of a given document among a closed set of authors. Most of the submitted solutions of PAN 2019 were SVM variations and no even one participant used a deep learning based approach. The main differences between a traditional machine learning and a deep learning application from the development perspective are the hardware requirements, the amount of data to handle, and the ability to handle raw data as input [14]. Conventional machine learning techniques need extensive feature engineering to transform the raw data in order to use it for training and testing a model.

Deep learning algorithms were used in PAN 18 authorship attribution shared task [12], but as far as we know there is no much research on Siamese networks to solve these problems. In [17], a Siamese architecture composed of a character embeddings input layer, a convolution [15], and LSTM [7] networks was tested on an English books corpus. This architecture achieved good performance on a Gutenberg [2] project corpus specifically designed to test authorship verification methods. The main differences with our architecture are the input which consists of n-grams, the way of extracting the embedded vector from each input, and the compare method which originally uses euclidean distance.

3 Datasets

The PAN 2020 authorship verification organization provided two datasets: large and small. The purpose of these two corpus was to train two models, one for each dataset, to compare a hungry data method as a deep learning model with a symbolic machine learning approach. As it is known, one of the main problems for training a neural network is to get enough data to achieve good performance. So, we decided to train two models with the aim to compare how much the amount of data affects the performance of our method.

Table 1 shows the results of a basic data analysis over both datasets. The *Samples* column shows the number of documents pairs, the *Different Texts* column shows the number of different documents in the corpus, and the last three columns show the characters statistics of the documents. The small dataset corresponds to 19% of the size

Dataset	Samples	Positive Samples	Different Texts	Max Characters	Min Characters	Mean Characters
Large	275565	147778	494257	943947	20355	21426.08
Small	52601	27834	93667	296887	20670	21424.93

Table 1: Statistics measures of PAN 2020 datasets.

of the large dataset. Most of the documents are around 21000 characters and there are almost no duplicated documents. The details of the datasets are available in [10].

4 Methodology

In the machine learning literature, there is a wide variety of deep learning architectures, as far as we know, conventionally most of these receive as input raw data. In other words, deep learning models learn to extract relevant features from raw data to reach good performance. Despite this, our neural architecture approach receives character n-grams as inputs. So in order to train our network, the first step is to transform the texts dataset into character n-grams (with n varying from 1 to 3) frequency vectors (the dimensions of the vectors correspond to the frequency of each n-gram).

4.1 Features Extraction

In order to obtain the n-grams of the documents pairs, we use the Scikit Learn Python module [19]. It extracts all the n-grams (with n varying from 1 to 3) within the documents and produces a vector representation with the frequency of such n-grams. With the purpose of optimizing time and memory resources and without compromising the classifier performance we performed basic hyperparameters tuning during the n-gram features extraction process. The hyperparameters we tune are the minimum document frequency and the maximum document frequency. To archive this we train the same Siamese architecture on different n-grams sets. Due to time restriction, we were not able to perform a complete grid search over all hyperparameters. Our best model on the training dataset receives as input n-grams (with n varying from 1 to 3) that appear at least in the 1% of all documents (min document frequency) and with no restriction on the maximum document frequency. Finally, ℓ_1 normalization was performed over the n-grams vectors.

4.2 Network Architecture and Training

A siamese network can be seen generally as a comparison network, its main components are a set of identical subnetworks and a final layer to contrast the subnetworks outputs [4]. The main idea behind this architecture is to extract a set of features from each input using a subnetwork and in the same way train another network with the aim to compare the two outputs of the subnetworks.

Figure 1 shows the network architecture. The architecture has only one extracting subnetwork which receives each input n-gram vector separately from each document to

compare. The first layers of the extracting subnetwork are Batch normalization (BN), Gaussian Noise (GN), and Dropout (DP). These layers were set at the input with the aim to speed up the training and achieve a better validation score.

Then, a residual network converts the input into a feature space of 512 dimensions. The first dense layer inside the residual network acts as an adapter, it allows to connect the input to the residual network. This residual network has a depth of 8 and all the internal layers have a feature space of 512. Some differences from the ResNet [6] are the connections inside the residual network which perform a subtraction instead of an addition, and the use of dense layers instead of convolutional layers. Finally, the classifier network (Dense network with binary output) takes the absolute difference of both subnetworks outputs and performs a binary classification with a 3 layers dense network. In order to speed up training, we also use BN between the dense layers of the classifier network. We use the elu [5] activation, and Radam [16] optimization to train the complete Siamese network.

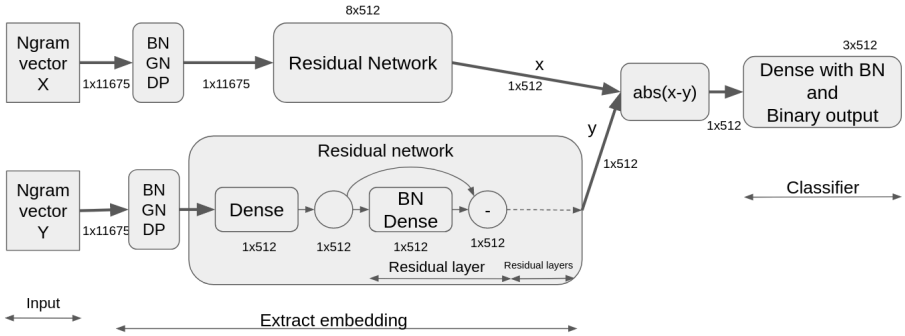


Figure 1: Final architecture trained. "BN" means batch normalization, "GN" refers to Gaussian noise, "Dense" means dense layer, and "abs" means absolute value.

We trained two models, one with the small dataset and the other with the large one. In order to speed up the n-gram extraction process, we only used the first 10000 (ten thousand) characters from each document. Both datasets were split on training and validation with 70 and 30 percent of samples respectively. To train the network we used the default value of the alpha learning rate which is 10^{-3} and the n-grams range was set from 1 to 3. Table 2 shows the examined hyperparameters of the complete architecture. For our final submission, we used the parameters that achieved the best classification performance in terms of AUC on 30% of both large and small training datasets. The final hyperparameters are shown in Table 2 in bold.

Both models were trained on a 2070super GPU with 32bits configuration, 64GB RAM. The training time was around 6 and 14 hours on the small and large datasets respectively. We used the pipelines of TensorFlow[5].

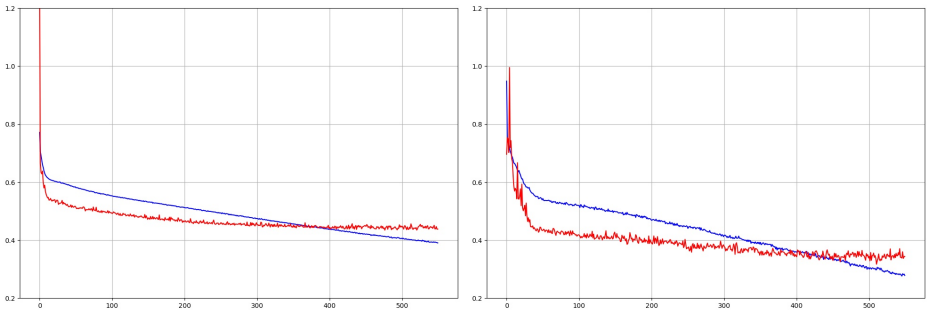
[5] https://www.tensorflow.org

Gaussian Noise	Dropout	Activation	Min Document Frecuency
0	0.1	relu	0.05
10^{-3}	0.3	**elu**	**0.01**
10^{-4}	0.7		
$\mathbf{10^{-5}}$	**0.9**		

Table 2: Examined system configurations.

5 Results

We trained a model for each dataset (large and small). Figure 2 shows the loss and validation loss obtained by training. We set the same hyperparameters for training, preprocessing, and architecture in both models. Figure 2a (Large dataset model) is smoother due to the amount of data available for training the model. It can be observed that the model trained on the large dataset converges faster.



(a) Large dataset (b) Small dataset

Figure 2: Both figures show the number of epochs in the horizontal axis. Training loss is shown as a blue line. Validation loss is shown as a red line.

Table 3 shows the validation scores of the models trained on the small and large datasets. Both datasets were split in Training and Validation following a 70%-30% proportion. In this way, we evaluated two times each model, first on the same dataset but with different examples (30% validation set extracted from the same dataset), second, with a different dataset (30% validation set comes from the other dataset). In this sense, we obtain four validation scores from the two models and the two validation sets. It can be observed that the AUC score is 1.0 when the Training and Validation sets come from the same dataset. On the other hand, the small dataset model achieves only 0.823 AUC on the validation set that comes from the large training set. We believe that the reason for this behavior is that the same document can appear in both training and validation sets because we only divided the document pairs samples.

Training Dataset	Validation Dataset	AUC	c@1	f_05_u	F1	overall
Large (70%)	Small (30%)	0.964	0.882	0.858	0.894	0.899
Large (70%)	Large (30%)	1.000	0.993	0.990	0.993	0.994
Small (70%)	Small (30%)	1.000	0.987	0.981	0.988	0.989
Small (70%)	Large (30%)	0.823	0.748	0.773	0.745	0.772

Table 3: Validations exchanging datasets.

Finally, Table 4 shows the final results of the PAN 2020 authorship verification task evaluated on the TIRA platform [21]. It can be observed, that the overall score shows that the small dataset model achieves better results than the large dataset model, even though the performance of both models are very similar, for example, the AUC difference between both results is 0.015.

Training Dataset	AUC	c@1	f_05_u	F1	overall
Large	0.859	0.751	0.745	0.800	0.789
Small	0.874	0.770	0.762	0.811	0.804

Table 4: Final results of PAN 2020 authorship verification task.

6 Conclusion

In this paper, we presented a siamese network approach to solve the PAN 2020 authorship verification problem. The network receives document pairs as character n-grams (with n varying from 1 to 3) as input and learns to identify if these documents are written by the same author. The network hyperparameters were adjusted on the two datasets provided by the task organizers. The final configuration submitted to the conference achieved the best performance during training time on both, large and small datasets. Our experiments showed that both models achieve similar results when training on different data and amount of examples.

Even while using as few as 19% of the size of the large dataset, the model trained on the small dataset achieved similar results on the validation set. On the other hand, the AUC score reached 1.0 at validation if the training and validation sets come from the same dataset. In the case of PAN 2020 test dataset, the small dataset model outperformed the large dataset model but only by 0.015 of AUC score. The validation of the small dataset model on the large validation set achieves a score near to the score obtained on the PAN 2020 test dataset.

We demonstrated that this architecture can achieve good results at the PAN 2020 verification task, even when trained on a small dataset and using only 10000 characters per document. Besides, our approach does not implement score corrections and we only

tune our models with the AUC score as reference. As future research directions, we plan to perform a better tuning of the described hyperparameters using a genetic-based parameter tuning [22].

Aknowledgements

This work has been partially supported by PAPIIT-UNAM projects TA100520.

References

1. Organization (2020), https://pan.webis.de/
2. Project gutenberg (2020), www.gutenberg.org. urldate = 2020-06-29
3. Abbasi, A., Chen, H.: Applying authorship analysis to extremist-group web forum messages. In: Homeland Security (2005)
4. Bromley, J., Guyon, I., LeCun, Y., Sickinger, E., Shah, R.: Signature verification using a "siamese" time delay neural network. In: American Telephone and Telegraph Company papers (1994)
5. Clevert, D.A., Unterthiner, T., Hochreiter, S.: Fast and accurate deep network learning exponential linear units (elus) (2016)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition (2015)
7. Hochreiter, S., Schmidhuber, J.: Long short-term memory (1997)
8. Jasim-Basim, Y.: Author attribution in suicide notes: Evidence from applied linguistics. In: Comparative Legilinguistics (2012)
9. Juola, P.: Especial Problems of Linguistic Forensics (2008)
10. 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 (Sep 2020)
11. Kestemont, M., Stamatatos, E., Manjavacas, E., Daelemans, W., Potthast, M., Stein, B.: Overview of the cross-domain authorship attribution task at pan 2019 (2019)
12. Kestemont, M., Tschuggnall, M., Stamatatos, E., Daelemans, W., Specht, G., Stein, B., Potthast, M.: Overview of the Author Identification Task at PAN-2018 cross-domain authorship attribution and style change detection (2018)
13. Koppel, M., Winter, Y.: Determining if two documents are written by the same author. In: Journal of the American society for information science and technology (2013)
14. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning (2015)
15. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. In: Neural computation (1989)
16. Liu, L., Jiang, H., Pengcheng He, W.C., Xiaodong Liu, J.G., Han, J.: On the variance of the adaptive learning rate and beyond (2020)
17. Lopez-Velasco, F.: Verificacíon de autoría en textos mediante redes neuronales convolucionales y recurrentes (2018)
18. Martín-Del-Campo-Rodríguez, C., Gómez-Adorno, H., Sidorov, G., Batyrshin, I.: Cic-gil approach to cross-domain authorship attribution: Notebook for pan at clef 2018. In: PAN working notes (2018)

19. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830 (2011)
20. Potha, N., Stamatatos, E.: Improving author verification based on topic modeling (2019)
21. Potthast, M., Gollub, T., Wiegmann, M., Stein, B.: TIRA Integrated Research Architecture. In: Ferro, N., Peters, C. (eds.) Information Retrieval Evaluation in a Changing World. Springer (Sep 2019)
22. Sanchez-Perez, M.A., Gelbukh, A., Sidorov, G., Gómez-Adorno, H.: Plagiarism detection with genetic-based parameter tuning. International Journal of Pattern Recognition and Artificial Intelligence 32(01), 1860006 (2018)
23. Teahan, W.J., Harper, D.J.: Using compression-based language models for text categorization (2003)
24. Zaharia, A.: 300+ terrifying cybercrime and cybersecurity statistics & trends [2020 edition] (2020), https://www.comparitech.com/vpn/cybersecurity-cyber-crime-statistics-facts-trends/