

Identifying Author Profiles Containing Irony or Spreading Stereotypes with SBERT and Emojis

Notebook for PAN at CLEF 2022

Narjes Tahaei, Harsh Verma, Parsa Bagherzadeh, Farhood Farahnak, Nadia Sheikh and Sabine Bergler

Concordia University, Montreal

Abstract

The Profiling Irony and Stereotype Spreaders on Twitter Shared Task at CLEF 2022 asks to analyze a set of tweets from an author in order to determine whether the author spreads irony and stereotypes. Our approach is to feed all tweets from an author to SBERT as a single input batch and feed all output vectors from the model to an additive attention layer to produce a vector representation per author. This vector representation is the input to a linear binary classifier. We placed second among 40 participants with a test accuracy of 97.8. We selected the best model using 5 fold cross-validation.

Keywords

SBERT, Tokenization, Emojis,

1. Introduction

Irony is the expression of one's meaning by using language that normally signifies the opposite [1]. *Stereotype* is a fixed, over-generalized belief about a particular group that can propagate false biases regarding that group [2]. Twitter is a widely used communication platform with a high percentage of tweets using irony and stereotypes. The Profiling Irony and Stereotype Spreaders on Twitter Shared Task [3] is to classify an author as someone who spreads irony and stereotypes from so-called *Profiles* of 200 tweets for each of 420 authors.

Current best practice for NLP tasks is to feed sentence input to a BERT-like pre-trained language model [4] and use the CLS output token as input to a classifier. If this task was to classify individual tweets as containing ironic or stereotypical content, this would be a satisfactory approach and would be expected to perform well. But this task calls for the classification of Profiles with 200 tweets each, with each tweet providing important context. We thus batch the data with batch size 200 to model the Profiles.

We fine-tune SBERT [5] to construct individual vector representations for each tweet from an author followed by an additive attention layer to calculate a vector representation for each Profile. Profile vectors feed into a linear classifier.

CLEF 2022 – Conference and Labs of the Evaluation Forum, September 5–8, 2022, Bologna, Italy

✉ narjesossadat.tahaei@mail.concordia.ca (N. Tahaei); farhood.farahnak@mail.concordia.ca (F. Farahnak);

bergler@cse.concordia.ca (S. Bergler)

🆔 0000-0001-9987-6747 (S. Bergler)



© 2022 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)

We submitted two runs which differ only in the number of epochs, for which the model ran. *narcis* ran for six epochs and obtained rank 10 with an accuracy of 0.93; *harshv* ran for seven epochs and obtained rank 2 with an accuracy of 0.98 [6]. The results were submitted to the TIRA platform [7].

2. Related Work

The authors of the top ranked system on SemEval15 Task 11 on Sentiment Analysis of Figurative Language in Twitter (Task 11) [8, 9] did not make any adaptations to a sentiment analysis pipeline aside from training on the training data. This suggests that sentiment (which is often inverted in ironic language) co-occurs with other cues that are strong enough for a normal sentiment analysis system to succeed.

In [10], the authors applied supervised machine learning to a set of Twitter corpora that have been used earlier for the irony and sarcasm detection to identify ironic tweets. Different groups of features have been used. Structural features used to detect common patterns of the ironic tweets including length, type of punctuation, and emoticons. Other features used to capture information about affect, such as semantic lexicons and dictionaries of affective terms. Affective features outperformed in distinguishing among ironic and non-ironic tweets in all Twitter corpora.

The authors in [11] propose a model to detect hate spreaders on Twitter. BERTweet is used to embed each tweet into a vector representation. Two methods are used to construct an author's profile. First, a graph neural network is used to relate information from different posts to their authors. Second, a sequence of encoded tweets is given to an additive attention-based, fully connected neural network. The single vector coming from adding all weighted vectors is used by a classifier. To classify whether an author is irony and stereotype spreader or not a version of impostor method is introduced. In [12] the Impostors method is used to determine whether two documents are from the same author. The method checks whether X (from a hate spreader set), is closer to Y (from non-hate spreader set) than to each one of impostors, which is a similarity function here. The system proved useful for the hate speech spreaders identification shared task at PAN 2021. The second method, which is a sequence based Profile modeling used similar additive attention approach. They first applied the additive attention to obtain a vector representation for each tweet and then added them together to have a Profile representation. In contrast, we have all tweets of an author available for fine tuning and it is the additive attention layer which gives us a representation for a Profile.

3. Data

The training dataset consists of 420 XML files, corresponding to authors, and each XML file contains 200 tweets from its author. The dataset contains anonymized URLs, hashtags, and user mentions. The URLs, hashtags, and user mentions are replaced with tags like #URL#, which are frequently repeated at the beginning or end of tweets. These repetitions are retained.

Tweets also frequently contain emojis, in fact many tweets contain many different and even repetitions of the same emojis. Emojis are retained.

4. Model

For comparison, we feed the data to two classifiers, a SVM classifier using authors’ TF-IDF vectors and a fine-tuned SBERT model.

For the SVM baseline, we construct TF-IDF vectors for all Profiles. Tokens that occur fewer than 30 times in the whole corpus were removed from the vocabulary. The final size of the vocabulary is 4128. The SVM baseline yields 86% accuracy on our evaluation set.

Our main model is based on SBERT [5], a modification of BERT that is fine-tuned for sentence representations on NLI data to produce semantically meaningful sentence embeddings that can be compared using cosine-similarity. SBERT adds a pooling layer on top of BERT’s 12 transformer layers, which produces a fixed size sentence embedding. For fine-tuning there is a siamese and triplet networks to update weights. During training process of finding the most similar sentence pairs, each sentence in a pair are inputs to the networks, which outputs two sentence embeddings. Then, the similarity of these embeddings is computed using cosine similarity between vectors. The resulting sentence embeddings can be used for different tasks.

SBERT used mean pooling for the semantic similarity task [5]. We experimented with combining the CLS token output with mean pooling, max pooling, or additive attention. For mean pooling and max pooling, we compute the mean and maximum of all 200 CLS outputs for a Profile. The additive attention on the 200 CLS outputs outperformed the other two methods.

Each training sample corresponds to a Profile and comprises 200 tweets. We use SBERT and input tweets with a batch size of 200:

$$H = [h_{CLS_1}; \dots; h_{CLS_{200}}] = \text{SBERT}(Tweet_1, \dots, Tweet_{200}) \quad (1)$$

where $H = [h_{CLS_1}; \dots; h_{CLS_{200}}]$ is the row-level concatenation of the [CLS] representations of the 200 Profile tweets. To obtain a single representation for the authors’ profile we then use additive attention:

$$h = \text{softmax}(W_{att}H^T)H \quad (2)$$

where $W_{att} \in \mathbb{R}^{1 \times 384}$ is a learnable parameter. The additive attention assigns importance weights to tweets and provides a weighted sum of the [CLS] representation. The profile representation h is then used for the final classification:

$$p = \text{softmax}(hW + b) \quad (3)$$

where $W \in \mathbb{R}^{384 \times 2}$ is a linear transformation that characterizes the classifier and $p \in \mathbb{R}^2$ represents the class probabilities.¹ We calculate loss using Cross-Entropy loss and optimize the network using the Adam [13] optimizer with $lr = 5e - 6$.

4.1. Input representations

There are two ways to input the data into the model. One way is to assign the Profile’s label to all 200 of its tweets and classify individual tweets. When we have assigned predictions to all

¹Note that the task is a binary classification problem.

tweets of a profile, they have to be aggregated, for instance with a majority vote or an additive attention layer to obtain Profile labels.

We chose instead to represent Profiles in batches of size 200, putting all tweets from one Profile into a single batch. The Profile vector is obtained using an attention layer over the CLS tokens of all tweets in a batch. This high-level author encoding vector is used as input to the final layer for the classification.

We use the batch method for our system. During training, cost function and gradient descent are calculated for Profiles only. This is intended to avoid the noise created by over-assigning the Profile label to tweets that do not contribute to its assignment² and produces better gradient descent and finally a richer final representation for a Profile.

5. Model Architecture

The BERT tokenizer³ prepares the input for the model. It splits input into sub-word token strings, converts tokens to their ids, adds new tokens to the vocabulary, manages special tokens, pads, and truncates vectors. For each input batch, we first called the tokenizer along with the model. It turns out that the computational overhead of the subword tokenizer restricts the model to run properly on the large batch size. This was addressed by transferring the tokenizer outside the model.

We used PyTorch⁴ and the SBERT architecture implemented by Huggingface⁵, and ran models on one GPU. The longest tweet has over 600 tokens, which made it impossible to give all Profile tweets as one batch to the SBERT model. Since only 25 tweets have between 200 to about 600 tokens, we truncated tweets to a maximal length of 200 tokens and set padding to True. Thus the model has the dimensions of 200 tweets * 200 tokens * 384 hidden layers (SBERT's default).

5.1. Feature Exploration

5.1.1. Emojis

In many texts, specifically in the context of short texts like tweets, emojis serve as a proxy for the emotional contents of the text [14]. When using irony, authors may add emojis to the text to ensure that the double entendre does not go unnoticed. Emojis both, text descriptions and UNICODE values⁶. We choose UNICODE values, to keep emojis distinct from text.

In order to use emojis in SBERT, we add their UNICODEs to the SBERT vocabulary with random weight vectors.

²An irony spreader may have a majority of tweets that are neutral, factual, and objective.

³https://huggingface.co/transformers/v3.3.1/main_classes/tokenizer.html

⁴<https://pytorch.org/>

⁵<https://huggingface.co/transformers/v3.3.1/index.html>

⁶<https://unicode.org/emoji/charts/full-emoji-list.html>

Table 1

F1 scores for positive class and the accuracy of predicting irony and stereotype profiles on validation data using BERT and SBERT models. Maximum length of tokens is 100.

Model	accuracy	f1
BERT	87	87
SBERT	94	95

Table 2

Effect of adding emojis to the vocabulary. The f1 score and accuracy of predicting irony and stereotype Profiles on validation data based on SBERT models from 5 fold cross-validation on epoch 6.

Model	accuracy	f1	std
SBERT with emojis	92.3	93.2	2.99
SBERT	91.6	91.6	3.49

6. Experiments

We used 5-fold cross validation during training for fine-tuning and to select the best performing model. The splits were fixed by setting random state values to integer values. Adding emojis to the SBERT vocabulary improved performance during development and was retained for the submitted system.

Table 2 shows the results of feeding all tweets from a user as a batch to BERT and SBERT. For this table, we truncated the maximum length of tokenized tweet to 100, due to the much higher dimension of hidden units in BERT which is 768. We limited the maximum length of tokenized tweet in SBERT to 100 as well. The result confirms that SBERT outperforms the BERT model. Table 2 shows that adding emojis slightly increases accuracy and most interestingly, reduces standard deviation.

7. Results

Table 3 shows our two competition runs, of the same model. *narcis* ran for six epochs and obtained rank 10 with an accuracy of 0.93; *harshv* ran for seven epochs and obtained rank 2 with an accuracy of 0.98.

Table 3

Competition results

submission	epoch	accuracy
narcis	6	0.93
harshv	7	0.98

7.1. Analysis

To explore the features of the most important tweets for detecting authors who use irony, we selected the top 15 tweets according to attention scores.

We counted the number of hashtags in the 15 tweets with highest attention scores across ironic and non ironic Profiles. Although all hashtags were mapped to HASHTAG, they are still good indicators for ironic profiles. There are authors in both groups who have multiple hashtags in their top ranked tweets, but ironic authors used them more frequently compared to others.

We also find that authors of ironic Profiles write more short tweets than non-ironic authors.

8. Conclusion

We focused on trying to preserve the context of an entire Profile and chose to encode Profiles as single batches of size 200 with additive attention. In experiments on the training data, this performed better than the alternative classification of individual tweets. We also saw an improvement in performance when adding all emojis to the language model vocabulary. And finally, SBERT, trained for sentence similarity, yielded better performance than BERT on the training data. The high performance of our system (run for 7 epochs: rank 2) indicates that the training data foreshadows the test data well. The performance of our system run for 6 epochs and rank 10 shows that subtle parameters can be more influential than system design.

References

- [1] K. Buschmeier, P. Cimiano, R. Klinger, An impact analysis of features in a classification approach to irony detection in product reviews, in: Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, Association for Computational Linguistics, Baltimore, Maryland, 2014.
- [2] R. Pujari, E. Oveson, P. Kulkarni, E. Nouri, Reinforcement guided multi-task learning framework for low-resource stereotype detection, in: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Dublin, Ireland, 2022.
- [3] 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: M. D. E. F. S. C. M. G. P. A. H. M. P. G. F. N. F. Alberto Barron-Cedeno, Giovanni Da San Martino (Ed.), 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.
- [4] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, 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), Association for Computational Linguistics, Minneapolis, Minnesota, 2019.

- [5] N. Reimers, I. Gurevych, Sentence-BERT: Sentence embeddings using Siamese BERT-networks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, Hong Kong, China, 2019.
- [6] O.-B. Reynier, C. Berta, R. Francisco, R. Paolo, F. Elisabetta, Profiling Irony and Stereotype Spreaders on Twitter (IROSTEREO) at PAN 2022, in: CLEF 2022 Labs and Workshops, Notebook Papers, CEUR-WS.org, 2022.
- [7] M. Potthast, T. Gollub, M. Wiegmann, B. Stein, TIRA Integrated Research Architecture, in: N. Ferro, C. Peters (Eds.), Information Retrieval Evaluation in a Changing World, The Information Retrieval Series, Springer, Berlin Heidelberg New York, 2019.
- [8] C. Özdemir, S. Bergler, A comparative study of different sentiment lexica for sentiment analysis of tweets, in: Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2015), 2015.
- [9] C. Özdemir, S. Bergler, CLaC-SentiPipe: SemEval2015 Subtasks 10 b,e, and Task 11, in: Proceedings of SemEval 2015 at NAACL/HLT, 2015.
- [10] D. I. H. Fariás, V. Patti, P. Rosso, Irony detection in twitter: The role of affective content, *ACM Trans. Internet Technol.* 16 (2016).
- [11] R. Labadie Tamayo, D. Castro Castro, R. Ortega Bueno, Deep Modeling of Latent Representations for Twitter Profiles on Hate Speech Spreaders Identification Task—Notebook for PAN at CLEF 2021, in: G. Faggioli, N. Ferro, A. Joly, M. Maistro, F. Piroi (Eds.), CLEF 2021 Labs and Workshops, Notebook Papers, CEUR-WS.org, 2021.
- [12] S. Seidman, Authorship verification using the impostors method, in: Notebook for PAN at CLEF 2013, 2013.
- [13] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, *CoRR* abs/1412.6980 (2015).
- [14] B. Felbo, A. Mislove, A. Søgaard, I. Rahwan, S. Lehmann, Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Copenhagen, Denmark, 2017.