Università di Pisa

Dipartimento di Informatica

Master Degree in Data Science and
Business Informatics

Text Analytics Project

Irony and Sarcasm Detection on Tweets

2021/2022

Autori

Andrea Carnevale (560166)
Giulia Marcoccio (624020)
Saverio Telera (626613)

Contents

1 Introduction

Philosophers and rhetoricians have been interested in irony and sarcasm for over 2500 years. In recent times, they have been popular issues discussed; in this report will be taken an Irony and Sarcasm detection on Tweets provided by an Evalita Challenge, called IronITA (Irony Detection in Italian Tweets) [Cignarella et al., 2018] (*ironita@evalita 2018* [4]).

First of all, it's important to underline the difference between the two terms in order to better understand the aim of the project.

"Irony, in its broadest sense, is a rhetorical device, literary technique, or event in which what on the surface appears to be the case or to be expected differs radically from what is actually the case." [6]

Considering the majority of state-of-the-art studies in Computational Linguistics, "irony" is as an umbrella term which includes satire, sarcasm and parody due to fuzzy boundaries between them, and especially sarcasm is defined as sharp or cutting ironic expressions with the intent to convey scorn or insult [Gibbs, 2000].

Sarcasm is a form of verbal irony that is intended to express contempt or ridicule, in other words an expression of personal dissatisfaction or smugness in humiliating others. The importance to detect irony and sarcasm is interesting per se, but also due to the possibility of reaching better predictions in sentiment analysis and for understanding what is the real opinion and orientation (positive or negative) of users about a specific subject.

In this paper our main goals are:

- TASK 1: Binary classification ironic / not ironic
- TASK 2: Multi Class Classification: not ironic / ironic but not categorized as sarcastic / sarcastic

For the development of these tasks initially we have undertaken a Data Understanding analysis trying to figure out our data, later through a Data Preprocessing we manipulated the raw text we had, in order to obtain different types of it. Finally we did some analysis like distribution of words, using a WordCloud to have a representative visualization.

Following, a classification analysis with simple classifiers has carried out (SVM, Naive Bayes and Decision Tree) both binary and multi class and we have tried to improve the classification results by integrating sentiment and emotions features. Finally, we developed LSTM and BERT to have an exhaustive overview on the classification task with models specifically suited for NLP.

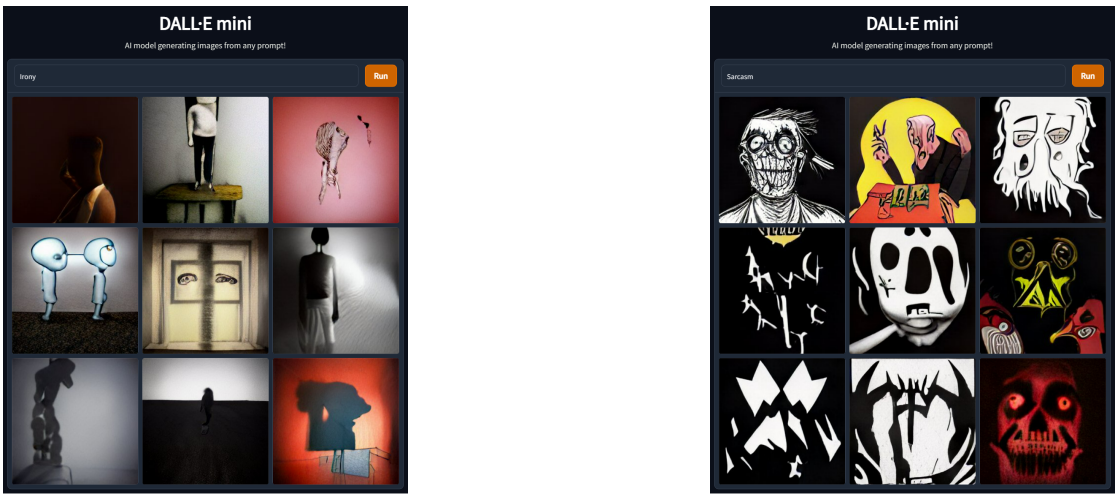

Fig. 1: Machine learning representation of an abstract concept as Irony (left) and Sarcasm (right) using DALL-E," *a model intended to be used to generate images based on text prompts for research and personal consumption*"[2]

2 Data Understanding

The dataset used during this analysis is divided in two parts: the training set composed by 3977 rows and the test set composed by 872 rows. The tables have originally 5 columns:

- **id**: the tweet id
- **text**: the corpus of each tweet
- **irony**: the class specifying if a text is ironic (1) or not (0)
- **sarcasm**: the class specifying if a text is sarcastic (1) or not (0)
- **topic**: the Tweets are extracted following 5 categories:
    1. **TW-BS**: TWitterBuonaScuola is a corpus of Italian tweets on the topic of national educational and training systems. The tweets were extracted from a specific hashtag #labuonascuola, the nickname of an education reform (translating to the good school) and a set of related keywords: "la buona scuola" (the good school), "buona scuola" (good school), "riforma scuola" (school reform), "riforma istruzione" (education reform)
    2. **TW-SPINO**: TWSpino contains tweets from Spinoza, a popular satirical Italian blog on politics
    3. **TW-SENTIPOLC**: The SENTIment POLarity data were gathered from TWITA and Senti-TUT*
    4. **HSC**: Italian Hate Speech Corpus is a corpus of hate speech on social media towards migrants and ethnic minorities, in the context of the Hate Speech Monitoring Program of the University of Turin. The tweets were collected according to a set of keywords: invadere (invade), invasione (invasion), basta (enough), fuori (out), comunist (communist), african (African), barcon (migrantsboat)
    5. **TWITA**: Random tweets scraped from the Italian streams of tweets

*(**Senti-TUT**: composed by TWNews that contains tweet retrieved by querying the Twitter search API with a series of hashtags related to Mario Monti and TWSpino)

Some examples of the different combination of the labels are reported in the box.

(Not Ironic)
Frosinone – Ora i profughi ci portano anche la tubercolosi, caso accertato a Fiuggi https://t.co/AhH2nnpTgx

(Ironic, not sarcastic)
Ghedini: "Mezza Italia mi odia". Ora prova a chiedere all'altra metà. [fedgross]

(Sarcastic)
Striscioni razzisti, il sindaco di Brembate prende le distanze: ce ne vuole uno ogni dieci metri. [misterdonnie]

2.1 Topic Distribution

First of all we wanted to observe the distribution of the topics, in order to understand the most predominant for test and train with the relative percentages.

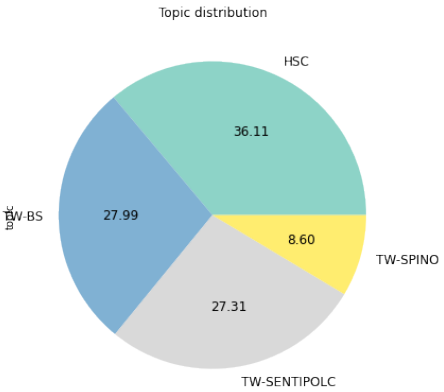

Fig. 2: Train Set

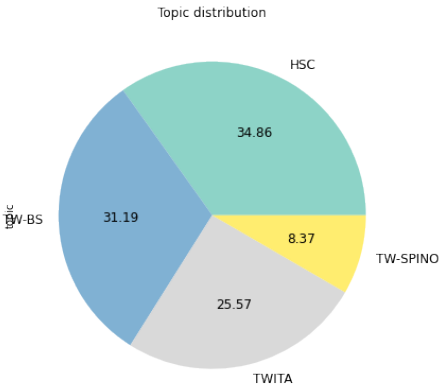

Fig. 3: Train Set

We can see that in our context **TWITA** topic is included just in the test set, while **TW-SENTIPOLC** just in the train set and in both of them there is a prevalence of TW-BS and HSC.

The distribution of Irony and Sarcasm per topic is shown on Figure 4:

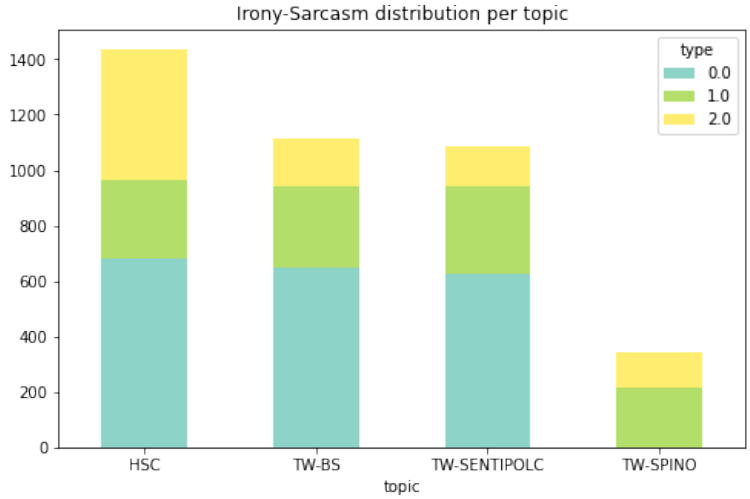

Fig. 4: Topic Distribution
0: not ironic, **1**: ironic but not categorized as sarcastic, **2**: sarcastic

As we know, sarcasm is a subcategory of irony and in this plot the yellow part is the sarcastic one for each Topic, while the green slice is the ironic and not sarcastic one. The TW-SPINO Topic is just Ironic cause the Spinoza Blog on twitter is a satirical blog, while it's possible to notice that the Hate Speech language has a remarkable prevalence of Sarcastic in the irony part, in fact sometimes aggressive comments are disguised as sarcastic.

In Figure 5 we exploit UMAP Corpus Visualization in order to discover some intrinsic properties in tweets topic. The UMAP algorithm [7] has allowed us to embed the tf-idf weighted lemmatized text in two dimensions (further information on lemmatization text can be found in section 3.1).
In this projection tweets are highlighted by topic. As expected HSC, TW-BS and TW-SENTIPOLC tweets are grouped in dense cluster being scraped with keywords referred to the same subject, while TWITA and TW-SPINO are quite sparse because they are not filtered by any keyword/hashtag, but more general, because TWITA is extracted randomly and TW-SPINO covers many topics.

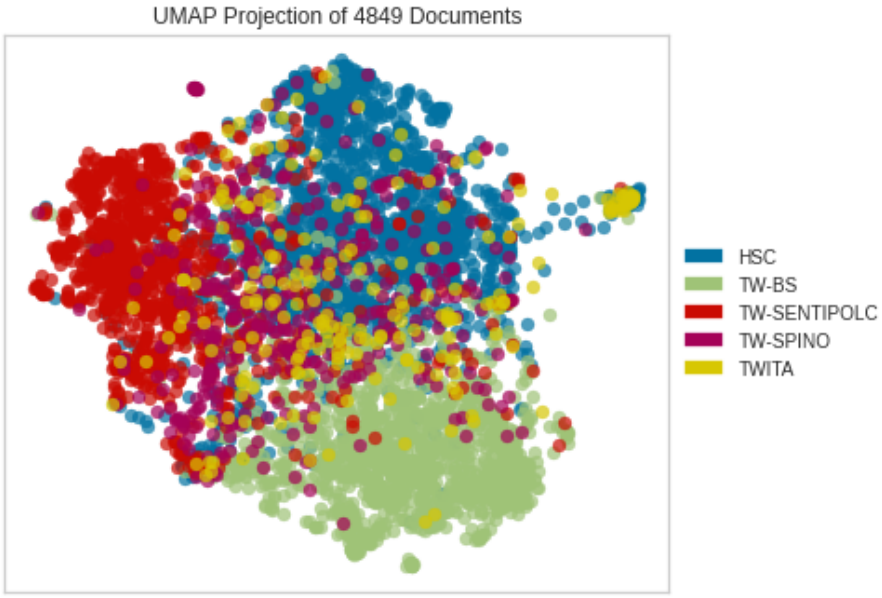

Fig. 5: Dimensional representation of topics

Then we have observed the words count distribution for Ironic, Sarcastic and Not Ironic tweets, showed in Figure 6.

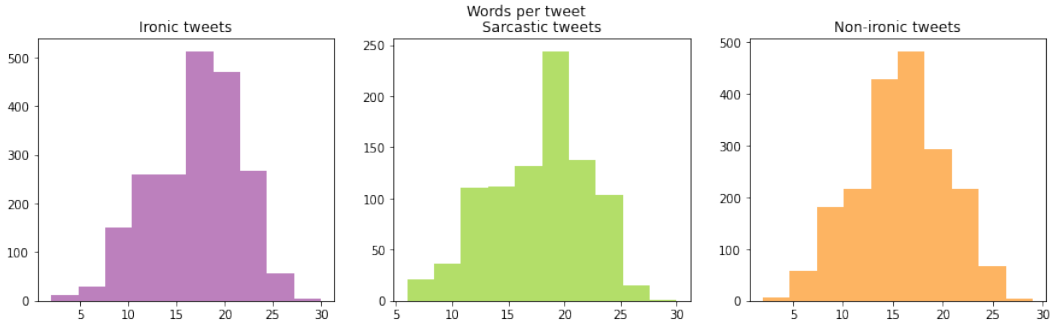

Fig. 6: Distribution of words per tweet

As we can see, Ironic tweets, with a mean of $\approx$ 17 words, tend to have more words than its counterpart, not ironic, with $\approx$ 16 . While for the Sarcastic tweets, the mean is $\approx$ 18 but we can notice that it is different from the ironic ones because we can find less tweets with an high number of words (higher kurtosis) ($\approx$ 250)

2.2 Emoji Analysis

Since emojis are usually used online to express an emotion and sometimes matches the literal meaning of the sentence, we decided to analyze which emojis may be used ironically. In Figure 7 is shown the distribution of emojis for ironical tweets.

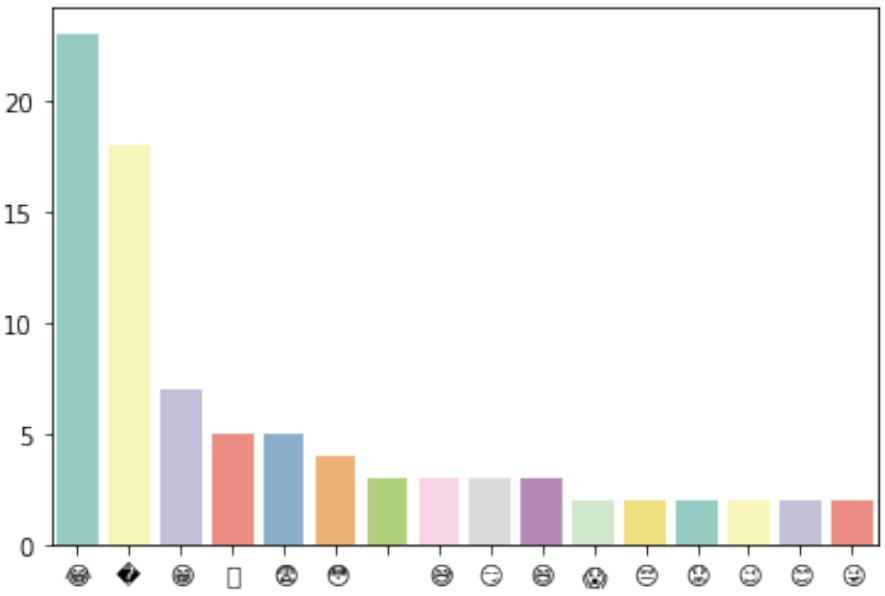

Fig. 7: Emoji distribution

Of course stands out the laughing emoticon, which is the most used, followed by others mixed emojis that could be significant or not, a limit is in the fact that some of them are not visualized or belong to the iOS emoji encoding.

Then we have analyzed the emoticon distribution between ironic and not ironic tweets. We have noticed that most of emoji are used in ironic tweets, furthermore we have seen that there are sets of emoji mostly used in ironic tweets instead of their counterpart. In conclusion we can assert that emoji are mostly used in ironic text and that there are particular emoji used for ironic and not ironic tweets.

3 Data Preprocessing

Later, a Data Preprocessing has been made in order to elaborate and analyze our tweets before applying some models.

3.1 Text processing

In this phase we have generated three different versions of the raw text that has been provided: *lemmatized_text*, *cleaned_text* and *cleaned_text_mention_hashtag*.

For all the types of text we have followed some common steps:

- removal of punctuation
- replacement of emoji with *@emoji*
- replacement of website urls with *@url*
- replacement of numbers with *@number*
- removal of the over spaces present
- removal of Italian stopwords provided by NLTK, adding to that set also some infinitive verbs that were more frequent in lemmatized sentences('essere', 'avere', 'fare', 'stare').

Later, a lemmatized version of the text ("*lemmatized text*") has been generated in order to reduce some inconsistencies especially on verbs, lemmatization takes into consideration the morphological analysis of the words and consequently several conjugated verbs have been converted to the infinitive

form. It was applied on the raw text since we have noticed that keeping the uppercase text returned results more accurate, for example taking into account the proper nouns.

For the third text version we have decided to exclude all the keywords that might generate some biases during the training phase, it is the cleaned text without mentions and hashtags, so we did the following steps in addition to the others:

- replacement of mentions with *@mention*
- replacement of hashtags with *@hashtag*

3.2 Part Of Speech Distribution

We decided to analyze the Part of Speech distribution through a POS Tagging on the 'cleaned_text' of Tweets. As it is possible to notice there is a predominance of nouns and adjectives, followed by the different types of verbs, especially the verbs are at the present tense because on Twitter it's more likely to write about contemporary events.

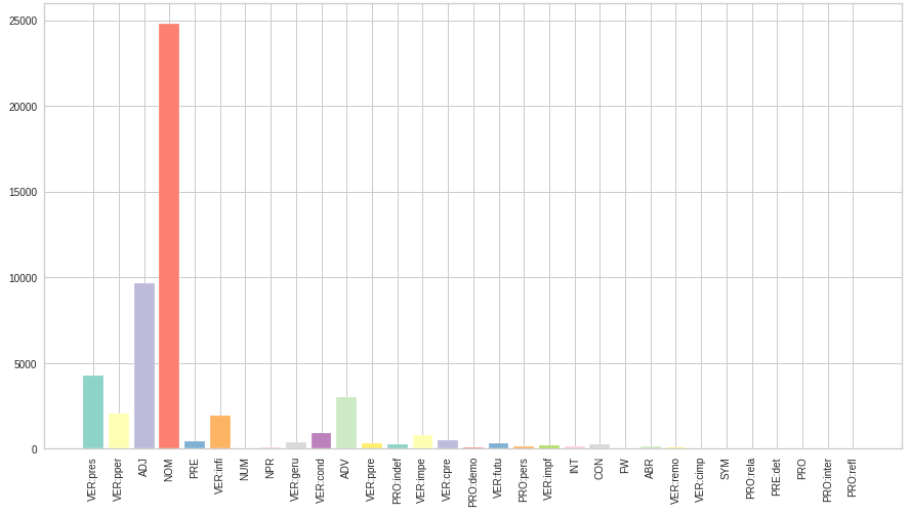

Fig. 8: Part Of Speech Distribution

Then we tried to observe how the distribution of the parts of speech changes in ironic and not ironic tweets but we haven't find any clear difference.

3.3 Words Distribution

Moving forward in the analysis, we have decided to look at words distribution. For this task we decided to use the lemmatized version of our tweets, in order to better understand how words are distributed regardless of conjugation of a verb or gender of a noun.

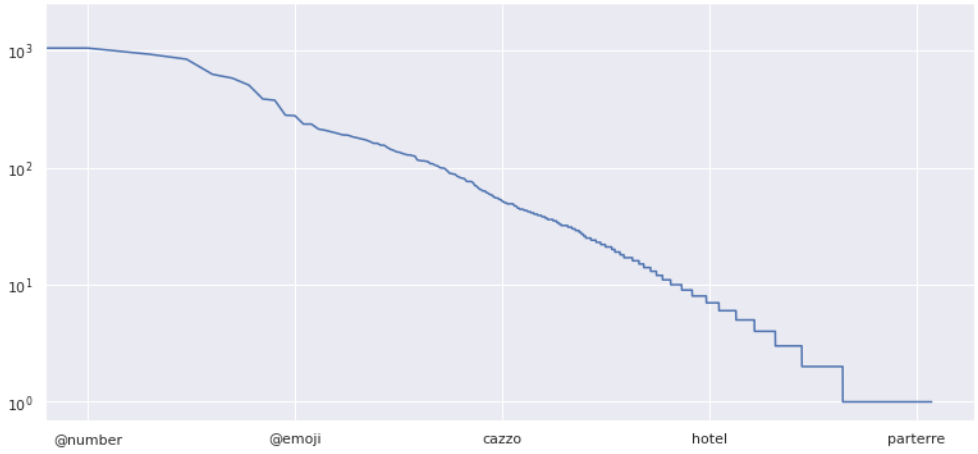

Fig. 9: Log Scale Distribution

As we can see from the log log scale distribution in Figure 9 of our words, it follows the Zipf law, where frequency of use is inversely proportional to rank by frequency.

Analysing the words distribution obtained in Figure 10 it's possible to highlight a subset of words with an high frequency:

1. *@url*, frequency: 1743
2. *@number*, frequency: 1047
3. *#labuonascuola*, frequency: 926
4. *scuola*, frequency: 841
5. *governo*, frequency: 625

As expected most frequent words are keywords used for replace some special tokens and keywords used from the authors of the dataset for the scraping purpose. Since those words are not relevant for our work, we decided to temporally remove them for the analysis distribution. However we choose to keep them for the classification, allowing the features reduction algorithm to select which is relevant. According to our assumption, *@url* will prove to be the most important one.

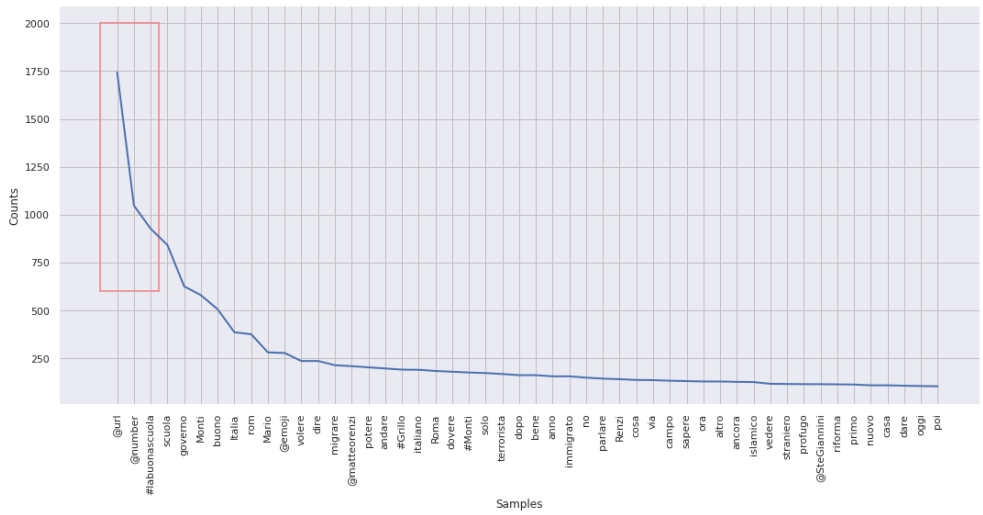

Fig. 10: Words Distribution

3.4 Word Cloud

The Word Clouds have been generated in order to graphically visualize the words divided into topics and understand the different reference contexts.

This visualization helps to have a better interpretation of the text and get the most important information, that will appear bigger than the others.

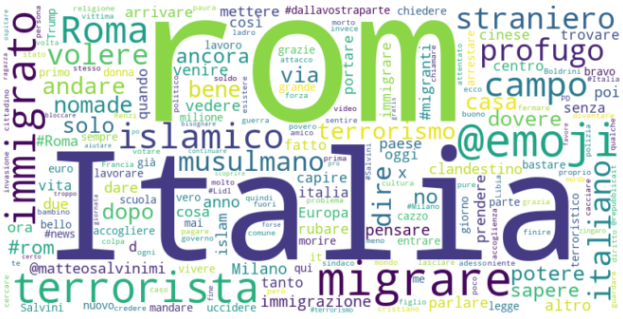

Fig. 11: Italian Hate Speech Corpus

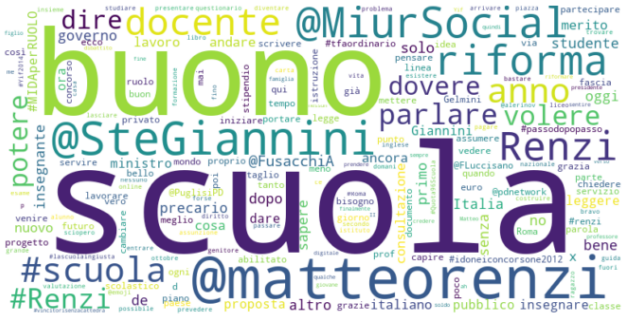

Fig. 12: TWitterBuonaScuola

Fig. 13: Random tweets from the Italian streams of tweets

Fig. 14: TWSpino contains tweets from Spinoza

Fig. 15: SENTIment POLarity data

As we can see, it is possible to delineate subjects from each set of tweets. **TWSpino** and **SENTIment POLarity** talk about Italian politics, in particular we can spot Italian politicians such as Monti, Berlusconi, etc. and some words closely related to the politics world. From **TWitterBuonaScuola** clearly emerges that the topic talks about the school reform at the time, together with the authors of the reform Stefania Giannini and Matteo Renzi. From **Italian Hate Speech** the words used for scraping are the most visible and as we could imagine are related to immigration in a discriminatory way. Finally as we expected it is impossible to delineate a clear topic in the **random tweets** from the Italian streams of tweets. An observation that can be done refers to the fact that we have the

same words in different form with minimal changes as an hashtag or upper case, i.e. the word Monti: #Monti, #monti, monte, Monti in Figure 15.

Moreover, the bigrams generated from the tweets (using lemmatized_text) have been analyzed with their frequencies and scores. Some examples are reported below.
The WordCloud showing the addition of bigrams at our original stream could be seen in Figure 16.

Bigram	Frequence	Score
buono scuola	438	50.94
governo Monti	313	42.53
Mario Monti	229	69.04
governo #Monti	116	70.71
campo rom	74	69.65

Tab. 1: Bigram Frequence and Score

Fig. 16: Bigram WordCloud

4 Simple Classifiers

4.1 Binary Classification

In this section some classification models have been tested such as Naive Bayes, Decision Tree and SVM enabling us to investigate the irony of a tweet. Data have been preprocessed with the following pipeline:

- **CountVectorizer**, through which senteces are tokenized using TweetTokenizer library (able to preserve mention and hashtag) and extended with bi and tri-gram.
- **SelectKBest** (score function *chi2*), narrow it down at 200 features, except for the decision tree, where we used an higher value (500), because its implementation already include its own feature selection
- **TfidfTranformer**, in order to give a weight to the tokens

For each classifier the three types of cleaned text have been submitted in order to better understand which is the best one for the different models.

The algorithm GridSearchCV has been useful to grasp the best parameters for the different models. In Table 2 it's possible to observe the **F1-score** results given by the algorithms applied. This measure is chosen in order to have a benchmark on the results obtained by our model versus the other challenge participants.

Text	**Model**	**F1-score**					
		TR			TS		
		not-ironic	*ironic*	*avg*	*not-ironic*	*ironic*	*avg*
cleaned_text	SVM	0.73	0.78	0.75	0.62	0.71	0.67
	Naive Bayes	0.77	0.76	0.76	0.62	0.64	0.63
	Decision Tree	0.73	0.76	0.75	0.62	0.70	0.66
lemmatized_text	SVM	0.73	0.78	0.76	0.62	0.71	0.67
	Naive Bayes	0.75	0.75	0.75	0.60	0.67	0.63
	Decision Tree	0.72	0.76	0.74	0.61	0.70	0.66
cleaned_text_ mention_hashtag	SVM	0.72	0.74	0.73	0.65	0.68	0.67
	Naive Bayes	0.73	0.73	0.73	0.63	0.65	0.64
	Decision Tree	0.67	0.74	0.71	0.64	0.70	0.67

Tab. 2: Binary Classification Results

As we can notice best results are returned by *cleaned_text_mention_hashtag*, in particular the highest F1 score macro avg is reached by the Decision Tree. We can assume that these higher results, obtained removing hashtags and mentions, are due to the fact that the models can improve its generalization on the test set. On the flip side of the coin, we can also point out that looking at the ironic F1 column for the others processed type of text (cleaned_text lemmatized_text) ironic results are higher.

The worst model is the Naive Bayes, these low results may be explained by the strong (*naive*) independence assumptions between the features on which the model is built, in particular dealing with text where the words usage is strictly correlated.

4.2 Multi class Classification

As we had a double target value we also applied a Multi class classification with the same algorithms used for the binary one (SVM, Naive Bayes and Decision Tree). But in this case we used two types of Naive Bayes: Multinomial and Complement, an adaptation of the first one particularly suited for imbalanced case like the Sarcasm label, which employs complement classes probability. We have summed up the columns values of 'irony' and 'sarcasm' obtaining in this way three labels:

- 0 for a not ironic tweet
- 1 for an ironic tweet
- 2 for a sarcastic tweet (because a sarcastic tweet is both ironic and sarcastic)

Also in this case we have used the GridSearchCV algorithm in order to obtain the best F1-score values from our data, showed in the following Table

Text	Model	F1-score							
		TR				TS			
		not-ironic	*ironic*	sarcastic	*avg*	*not-ironic*	*ironic*	sarcastic	*avg*
cleaned_text	SVM	0.78	0.57	0.61	0.66	0.66	0.36	0.32	0.45
	MNB	0.74	0.43	0.52	0.56	0.66	0.26	0.32	0.41
	CNB	0.72	0.43	0.54	0.56	0.64	0.25	0.38	0.42
	DT	0.75	0.49	0.49	0.58	0.64	0.38	0.31	0.44
lemmatized_text	SVM	0.80	0.61	0.63	0.68	0.65	0.41	0.28	0.45
	MNB	0.74	0.43	0.51	0.56	0.65	0.26	0.29	0.39
	CNB	0.71	0.44	0.53	0.56	0.62	0.20	0.37	0.40
	DT	0.74	0.50	0.39	0.54	0.65	0.39	0.22	0.42
cleaned_text_ mention_hashtag	SVM	0.79	0.59	0.60	0.66	0.68	0.40	0.31	0.47
	MNB	0.73	0.41	0.49	0.54	0.65	0.25	0.29	0.40
	CNB	0.69	0.41	0.51	0.54	0.65	0.25	0.39	0.43
	DT	0.75	0.51	0.42	0.56	0.65	0.38	0.28	0.44

Tab. 3: Multi class Classification Results

As expected we have overall lower results, this is due to the multiple values that target variable can assume. Furthermore it is interesting to observe how the general result of F1-score on not ironic label are higher than previous binary task. We can assume that inferring not ironic tweets is identical to the previous task. Instead the ironic label can be further split into two different scenarios: sarcastic and not sarcastic, making the imputation more complex.

The best results are reached again by the *cleaned_text_mention_hashtag*, for the same previous reasons, but in this case, instead, SVM perform better than Decision Tree.

5 Irony, Sarcasm and Sentiment Analysis

Since irony and sarcasm are two attitudes related to the feelings of a person, and in our case, for example, a person could decide to write a tweet according to the sentiment he/she is perceiving, we decided to undertake a sentiment analysis on Tweets.

In such circumstance, we have used the **FEEL-IT Python Package**[5], an UmBERTo fine-tuning for predicting emotion and sentiment in Italian language.

This model allows to find two new attributes:

- **Sentiment features**: [*positive*, *negative*]
- **Emotion features**: four basic emotions [*anger*, *fear*, *joy*, *sadness*]

5.1 Binary Classification

Once retrieved for each tweet in our dataset these two attributes we decided to expand the classification task with a new model, adding these features in the training process.

We took the simple classifier model that reached the highest *F1 score* in the binary classification, the Decision Tree, and extended it with our new features. We kept the same pipeline applied in the previous task, and executed a new grid search in order to tune hyper-parameters. From the results obtained, we noticed that the scores are similar to the previous ones.

To better analyze the results, we plotted the features importance of the decision tree in order to see if the tree uses the new added features.

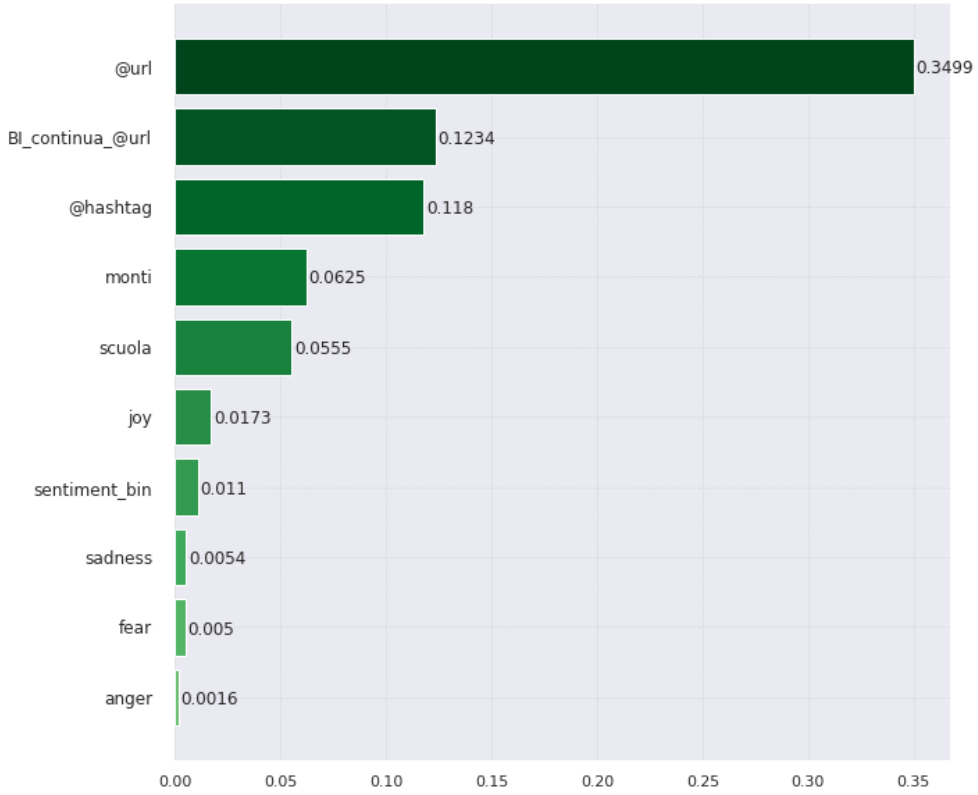

Fig. 17: Binary decision tree feature importance

In the plot above is displayed the feature importance of the 5 most relevant features (*@url, BI_continua_@url, @hashtag, monti, scuola*) and the 5 new features (*sentiment_bin, joy, sadness, fear, anger*).

The most important features are "**@url**" and "**BI_continua_@url**", this high value can be explained since many tweets talking about news contain the link to the full newspaper article, references to some information website or YouTube videos, so these two features help the model to discern news tweets that are obviously not ironical. Our model allocate low feature importance value to the sentiment and emotions features, not enough to change the previous results.

5.2 Multi class classification

For the sake of completeness, we decided to analyze the influence of our new features on the multi class classification. We kept the same model, the Decision Tree, and applied it on training and test expanded with sentiment and emotions features. As in the previous task, results haven't changed much and we decided to observe the features importance of our decision tree.

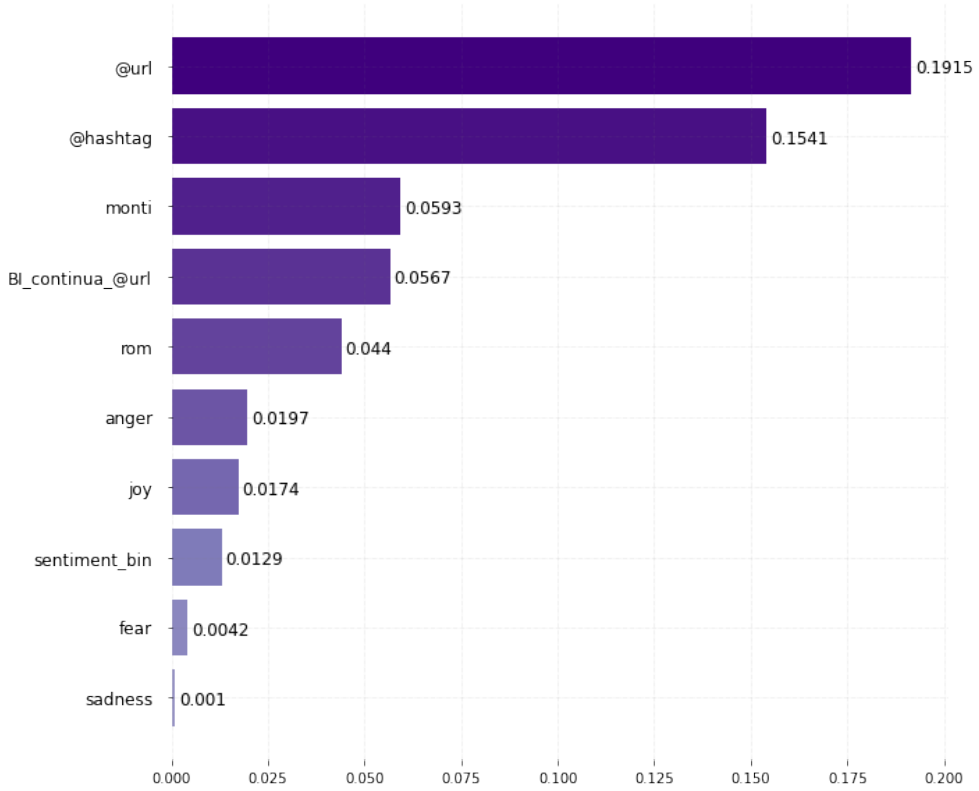

Fig. 18: Multi class decision tree feature importance

In the plot, as previously, we took the top 5 most important features (*url, @hashtag, monti, BI_continua_url, rom*) and our 5 sentiment and emotion features (*sentiment_bin, joy, sadness, fear, anger*).

We can see that feature importance of the added features is increased, in particular we can notice that the feature that mostly changed its importance is **anger**. It was in the last position, among the added features, during the binary classification and in the first in the multi class classification. Seems like that Angry expressions helps our classifier to improve a little bit the performances when we introduce the sarcastic / notsarcastic label. It might be explained starting from the formal definition of sarcasm: "*Bitter or caustic irony, an expression of personal dissatisfaction or smugness in humiliating others*", angry sentiment can be involved to express sarcastic sentences, especially when coming out from a situation of self-dissatisfaction.

In addition, it is curios to observe how these results are related in some way to the images generated by DALL-E (Figure 1) where the pictures yielded by the machine learning model can be seen as a visual representations of anger.

6 LSTM

LSTM is a special kind of RNN, therefore works with feedback loops, and capable of learning long-term dependencies. Before applying the model to our *'cleaned_text'*, that has be chosen because we want to keep conjugation of verbs and allow the model to learn tweets meaning, some steps have been performed:

- Encoding the sentences in a list of integers, each mapping a word.
- Setting the maximum dictionary length to 5000 word
- Padding the resulting lists to match size 200

Regarding the model development the Keras library has been used, which allowed us to build the NN architecture stacking the following layers:

- *Embedding layer*, taking as input dimension the vocabulary size ('input_dim' = 5000), while output dimension have to be tuned ('output_dim').
- *LSTM layers*, using cuDNN requirements for the parameters' choice for speed up the learning phase.
- *Dense layer*, made up of one unit with sigmoid activation function for the binary classification. Three units with softmax activation function for the multi class task.

The NN weights are taken from a random normal distribution with mean 0 and standard deviation 0.2, moreover they are initialized with a seed in order to guarantee reproducibility. Both the LSTM and Dense layers are regularized with a Ridge penalty term.
The training phase employs Adam optimizer with adaptive learning rate of which we have to tune the initial value and the beta1/beta2, trying to minimize crossentropy loss (binary/categorical based on the task).

We implemented a grid search with 4-Fold CrossValidation to find a good hyperparameters setting. The parameters grid is showed in the Table 4, with a wide range arising from multiple initial trials. We also noticed that using 2 layers enables us to achieve better generalization performances instead of using 1 or more than 2 layers, but it has not been possible to ran an exhaustive grid for each of these configurations due to a limited GPU usage time.

Param	**Range**	**BinaryLSTM**	**BinaryBiLSTM**	**MultiLSTM**	**MultiBiLSTM**
units	[10, 20, 50]	50	20	50	20
output_dim	[20, 50, 100]	100	100	50	20
eta	[0.01, 0.001, 0.0005]	0.01	0.01	0.01	0.01
lambd	[0.1, 0.01, 0.001]	0.001	0.001	0.001	0.001
beta_1	[0.9, 0.8, 0.6]	0.9	0.9	0.8	0.8
beta_2	[0.999]	0.999	0.999	0.999	0.999
weight_init	[0.2]	0.2	0.2	0.2	0.2

Tab. 4: LSTM Parameters

We trained our NN for 300 epochs using Early Stopping approach (with 50 epochs of patience), monitoring the validation loss in order to avoid overfitting and speed up the search. We stored the mean training loss (over the 4 fold) of each hyper parameters configuration, so we could use it when we retrained the model on the whole training set to halt its learning.

Furthermore bidirectional version of the model has been developed which takes into account both the left and right context and, as it is possible to observe in Table 5, we have obtained a gain in performance for the multi class task with respect to unidirectional one.

	TR			TS		
	not-ironic	ironic	avg	not ironic	ironic	avg
LSTM binary	0.89	0.89	0.89	0.59	0.63	0.61
BiLSTM binary	0.88	0.90	0.89	0.52	0.67	0.60

Tab. 5: Binary LSTM Results

	TR				TS			
	not-ironic	ironic	sarcastic	avg	ironic	not ironic	sarcastic	avg
LSTM multiclass	0.99	0.89	0.89	0.92	0.55	0.30	0.37	0.41
BiLSTM multiclass	0.88	0.83	0.79	0.83	0.67	0.29	0.33	0.43

Tab. 6: Multi class LSTM Results

7 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a computational model developed in 2018 by researchers at Google, in particular it is an Encoder stack of transformer architecture.

We have found on Hugging Face[1] the *bert-base-italian-cased* model that best fits to our Italian data, developed from dbmdz (*MDZ Digital Library team*). [3]

BERT is applied on the raw text, since its formulation enable to understand sentences as they are. The following steps have been performed:

1. Tokenization with AutoTokenizer using bert-base-italian-cased and transforming in lower case the words
2. Insertion of the special token [CLS] at the beginning of each tweet
3. Insertion of [SEP] at the end of each tweet in order to separate them
4. Setting of Maximum Train and Test Len (128 and 512) because BERT has a maximum number of tokens we can pass
5. Encoding the sentences in list of integers each mapping a word
6. Creation of Attention Masks, that indicate to the model which tokens should be attended to, and which should not.
7. Creation of a validation set
8. Transformation of the Dataset in Tensor in order to use the GPU
9. Initialization of a DataLoader to iterate over batches during the training phase.

After the model building, the parameters have been tested in order to find the best combination. In Table 7 it is possible to take a view of the best parameters both for Binary and Multi class and the results obtained training the models on our data.

Param	**Range**	**BinaryBERT**	**MulticlassBERT**
epochs	[2, 3, 4]	2	3
weight_decay	[0.1, 0.01, 0.001]	0.1	0.1
lr	[2e-5, 3e-5, 4e-5]	2e-5	4e-5
F1-score	Train	0.7801	0.6529
	Test	0.7106	0.5151

Tab. 7: BERT Parameters and Results

7.1 BERT Evaluation and Conclusions

The F1-score results obtained reach a better score with respect to the Simple Classifiers and LSTM applied previously.
This output is given by the fact that BERT is a more powerful model, and it is able to take into account word's context, unlike simple Classifiers, thanks to the Attention mechanism. LSTM is able to understand sentences meaning too by using cell state, but since BERT is pre-trained on an huge number of data, is able to build a proper language model and in this way outperform LSTM in this tasks.

References

[1] *Bert Model*. URL: https://huggingface.co/dbmdz/bert-base-italian-cased.

[2] *DALL-E model*. URL: https://huggingface.co/dalle-mini/dalle-mini.

[3] *dbmdz BERT models*. URL: https://github.com/dbmdz/berts.

[4] *Evalita*. URL: http://www.di.unito.it/~tutreeb/ironita-evalita18/index.html.

[5] *FEEL-IT: Emotion and Sentiment Classification for the Italian Language*. URL: https://huggingface.co/MilaNLProc/feel-it-italian-sentiment.

[6] *Irony*. URL: https://en.wikipedia.org/wiki/Irony.

[7] Leland McInnes, John Healy, and James Melville. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. 2018. DOI: 10.48550/ARXIV.1802.03426. URL: https://arxiv.org/abs/1802.03426.