

Heterogeneous-Graph Convolutional Network for Authorship Verification

Notebook for PAN at CLEF 2023

Andric Valdez-Valenzuela¹, Jorge Alfonso Martinez-Galicia¹ and Helena Gómez-Adorno²

¹Posgrado en Ciencia e Ingeniería de la Computación, Universidad Nacional Autónoma de México, CDMX, México

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

Abstract

Authorship Analysis (AA) is a research area that aims to investigate and identify characteristics in the writing style in text documents to extract relevant information about the author. One of the most common tasks in recent years within this area is Authorship Verification (AV). AV aims to determine if two given text documents were written or not by the same author by measuring different characteristics. The PAN@CLEF 2023 AV challenge [1][2] requires solving the task on a cross-discourse type and open-set collection of essays, emails, interviews, and speech transcriptions. We model a text graph representation and build a Graph Neural Network (GNN) to extract relevant features from the text documents. Specifically, we use a heterogeneous text graph representing the entire corpus, then use it as input for a Graph Convolutional Network (GCN) to obtain learning vectors (embeddings), and finally use these vectors to train a classification model (fully connected network).

Keywords

Authorship verification, Text graph, Heterogenous graph, Graph Neural Networks

1. Introduction

The Authorship analysis research field studies the characteristics that help to define an author's writing style. The features can be extracted using text samples of the authors. This research area includes authorship attribution, author profiling, author clustering, and plagiarism detection [3]. The AV task aims to determine if two given text documents were written or not by the same author.

To approach the AV task at PAN 2023, we use a heterogeneous text graph representing the entire corpus, then use it as input for a Graph Convolutional Network (GCN) to obtain learning vectors (embeddings), and finally use these vectors to train a classification model (fully connected). We also evaluated three strategies (short, medium, and full) for representing texts as graphs based on the relation of the Part of Speech (POS) labels and the co-occurrence of

CLEF 2023: Conference and Labs of the Evaluation Forum, September 18–21, 2023, Thessaloniki, Greece


✉ andric.valdez@gmail.com (A. Valdez-Valenzuela); jorgetonatiuhmg@gmail.com (J. A. Martinez-Galicia);

helenagomez@iimas.unam.mx (H. Gómez-Adorno)

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

🆔 0000-0002-6966-9912 (H. Gómez-Adorno)

© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

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 our GitHub repository ¹

This paper is structured as follows: Section 2 summarizes related works on Authorship Analysis. Section 3 describes the dataset used for the task. Section 4 presents the Heterogeneous Graph representation used to feed the Graph Neural Network. Section 5 describes the Graph Convolutional Network architecture, while the experiments and the obtained results are presented in Section 6. 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 [4]. Some of the most commonly used supervised classification algorithms used in AV analysis are discriminant analysis, support vector machines, decision trees, neural networks, and genetic algorithms [5].

Another option that can effectively model relationships and structural information is the mathematically constructed graph representation. This representation can be possible using feature terms as vertices and significant relations between the feature terms as edges. 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 [6].

Some research works applied this text graph representation approach to solving different problems related to text analysis or classification tasks. For example, Liang Yao et al. [7] implemented a text classification model (datasets: Movie Review, 20-Newsgroups, etc.) using a GNN architecture with convolution layers which they trained using a text graph (constructed from the corpus) based on in the co-occurrences of words and the relationship of words in documents. When evaluating the model, its results present a performance improvement compared to traditional classification methods, and in addition, they require less training data to achieve good performance. Another example for Embarcadero-Ruiz et al. [8] presented a work focused on the AV task; they implemented a Siamese Network architecture composed of GCNs and pooling and classification layers. The main idea was to represent the text documents of the corpus as text graphs, then use them as input to the GCN to extract relevant features from the graph, and finally apply a classification layer that verifies the authorship of a document. A collection of fanfiction texts provided by PAN@CLEF 2021 was used to evaluate this approach. Their results show performance similar to state-of-the-art performance and tend to be better than the traditional ones.

¹<https://github.com/PLN-disca-iimas/AuthorshipVerification-heterogeneousGraph>

Table 1

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

<i>Split</i>	<i>Total</i>	<i>Positive</i>	<i>Texts</i>	<i>Authors</i>
Train	8024	4012	786	48
Validation	372	186	48	4
Test	440	220	52	4

3. Authorship Verification Dataset at PAN 2023

For the Authorship Verification task, the training dataset provided by the PAN@CLEF 2023 [1] organization consists of four cross-discourse types (DT): essays, emails (as written language), interviews, and speech transcriptions (as spoken language). The corpus comprises texts from 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 2023 is 8,836. Each problem is composed of two texts belonging to two different DTs.

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 model’s hyperparameters. The testing 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 8,836 problems and only 56 authors, splitting the dataset in an author-disjoint way yielded unbalanced splits. As a result, we got more positive problems than negative ones since the pairs with authors from different partitions were removed. New positive instances (same author) and new negative instances (different authors) were generated to balance the partitions. To achieve this, we applied the following 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 randomly selected positive and negative instances from P and N sets to balance the training, validation, and test partitions.

The new dataset has a balanced proportion of true and false problems. Table 1 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.

4. Modeling text as Graph

First steps before obtaining our graphic representation, We perform a text pre-processing consisting of the following steps:

- Normalize text to lowercase.
- Substitution of non-ASCII characters.²
- Remove punctuation and stop words.
- Tokenize and obtain the POS labels.

Then, We applied the approach proposed by Embarcadero-Ruiz et al. [8], where three types of word groups are built based on the POS labels of the word tokens: Short, Medium, and Full. For POS tags, we use a Python package called NLTK³ package which uses the PENN-Treebank POS labels [9], 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 generate these word groups, we defined the following set of POS labels, and denote it as REDUCE_LABELS (that is, group by word POS using this set):

```
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]
```

Then We apply the following rule/formula to group the word/tokens by these POS labels, for this: 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 $P[i] = (\text{word}, \text{pos})$ in P , we can define

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

where M is the list defined by the tuples masked as explained.

Finally, after applying these We build a heterogeneous word document graph for a whole corpus to represent text as a graph [7]. In this graph type, words and documents are represented as nodes, and the relation between word to document and word to word as edges. Specifically, the edge between two-word nodes is built by word co-occurrence information (when two words appear together in the text) and the relation between word-document nodes is built if the word exists in that text document. As weights/attributes, the word-to-document relation has the Term Frequency-Inverse Document Frequency (TFIDF) measure, and the word-to-word relation has the Point-wise Mutual Information (PMI) measure (see figure 1).

²<https://github.com/avian2/unidecode>

³<https://www.nltk.org>

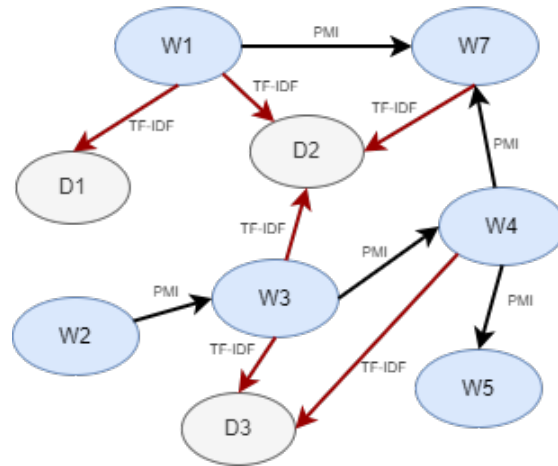


Figure 1: Heterogeneous Graph structure. The words (W1 to W7) and documents (D1 to D3) are represented as nodes. The relations word-word (black arrow with PMI metric as weights) and document-word (red arrow with TF-IDF metric as weights) are represented as edges.

5. Graph Convolutional Network

The model for the graph structure is a Graph Convolutional Network proposed in [7][10]. Originally, this network was used for text classification, but it will be used as the feature extraction stage for texts in this case.

The model consists of a multilayer neural network that operates directly on a graph and induces node vector embedding based on their neighborhoods' properties. A Graph Convolutional Network with two layers enables the transmission of messages between nodes that are at most two steps apart. This means that even though there are no direct connections between documents in the graph, the two-layer GCN facilitates the exchange of information between pairs of documents. In this case, experiments with N layers were performed to find the right hyperparameter for the corpus and the task. The model includes a reduction step where the embeddings corresponding to an authorship verification problem are concatenated, and a fully connected network is used for classification (dense and linear hidden layers with ReLU activation function, and an output layer with Sigmoid activation function). Figure 2 illustrates the model architecture.

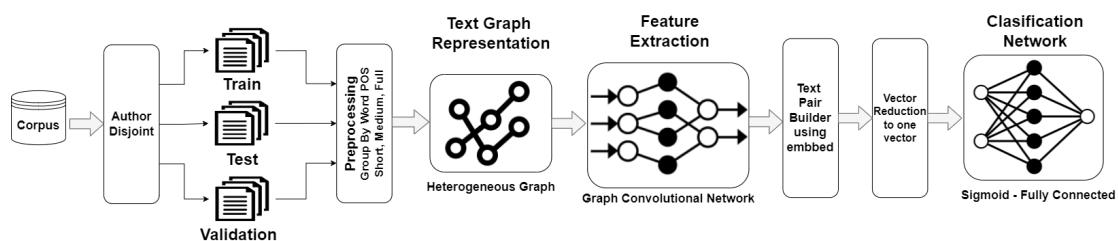


Figure 2: Graph Convolutional Network Model

Table 2

Validation and Test scores varying graph type (short, medium, full), window size, function words, and punctuation marks. Highlighted in bold are the best validation and test scores for each graph type.

Functional Words	Punctuation marks	Window Size	Short Graph		Medium Graph		Full Graph	
			Validation	Test	Validation	Test	Validation	Test
Yes	Yes	10	0.6605	0.7046	0.659	0.6902	0.6223	0.6623
		20	0.6614	0.7068	0.5971	0.6862	0.6146	0.6742
		30	0.6701	0.6823	0.6102	0.6749	0.6334	0.6644
	No	10	0.702	0.6795	0.6674	0.6497	0.554	0.564
		20	0.6741	0.6823	0.6638	0.6475	0.543	0.59
		30	0.6761	0.6706	0.6503	0.6565	0.5231	0.5941
No	Yes	10	0.7189	0.7044	0.6756	0.6928	0.5878	0.5778
		20	0.6481	0.6933	0.6717	0.7	0.5187	0.5986
		30	0.7051	0.6944	0.6757	0.6844	0.345	0.6
	No	10	0.6654	0.6632	0.702	0.6251	0.5622	0.5422
		20	0.6789	0.6823	0.6961	0.6559	0.5178	0.5768
		30	0.6945	0.7033	0.6847	0.6647	0.5906	0.5876

6. Results

We trained the network using the cross-entropy loss function for all our experiments. To measure the performance of all models, we use five metrics: Area Under the Receiver Operating Characteristic curve (AUC ROC), F1 score, Brier score [11], F0.5u score [12], and C@1 score [13]. 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 (from 100 to 200) 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 dropout of 0.2 and a learning rate of 0.0001.

In Table 2, we show the average score obtained by varying graph types (short, medium, full), window sizes (10, 20, 30), function words ⁴ (yes, no), and punctuation marks ⁵ (yes, no). All these experiments were made with 4 classification layers and 8 convolutional layers of type GraphConv which is a basic implementation of the graph neural network model described by Morris et al. [14].

7. Conclusions

This paper presents a heterogeneous text graph representing the entire corpus, then use it as input for a Graph Convolutional Network to obtain learning vectors, and finally uses these vectors to train a classification model. We also evaluated three graph strategies (short, medium, and full) based on the relation of the Part of Speech (POS) labels and the co-occurrence of words.

For the dataset, we made splits (Train, Test, Validations) author-disjoint, which is no text in one partition has the same author as any text in a different partition. After some experiments,

⁴Defined for the NLTK Python package: `nltk.download('stopwords')`

⁵Use of a regular expression deleting all characters different from letters and numbers

we obtained the best results (for validation and test partitions) for the short graph representation compared to the other graph types (med, full).

For the PAN@CLEF 2023 AV challenge, We presented 3 final runs: 1 Short Graph, 1 Medium Graph, and 1 Full Graph. For each one, We set the parameters based on the results obtained from our test score (as shown in the table 2).

Acknowledgments

This work was supported by the Graduate Science and Engineering Computing (UNAM) and the CONACYT scholarship program (CVU: 927264). Also, the authors thank CONACYT for the computing resources provided through the Deep Learning Platform for Language Technologies of the INAOE Supercomputing Laboratory.

References

- [1] J. Bevendorff, I. Borrego-Obrador, M. China-Ríos, M. Franco-Salvador, M. Fröbe, A. Heini, K. Kredens, M. Mayerl, P. Pezik, M. Potthast, F. Rangel, P. Rosso, E. Stamatatos, B. Stein, M. Wiegmann, M. Wolska, , E. Zangerle, Overview of PAN 2023: Authorship Verification, Multi-Author Writing Style Analysis, Profiling Cryptocurrency Influencers, and Trigger Detection, in: A. Arampatzis, E. Kanoulas, T. Tsirikika, A. G. Stefanos Vrochidis, D. Li, M. Aliannejadi, M. Vlachos, G. Faggioli, N. Ferro (Eds.), *Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Fourteenth International Conference of the CLEF Association (CLEF 2023)*, Lecture Notes in Computer Science, Springer, 2023.
- [2] E. Stamatatos, K. Kredens, P. Pezik, A. Heini, J. Bevendorff, M. Potthast, B. Stein, Overview of the Authorship Verification Task at PAN 2023, in: *CLEF 2023 Labs and Workshops, Notebook Papers*, CEUR-WS.org, 2023.
- [3] J. Bromley, I. Guyon, Y. LeCun, E. Säcker, R. Shah, Signature verification using a "siamese" time delay neural network, *Advances in neural information processing systems* 6 (1993).
- [4] E. Stamatatos, A survey of modern authorship attribution methods, *Journal of the American Society for information Science and Technology* 60 (2009) 538–556.
- [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, in: *CLEF 2014 Evaluation Labs and Workshop Working Notes Papers*, Sheffield, UK, 2014, 2014, pp. 1–21.
- [6] 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).
- [7] L. Yao, C. Mao, Y. Luo, Graph convolutional networks for text classification, in: *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 2019, pp. 7370–7377.
- [8] D. Embarcadero-Ruiz, H. Gómez-Adorno, A. Embarcadero-Ruiz, G. Sierra, Graph-based siamese network for authorship verification, *Mathematics* 10 (2022) 277.

- [9] M. A. Marcinkiewicz, Building a large annotated corpus of english: The penn treebank, *Using Large Corpora* 273 (1994).
- [10] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, *arXiv preprint arXiv:1609.02907* (2016).
- [11] G. W. Brier, et al., Verification of forecasts expressed in terms of probability, *Monthly weather review* 78 (1950) 1–3.
- [12] 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.
- [13] 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).
- [14] 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.