Proposal for PAN 2021 Profiling Hate Speech Spreaders

Lunex Team (Amistad con los pueblos)

Considering the method proposed to capture the relations among the terms within the tweets, the Architecture we proposed is necessarily modular (i.e has an encoder module and a prediction module). This is because the text representation will be computed employing a Transformer Language Model **(LM)**, specifically BETO for the Spanish language and BERTweet for the English language, which introduces a limit to the input sequence length.
Some proposed ideas related to using the transition point, due to the short remaining time and the amount of tuning combinations required by the hyper-parameters, will be considered after conducted the following ideas, which still being promising. In the succeeding paragraphs, we describe the strategies proposed for training the Encoder module.

Encoder Module

One of the main challenges which brings this modular Architecture, because of the structure and the annotations coming with the data, is the selection of the targeted task to train the Encoder module. For this, the first idea that we propose to experiment with is applying fine-tuning over the LM using Masked Language Modeling task **(MLM)** for introducing some bias towards the specific language relations existing within the proposed data and which are related to hate spreading.

Another way to relate the encoding of the tweets with the actual task is fine-tuning the LM onto the specific task of predicting whether a tweet spreads hate or not, but the annotation on the provided data involves labeling only at the account level (i.e a hate spreader account may contain tweets without hate speech). To address this problem we could use external data, as the one from Hateval@Semeval 2019, assuming the risk of using data from different origin and domain.
Having this into account we propose to train the Encoder for predicting whether a message comes from the same author or not, in this way we can capture specific features that characterize the authors' style. To this end, we may use a Siamese Neural Network **(SNN)** fixing the LM and adding some dense layers which learn representations that get closer tweets from the same person and take away the ones from different accounts w.r.t a distance or similarity function. Also before fixing the LM weights, we can fine-tune it with one of the ideas described above to relate it with the task language, and in case of choosing the second one, we can use the IMF technique to give relevance to the output

vectors of the LM by using an attention mechanism.

Nonetheless, with this approach come the challenges of choosing the pairs to train the SNN correctly and somehow keep the data from growing exaggeratedly, to address these problems we propose using an Object Reduction technique described as :

- Select seed for Hate prototype
- Select seed for No_Hate prototype
- Get Hate prototype subset complement
- Get Hate prototype subset complement

Finally, at this point, we have each tweet representation, but in order to represent a whole profile, we need to obtain a dense vector combining those tweets information. There are some ideas such as: take the normalized sum of them, apply max-pooling, or simply concatenate them.

Also, we could use a Graph Neural Network (GNN)[1], but here we have the same annotation problem again, since for obtaining a representation that relates a node (tweets) w.r.t their neighbors (the other profile tweets) we need to target some task, depending on which the GNN will learn to predict a value for a node having in account their surrounding nodes and their relationships.

[1] https://medium.com/dair-ai/an-illustrated-guide-to-graph-neural-networks-d5564a551783

Prediction Module

For this module we propose employ the **Seidman Method** described as:

Algorithm 1 Seidman Method

Input:
Hate Profiles **A**, NO_Hate Profiles **B**, Unknown Profile **U**

procedure SEIDMAN_METHOD(A, B, U)
  Select non randomly from A a subset A^*
  for all object a from A^*: **do**
    Select non randomly from B a subset B^*
    for all object b from B^*: **do**
      loop: 1 to k:
        Chose non randomly subset of features
        Represent $\bar{a}$, $\bar{b}$, $\bar{U}$ from a, b, U respectively
        if Sem($\bar{a}$, $\bar{U}$) *criterion* Sem($\bar{b}$, $\bar{U}$) **then**
          ksuma_bar ++
        else:
          ksumb_bar++
        end if
      end loop
      suma_bar += (ksuma_bar >ksumb_bar)
    end for
    $\hat{y}$ += (suma_bar > $\frac{|B^*|}{2}$)
  end for
  $\hat{y} = (\hat{y} > \frac{|A^*|}{2})$ ▷ 1: Hate Speech Spreader 0: No Hate Speech Spreader
end procedure

Output: $\hat{y}$

Here, *Sem* stands for a function which given the vectorial representation of two objects yields a numerical value describing some relation, such as how likely they are or the probability of both belonging to a certain class.
Also x *criterion* y, is a placeholder to decide after comparing x and y if assume lager values as convenient or lower ones, this depends on *Sem* (e.g a large similarity between x and y is convenient whereas a short distance it is) .

We think using another SNN based on a simple FNN as *Sem* which predicts the probability of two objects (profile representations) belong to the same class could be interesting. Also here we can combine each individual tweet representation from the LM by an attention mechanism.
Nevertheless, we will test the Method performance based on conventional metrics. Note that we have only a few examples of profiles (Hate Speech Spreaders o not) to train a neural model.

Having into account that a dense representation of a profile extracted by NNs contains features not necessarily independent and mostly uninterpretable, unlike conventional hand-crafted representations, referring to the Seidman Method we propose to avoid the features reduction section (the 1 to k loop).