Using Vector Embeddings and Feature Vectors to Humor Identification

María Carmen Aguirre-Delgado[*,†], Angel Eduardo Cadena-Bautista[†]

Posgrado en Ciencia e Ingeniería de la Computación, Universidad Nacional Autónoma de México (UNAM), Mexico

Abstract

Expressing prejudice is the most common strategy to harm minority groups. Prejudice is defined as "the formation of a negative concept or judgment in advance about members of a race, religion, or any other significant social group, despite facts contradicting it." This paper proposes two different approaches for the classification of tweets aiming to generate humor by expressing prejudice, detecting the target group to which the prejudice is being expressed, as well as the prejudice score, ranging from 1 to 5, contained in the tweet. The approaches to solve these tasks are from two perspectives: one focusing on if there are some lexical and morphological characteristics to discriminate tweets with humor and those who don't, and the other utilizing word embeddings to tackle the same tasks.

Keywords

Prejudice, NLP, features, Embeddings, Humor, feature vectors

1. Introduction

A figurative language is a tool that enriches a language by providing it with greater expressiveness and connotation. However, its effectiveness is disrupted when this language is used with the intention of hurting, causing harm, or expressing prejudices. Prejudice refers to the preconceived negative judgment or concept about members belonging to a social minority group, despite evidence contradicting it. The types of figurative language commonly used for such purposes are sarcasm and humor. Sarcasm can be defined as saying the opposite of what one truly intends to convey, with the aim of hurting the other person, while humor seeks to provoke laughter and, in this case, avoid the moral judgment that penalizes discriminatory and prejudiced behaviors. The automatic identification of these types of language has become an emerging task due to the amount of information disseminated through the internet, which often contains prejudices disguised with sarcasm and humor. Various approaches have attempted to address this complex problem. Barbieri et al. in [1] aims to identify sarcasm in tweets by using textual, morphological, and sentiment feature vectors to find patterns, characterize, and identify sarcasm. The authors from [2] seek to use different features for the textual representation of humorous texts and detect which are the characteristics that distinguish non-offensive jokes

IberLEF 2023, September 2023, Jaén, Spain

*Corresponding author.

†These authors contributed equally.

aguirre.mcarmen@comunidad.unam.mx (M. C. Aguirre-Delgado); angelcaden@hotmail.com (A. E. Cadena-Bautista)

from highly offensive ones. We are focusing on the tasks provided by Labadie et al. in [3]. Those tasks focus on three main objectives:

- Subtask 1: Identify whether the tweet text uses humor to express prejudice.
- Subtask 2A: Identify which of the following groups the prejudice is being expressed towards:
  - Women and feminists
  - LGBTIQ community
  - Immigrants and racially discriminated individuals
  - Overweight individuals
- Subtask 2B: Determine the level of prejudice in the text on a scale from 1 to 5.

1.1. Tasks

- Subtask 1:
  HUrtful HUmour Detection: The first subtask consists of determining whether a prejudiced tweet intends to be humorous. Participants will have to distinguish between tweets that express prejudice using humor and tweets that express prejudice without using humor. Systems will be evaluated using the F1 measure for the positive class.
- Subtask 2A:
  Prejudiced Target Detection: Taking into account the analyzed minority groups, namely women and feminists, LGBTIQ community, immigrants, racially discriminated individuals, and overweight individuals, participants are asked to identify the target groups in each tweet as a multi-label classification task. The evaluation metric used for this subtask will be macro-F1.
- Subtask 2B:
  Prejudice Degree Prediction: The third subtask consists of predicting the degree of prejudice in a continuous scale from 1 to 5 for the messages targeting the minority groups. The predictions submitted will be evaluated using the Root Mean Square Error (RMSE).

1.2. Evaluation Measures

1.2.1. F1

To define the F1 measure, we need to use two widely used measures for evaluating the performance of information retrieval systems, classification, and other tasks. Precision is defined as the proportion of correct predictions among the total predictions of a class. In other words, it is the proportion of true positives out of all positive predictions.

$$\text{precision} = \frac{\text{TP}}{\text{FP} + \text{TP}} \tag{1}$$

Recall, also known as sensitivity or true positive rate, is the proportion of retrieved items out of the total relevant items. In other words, it is the proportion of true positives for a class out of all positive instances.

$$\text{recall} = \frac{\text{TP}}{\text{FN} + \text{TP}} \tag{2}$$

Thus, the F1 measure is the harmonic mean that combines precision and recall values.

$$F1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{3}$$

The F1 measure ranges between 0 and 1, where a value closer to 1 indicates better classification performance.

1.2.2. Macro-F1

The macro-averaged F1 score (or macro-F1 score) is calculated by taking the arithmetic mean (i.e., the unweighted mean) of all class-wise F1 scores. This method treats all classes equally, regardless of their support values.

1.3. Root Mean Square Error (RMSE)

To calculate RMSE, the residual (difference between the prediction and the truth) of each data point is computed, the norm of the residual for each data point is calculated, the average of the residuals is computed, and the square root of that average is obtained.

$$\text{RMSE}(y, \hat{y}) = \sqrt{\frac{\sum_{i=0}^{N-1}(y_i - \hat{y}_i)^2}{N}} \tag{4}$$

The best possible score is 0, and a lower value indicates better performance.

2. Methodology

2.1. Description of the dataset

The organizers [3] provided a dataset, this dataset was divided into two parts: the training set and the evaluation set.

The training set consists of 2671 entries with 8 columns. Figure 2 shows an example of the data:

- Index: Unique identifier for the tweet.
- Tweet: Text of the tweet in question.
- Humor: Binary, 1 if the tweet is considered humorous, 0 if not.
- Prejudice_woman: Binary, 1 if the tweet exhibits prejudice towards women, 0 if not.
- Prejudice_lgbtiq: Binary, 1 if the tweet exhibits prejudice towards the LGBTIQ community, 0 if not.
- Prejudice_inmigrant_race: Binary, 1 if the tweet exhibits prejudice towards immigrants or racialized individuals, 0 if not.
- Gordophobia: Binary, 1 if the tweet exhibits prejudice towards overweight people, 0 if not.

	index	tweet	humor	prejudice_woman	prejudice_lgbtiq	prejudice_inmigrant_race	gordofobia	mean_prejudice
0	72157	Mi celular tiene una aplicación que te hace ve...	1	0	0	0	1	3.0
1	68084	En esta vida me tocó tener mala suerte, espero...	1	0	0	0	1	2.8
2	69089	Tu mamá es taaan taan obesa, que cuando pasa f...	1	0	0	0	1	3.6
3	69190	Mi tía me dijo: \n- tengo memoria de Elefante....	1	0	0	0	1	3.4
4	70474	- Mamá, en el colegio me dicen gorda.\n- ¡Ay M...	1	0	0	0	1	3.0

Figure 1: First 4 rows of the training data set

- Mean_prejudice: Continuous value between 1 and 5, representing the degree of prejudice in the tweet.

For the test dataset the organizing committee provided a set of 778 tweets with the following form.

- index: Unique indicator for the tweet
- tweet: Text of the tweet in question

	index	tweet
0	52830	-Mamá en la escuela me dicen gorda -Pobresilla...
1	78883	No te sientas diferente, da igual si eres negr...
2	78926	Si esta asi.. SUPER SI.. y que se pongan celos...
3	61844	—Bebé ¿Me veo gorda con este vestido?\n—¡No mi...
4	78830	Las mujeres solo desean 2 cosas en la vida: co...

Figure 2: First 4 rows of the test dataset

2.2. Data preprocessing

We applied basic preprocessing techniques for feature extraction and TF-IDF weighting on the tweets, following these steps:

1. Removal of pre-existing marks on the dataset such as "URL", "HASHTAG", and "MENTION".
2. Removal of line breaks, special characters, and double spaces.
3. Removal of stop words.
4. Tokenization of words and sentences.

For the use of word embeddings, the same procedure was followed, except for the removal of stop words. It was observed that removing stop words resulted in lower performance of the classifiers.

2.3. Feature engineering

We aimed to identify features that would allow us to distinguish between humorous and non-humorous tweets. To achieve this, we performed textual, grammatical, and morphological feature extraction on the tweets to capture humor-related characteristics.

A total of 19 features were extracted from the text, as follows:

- Textual Features
  - Number of characters per document
  - Number of digits
  - Number of words per document
  - Number of characters per word
  - Number of uppercase characters per document
  - Number of special characters (-, :, !, ¡, ¿, ?)
  - Number of emoticons (:),;/, <3, etc.)
  - Number of emojis
  - Flesch-Kincaid Grade Level
- Morphological Features
  - Number of verbs
  - Number of adjectives
  - Number of nouns
  - Number of pronouns
- TF-IDF Weighting
  - Bag of Words
  - Word bigrams
  - Word trigrams
  - POS tag bigrams
- TF Weighting
  - Bag of Words
  - Word bigrams
  - POS tag bigrams
- Embeddings
  - fastText
  - Word2Vec
- Lexicons of offensive words
  - Hurtlex [4] + SHARE [5]

Feature vectors were computed for each tweet, and the difference between the humor and non-humor categories was examined.

2.4. Word Embeddings

We defined a feature vector space for training and evaluation composed of unsupervised word embedding vectors. A set of word embedding vectors represents the ideal semantic space of words in a continuous vector space of real values, where the relationships between word vectors reflect linguistic relationships between words. Word embeddings provide a dense representation of the meaning of a word, with each word linked to a continuous vector of real values with a defined dimension. Word embeddings can be generated using pre-trained embeddings such as Word2Vec [6], GloVe [7], and fastText [8].

In this study, we utilized the average of pre-trained word embeddings in Spanish. Specifically, we used a set of pre-trained word embedding vectors of 300 dimensions from fastText, trained on an unannotated Spanish corpus. These vectors were obtained using the skip-gram model. They were trained on 15 different sources, including Spanish Wikis, ParaCrawl, EUBookshop, MultiUN, OpenSubtitles, among others. The official embeddings were trained solely on Common Crawl and Wikipedia sources.

Additionally, we also experimented with pre-trained word embeddings in Spanish from Word2Vec, which are 300-dimensional embeddings generated using the skip-gram model. The training corpus used for these embeddings is an unannotated Spanish language corpus of nearly 1.5 billion words, compiled from various corpora and web resources. These embeddings were evaluated by translating the word analogy test set from the original Word2Vec [9] into Spanish. Both are available in the following repository:
https://github.com/dccuchile/spanish-word-embeddings.

2.5. Classification Models

2.5.1. Model configuration using features

For subtasks 1 and 2A using only features, three sklearn models were used with the following parameters:

- Logistic Regression: solver = "lbgs", tol $= 0.001$, $c = 0.01$, class_weight = "balanced"
- Support Vector Machines: $c = 0.01$, kernel = "linear", class_weight = "balanced"
- Random Forests: max_depth $= 10$, random_state $= 0$, class_weight = "balanced"

The configuration for subtask 2B, being a regression one, the following models were used:

- Ridge: alphas $= [1e^{-3}, 1e^{-2}, 1^{e-1}, 1]$, cv$= 10$
- Lasso: cv $= 2$
- SVM: $c = 1.0$, epsilon $= 0.2$, gamma = "scale", kernel = "linear"
- DTR: random_state $= 0$

2.5.2. Model configuration for models using word embeddings

For the use of word embeddings, the following configurations were used:

For subtask 1, the following models were tested using RandomizedSearchCV with 5 folds for parameter exploration. The next parameter options were passed to each model to select the best ones using cross-validation. Those in which no parameters are indicated were initialized with default parameters2:

- Logistic Regression (LR): C: expon(scale=100), 'penalty': ['l1', 'l2', 'elasticnet', None]
- Support Vector Machines (SVM): class_weight='balanced', kernel: [linear, poly, rbf, sigmoid, precomputed], C: expon(scale=100)
- XGBoost: n_estimators: stats.randint(150, 500), learning_rate: stats.uniform(0.01, 0.07), subsample: stats.uniform(0.3, 0.7), max_depth: $[3, 4, 5, 6, 7, 8, 9, 10]$, colsample_bytree: stats.uniform(0.5, 0.45), min_child_weight': [1, 2, 3, 4]
- Random Forest: max_depth $= 10$, random_state $= 0$, class_weight = "balanced"
- Gaussian Naive Bayes (GaussianNB)
- Bernoulli Naive Bayes (BernoulliNB)

For subtask 2A, the following models were used. Those in which no parameters are indicated were initialized with default parameters. We used cross-validation with 5 folds:

- Logistic Regression (LR)
- Support Vector Machines (SVM): $c = 1.0$, epsilon $= 0.2$, gamma = "scale", kernel = "linear"
- XGBoost: objective='binary:logistic'
- Random Forest: max_depth $= 10$, random_state $= 0$, class_weight = "balanced"

For subtask 2B, the following models were used:

- Ridge: alphas $= [1e^{-3}, 1e^{-2}, 1^{e-1}, 1]$, cv$= 5$
- Lasso: CV = 5
- XGBoost Regressor: n_estimators: 500, max_depth: 4, min_samples_split: 5, learning_rate: 0.01, loss: squared_error
- LGBM Regressor: task: train, boosting: gbdt, objective: regression, num_leaves: 10, learnnig_rage: 0.05, metric: {l2, l1 }
- Support Vector Machines (SVM): C$= 1.0$, epsilon$= 0.2$, gamma$=$ 'scale', kernel$=$ 'linear'

2.6. Feature Engineering

After extracting the feature vectors, the F1 score was calculated for each of them using three different classifiers. This calculation was performed using stratified 10-fold cross-validation.

For subtask 1, we hypothesized that if a feature demonstrated outstanding individual performance in classification, then including that feature would improve the classification

process. Consequently, we proceeded to combine the features that showed the best individual performance, calculating the F1 score for each of the obtained combinations.

In a second experiment, an exhaustive search was conducted to identify the combination that offered the best performance in terms of the F1 score in the classification.

Regarding subtasks 2A and 2B, we selected the combination of features that yielded the best results in subtask 1 and used that combination in those subtasks.

Regarding the embeddings, the following combinations were tested:

- **Subtask 1**: Pretrained Word2Vec and fastText embeddings were tested individually, along with TF-IDF vectorization, character n-grams using count vectorizer from sklearn.
- **Subtask 2A**: TF-IDF vectorization, TF-IDF + fastText vectorization, TF-IDF + fastText + character Bag of Words, TF-IDF + fastText + character Bag of Words + count of aggressive words using the SHARE lexicon [5] were tested.
- **Subtask 2B**: Same configuration as subtask 2A.

3. Evaluation

We can observe that the best results for individual feature vectors across three classifiers on the training set (evaluated through CV) are obtained using the Bag of Words approach with term frequency weighting, as shown in Table 1.

Subsequently, the combination of characteristics was made. During the process of feature combination for each classifier, it is observed that the combinations differ from each other. The combination of features was made taking into account the individual performance and doing it forward. In the case of logistic regression, the combination with better performance in the train and evaluation set includes term frequency weighting and TF-IDF weighting for words. For SVM, all available features are used, while for Random Forest, in the training set, the top 12 individual features were better, and in the evaluation set, the best combination was term frequency weighting and TF-IDF for words the results are in Table 2.

However, it is important to note that the results obtained on the evaluation set do not necessarily coincide with the best combination found in the testing phase. This means that a combination that performs well on the training set may not necessarily have the best performance on the evaluation set.

It is crucial to conduct a thorough analysis of the results on both sets to determine the optimal combination of features that maximizes the overall model performance. This highlights the importance of comprehensive evaluation and validation to ensure reliable and consistent results.

Table 1
F1 scores for subtask 1 evaluated across different classifiers: Logistic Regresion, Support Vector Machine, and Random Forest; the best results are highlighted in red, were achieved using the Bag of Words approach.

F1 subtask 1					
LR		SVM		RF	
0.71	BoW	0.67	BoW	0.73	BoW
0.71	TFIdFBoW	0.67	TFIdFBoW	0.73	TFIdFBoW
0.57	Bigram_POS	0.59	LenDoc	0.60	LenDoc
0.57	Bigram_TFIdF_POS	0.57	PalsDoc	0.56	PalsDoc
0.53	FLSKGL	0.57	Bigram_POS	0.54	NOUNCnt
0.53	PalsDoc	0.57	Bigram_TFIdF_POS	0.52	Bigram_POS
0.52	NOUNCnt	0.53	FLSKGL	0.52	Bigram_TFIdF_POS
0.52	ADJCnt	0.53	NOUNCnt	0.52	EmojisCnt
0.51	LenDoc	0.52	ADJCnt	0.51	PRONCnt
0.47	CharsPal	0.50	VERBCnt	0.51	ADJCnt
0.46	MayusDoc	0.49	PRONCnt	0.47	VERBCnt
0.45	VERBCnt	0.49	MayusDoc	0.46	CharsPal
0.42	SpecCharCnt	0.48	Bigram_BoW	0.43	MayusDoc
0.39	PRONCnt	0.46	CharsPal	0.39	FLSKGL
0.37	Bigram_TFIdF_BoW	0.40	LexRich	0.36	LexRich
0.37	LexRich	0.40	Bigram_TFIdF_BoW	0.32	SpecCharCnt
0.35	Bigram_BoW	0.38	SpecCharCnt	0.32	Bigram_BoW
0.23	DigitsCnt	0.09	DigitsCnt	0.31	Bigram_TFIdF_BoW
0.09	EmojisCnt	0.05	EmojisCnt	0.12	DigitsCnt

On the other approach taken into account to subtask 1, performance comparison of average embeddings from fastText and Word2Vec used with various classifiers. FastText shows slightly better performance compared to Word2Vec, which is why it was used in later configurations. These results are in Table 3. On the other hand, Table 4 presents a comparison of TF-IDF weighting against the use of TF-IDF by character.

For subtasks 2A and 2B, the results using word embeddings are shown in Table 5 and Table 8. We observed that the best results for subtask 2a are obtained using the combination of TF-IDF weighing, bag of words, FastText embeddings and are further improved by adding the aggressive word count. On the other hand, for substask 2b almost all classifiers show an over-fit, as their performance worsens in the test data set. However, the best results are obtained with the combination of bag-of-words, heavy TF-IDF, fastTex emmbedings, and aggressive word counts. LGBM Regressor and SVM show less overfitting compared with the rest.
For subtasks 2A and 2B, the matrix of the best feature combination from subtask 1 was used, which consisted of the top 12 best-performing features for the Random Forest algorithm. The results for subtask 2A using feature combination are shown in Table 6. And for the subtask 2B is shown in Table 7

Table 2
F1 scores for subtask 1 with feature combination. The combinations were made forward, taking into account all of the above characteristics.

F1 subtask 1 Feature Combination					
LR		SVM		RF	
	BoW		BoW		BoW
0.71	TFIdFBoW	0.67	Bigram_POS	0.74	TFIdFBoW
0.69	Bigram_POS	0.67	FLSKGL	0.72	Bigram_POS
0.66	FLSKGL	0.66	LenDoc	0.73	PalsDoc
0.66	Bigram_TFIdF_POS	0.66	PalsDoc	0.69	Bigram_TFIdF_POS
0.66	EmojisCnt	0.66	NOUNCnt	0.71	NOUNCnt
0.66	ADJCnt	0.66	EmojisCnt	0.71	LenDoc
0.66	LenDoc	0.66	ADJCnt	0.71	FLSKGL
0.66	PalsDoc	0.65	VERBCnt	0.69	EmojisCnt
0.66	CharsPal	0.65	CharsPal	0.71	PRONCnt
0.66	NOUNCnt	0.66	Bigram_TFIdF_POS	0.71	SpecCharCnt
0.65	VERBCnt	0.66	PRONCnt	0.72	VERBCnt
0.66	SpecCharCnt	0.66	SpecCharCnt	0.71	ADJCnt
0.66	LexRich	0.66	MayusDoc	0.71	MayusDoc
0.66	MayusDoc	0.66	Bigram_BoW	0.74	CharsPal
0.66	PRONCnt	0.67	DigitsCnt	0.73	LexRich
0.66	Bigram_TFIdF_BoW	0.70	TFIdFBoW	0.70	Bigram_TFIdF_BoW
0.66	Bigram_BoW	0.70	LexRich	0.69	Bigram_BoW
0.66	DigitsCnt	0.70	Bigram_TFIdF_BoW	0.69	DigitsCnt

Table 3
F1 score for subtask 1: Performance comparison embeddings from fastText and Word2Vec.

	Word2Vec		fastText	
	Train	Test	Train	Test
LR	0.77	0.59	0.76	0.6
SVM	0.99	0.61	0.99	0.64
XGBoost	0.99	0.61	0.99	0.6
RandomForest	0.9	0.51	0.87	0.48
GaussianNB	0.6	0.58	0.62	0.59
BernoulliNB	0.57	0.54	0.61	0.63
Mean	0.835	0.585	0.815	0.6

Table 4
F1 in subtask 1 evaluated on different classifiers and using TF-IDF weighing by word and the combination of TF-IDF weighing by word and character.

	TF-IDF		TF-IDF char_wb	
	Train	Test	Train	Test
LR	0.99	0.59	0.6	0.52
SVM	0.99	0.46	0.61	0.56
XGBoost	0.86	0.59	0.99	0.51
RandomForest	0.56	0.55	0.76	0.33

Table 5
Macro-F1 in subtask 2A: Combinations of TF-IDF, fastText embeddings, Bag of Words, and aggressive word count applied to different classifiers. The best results are highlighted in red.

	TF-IDF		fastText + TF-IDF		CBOW + TFIDF + fastText		CBOW + TFIDF + fastText + AgressiveWords	
	Train	Test	Train	Test	Train	Test	Train	Test
LR	0.72	0.67	0.76	0.45	0.99	0.85	0.99	0.85
SVM	1	0.85	1	0.87	1	0.88	1	0.89
XGBoost	0.93	0.84	1	0.82	1	0.82	1	0.81
RandomForest	1	0.85	1	0.33	1	0.39	1	0.41

Table 6
Macro-F1 for subtask 2A: Results for the best feature combination (which consisted of the top 12 best-performing features for the Random Forest classificator, see Table 2) now applied to subtask 2A.

TRAIN			TEST		
LR	**SVM**	**Random Forest**	**LR**	**SVM**	**Random Forest**
0.67	0.87	0.79	0.58	0.70	0.50

Table 7
RSME for subtask 2B: Performances for the best feature combination now applied to subtask 2B.

TRAIN				TEST			
Ridge	**Lasso**	**SVM**	**DTR**	**Ridge**	**Lasso**	**SVM**	**DTR**
0.68	0.59	0.71	1.14	0.94	0.71	0.99	1.42

Table 8
RSME for subtask 2B: Performances for combinations of TF-IDF, FastText emmbedings, bag-of-words and aggressive word counts from LEXICONS applied in different classifiers. In red the best results.

	TF-IDF		fastText + TF-IDF		CBOW + TFIDF + FastText + Agressive Words	
	Train	Test	Train	Test	Train	Test
Ridge	0.4	0.73	0.4	0.72	0.04	0.73
Lasso	0.64	0.75	0.67	0.74	0.6	0.76
XGBoost	0.67	0.74	0.62	0.74	0.6	0.75
LGBM Regressor	0.69	0.74	0.59	0.73	0.73	0.73
SVM	0.33	0.76	0.1	0.56	0.75	0.76

4. Conclusions

4.1. Feature Vectors

The best performance for subtask 1 is achieved with an F1-score of 0.73 for a Random Forest classifier, using a combination of Bag of Words (BoW) and TF-IDF. In subtask 2A, the best macro F1-score is 0.69 for SVM, with the combination that performs best on the training set. For subtask 2B, our best score was 0.71 using Lasso regression.

By following the proposed methodology of combining feature vectors based on their individual performance, the expected result is not obtained. In addition, features extracted from the text that we assumed would provide more information to the classifier did not yield the expected results. Furthermore, these features were able to help with classification for subtask 1 but not so much for subtasks 2A and 2B. A better approach could be to improve the use of lexicon to identify the target group of aggressive words to help classify better for subtask 2A and this would improve subtask 2B as well. The evaluation metric depends on the order in which the feature vectors are combined and thus, the best combination for one subtask does not necessarily work well for the others.

4.2. Embeddings and Feature Vectors

The best embedding for addressing the problem is fastText, although Word2Vec also performs well, in fact, quite similar to fastText.

In subtask 2A, including the aggressive word count improves the performance compared to using just embeddings and vectorization. In subtask 2A, a macro-F1 score of 0.81 is achieved using SVM with TF-IDF, CBOW, fastText, and aggressive word count.

When using embeddings and vectorizations, subtask 2B shows overfitting in most of the regressors, except for LGBM Regressor and SVM, wich achieves 0.73 and 0.73 RMSE. The best results were obteined using Ridge regresion and XGBoost regressor with 0.73 and 0.75.

One thing that all successful configurations have in common is the use of emmbeddings, lexicons, and weights, which proves to improve the performance of the classifiers when used together. Due to the structure and versatility of lexicons, their use could be improved to obtain other metrics, as they proved to be useful when used for counting aggressive words.

Future works

- Explore deep learning techniques, such as CNN and Transformers.
- Utilize embeddings from pretrained transformer models.
- Improve the use of lexicons and test them for subtasks 1 and 2A.
- Use heuristics to find the best combination of feature vectors.
- Try data augmentation techniques, as the dataset is imbalanced.

References

[1] F. Barbieri, H. Saggion, Automatic detection of irony and humour in twitter, in: International Conference on Innovative Computing and Cloud Computing, 2014.

[2] L. I. Merlo, B. Chulvi, R. Ortega, P. Rosso, When humour hurts: linguistic features to foster explainability, Procesamiento del Lenguaje Natural (2023) 85–98. doi:10.26342/2023-70-7.

[3] R. Labadie-Tamayo, B. Chulvi, P. Rosso, Everybody hurts, sometimes. overview of hurtful humour at iberlef 2023: Detection of humour spreading prejudice in twitter, in: Procesamiento del Lenguaje Natural (SEPLN), volume 71, 2023.
[4] E. Bassignana, V. Basile, V. Patti, Hurtlex: A multilingual lexicon of words to hurt, volume 2253, 2018. doi:10.4000/books.aaccademia.3085.
[5] F. M. Plaza-Del-Arco, A. B. P. Portillo, P. López-Úbeda, B. B. Gil, M. T. Martín-Valdivia, Share: A lexicon of harmful expressions by spanish speakers, 2022.
[6] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space (2013). URL: http://arxiv.org/abs/1301.3781.
[7] J. Pennington, R. Socher, C. Manning, GloVe: Global vectors for word representation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1532–1543. URL: https://aclanthology.org/D14-1162. doi:10.3115/v1/D14-1162.
[8] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information, Transactions of the Association for Computational Linguistics 5 (2017) 135–146. URL: https://aclanthology.org/Q17-1010. doi:10.1162/tacl_a_00051.
[9] C. Cardellino, Spanish Billion Words Corpus and Embeddings, 2019. URL: https://crscardellino.github.io/SBWCE/.