Machine Learning Project
Sarcasm and Irony detection on Twitter

Lucrezia Labardi
Master in Digital Humanities (Language Technologies)
University of Pisa, Italy
University of Antwerp, Belgium

1. Introduction

Sarcasm and irony are two attitudes that pervade language, especially that of social networks and in particular that used on Twitter. In this research, a dataset of approximately 90 thousand tweets, collected in the period between July and September 2015, is used, with the specific aim of being able to classify irony and sarcasm as two independent classes, a third class containing both and finally a class of regular tweets. The present study shows that the classification of regular tweets and split irony and sarcasm is a simple task that achieves more than optimal results, whereas the class containing both linguistic figures fails to be identified by either classical or deeper models. After a brief introduction to the related research in section 1.1, the experiments are illustrated (section 2, from the preprocessing phase to the results. In the end, in section 3 there is a discussion about what was or was not found, followed by the general conclusion in section 4.

1.1. Related research

Sentiment analysis is an NLP technique used to determine the emotional tone expressed in a piece of text. The detection of irony and sarcasm is a complex task, even for humans, and can cause many problems in an automatic setting. This research introduces an important novelty: it attempts to classify sarcasm and irony as independent classes, whereas in the literature they are always considered either separately or as synonyms. The difficulties of this task are posed by two factors: the first is that the poor structure and lack of context of the tweets do not allow the target of irony or sarcasm to be easily identified, and the second is that the text-based user-generated content alone is quite limiting to sarcasm and irony detection, compared to other modes of communication [1]. The task of classifying figurative language is not new and has been approached with different methodologies. One of the most interesting is the assignment of a score based on how critical the meaning of the tweet is [2]. In our case, however, only the hashtags entered by the user are used.

2. Experiments

Several experiments were implemented for this project, to compare the performance of different types of Machine Learning models and determine which one could classify tweets more accurately. Section 2.1 describes in detail the starting dataset, which steps were included in the pre-processing and which features were used for the final analysis. Section 2.2 describes the methods and the pipeline used during the analysis and explains the evaluation metrics. Section 2.3 shows the hyperparameters used for each model and both the partial and the general results.

2.1. Data

This research uses the 'Sarcasm and Irony' dataset, first used in [3], in its reduced version available on Kaggle[1]. The dataset consists of

1. Link to the Kaggle repository: https://www.kaggle.com/datasets/nikhiljohnk/tweets-with-sarcasm-and-irony

tweets belonging to four classes: figurative, sarcasm, irony and regular. The dataset was already divided into train and test sets, and the proposed division was maintained. The tweets were originally classified using the hashtags within them. The initial phase dealt with pre-processing and exploration of the data: tweets had to be cleaned and linguistic and stylistic features had to be extracted. The test set had a few missing elements and the training set had 49 duplicates. All these elements were removed. The tweets still contained URLs, tags, hashtags and other special characters (e.g. emojis). Since much of this information would have skewed the linguistic analysis, it was decided to encode it as additional features. The characteristics investigated and counted were:

- URLs: encoded as a binary feature the removed;
- Tags: counted and encoded as a feature, then removed;
- Hashtags: counted, analysed for finding patterns and finally removed;
- Special characters: escape characters were removed, special characters (e.g. &) were counted and then removed;
- Emojis: counted using a specific library [2], then removed;
- Exclamation and question marks: counted since they could have been used as emotional indicators.

After a final cleaning of the texts, which consisted of eliminating all the characters that were not numbers or letters, some linguistic features were added to the dataset:

- Number of characters in the tweet;
- Number of words in the tweet;
- Average length of words in the tweet.

The final size of the dataset is shown in table 1. As can be seen, the test set is about one-tenth of the training set. This information was useful for generating an appropriate validation set. The records belonging to each class are well-balanced. After encoding the class variable from categorical to numeric, in ascending order from figurative=3 to regular=0, the final correlation of the variables was plotted in a heatmap (see figure 1). Focusing on the

2. Link to the reference of the emojis library: https://pypi.org/project/emoji-data/

TABLE 1: Train and test set sizes divided by class

	Training set	**Test set**
Figurative	21.234	2.111
Sarcasm	20.874	2.105
Irony	20.677	2.044
Regular	18.569	1.859
Total	81.354	8.119

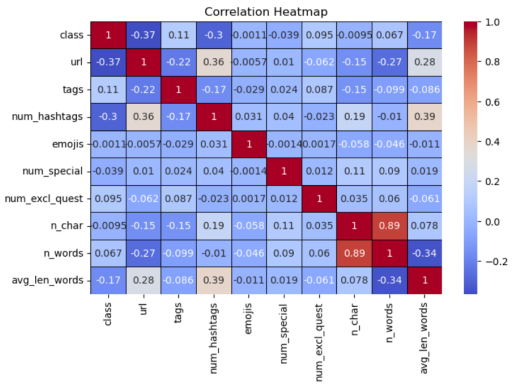

Figure 1: Correlation between the features

features correlated with 'class', it can be seen that none of them is strongly positively correlated (it means that the 'figurative' class is probably not related to any other feature). On the contrary, a lot of features are negatively correlated: this means that the 'regular' class is related mainly to the presence of URLs (-0.36), the number of hashtags (-0.29) and a bigger average length of the words in the tweet (-0.16). This can indicate that the classification of the regular class could be easier than the classification of the figurative class.

2.2. Methods and system description

The research was divided into two main parts: during the first part, three classical models, Stochastic Gradient Descent Classifier (SGDC), Decision Tree and KNN Classifier were used for the task. In the second part, it was decided to use some deeper models: two Deep Neural Networks, a Recurrent Neural Network (RNN) and a Long Short Term Memory (LSTM). As a baseline, a Dummy Classifier was used, which simulates classifying only the majority class well. As evaluation metrics, both those related to individual classes — precision, recall, and F1-score — and general

metrics related to the entire dataset — accuracy and macro-average — were chosen. Indeed, it was considered that, given that this problem is a multiclass classification, it would be relevant to assess both the performance on the entire dataset and individual classes. This approach aims to investigate the reasons behind any potential errors. Regarding the system used, since the processed dataset contained 9 numerical features and one textual feature, a decision was made to create a pipeline. This choice ensures equivalent processing before feeding the data to each model. The constructed pipeline applied MaxAbsScaler to the numerical features to achieve normalisation, and the CountVectorizer with words to the textual feature (fully cleaned text). For each model, GridSearch was applied with cross-validation in 5 folds to perform the tuning of the hyperparameters. This pipeline was not applied to RNN and LSTM, where just the textual information was used.

2.3. Results

In the first part, three classical Machine Learning models were tested. The first was an SGD Classifier, whose best parameters proved to be loss='perceptron' and penalty='elasticnet'. The results divided by class are shown in table 2. As can be seen, the figurative class can't be classified well, while the others, especially regular, are very well identified. The second model was a Decision Tree, chosen specifically for its transparency in the decision. The best parameters were criterion='entropy' and max_depth=4. The most important features, the ones that appear in the first splits, obtained by plotting the tree using just the numeric features, were urls and number of hashtags: the same that had a major correlation with the data. The results are very similar to the previous ones, but in this case, the F1-score for the figurative class is 0, which means that the records for that class are always misclassified. The third model was a KNN Classifier, whose best parameter was n_neighbors=5. The results were almost the same as the SGDC, with a very poor performance for the figurative class (F1-score is 15%) and a general decrease for the irony and sarcasm classes (F1-score is 65% and 56%).

TABLE 2: Metrics by class with SGD Classifier

SGDC	*Precision*	*Recall*	*F1*
Figurative	28%	12%	17%
Sarcasm	67%	84%	75%
Irony	66%	85%	75%
Regular	100%	99%	100%

For the second part, it was decided to try three neural networks, from linear to deep, a RNN and a LSTM. The preprocessing for the neural networks followed the pipeline explained in 2.2. Since a validation set was necessary, the 10% of the training set was used and it turned out to be approximately the same size as the test set, as expected. In this way, the test set used for all the experiments was the same. A Linear Network was first tried. The loss used for evaluation was CrossEntropyLoss, the optimizer was an SGD, the learning rate was set equal to 0.01 and the training was done for 100 epochs, with early stopping. The loss curves for training and test sets were very smooth and began to separate around epoch 30. Results are not much better than the classical models, especially considering that the figurative class is always misclassified. To obtain a more precise result, a new model was created adding a ReLU layer with 50 neurons to the previous one. Hyperparameters were the same as the linear model, and the early stopping occurred at epoch 20. Results by class are shown in table 3. The performance for all classes remains practically unchanged compared to Linear NN, and even if its precision improved by 20%, the figurative class is always misclassified. Sarcasm and irony classes reached an F1-score of 80%, better than all the classical models. As a final attempt, another layer with 50 neurons was added, for a total of two. Hyperparameters were the same and the early stopping occurred at epoch 11. The

TABLE 3: Metrics by class with Deep Neural Network

Deep NN	*Precision*	*Recall*	*F1*
Figurative	50%	0%	0%
Sarcasm	68%	100%	80%
Irony	67%	100%	80%
Regular	99%	100%	99%

results did not change. Several attempts have been made either by increasing the number of neurons in the hidden layer(s), decreasing the learning rate or increasing the number of epochs, but the figurative class is never detected while the other three are detected almost perfectly.

Since the features that were created did not give a satisfactory result, it was decided to focus on the text and its characteristics. Two more models were tested: a simple Recurrent Neural Network (RNN) and a Long Short Term Memory (LSTM). Since those models are trained with embeddings, only the cleaned text was used as data. The validation set was the same used for the Neural Networks. The text was tokenized and a reverse word index, with four more special tokens, was created. Each tweet was transformed into a sequence of numbers, corresponding to the same words of the vocabulary. As hyperparameters, CrossEntropy loss was used, with an Adam optimizer and a learning rate of 0.0001. Results by class for RNN are shown in table 4. As can be seen, the results for sarcasm, irony and regular classes are really good, while the figurative class is still misclassified. Only 2% of the records are detected correctly, which is not enough.

TABLE 4: Metrics by class with RNN

RNN	*Precision*	*Recall*	*F1*
Figurative	20%	1%	2%
Sarcasm	68%	99%	80%
Irony	67%	97%	79%
Regular	99%	100%	99%

For the LSTM, three different types of embeddings were tried: in the initial experiment, the previous word index was utilized, while in the second experiment, GloVe pre-trained embeddings with 50 dimensions were employed. In the third experiment, random embeddings were used. The hyperparameters were the same used for RNN, and unfortunately, also the results were almost the same, with the figurative class not being recognized. The results by class have been reported just once, since they all look alike, in table 5. In table 6 the accuracy and the macro average scores for all the models are reported. All the models are better than the baseline, but the first thing that appears is that the deeper models are not better than the classical ones. The model that detects better the figurative class is SGD Classifier, but it only reaches 17%, while most of the other models are stuck to 0%.

TABLE 5: Metrics by class with LSTM

LSTM	*Precision*	*Recall*	*F1*
Figurative	29%	0%	0%
Sarcasm	68%	99%	80%
Irony	67%	100%	80%
Regular	99%	100%	99%

TABLE 6: General metrics

Models	**Accuracy**	**Macro avg**
Dummy baseline	25%	10%
SGD Classifier	70%	**66%**
Decision Tree	**75%**	65%
KNN Classifier	56%	57%
Linear NN	**75%**	65%
Deep NN	**75%**	65%
Deeper NN	**75%**	65%
RNN	74%	65%
LSTM	**75%**	65%

3. Discussion

As was shown in section 2.3, of the four classes that were considered in this experiment, three (regular, sarcasm and irony) were always interpreted correctly, with F1-score percentages around 80%. In contrast, the fourth class (figurative) was not interpreted correctly by any of the proposed models. Wanting to make an error analysis, we refer to figure 2, which shows the confusion matrix obtained with SDG Classifier, which reached the highest percentage of F1-score for class figurative (17%). As can be seen, most of the records in the figurative class are classified as sarcasm or irony. This is the case for all models. It can also be deduced from the matrix that the classification of the other three classes is almost perfect and probably very easy to achieve. Regarding the reasons for the common error, two hypotheses were

Figure 2: Confusion matrix obtained with SGDC

generated: classification of the figurative class may be a difficult task itself, or the error may arise from the data collection: the tweets were classified according to the hashtag in it, which was given directly by the user. This method may be flawed, especially considering how often the terms 'sarcasm' and 'irony' are used as synonyms. As we saw from the correlation matrix, the features that turned out to be most important were not linguistically significant, an indication that it is not clear what characteristics a phrase must have to be correctly classified. Furthermore, the same experiments carried out with LSTMs were also tested on the unprocessed tweets, obtaining the same result.

4. Conclusion

This research focused on comparing different algorithms for the classification of sarcastic and/or ironic tweets. Three classical models were tested (SGDC, Decision Tree and KNN), followed by three Neural Networks, a RNN and a LSTM. It was shown that the absence of these attitudes or their independent presence is easily recognisable by the classifiers, whereas their simultaneous presence is difficult to detect. The reasons for the misclassification probably belong to the data collection system and the overconfidence in the linguistic competence of the users. Surely this task can be implemented more accurately, by trying to find new linguistic or stylistic features that are significant for a more successful classification.

References

[1] M. Sykora, S. Elayan, and T.W. Jackson, *A qualitative analysis of sarcasm, irony and related #hashtags on Twitter.*, in Big Data & Society, 7(2), (2020), https://doi.org/10.1177/2053951720972735

[2] A. Ghosh, G. Li, T. Veale, P. Rosso, E. Shutova, J. Barnden, and A. Reyes, *SemEval-2015 Task 11: Sentiment Analysis of Figurative Language in Twitter.*, in P. Nakov, T. Zesch, D. Cer, and D. Jurgens (A c. Di), Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015) (pp. 470–478), Association for Computational Linguistics, 2015, https://doi.org/10.18653/v1/S15-2080

[3] J. Ling and R. Klinger, *An Empirical, Quantitative Analysis of the Differences Between Sarcasm and Irony*, in Extended Semantic Web Conference, 2016, https://api.semanticscholar.org/CorpusID:29174571