Ingeotec at DA-VINCIS: Bag-of-Words Classifiers

Mario Graff[1,2,*], Daniela Moctezuma[3], Eric Tellez[1,2,4] and Sabino Miranda[1,2]

[1]*INFOTEC Centro Público de Investigación en Tecnologías de la Información y Comunicación, 112 Circuito Tecnopolo Sur, Parque Industrial Tecnopolo 2, Aguascalientes, 20326, México.*

[2]*CONAHCYT Consejo Nacional de Humanidades, Ciencia y Tecnología,1582 Insurgentes Sur 1582, Crédito Constructor, Ciudad de México, 03940 México*

[3]*CentroGEO Centro de Investigación en Ciencias de Información Geoespacial, Circuito Tecnopolo Norte No. 117, Col. Tecnopolo Pocitos II, C.P., Aguascalientes, Ags 20313 México*

[4]*CICESE Centro de Investigación Científica y de Educación Superior de Ensenada. Carretera Ensenada - Tijuana No. 3918, Zona Playitas, CP. 22860, Ensenada, B.C. México.*

Abstract

Violence is a serious problem that can have a devastating impact on individuals and communities. In some cases, the virtual world is prone to aggressive expressions and insults, among other violent expressions. Also, social media platforms provide a valuable source of information for detecting and monitoring violent events, as people often share posts about them in real time. This information can be used to improve responses to violence in the world and to design better crime prevention policies. In this sense, the DA-VINCIS task at IberLEF 2023 is a competition to develop multimodal models to detect violent incidents on Twitter. The task has two tracks: i) violent event identification and ii) violent event category recognition. This manuscript describes our solution system for the (i) track using only text-based features. More particularly, we used EvoMSA, our multilingual text classification framework based on stacked generalization, to develop competitive models against more complex approaches achieving an F1 score of 0.89, just 3 hundredths behind the best model in the final ranking.

Keywords

Identification of violent events, Models based on a bag of words, Pre-trained vocabularies, Stacked generalization

1. Introduction

Physical and psychological violence victims can suffer depression, anxiety, and post-traumatic stress disorder. Social media platforms provide a valuable source of information for detecting and monitoring violent events, as people often share posts about them in real-time. It is possible to create models to detect automatically violent incidents in social media for better responses in the real world and improve the design of crime prevention policies. This problem is particularly

IberLEF 2023, September 2023, Jaén, Spain

*Corresponding author.

mario.graff@infotec.mx (M. Graff); dmoctezuma@centrogeo.edu.mx (D. Moctezuma); eric.tellez@infotec.mx (E. Tellez); sabino.miranda@infotec.mx (S. Miranda)

https://mgraffg.github.io/ (M. Graff); https://www.centrogeo.org.mx/areas-profile/dmoctezuma (D. Moctezuma); https://sadit.github.io/ (E. Tellez); https://www.infotec.mx/es_mx/Infotec/sabino-miranda-jimenez (S. Miranda)

0000-0001-6573-4142 (M. Graff); 0000-0003-3038-4642 (D. Moctezuma); 0000-0001-5804-9868 (E. Tellez); 0000-0002-0595-2401 (S. Miranda)

difficult since, in some cases, even human beings disagree about what is and what is not an insult or violent message in text [1].

As an effort to attend this problem, the DA-VINCIS task [2] at IberLEF 2023 asks for models to detect violent incidents on Twitter using images and text. It asks for methods that classify tweets as mentioning a violent event. The shared task has two tracks: i) violent event identification and ii) violent event category recognition. Our solution focuses on the first sub-task using the provided text, i.e., we provide a model using natural language processing features.

Arellano et al. [3] presented the previous edition of the challenge focusing on Twitter messages with two sub-tasks: binary and multi-label classification tasks. DA-VINCIS 2022 solutions were all Transformer-based.

Nevertheless, there is more extensive literature related to violence detection on Twitter; for instance, in [4] is proposed a model based on natural language processing to identify intimate partner violence (IPV) on Twitter where achieved scores of F1-score of 0.76 on IPV class and 0.97 on the non-IPV class. This issue is not tackled in either English or Spanish languages; it has been addressed in other languages such as Arabic; in [5] sentiment analysis is used based on lexicon to classify violence on social networks. They used two social media platforms, Facebook and Twitter, to acquire messages to be classified. The results get a performance metric of F1-score between 0.75 and 0.8 values.

This manuscript describes our system solution to the DA-VINCIS 2023, which is based on our EvoMSA framework (evomsa.readthedocs.io) [6]. EvoMSA is a multilingual text classification framework built on stacked generalization [7], which effectively combines the output of multiple models into a single prediction. Our approach uses lexical and semantic features, all computed as bags of words. Compared to more complex deep learning approaches, the resulting system is competitive for violent event identification, even only using features from pure text messages.

This manuscript is organized as follows, Section 2 details our proposed system sent to the DA-VINCIS challenge. Section 3 presents our results, and some conclusions are given in Section 4. Finally, the appendix has the EvoMSA code to use our models.

2. System description

Since we focus on text messages, violent event identification can be seen as a text classification task - in fact, many of the tasks encountered at IberLEF2023 [8] can be posed as text categorization problems. Given the content of a tweet written in natural language, our model must identify whenever a violent event is mentioned.

A standard approach to tackle text classification problems is to pose it as a supervised learning problem. In supervised learning, everything starts with a dataset composed of pairs of inputs and outputs; in this case, the inputs are texts, and the outputs correspond to the associated labels or categories. The aim is that the developed algorithm can automatically assign a label to any given text independently, whether it was in the original dataset. The feasible categories are only those found on the original dataset. In some circumstances, the method can also inform the confidence it has in its prediction so the user can decide whether to use or discard it.

Following a supervised learning approach requires that the input is in amenable representation for the learning algorithm; usually, this could be a vector. One of the most common methods to

represent a text into a vector is to use a Bag of Words (BoW) model, which works by having a fixed vocabulary where each component represents an element in the vocabulary and the presence of it in the text is given by a non-zero value.

The core idea of a BoW is that after the text is normalized and tokenized, each token t is associated with a vector $\mathbf{v_t} \in \mathbb{R}^d$ where the i-th component, i.e., $\mathbf{v_t}_i$, contains the Inverse-Document-Frequency (IDF) value of the token t and $\forall_{j \neq i} \mathbf{v_t}_j = 0$. The set of vectors $\mathbf{v}$ corresponds to the vocabulary, there are d different tokens in the vocabulary, and by definition $\forall_{i \neq j} \mathbf{v_i} \cdot \mathbf{v_j} = 0$, where $\mathbf{v_i} \in \mathbb{R}^d$, $\mathbf{v_j} \in \mathbb{R}^d$, and $(\cdot)$ is the dot product. It is worth mentioning that any token outside the vocabulary is discarded.

Using this notation, a text x is represented by the sequence of its tokens, i.e., $(t_1, t_2, \ldots)$; the sequence can have repeated tokens, e.g., $t_j = t_k$. Then each token is associated with its respective vector $\mathbf{v}$ (keeping the repetitions), i.e., $(\mathbf{v_{t_1}}, \mathbf{v_{t_2}}, \ldots)$. Finally, the text x is represented as:

$$\mathbf{x} = \frac{\sum_t \mathbf{v_t}}{\|\sum_t \mathbf{v_t}\|}, \tag{1}$$

where the sum goes for all the elements of the sequence, $\mathbf{x} \in \mathbb{R}^d$, and $\|\mathbf{w}\|$ is the Euclidean norm of vector $\mathbf{w}$. The term frequency is implicitly computed in the sum because the process allows token repetitions.

The second representation developed relies on using a dense representation based on BoW. The idea is to represent a text in a vector space where the components have a more complex meaning than the BoW model. In BoW, each component's meaning corresponds to the associated token, and the IDF value gives its importance. The complex behavior comes from associating each component to the decision value of a text classifier (e.g., BoW) trained on a labeled dataset that is different from the task at hand, albeit nothing forbids to be related to it. The datasets from which these decision functions come can be built using a self-supervised approach or annotating texts.

Without loss of generality, it is assumed that there are M labeled datasets, each one contains a binary text classification problem; noting that if a dataset has K labels, then this dataset can be represented as K binary classification problems following the one versus the rest approach, i.e., it is transformed to K datasets.

For each of these M binary text classification problems, a BoW classifier is built using the default parameters (a pre-trained BoW representation and a linear Support Vector Machine (SVM) as the classifier). Consequently, there are M binary text classifiers, i.e., $(c_1, c_2, \ldots, c_M)$. Additionally, the decision function of c_i is a value where the sign indicates the class. The text representation is the vector obtained by concatenating the decision functions of the M classifiers and then normalizing the vector to have length 1.

A text x is represented with vector $\mathbf{x}' \in \mathbb{R}^M$ where the value $\mathbf{x}'_i$ corresponds to the decision function of c_i. Given that the classifier c_i is a linear SVM, the decision function corresponds to the dot product between the input vector and the weight vector $\mathbf{w}_i$ plus the bias $\mathbf{w}_{i_0}$, where the weight vector and the bias are the parameters of the classifier. That is, the value $\mathbf{x}'_i$ corresponds to

$$\mathbf{x}^{'}_{i} = \mathbf{w}_i \cdot \frac{\sum_t \mathbf{v_t}}{\|\sum_k \mathbf{v_k}\|} + \mathbf{w}_{i_0}, \tag{2}$$

where $\mathbf{v_t}$ is the IDF vector associated to the token t of the text x. In matrix notation, vector $\mathbf{x}'$ is

$$\mathbf{x}^{'} = \mathbf{W} \cdot \frac{\sum_t \mathbf{v_t}}{\|\sum_k \mathbf{v_k}\|} + \mathbf{w_0}, \tag{3}$$

where matrix $\mathbf{W} \in \mathbb{R}^{M \times d}$ contains the weights, and $\mathbf{w_0} \in \mathbb{R}^M$ is the bias. Another way to see the previous formulation is by defining a vector $\mathbf{u_t} = \frac{1}{\|\sum_k \mathbf{v_k}\|}\mathbf{W}\mathbf{v_t}$. Consequently, $\mathbf{x}'$ is defined as:

$$\mathbf{x}' = \sum_t \mathbf{u_t} + \mathbf{w_0}, \tag{4}$$

vectors $\mathbf{u} \in \mathbb{R}^M$ correspond to the tokens; this is the reason we refer to this model as a dense BoW. Finally, the vector representing the text x is the normalized $\mathbf{x}^{'}$, i.e., $\mathbf{x} = \frac{\mathbf{x}^{'}}{\|\mathbf{x}'\|}$.

2.1. BoW parameters

Different BoW representations were created and implemented following the approach described in [9]. The first step was to set all the characters to lowercase and remove diacritics and punctuation symbols. Additionally, the users and the URLs were removed from the text. Once the text is normalized, it is split into bigrams, words, and q-grams of characters with $q = \{2, 3, 4\}$.

The pre-trained BoW is estimated from 4,194,304 (2^{22}) tweets randomly selected in a larger collection of messages. The IDF values were estimated from the collections, and some tokens were selected from all the available ones found in the collection. Two procedures were used to select the tokens; the first corresponds to selecting the d tokens with the highest frequency, and the other to normalize the frequency w.r.t. their type, i.e., bigrams, words, and q-grams of characters. Once the frequency is normalized, one selects the d tokens with the highest normalized frequency. The value of d is 2^{17}; however, one can also find in the library models for $2^{13}, 2^{14}, \ldots, 2^{17}$.

It is also possible to train the BoW model using the training set; in this case, we used the default parameters. The only difference is that vocabulary size d is unlimited, containing all the tokens as found in the training set.

2.2. Dense representation parameters

The dense representations start by defining the labeled datasets used to create them. These datasets are organized in three groups. The first one is composed of human-annotated datasets; we refer to them as *dataset*. The second groups contain a set of self-supervised dataset where the objective is to predict the presence of an emoji as expected; these models are referred to as *emoji*. The final group is also a set of self-supervised datasets where the task is to predict the presence of a particular word, namely *keyword*. This final group includes a set of dense representations where the words are selected from the training set; we refer to these representations as *tailored*,

given that these were designed based on the problem at hand. The words were the most discriminative based on the BoW classifier; these were the ones with the highest absolute value between the product of the coefficient computed by the linear SVM and the IDF coefficient.

Following an equivalent approach used in the development of the pre-trained BoW, different dense representations were created; these correspond to varying the size of the vocabulary and the two procedures used to select the tokens. The vector space create by the dataset representation is $\mathbb{R}^{57}$, for the emoji models is $\mathbb{R}^{567}$, and, finally, for the keyword is $\mathbb{R}^{2048}$.

2.3. Configurations

We tested 13 different algorithms for each task. The configuration having the best performance was submitted to the contest. The best performance was computed using k-fold cross-validation ($k = 5$).

The different configurations tested in this competition are described below. These configurations include BoW and a combination of BoW with dense representations. Stack generalization combines the different text classifiers, and the top classifier was a Naive Bayes algorithm. The specific implementation of this configuration can be seen in EvoMSA's documentation, particularly in the Section Competition. The implementation of the best configuration for each problem is also described in the appendix.

bow Pre-trained BoW where the tokens are selected based on a normalized frequency w.r.t. its type, i.e., bigrams, words, and q-grams of characters.

bow_voc_selection Pre-trained BoW where the tokens correspond to the most frequent ones.

bow_training_set BoW trained with the training set; the number of tokens corresponds to all the tokens in the set.

stack_bow_keywords_emojis Stack generalization approach where the base classifiers are the BoW, the emojis, and the keywords dense BoW.

stack_bow_keywords_emojis_voc_selection Stack generalization approach where the base classifiers are the BoW, the emojis, and the keywords dense BoW. The tokens in these models were selected based on a normalized frequency w.r.t. its type, i.e., bigrams, words, and q-grams of characters.

stack_bows Stack generalization approach where the base classifiers are BoW with the two token selection procedures described previously (i.e., bow and bow_voc_selection).

stack_2_bow_keywords Stack generalization approach where with four base classifiers. These correspond to two BoW and two dense BoW (emojis and keywords), where the difference in each is the procedure used to select the tokens, i.e., the most frequent or normalized frequency.

stack_2_bow_tailored_keywords Stack generalization approach with four base classifiers. These correspond to two BoW and two dense BoW (emojis and keywords), where the difference in each is the procedure used to select the tokens, i.e., the most frequent

or normalized frequency. The second difference is that the dense representation with normalized frequency also includes models for the most discriminant words selected by a BoW classifier in the training set. We refer to these latter representations as *tailored keywords*.

stack_2_bow_all_keywords Stack generalization approach with four base classifiers equivalent to stack_2_bow_keywords where the difference is that the dense representations include the models created with the human-annotated datasets.

stack_2_bow_tailored_all_keywords Stack generalization approach with four base classifiers equivalent to stack_2_bow_all_keywords, where the difference is that the dense representation with normalized frequency also includes the tailored keywords.

stack_3_bows Stack generalization approach with three base classifiers. All of them are BoW; the first two correspond pre-trained BoW with the two token selection procedures described previously (i.e., bow and bow_voc_selection), and the latest is a BoW trained on the training set (i.e., bow_training_set).

stack_3_bows_tailored_keywords Stack generalization approach with five base classifiers. The first corresponds to a BoW trained on the training set, and the rest are used in stack_2_bow_tailored_keywords.

stack_3_bow_tailored_all_keywords Stack generalization approach with five base classifiers. It is comparable to stack_3_bows_tailored_keywords being the difference in the use of the tailored keywords.

3. Results

Table 1 presents the performance, in terms of F1, of the different configurations in the k-fold cross-validation ($k = 5$). It also includes the performance of our submission in the competition and the competition's winner. The table includes two columns *Tailored* and *Dense*; these provide an overview of the systems. The tailored column indicates whether the training set was used to define the words in the self-supervised datasets or the BoW vocabulary was obtained from it. The dense column indicates the use of a dense representation.

It can be observed that, for edition 2023, the best configuration is a stack generalization algorithm with four base classifiers; two BoW and two dense representations. One of the dense models also includes tailored keywords. The best configuration for edition 2022 is quite similar to the one for 2023; the difference is that the 2022 configuration does not use tailored keywords. The configuration used in 2023 is the second-best configuration of 2022, according to the performance obtained in the k-fold cross-validation approach.

Another comparison that one can make with the Table's data is computing the difference between the best performance in k-fold cross-validation and the worst; it can be observed that for the 2023 edition, the difference is 1.4%, and for 2022 is 13.3%. The difference of 1.4% indicates that following this approach will be complicated to improve. On the other hand, for the edition 2022, there might be room for improvement following the presented approach.

Table 1
Performance, in terms of F1, of different configurations on a k-fold cross-validation. The second block shows the performance of our submission (INGEOTEC) and the competition's winner in each edition. The best performance is in boldface.

Configuration	Tailored	Dense	DAVINCIS 2023	DAVINCIS 2022
stack_2_bow_tailored_all_keywords	X	X	**0.8984**	0.8361
stack_2_bow_all_keywords		X	0.8951	**0.8447**
stack_3_bows_tailored_keywords	X	X	0.8971	0.7555
stack_3_bow_tailored_all_keywords	X	X	0.8968	0.8219
stack_2_bow_tailored_keywords	X	X	0.8966	0.7572
stack_2_bow_keywords		X	0.8955	0.7525
stack_3_bows	X		0.8931	0.7329
bow_voc_selection			0.8907	0.7342
bow			0.8894	0.7324
bow_training_set	X		0.8892	0.7337
stack_bows			0.8879	0.7329
stack_bow_keywords_emojis		X	0.8863	0.7595
stack_bow_keywords_emojis_voc_selection		X	0.8859	0.7588
Competition				
Winner			**0.9264**	**0.7817**
INGEOTEC			0.8903	0.7510
Difference			4.1%	4.1%

The second block in the table presents the performance of our participation as INGEOTEC in the competition and the performance of the competition's winner. The last row shows the difference in percentage between these two values. As can be seen, the difference in both competitions is 4.1%, indicating that the approach presented is competitive with the winner. However, there is more viability in the 2022 competition than in 2023, as seen in the difference of performance on the different configurations tested.

4. Conclusions

We have presented our system solution and results for detecting violent incidents on Twitter using only text-based features in the context of the DA-VINCIS 2023 challenge. Our system is based on the ensemble of multiple models using stacked generalization, more precisely with our EvoMSA framework. Our approach uses lexical and semantic features, all computed as bags of words. We achieved an F1 score of 0.89, just 3 hundredths behind the best model in the final ranking.

Our results show that developing competitive models for violent event identification is possible using only text-based features and, even more, bag-of-words-based models. This is important because it means that we can use similar methodologies for languages without large language models or when it is impossible to afford the prediction costs of transformer-based solutions.

References

[1] J. Guberman, C. Schmitz, L. Hemphill, Quantifying toxicity and verbal violence on twitter, in: Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion, CSCW '16 Companion, Association for Computing Machinery, New York, NY, USA, 2016, p. 277–280. URL: https://doi.org/10.1145/2818052.2869107. doi:10.1145/2818052.2869107.

[2] H. Jarquín-Vásquez, D. I. Hernández Farías, J. Arellano, H. J. Escalante, L. Villaseñor-Pineda, M. Montes y Gómez, F. Sanchez-Vega, Overview of da-vincis at iberlef 2023: Detection of aggressive and violent incidents from social media in spanish, Procesamiento del Lenguaje Natural 71 (2023).

[3] L. J. Arellano, H. J. Escalante, L. Villaseñor Pineda, M. Montes y Gomez, F. Sanchez Vega, Overview of DA-VINCIS at IberLEF 2022: Detection of Aggressive and Violent Incidents from Social Media in Spanish, Procesamiento de Lenguaje Natural (2022) 207–215. URL: https://scholar.google.com/scholar?hl=es&as_sdt=0%2C5&q=Overview+of+DA-VINCIS+at+IberLEF+2022%3A+Detection+of+Aggressive+and+Violent+Incidents+from+Social+Media+in+Spanish&btnG=.

[4] M. A. Al-Garadi, S. Kim, Y. Guo, E. Warren, Y.-C. Yang, S. Lakamana, A. Sarker, Natural language model for automatic identification of intimate partner violence reports from twitter, Array 15 (2022) 100217.

[5] M. Khalafat, S. A. Ja'far, R. Al-Sayyed, M. Eshtay, T. Kobbaey, Violence detection over online social networks: An arabic sentiment analysis approach, iJIM 15 (2021) 91.

[6] M. Graff, S. Miranda-Jiménez, E. S. Tellez, D. Moctezuma, EvoMSA: A Multilingual Evolutionary Approach for Sentiment Analysis, Computational Intelligence Magazine 15 (2020) 76 – 88. URL: http://arxiv.org/abs/1812.02307.

[7] D. H. Wolpert, Stacked generalization, Neural Networks 5 (1992) 241–259. URL: https://www.sciencedirect.com/science/article/pii/S0893608005800231. doi:10.1016/S0893-6080(05)80023-1.

[8] S. M. Jiménez-Zafra, F. Rangel, M. Montes-y Gómez, Overview of IberLEF 2023: Natural Language Processing Challenges for Spanish and other Iberian Languages, Procesamiento del Lenguaje Natural 71 (2023).

[9] E. S. Tellez, S. Miranda-Jiménez, M. Graff, D. Moctezuma, R. R. Suárez, O. S. Siordia, A simple approach to multilingual polarity classification in Twitter, Pattern Recognition Letters 94 (2017) 68–74. doi:10.1016/j.patrec.2017.05.024.

A. Library Usage

This appendix aims to illustrate how the best configurations were implemented using EvoMSA (evomsa.readthedocs.io) [6]. The first step is to install the library, which can be done using the Anaconda package manager with the following instruction.

```
conda install -c conda-forge EvoMSA
```

A more general approach to installing EvoMSA is through the use of the command pip, as illustrated in the following instruction.

```
pip install EvoMSA
```

Once EvoMSA is installed, one must load a few libraries; the libraries used in the best configurations are the following.

```
from EvoMSA import BoW, DenseBoW
from EvoMSA import StackGeneralization as Stack
from EvoMSA.utils import Linear
from microtc.utils import tweet_iterator
```

The best configuration found for DAVINCIS 2023 corresponds to a stack generalization with four base classifiers, two BoW, and two dense BoW, where one dense BoW contains models identified in the training set. The following function implements it. It can be observed that the second and eighth lines create instances of BoW. The third line creates the instance of one of the dense BoW; the fifth line has the name of the tailored keywords, which are downloaded and loaded on line sixth line. The ninth line initializes the second dense BoW corresponding to the vocabulary with the most frequent tokens. The thirteenth line creates the stack generalization, and the last line performs the predictions.

```
def stack_2_bow_tailored_all_keywords(lang, tr, vs):
    bow = BoW(lang=lang)
    keywords = DenseBoW(lang=lang, n_jobs=-1,
                        skip_dataset=set(['davincis2022_1']))
    tailored = 'IberLEF2022_DAVINCIS_task1_Es.json.gz'
    keywords.text_representations_extend(tailored)
    keywords.select(D=tr)
    bow2 = BoW(lang=lang, voc_selection='most_common')
    keywords2 = DenseBoW(lang=lang, n_jobs=-1,
                         skip_dataset=set(['davincis2022_1']),
                         voc_selection='most_common')
    keywords2.select(D=tr)
    stack = Stack(decision_function_models=[bow, bow2, keywords,
                                            keywords2]).fit(tr)
    return stack.predict(vs)
```

DAVINCIS 2022 best configuration is almost equivalent to the 2023 best configuration. The only difference is that the former edition did not use the tailored keywords. The following function implements the 2022 configuration; the absence of the line *keywords.text_representations_extend* can be observed, which is responsible for incorporating the tailored keywords.

```
def stack_2_bow_all_keywords(lang, tr, vs):
    bow = BoW(lang=lang)
    keywords = DenseBoW(lang=lang, n_jobs=-1,
                        skip_dataset=set(['davincis2022_1']))
    keywords.select(D=tr)
    bow2 = BoW(lang=lang, voc_selection='most_common')
    keywords2 = DenseBoW(lang=lang, n_jobs=-1,
                         skip_dataset=set(['davincis2022_1']),
```

```
                    voc_selection='most_common')
keywords2.select(D=tr)
stack = Stack(decision_function_models=[bow, bow2, keywords,
                                        keywords2]).fit(tr)
return stack.predict(vs)
```