Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022), pages 2196–2204
Marseille, 20-25 June 2022

Unsupervised Embeddings with Graph Auto-Encoders for Multi-domain and Multilingual Hate Speech Detection

Gretel Liz De la Peña Sarracén,[1,2] Paolo Rosso[1]
[1]Universitat Politècnica de València, [2]Symanto Research
gredela@posgrado.upv.es, prosso@dsic.upv.es

Abstract

Hate speech detection is a prominent and challenging task, since hate messages are often expressed in subtle ways and with characteristics that may vary depending on the author. Hence, many models suffer from the generalization problem. However, retrieving and monitoring hateful content on social media is a current necessity. In this paper, we propose an unsupervised approach using Graph Auto-Encoders (GAE), which allows us to avoid using labeled data when training the representation of the texts. Specifically, we represent texts as nodes of a graph, and use a transformer layer together with a convolutional layer to encode these nodes in a low-dimensional space. As a result, we obtain embeddings that can be decoded into a reconstruction of the original network. Our main idea is to learn a model with a set of texts without supervision, in order to generate embeddings for the nodes: nodes with the same label should be close in the embedding space, which, in turn, should allow us to distinguish among classes. We employ this strategy to detect hate speech in multi-domain and multilingual sets of texts, where our method shows competitive results on small datasets.

Keywords: Hate Speech Detection, Unsupervised Embeddings, Graph Auto-Encoders

1. Introduction

In this paper we investigate an unsupervised, graph-based approach to learn embeddings for hate speech detection. According to (Fortuna and Nunes, 2018), hate speech can be defined as a language that attacks, diminishes or incites violence against groups based on specific characteristics. Accordingly, the aim of hate speech detection is to discriminate texts that contain hate from those that do not. This is a widely studied task that involves a number of challenges (Poletto et al., 2020). Specifically, we study the problem of data-poor settings that appears in low-resource domains and languages – i.e., in those settings where supervised approach are not able to generalise well.

For this, we make use of Graph neural networks (GNNs). They are a framework based on deep learning to operate on graphs. They follow a recursive neighborhood aggregation scheme, called message passing, where each node aggregates feature vectors of its neighbors to compute its new feature vector (Xu et al., 2018). After a number of iterations, a node is represented by its transformed feature vector, which captures the structural information within its neighborhood. Therefore, GNNs have been effective at tasks thought to have rich relational structure since they can preserve global structure information of a graph in embeddings. (Wu et al., 2021) provide an overview of recent studies on deep learning approaches for graphs and discuss the applications of GNNs in several areas. (Zhou et al., 2020) present some GNN-based methods that have been applied to text classification, and point out that representing texts as graphs can effectively capture semantics among words. (Yao et al., 2019), for instance, use GNNs for text classification. The authors propose a strategy to represent texts as graphs and use a convolutional graph neural network to learn embeddings of words and documents. As a result, they show that the improvement of their graph-based model over state-of-the-art models becomes more prominent as lower the percentage of training data is.

Cross-lingual transfer learning is one of the strategies used to leverage already existing data from higher-resource languages (Ranasinghe and Zampieri, 2020; Stappen et al., 2020; Bigoulaeva et al., 2021). However, this approach can introduce other problems related to the variability between different languages. Moreover, the resulting datasets are heterogeneous not only in terms of languages, but also in terms of domains, which can affect the learning of the models.

We aim to study the performance of hate speech detection with GNNs under the motivation of improving this task in data-poor settings without the need of external data. For this, we propose a graph auto-encoder framework that allows us to learn a latent representation from a set of texts in an unsupervised way. In this representation the texts from the same class are close, hence we can use embeddings from this space to efficiently distinguish instances of different classes. Our framework builds a graph with the texts, encodes the nodes in a low-dimensional space (the latent space) and then reconstructs the original graph with a decoder. The encoder is composed of a transformer layer, which introduces an attention mechanism, and a convolutional layer for generating the embeddings.

We evaluate the embeddings for hate speech detection as a classification task in small datasets. Moreover, we study the multi-domain and multilingual settings, to experimentally analyse whether graphs can jointly represent different types of information. Our contributions are the following ones[1]:

[1]We will make our codes freely available by the publication date of this work.

- We propose a graph auto-encoder for unsupervised representation learning on graph-structured data by reconstructing the initial graph. In this framework we incorporate a self-attention mechanism that allows us to adapt the strengths of the Transformer model (Vaswani et al., 2017) in the generation of embeddings.
- We use the embeddings generated with this framework for hate speech detection and show results which outperforms state-of-the-art models in data-poor settings, without using pre-trained word embeddings.
- We investigate the performance of our approach in multi-domain settings, where the small amount of data once more highlights the potentiality of our proposal.
- We extend the analysis for multilingual hate speech detection and use a strategy to aggregate prior knowledge about the language to obtain outstanding results.

2. Related Work

Hate Speech Detection. Most of the works done to detect hate speech is based on the analysis of textual instances. Other works have studied the phenomenon at the author level (Rangel et al., 2021), where the idea is to analyze a set of texts published by the same author to detect possible propagators of hate on the web. In general, the techniques used for hate speech detection range from traditional machine learning models to methods based on deep learning, such as convolutional neural network and recurrent neural networks, including attention mechanisms (Badjatiya et al., 2017; Gröndahl et al., 2018; Magalhaes, 2019). Due to the nature of the task, it is worth noting the models that take into account certain keywords that may indicate hateful content. The work (De la Peña Sarracén and Rosso, 2021b) proposed an approach for keyword extraction based on the attention mechanism of BERT and a reasoning on a word graph. Experimental results highlighted some points to consider when training models based on keywords. Moreover, the work (De la Peña Sarracén and Rosso, 2021a) studied how models learn bias towards relevant words in the training data. To extract the relevant words, the authors proposed a keyword extraction method based on the harmonic mean of relative frequencies and the discrimination between hateful and non-hateful texts. In recent years, the bidirectional encoder representations from Transformers (BERT) (Devlin et al., 2019), as well as other Transformer-based models such as RoBERTa (Liu et al., 2019) have been widely used due to their ability to capture language phenomena (Mozafari et al., 2020; Samghabadi et al., 2020). In fact, they have been used in most systems with outstanding results in shared tasks (Basile et al., 2019; Zampieri et al., 2020; Mandl et al., 2019). Moreover, (Mozafari et al., 2019) investigated the ability of BERT for detecting hateful content on social media and the results showed a considerable performance in comparison to other existing approaches. That is why we use this model to compare the results obtained with our framework.

Graph Neural Network for Abusive Language Detection. Regarding GNN-based models, the literature points out a number of strategies (Koncel-Kedziorski et al., 2019; Shi et al., 2021a). (Peng et al., 2018) proposed a graph-based deep learning model to convert texts to graphs of words, and then used graph convolution operations over the graph. (Yao et al., 2019) represented documents and words as nodes to construct a graph and used a convolutional graph neural network to learn embeddings of words and documents. As a result, the authors showed improvements over state-of-the-art models for text classification. However, very little has been studied to employ strategies based on GNNs to address the problem of hate speech detection. (Mishra et al., 2019) proposed a convolutional graph neural network for capturing the structure of online communities and the linguistic behavior of the users. They showed that the resulting heterogeneous graph significantly advanced the state of the art in abusive language detection. Thus, to the best of our knowledge, our method is the first proposal to learn embeddings in an unsupervised way for the specific problem of hate speech detection.

3. Graph Auto-Encoders for Hate Speech Detection

In this section, we describe the preliminaries of the framework, followed by details of our proposal.

3.1. Formalization

In this work we consider hate speech detection as a classification problem which involves the classes hate and not-hate. The data comprises N samples, where each sample is given by $\{t_i, y_i\}$. The set $\{t_i\}_{i=1}^{N}$ is composed of texts that are represented with numeric feature vectors $\{x_i\}_{i=1}^{N}$. In order to generate these feature vectors we use Term Frequency - Inverse Document Frequency (TFIDF) representation of each text in $\{t_i\}_{i=1}^{N}$.

TFIDF generates vectors from texts in such a way that often produces lower scores for high frequency function words and increases scores for terms that are more relevant in each text, it is well suited for tasks involving textual similarity. Thus, we represent the texts as vectors with TFIDF scores, for what we do not need external sources to train the initial vectors.

The set $\{y_i\}_{i=1}^{N}$ is composed of the labels 0 and 1, which indicate the presence or not of hate in each of the texts in $\{t_i\}_{i=1}^{N}$. Then, the aim of hate speech detection is to assign one of the labels to each t_i by using x_i.

Our aim in the present work is to learn embeddings from x_i in an unsupervised way to improve the performance in hate speech detection when N is small. Besides, we attempt to use this approach for multi-domain and multilingual hate speech detection, due to the suitability of graphs to jointly represent different types of information. In these cases $\{t_i\}_{i=1}^{N} = \cup_m \{t_i^m\}_{i=1}^{S_m}$, where m represents each of the M domains or languages and S_m its size, such that $N = \sum_{m=1}^{M} S_m$.
We address the problem by using a graph auto-encoder framework. Following, we describe our framework in details.

3.2. Background: Graph Auto-Encoders

Graph neural networks (GNNs) are models based on deep learning to operate on the graph domain. In particular, Graph Auto-Encoders (GAEs) are unsupervised learning frameworks which encode nodes or graphs into a latent vector space. Therefore, they can be used to learn embeddings. In general, they are trained with the aim of reconstructing their original graph input. First, an encoder takes a graph as its input and compresses it into a low-dimensional vector. Then, a decoder takes this vector representation and attempts to generate a reconstruction of the original input. Encoder-decoder pair is designed to minimize the loss of information between the input graph and the output graph (Wu et al., 2021).
Formally, let $G = (V, E)$ be a graph, where V and E represent the set of nodes and edges respectively. Let $X \in \mathbb{R}^{|V| \times d}$ be a matrix containing the features of the nodes, such that the i-th row is a d-dimensional feature vector of the i-th node. Moreover, let $A \in \mathbb{R}^{|V| \times |V|}$ be a matrix representation with a representative description of the graph structure, such as the adjacency matrix. A GAE takes as input the matrices X and A to learn a function $Z = enc(X,A)$ and produces a latent representation $Z \in \mathbb{R}^{|V| \times d'}$ (embeddings), where $d' < d$ is the number of features of the nodes in the latent representation. Then, Z is used to produce an approximate reconstruction output $\hat{A} = dec(Z)$ such that the error between A and $\hat{A}$ is minimized for preserving the global graph structure. Both functions $enc(\cdot, \cdot)$ and $dec(\cdot)$ are often defined through stacked layers.
Graph convolutional layer (GCL) re-defines the notion of convolution for graph data and are widely used as propagation operators for GNNs in general. The main idea is to operate directly on a graph and induce the embedding vectors of nodes based on the properties of their neighbors. A GCL takes as input the matrices X and A and generates a representation $H = f(X, A)$, where $f(\cdot, \cdot)$ is a propagation rule. (Kipf and Welling, 2017) introduced the propagation rule (1), where W is a weight matrix and $\sigma(\cdot)$ is an activation function. The matrix $\tilde{A} = A + I$ (I is the identity matrix) contains self-connections to aggregate, for each node, not only the information from its neighbors but also the node itself. Moreover, the matrix D is the diagonal node degree matrix of $\tilde{A}$, which is used for a symmetric normalization to deal with the problem of changing the scale of the feature vectors.

$$f(X, A) = \sigma(D^{-\frac{1}{2}} \tilde{A} D^{-\frac{1}{2}} X W) \quad (1)$$

Graph transformer layer (GTL) adapts the multi-head attention of Transformer (Vaswani et al., 2017) for graph learning. This was introduced in (Shi et al., 2021b) considering the case of edge features. Given the features vectors $X = \{x_i\}_{i=1}^{N}$, they generate new features vectors $\hat{X} = \{\hat{x_i}\}_{i=1}^{N}$ by calculating multi-head attention for each node i with its neighbors $\mathcal{N}(i)$:

$$q_{c,i} = W_{c,q} x_i \,,\; k_{c,j} = W_{c,k} x_j \,,\; v_{c,j} = W_{c,v} x_j$$
$$e_{c,ij} = W_{c,e} e_{ij}$$
$$\alpha_{c,ij} = softmax(\frac{q_{c,i}(k_{c,j} + e_{c,ij})}{\sqrt{d_c}})$$
$$r_i = W_r x_i$$
$$\hat{x_i} = r_i + \frac{1}{C} \sum_{c=1}^{C} [\sum_{j \in \mathcal{N}(i)} \alpha_{c,ij} (v_{c,j} + e_{c,ij})]$$

where c represents each head, d_c its hidden size and C the total number of heads. The vectors $q_{c,i}$, $k_{c,j}$ and $v_{c,j}$ correspond to the 'query', 'key', and 'value' vectors respectively, and $e_{c,ij}$ is a representation for the edge between i and j. $W_{c,q}$, $W_{c,k}$, $W_{c,v}$ and $W_{c,e}$ are the parameters in the head c. Notice that a term r_i is calculated to add a gated residual connection between layers.

3.3. Auto-Encoder Architecture

Figure 1 illustrates our auto-encoder. In order to generate the input for the model, we build the matrix X with the set of numeric feature vectors $\{x_i\}_{i=1}^{N}$, such that each vector is a row in X. On the other hand, we build the edges among nodes, to generate the matrix A, based on the inner product of the feature vectors. Then, the weight of each edge is defined by the inner product between the original vectors. We only consider edges between node pairs with values higher than a threshold (positive edges). The rest of node pairs are considered as non-existent edges (negative edges).

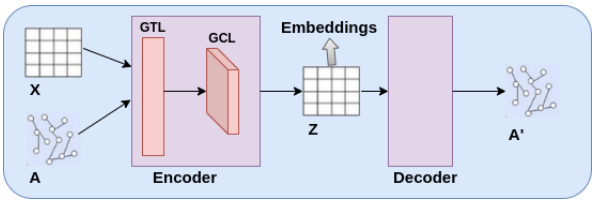


Figure 1: Auto-encoder architecture.

The encoder in our model stacks two layers. The first one is a GTL and the second one is a GCL. In particular, we use a GTL to enrich the node embeddings with attentive information propagation between nodes.

Thus, the encoder uses a GTL as the first layer to determine the relevance between nodes and their neighbors by leveraging the advantages of the attention mechanism among nodes. In this sense, we adopt the proposal in (Shi et al., 2021b) by considering only nodes and using an unique head. Hence, we transform the input X matrix as (2).

$$\begin{aligned} \alpha_{ij} &= softmax(\frac{(W_q x_i)^T W_k x_j}{\sqrt{d}}) \\ \hat{x_i} &= W_r x_i + \sum_{j \in \mathcal{N}(i)} \alpha_{ij}(W_v x_j) \end{aligned} \tag{2}$$

The second and final layer is based on the propagation rule (1) and the ReLU as the activation function. The input of this layer is composed of the new matrix $\hat{X}$ and the matrix A. Thus, we obtain the output of the encoder as (3), where W_c is a parameter matrix.

$$Z = enc(X, A) = ReLU(D^{-\frac{1}{2}}\tilde{A}D^{-\frac{1}{2}}\hat{X}W_c) \quad Encoder \tag{3}$$

The decoder implements the idea of the GAE in (Kipf and Welling, 2016). Thus, we base on the inner product of the embeddings to generate $\hat{A}$. The aim is to decode node relational information from the embeddings by reconstructing A as (4) defines. Then, the auto-encoder (5) is trained by minimizing the negative cross entropy given the real matrix A and the reconstructed matrix $\hat{A}$.

$$\hat{A} = dec(Z) = sigmoid(ZZ^T) \quad Decoder \tag{4}$$

$$\hat{A} = GAE(X, A) = dec(enc(X, A)) \quad Auto - Encoder \tag{5}$$

4. Experimental Design

In this section, we present our methodology for the empirical evaluation of the capability of our framework for unsupervised learning of embeddings. We also present the used dataset and details for the reproduction of the experiments.

4.1. Dataset

We evaluate our proposed auto-encoder framework on the XHate-999 dataset (Glavaš et al., 2020), which was built for abusive language detection. This dataset is composed of large training and validation sets of English texts, and a small multi-domain and multilingual test set. The test set contains text of six typologically diverse languages: English (EN), German (DE), Russian (RU), Turkish (TR), Croatian (HR) and Albanian (SQ). For each language there are three distinct domains: Fox News (Gao) with 99 samples, Twitter/Facebook (Trac) with 300 samples, and Wikipedia (Wul) with 600 samples, for a total of 999 samples per language. We only rely on the test set since our purpose is to study the data-poor settings as well as the multi-domain and multilingual perspective.

4.2. Experimental Setup

For each experiment we set the size of the vectors generated with the GTL in the encoder to 32 and the size of the output of the GCL to 16. The auto-encoder was trained using batches of 32 instances and the Adam optimizer with a learning rate of 0.01, in 200 epochs with the strategy of early stopping with patience of 10. For the threshold used in the generation of the matrix A, we searched in $\{0.01, 0.1, 0.5\}$, but realized that a value close to the average of the weights calculated for the pairs of vectors fitted in a better way, hence we set this value to 0.07.

In the evaluation, we first visualize the embeddings in the latent representation, generated with the encoder of our graph auto-encoder. We also visualize the initial vectors (TFIDF) to visually compare both representations.

Secondly, we evaluate the capacity of the embeddings on the task of node classification to study the performance of hate speech detection. In this sense, we use a classifier of two fully connected layers of 32 neurons with the ReLU activation function and the softmax function in a last layer to generate the predictions. This classifier was used to obtain prediction for the texts with the initial representation and on the other hand, with the embeddings obtained with our encoder. Hence, we can compare the results of classification between them. In both cases, the classifier was trained with a size of batch 32, using the Adam optimizer with a learning rate of 0.01, in 10 epochs. For test we separate the 30% of the data and the rest was used to train. That is, we used 30% from the test set of the XHate-999 dataset for testing and the other 70% for training. We run all the experiments five times and report the average scores.

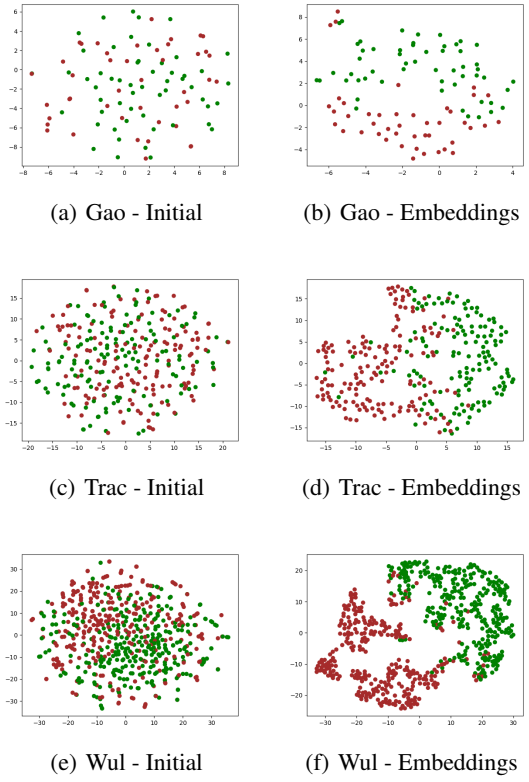
5. Embeddings Evaluation

In this section we analyze the mono-domain and monolingual evaluation. Therefore, we focus on each domain and each language separately.

5.1. Analysis of Latent Representation

In order to better analyze the generated embeddings by our encoder, we use t-SNE (Pezzotti et al., 2017) to visualize the initial and the latent representation of the vectors, corresponding to the hateful and non-hateful texts. As illustration, Figure 2 shows the results for the texts in English in each of the domains Gao, Trac and Wul. In each case, the representation for the initial vectors (with TFIDF) is visualized on the left, and on the right there is the latent representation (embeddings). We can see that our model can be used to distinguish both classes, since the separation between hateful

texts (red points) and non-hateful texts (green points) is more evident. Then, with this representation, a simple algorithm can be used to separate both types of texts. Similar behaviour was observed for the rest of the languages.



(a) Gao - Initial (b) Gao - Embeddings

(c) Trac - Initial (d) Trac - Embeddings

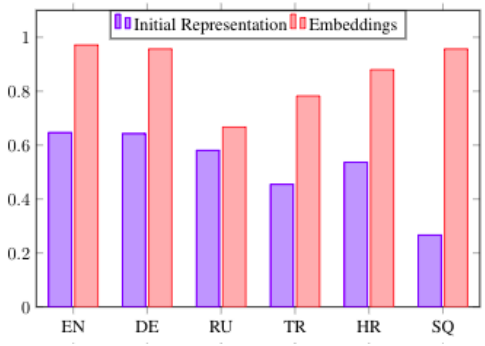
(e) Wul - Initial (f) Wul - Embeddings

Figure 2: Representation for English texts with t-SNE.

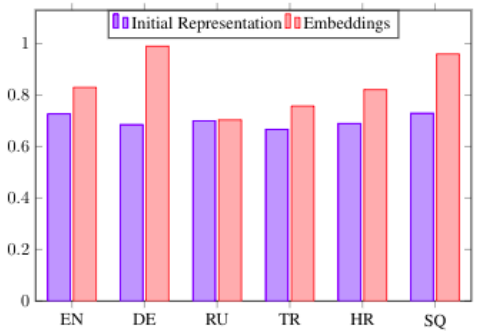
5.2. Evaluation for Hate Speech Detection

The results for hate speech detection, using both the initial vectors and the embeddings, are summarized in Figure 3. In general, we observe an improvement by using the embeddings as the input in the classifier. Thus, we can verify the suitability of the embeddings to discriminate among classes.

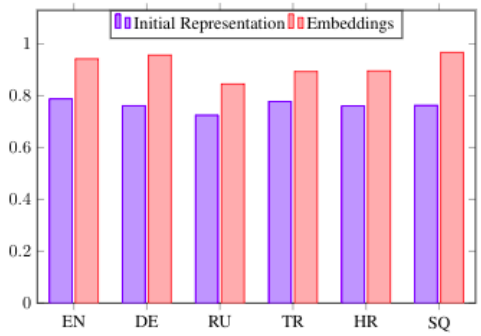
Notice that the results between both variants were similar only for the case of Russian texts in the Trac domain. Figure 4 illustrates the initial and latent representation for Russian, where we can see that in the Trac domain it is more difficult to learn embeddings that allow discriminating between classes. This suggests that in this domain and language, the hateful and non-hateful texts are more similar. In future work, we will try to increase the number of convolutional layers in the encoder to make a deeper propagation and analyze if this case improves. Anyway, for Gao and Wul in this same language, we observe better performance for the embeddings.



(a) Gao



(b) Trac



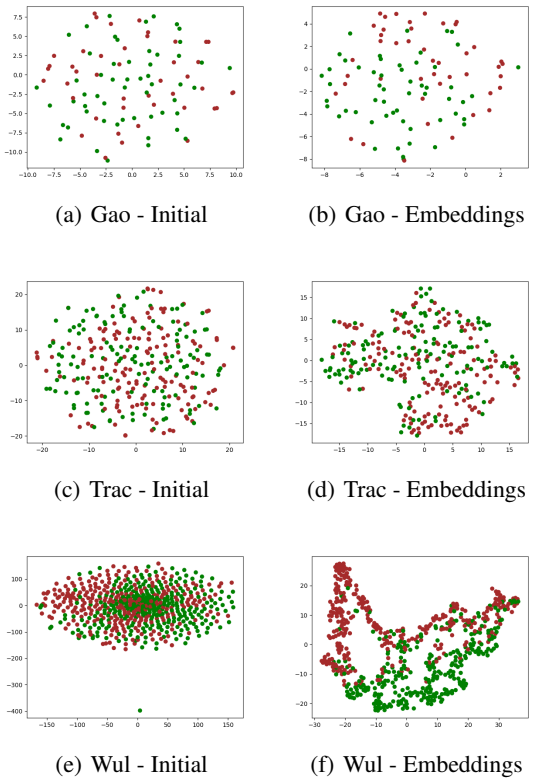
(c) Wul

Figure 3: F1 in Hate Speech Detection.

6. Multi-domain Evaluation

Besides the mono-domain evaluation, we focus on the multi-domain setting. In this case, the set of texts for the input of the auto-encoder is composed of three different types of data per language i.e. $\{t_i\}_{i=1}^{N} = \{t_i^{Gao}\}_{i=1}^{99} \cup \{t_i^{Trac}\}_{i=1}^{300} \cup \{t_i^{Wul}\}_{i=1}^{600}$.
In order to compare our classification results with a state-of-the-art model we use the multilingual BERT (mBERT) (Devlin et al., 2019) and XLM-R (Conneau et al., 2020). We used the HuggingFace Transformers framework[2] with the pre-trained models bert-base-multilingual-cased and xlm-roberta-base. For these models, the input is composed of the texts $\{t_i\}_{i=1}^{N}$ instead of the vectors $\{x_i\}_{i=1}^{N}$. For fine-tuning we used the same setup that we presented above for the classi-

[2]https://github.com/huggingface/transformers



(a) Gao - Initial (b) Gao - Embeddings

(c) Trac - Initial (d) Trac - Embeddings

(e) Wul - Initial (f) Wul - Embeddings

Figure 4: Representation for Russian texts with t-SNE.

fier that we use to evaluate our embeddings. The only difference is that we set the learning rate to 2e-5.

	GAE		
	Gao	Trac	Wul
EN	$0.9714_{.018}$	$0.8301_{.025}$	$0.9424_{.007}$
DE	$0.9565_{.022}$	$0.9900_{.017}$	$0.9569_{.011}$
RU	$0.6667_{.014}$	$0.7238_{.010}$	$0.8453_{.014}$
TR	$0.7826_{.008}$	$0.7567_{.030}$	$0.8936_{.021}$
HR	$0.8799_{.030}$	$0.8214_{.023}$	$0.8958_{.017}$
SQ	$0.9565_{.011}$	$0.9600_{.011}$	$0.9673_{.026}$
	XLM-R		
	Gao	Trac	Wul
EN	$0.5909_{.041}$	$0.7245_{.022}$	$0.8538_{.034}$
DE	$0.6857_{.031}$	$0.7272_{.014}$	$0.8635_{.049}$
RU	$0.5754_{.009}$	$0.7070_{.005}$	$0.8384_{.041}$
TR	$0.5171_{.011}$	$0.7371_{.017}$	$0.8199_{.036}$
HR	$0.5050_{.047}$	$0.6377_{.041}$	$0.8384_{.047}$
SQ	$0.5642_{.041}$	$0.7148_{.016}$	$0.8231_{.041}$

Table 1: F1 in Multi-domain Hate Speech Detection.

Note that in (Glavaš et al., 2020), the authors reported the results on the entire test set. In our experiments, we used a part of the test set 3 times. That is why we reproduced the experiments of that paper with mBERT and XLM-R using the same data that we used to evaluate our framework. Anyway, we observed that the reported results in that paper do not exceed 0.90 of F1.

	All		
	GAE	XLM-R	mBERT
EN	$\mathbf{0.8333}_{.010}$	$0.5642_{.047}$	$0.5111_{.053}$
DE	$\mathbf{0.9565}_{.014}$	$0.4545_{.038}$	$0.4850_{.045}$
RU	$\mathbf{0.8333}_{.001}$	$0.4923_{.061}$	$0.4527_{.031}$
TR	$\mathbf{0.8799}_{.004}$	$0.6192_{.043}$	$0.3864_{.057}$
HR	$\mathbf{0.8461}_{.012}$	$0.6459_{.039}$	$0.4545_{.044}$
SQ	$\mathbf{0.9565}_{.002}$	$0.4978_{.021}$	$0.4457_{.046}$

Table 2: F1 in Multi-domain Hate Speech Detection when using all domains.

Results and Discussion. Tables 1 and 2 summarizes the results of these experiments. The results obtained by using our embeddings are identified with the acronym GAE. In particular, Table 2 illustrates the results in the multi-domain setting per language. While Table 1 corresponds to the results obtained for each domain separately.

We observe two interesting points. First, GAE seems to be more stable in data-poor settings. We verify outstanding results even in Gao, which is the domain with the least amount of data. In contrast, the smaller the dataset, the worse the results obtained with XLM-R. Notice that for all the languages, the best results of XLM-R were in the Wul domain, which is the larger one. This confirms the findings in (Yao et al., 2019), where the authors use a model based on a convolutional graph neural network for text classification, and point out that the improvement over state-of-the-art models becomes more prominent as the percentage of training data is lower.

On the other hand, we note that XLM-R and mBERT obtain the worst results in the multi-domain setting. This suggests that heterogeneous data can affect the performance of these models. The behaviour is different for GAE. Although we do not see any gains moving from the mono-domain, we do not observe a considerable decrease. This suggests the suitability of our framework to deal not only with data-poor settings, but also with heterogeneous data.

7. Multilingual Evaluation

In order to evaluate the multilingual perspective, we consider the combination of all the languages. Therefore, the set of texts for the input of the auto-encoder is composed of six different languages per domain i.e $\{t_i\}_{i=1}^{N} = \cup_{m \in \mathrm{L}} \{t_i^m\}_{i=1}^{6 \times S}$, where S is the amount of samples in the specific domain and $\mathrm{L} = \{EN, DE, RU, TR, HR, SQ\}$. Then, we use three datasets, one per domain. The size of the dataset in Gao is 594 (6*99), in Trac is 1800 (6*300), and in Wul is 3600 (6*600).

For comparison we use mBERT and XLM-R as in the multi-domain evaluation.

Results and Discussion Table 3 illustrates the results of the multilingual evaluation. We observe that mBERT and XLM-R outperform the classifier that uses our embeddings. In fact, the results obtained with our framework decrease considerably. This makes sense, since no knowledge about the difference of the languages has been added in the learning of our embeddings. Whereas mBERT and XLM-R have been trained with large data collections that include the languages of the datasets.

For this reason, we use a strategy to incorporate language knowledge in the input of our graph auto-encoder with the embeddings from the Universal Sentences Encoder (USE)[3] (Cer et al., 2018). The results of this new variant (GAE-USE) are also shown in Table 3. In this way, the use of the embeddings once again improves the results obtained with mBERT and XLM-R.

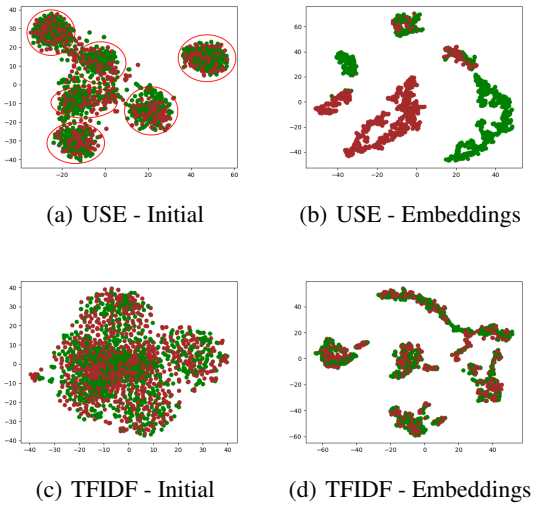
Domain	Gao	Trac	Wul
GAE	$0.3972_{.090}$	$0.6858_{.062}$	$0.6255_{.058}$
GAE-USE	$\mathbf{0.9308}_{.011}$	$\mathbf{0.9598}_{.005}$	$\mathbf{0.9491}_{.021}$
mBERT	$0.7047_{.054}$	$0.7952_{.080}$	$0.8939_{.072}$
XLM-R	$0.7349_{.015}$	$0.8585_{.012}$	$0.9303_{.008}$

Table 3: F1 in Multilingual Hate Speech Detection.

Adding Language Knowledge. In order to add information about the languages, we change the strategy to represent $\{t_i\}_{i=1}^{N}$ as $\{x_i\}_{i=1}^{N}$. In this case, instead of using TFIDF, we use the multilingual model of the universal sentences encoder (USE)[4]. This model was trained on several data sources and tasks to dynamically adapt a wide variety of natural language understanding tasks. The input is a text of variable length and the output is a 512 dimensional vector.

Figure 5 illustrates the representation of the initial vectors on the left and the latent representation (embeddings) on the right. This only corresponds to the case of Trac, but we observed similar behaviour in the other two domains. The two first Figures 5(a) and 5(b) show the representation when we add knowledge about languages with USE. In the initial representation we observe six groups (marked with red circles) that suggest the six different languages that the input contains. In the latent representation we observe a division among classes, similar to the ones obtained in the monolingual evaluation. Therefore, it seems that our framework can be useful to deal with multilingual datasets by adding prior knowledge of the languages.

On the other hand, Figures 5(c) and 5(d) show the representations obtained by using TFIDF. That is, the case where no knowledge about languages is added. An interesting phenomenon in the embeddings generated



(a) USE - Initial (b) USE - Embeddings

(c) TFIDF - Initial (d) TFIDF - Embeddings

Figure 5: Multilingual Representation with t-SNE for the Trac domain.

with our encoder is that separated groups of points with both types of points (red and green) can be identified. It seems that the encoder has learned the difference among languages instead of the difference between the classes hate and not-hate. In this sense we note that this representation is somewhat similar to the initial representation obtained with USE. Therefore, the embeddings learned with our framework could improve by adding more layers in the encoder. Therefore, we attempt to adapt the proposal in (Li et al., 2019) as future work. In that paper, the authors present a ways to successfully train very deep convolutional graph neural network.

8. Conclusion and Future Work

In this work, we proposed a graph auto-encoder framework to learn embeddings of a set of texts in an unsupervised way. The auto-encoder receives an initial vector representation of the texts and the relation among them in the form of a graph, to generate a low-dimensional representation. Then, the embeddings are extracted from this latent representation. In this sense we built the encoder with a sequence of a transformer layer and a convolutional layer to consider the information of the graph structure in the learning of the embeddings. We used this framework for hate speech detection by using the embeddings as input of a classifier. In the evaluation we considered multi-domain and multilingual settings with small datasets. We observed promising results by outperforming mBERT and XLM-R, one of the state-of-the-art models in hate speech detection. We noticed that the improvement by using the embeddings generated with our auto-encoder became more notable in small data, suggesting the suitability of our proposal to deal with data-poor settings. Moreover, we observed that the use of our embeddings was

[3] https://tfhub.dev/google/universal-sentence-encoder/4

[4] https://tfhub.dev/google/universal-sentence-encoder-multilingual-large/3

more stable in multi-domain settings. While in the case of multilingual settings, we had to add prior knowledge about languages to avoid a decrease in the performance. As future work, we will extend our analysis by building a deeper encoder. The idea is to investigate if it is possible to learn the embeddings for a multilingual setting without the need of prior knowledge about the languages. Moreover, we will adapt our graph auto-encoder to encode not only text, but also visual information. Thus, we will be able to deal with multimodal hate speech detection.

Acknowledgements

This research work was partially funded by the Spanish Ministry of Science and Innovation under the research project MISMIS-FAKEnHATE on Misinformation and Miscommunication in social media: FAKE news and HATE speech (PGC2018-096212-B-C31). The first author gratefully acknowledges the support of the Pro^{2}Haters - Proactive Profiling of Hate Speech Spreaders (CDTi IDI-20210776) and XAI-DisInfodemics: eXplainable AI for disinformation and conspiracy detection during infodemics (MICIN PLEC2021-007681) R&D grants. The work of the first author was also partially funded by the Centre for the Development of Industrial Technology (CDTI) of the Spanish Ministry of Science and Innovation under the research project IDI-20210776 on Proactive Profiling of Hate Speech Spreaders - PROHATER (Perfilador Proactivo de Difusores de Mensajes de Odio). Moreover, the work of the second author was partially funded by the Generalitat Valenciana under DeepPattern (PROMETEO/2019/121).

Bibliographical References

Badjatiya, P., Gupta, S., Gupta, M., and Varma, V. (2017). Deep Learning for Hate Speech Detection in Tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760.

Basile, V., Bosco, C., Fersini, E., Debora, N., Patti, V., Pardo, F. M. R., Rosso, P., Sanguinetti, M., et al. (2019). Semeval-2019 Task 5: Multilingual Detection of Hate Speech against Immigrants and Women in Twitter. In *13th International Workshop on Semantic Evaluation*, pages 54–63. Association for Computational Linguistics.

Bigoulaeva, I., Hangya, V., and Fraser, A. (2021). Cross-Lingual Transfer Learning for Hate Speech Detection. In *Proceedings of the First Workshop on Language Technology for Equality, Diversity and Inclusion*, pages 15–25. Association for Computational Linguistics.

Cer, D., Yang, Y., yi Kong, S., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y.-H., Strope, B., and Kurzweil, R. (2018). Universal Sentence Encoder. *CoRR*, abs/1803.11175.

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., and Stoyanov, V. (2020). Unsupervised Cross-Lingual Representation Learning at Scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451. Association for Computational Linguistics.

De la Peña Sarracén, G. L. and Rosso, P. (2021a). Automatic Keyword Extraction for Bias Analysis in Hate Speech Detection. *Natural Language Engineering (under review)*.

De la Peña Sarracén, G. L. and Rosso, P. (2021b). Offensive Keyword Extraction based on the Attention Mechanism of BERT and the Eigenvector Centrality using a Graph Representation. *Personal and Ubiquitous Computing*, pages 1–13.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 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)*, pages 4171–4186. Association for Computational Linguistics.

Fortuna, P. and Nunes, S. (2018). A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys (CSUR)*, 51(4):1–30.

Glavaš, G., Karan, M., and Vulić, I. (2020). XHate-999: Analyzing and Detecting Abusive Language Across Domains and Languages. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6350–6365.

Gröndahl, T., Pajola, L., Juuti, M., Conti, M., and Asokan, N. (2018). All You Need is" Love" Evading Hate Speech Detection. In *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*, pages 2–12.

Kipf, T. N. and Welling, M. (2016). Variational Graph Auto-Encoders. *arXiv preprint arXiv:1611.07308*, abs/1611.07308.

Kipf, T. N. and Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17.

Koncel-Kedziorski, R., Bekal, D., Luan, Y., Lapata, M., and Hajishirzi, H. (2019). Text Generation from Knowledge Graphs with Graph Transformers. In *NAACL*.

Li, G., Muller, M., Thabet, A., and Ghanem, B. (2019). Deepgcns: Can GCNs go as Deep as CNNs? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9267–9276.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*, abs/1907.11692.

Magalhaes, A. (2019). Automating Online Hate Speech Detection: A Survey of Deep Learning Approaches. Master's thesis, School of Informatics, University of Edinburgh.

Mandl, T., Modha, S., Majumder, P., Patel, D., Dave, M., Mandlia, C., and Patel, A. (2019). Overview of the HASOC Track at FIRE 2019: Hate Speech and Offensive Content Identification in Indo-European Languages. In *Proceedings of the 11th Forum for Information Retrieval Evaluation*, FIRE '19, page 14–17. Association for Computing Machinery.

Mishra, P., Del Tredici, M., Yannakoudakis, H., and Shutova, E. (2019). Abusive Language Detection with Graph Convolutional Networks. *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 1 2145-2150.*

Mozafari, M., Farahbakhsh, R., and Crespi, N. (2019). A BERT-based Transfer Learning Approach for Hate Speech Detection in Online Social Media. In *International Conference on Complex Networks and Their Applications*, pages 928–940. Springer.

Mozafari, M., Farahbakhsh, R., and Crespi, N. (2020). Hate Speech Detection and Racial Bias Mitigation in Social Media Based on BERT Model. *PloS one*, 15(8):e0237861.

Peng, H., Li, J., He, Y., Liu, Y., Bao, M., Wang, L., Song, Y., and Yang, Q. (2018). Large-Scale Hierarchical Text Classification with Recursively Regularized Deep Graph-CNN. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, page 1063–1072. International World Wide Web Conferences Steering Committee.

Pezzotti, N., Lelieveldt, B. P. F., Maaten, L. v. d., Höllt, T., Eisemann, E., and Vilanova, A. (2017). Approximated and User Steerable tSNE for Progressive Visual Analytics. *IEEE Transactions on Visualization and Computer Graphics*, 23(7):1739–1752.

Poletto, F., Basile, V., Sanguinetti, M., Bosco, C., and Patti, V. (2020). Resources and Benchmark Corpora for Hate Speech Detection: A Systematic Review. *Language Resources and Evaluation*, pages 1–47.

Ranasinghe, T. and Zampieri, M. (2020). Multilingual Offensive Language Identification with Cross-lingual Embeddings. pages 5838–5844.

Rangel, F., De la Peña, G. L., Chulvi, B., Fersini, E., and Rosso, P. (2021). Profiling Hate Speech Spreaders on Twitter Task at PAN 2021. In *Proceedings of the Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum, Bucharest, Romania*, volume 2936 of *CEUR Workshop Proceedings*, pages 1772–1789.

Samghabadi, N. S., Patwa, P., Srinivas, P., Mukherjee, P., Das, A., and Solorio, T. (2020). Aggression and Misogyny Detection Using BERT: A Multi-task Approach. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, pages 126–131.

Shi, Y., Zhang, S., Zhou, C., Liang, X., Yang, X., and Lin, L. (2021a). GTAE: Graph Transformer–Based Auto-Encoders for Linguistic-Constrained Text Style Transfer. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 12(3):1–16.

Shi, Y., Huang, Z., Wang, W., Zhong, H., Feng, S., and Sun, Y. (2021b). Masked Label Prediction: Unified Message Passing Model for Semi-supervised Classification. In *IJCAI*.

Stappen, L., Brunn, F., and Schuller, B. (2020). Cross-lingual Zero-and Few-shot Hate Speech Detection utilising Frozen Transformer Language Models and AXEL. *arXiv preprint arXiv:2004.13850*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. (2021). A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., and Jegelka, S. (2018). Representation Learning on Graphs with Jumping Knowledge Networks. In *International Conference on Machine Learning*, pages 5453–5462. PMLR.

Yao, L., Mao, C., and Luo, Y. (2019). Graph Convolutional Networks for Text Classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7370–7377.

Zampieri, M., Nakov, P., Rosenthal, S., Atanasova, P., Karadzhov, G., Mubarak, H., Derczynski, L., Pitenis, Z., and Çöltekin, Ç. (2020). SemEval-2020 Task 12: Multilingual Offensive Language Identification in Social Media (OffensEval 2020). pages 1425–1447.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. (2020). Graph Neural Networks: A Review of Methods and Applications. *AI Open*, 1:57–81.