IRONITA - Irony and Sarcasm detection in Italian Tweets

Gerardo Zinno [1] , Matteo Massetti [2]

[1] Università di Pisa (Artificial Intelligence); m.massetti2@studenti.unipi.it
[2] Università di Pisa (Artificial Intelligence); g.zinno@studenti.unipi.it

1. Introduction

The challenge we wanted to face is the challenge belonging to the 2018 edition of EvalTA called IronITA[1]. The challenge focuses on the role that irony plays in the analysis of sentiment in a text, in particular in tweets. Indeed, irony is a figurative language device that conveys the opposite of literal meaning profiling intentionally a secondary or extended meaning. Users on the Web usually tend to use irony like a good creative device to express their thoughts in context of short-texts like tweets, reviews, posts or commentaries. But irony as well as other figurative language devices (for example metaphor) is very difficult to deal with and precisely for its traits of recalling an other meaning or obfuscating the real meaning, it hinders correct sentiment analysis of texts and, therefore, correct opinion mining. Indeed, the presence of ironic devices in a text can work as an unexpected "polarity reverser" (one says something "good" to mean something "bad"), thus undermining systems' accuracy. Another factor that makes it difficult to understand irony is that it does not depend only on the sentence itself but also on the context in which it is placed. The term "irony" is an umbrella term which includes satire, sarcasm and parody due to fuzzy boundaries, so the objective of the challenge is not only to find out if the text contains irony, but also what kind of irony is present.

1.1. Tasks

The challenge comprehends two different tasks:

- Task A: two class classification task where the system has to predict whether a tweet is ironic or not.
- Task B: multi-class classification task where the system has to predict one out of three labels describing i) irony, ii) irony which can be categorized as sarcasm, iii) non irony.

We will face both of them.
The evaluations of these tasks are done using as a metric the F1 score computed as follows:

$$F1_A = Avg(F1_{ironic}, F1_{not-ironic})$$

$$F1_B = Avg(F1_{ironic-not-sarcastic}, F1_{sarcastic}, F1_{not-ironic})$$

2. Dataset

The dataset provided by the challenge authors includes both a training set and a test set, both labeled[1]. The dataset comprehends 4849 records of which 3977 belong to the training set and the other 872 to the test set.
Each record has the following structure:

- `id`: unique id of the record
- `text`: body of the tweet
- `irony`: binary attribute binary attribute that indicates the presence of irony
- `sarcasm`: binary attribute binary attribute that indicates the presence of sarcasm
- `topic`: indicates the collection from which the tweet comes

The 3977 tweets in the training set there are distributed into 1954 not ironic tweets, 1110 ironic but not sarcastic tweets and 913 sarcastic tweets. While in the test set there are 437 not ironic tweets, 216 sarcastic and 219 ironic but not sarcastic (figure 1).
The training set will be used for training and validation, while the test set will be used for final model evaluation and comparisons with other participants of the challenge.

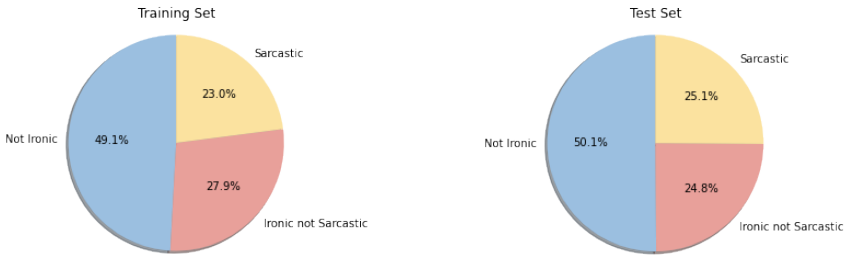


Figure 1. Class distribution in the Dataset

3. Preprocessing

In order to transform raw text in a format usable in a machine learning context, we have created a pipeline, represented in figure 2, through which each tweet passes through and is transformed.
The parts of this pipeline derive from empirical tests and from our decision, in particular they are:

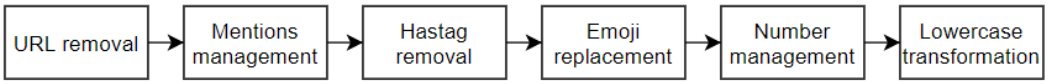


Figure 2. Preprocessing pipeline

- URL removal: delete urls using regular expressions.
- Mentions managements: all the twitter mentions are substituted with a particular token <*men*>
- Hashtag removal: discharge hashtags.
- Emoji replacement: use of the library *emoji* to replace all the detected emoji with the respective meaning in text form.
- Number management: replace all the numbers with a particular token <*num*>.
- Lowercase transformation: cast all the text to lowercase. This step is skipped by some models already pre-trained on cased datasets.

After this first preprocessing, the text in this form is not yet usable by machine learning models. It is indeed necessary to split each tweet into tokens and transform each one of these into a format suitable to be used by our models. Since we will use two different kind of models, we use two different methods.

The first method is Tokenization with the library *nltk* and representation of the words by a 128 elements vector using the word embedding provided by [2]. The special tokens introduced by the pipeline (<*men*> and <*num*>) are mapped to their respective meanings (*menzione* and *numero*), while all words that cannot be mapped to a vector (such as misspelled words) are mapped to a vector of zeroes.

The latter method instead is Tokenization and word representation using an implementation of BERT trained on a large collection of Italian texts [3].

BERT [4] (Bidirectional Encoder Representations from Transformers) is a transformer-based machine learning technique for natural language processing pretraining developed by Google and whose architecture is almost identical to the original Transformer implementation described in [5].

BERT is pretrained on two tasks: *language modelling* (where a percentage of tokens is masked and BERT is trained to predict them from context) and *next sentence prediction* (where BERT is trained to predict if a chosen next sentence is probable or not given the first sentence). As a result of the training process, BERT learns contextual embeddings for words. After pretraining, which is computationally expensive, BERT can be finetuned with less resources on smaller datasets to optimize its performance on specific tasks (Figure 3).

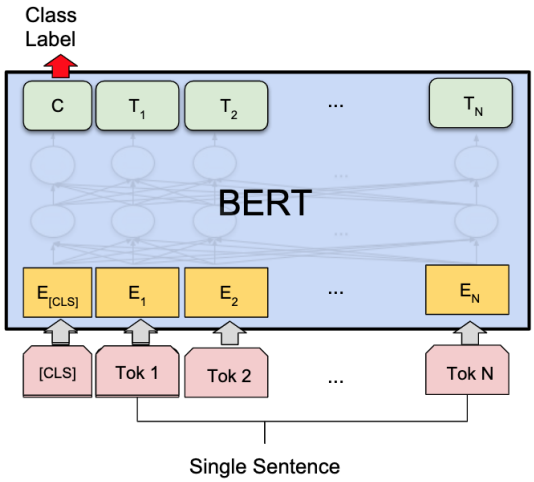


Figure 3. Single sentence classification task

As already mentioned the BERT model we used was pre-trained one, it has been trained on a recent dump of the Italian Wikipedia and various texts from the OPUS corpora collection [6].

In both methods we have decided to discard punctuation and stop words, since we believe they do not enrich the meaning of a tweet. We retrieved the set of Italian stop words using *nltk.corpus* and the punctuation using the *string.punctuation*.

4. Models

We decided to test and compare different approaches to both tasks, in particular we explored with the following:

- **Long Short Term Memory** [7]: is an artificial Recurrent Neural Network architecture. It's core concepts are the cell state and various gates (left image in figure 4). The state of the cell transports relative information all the way down to the sequence chain (the memory of the network), the gates instead are responsible to choose which information is allowed on the cell state and which has to be forgotten. Going into detail of these gates we find: a **Forget Gate**, that decides what information should be kept from previous hidden state and from the current input, an **Input Gate** that is used to update the cell stat and finally an **Output Gate**, which decides what the next hidden state should be.
- **Gated Recurrent Units** [7]: they are a gating mechanism in Recurrent Neural Networks. The GRU is like a long short-term memory (LSTM) with a forget gate but it has fewer parameters than LSTM, as it lacks an output gate. This architecture (right image in figure 4) is formed by two main gates: a **Reset Gate** which is responsible for the hidden state of the cell and an **Update**

Gate, which tells how much of the past information must be kept and passed forward. GRU's performance on certain tasks, including natural language processing, was found to be similar to that of LSTM. Furthermore GRUs have been shown to exhibit better performance on certain smaller and less frequent

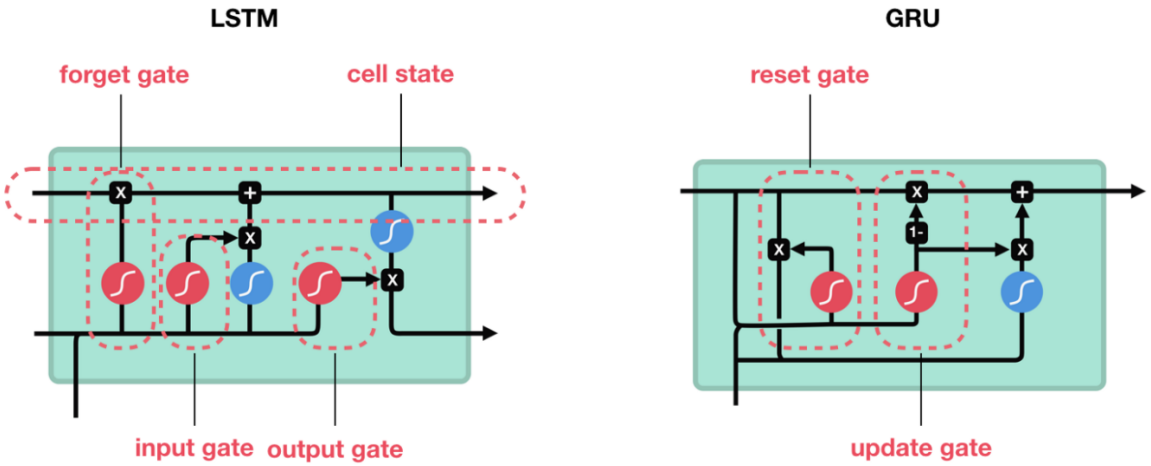


Figure 4. LSTM and GRU architectures

- **Convolution Networks** [8]: particular implementation of classical Neural Networks that uses one or more **filters**, also of different size, and the convolution operation instead of the classical Weights Matrix. This kind of networks are particularly suited for images, but also for other structured data like sequence and text.

4.1. Gridsearch

For each model we performed an exhaustive grid search on different hyperparameters of the models, such as number of layers, number of neurons, number of epochs and other parameters specific to different type of layers used, which are *GRU* (table 1), *LSTM* (table 2) and *1DConvolution* (table 3).

Hyperparameter	Values
Number of Hidden Layers	1, 2, 3
Number of GRU cells	16, 32, 64, 128
Dropout Hidden Layer	0.0, 0.2, 0.5
Epochs	10, 20

Table 1. Grid search for GRU

Hyperparameter	Values
Number of LSTM cells	8, 16, 32, 64
Neuron for Dense Hidden Layer	8, 16, 32, 64
Dropout Hidden Layer	0.25, 0.45
Epochs	5, 10

Table 2. Grid search for LSTM

Hyperparameter	Values
Number of Filters	32, 64
Kernel Size	3, 5
Neuron for Dense Hidden Layer	16, 24, 32
First Level Dropout	0.2, 0.5
Dropout Hidden Layer	0.0, 0.1
Epochs	10, 20

Table 3. Grid search for 1DConvolution

4.2. Model Evaluations

To chose the best configuration of each model and the best one among them, we reserved around 10% of training set for evaluation.
The choice of the former was based on the level of accuracy that each configuration was able to achieve

in the validation set, while the choice of the final model was driven by the best achieved value of $F1$ scores presented in section 1.1.

5. BERT-GRU model

The model that gives the best performance is the one built on BERT that uses GRU layers, its structure is shown in figure 5, and in particular it is formed by three **GRU** layers, respectively of 128, 64 and 16 units, two **Dropout** layers, 0.5 and 0.2, and a final **Dense** layers which consists in two output neurons.

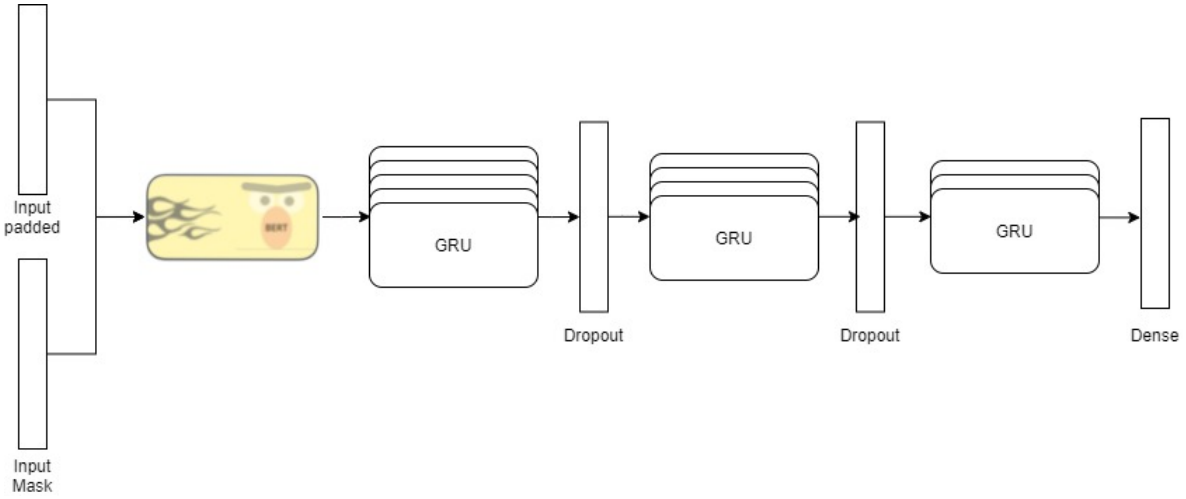
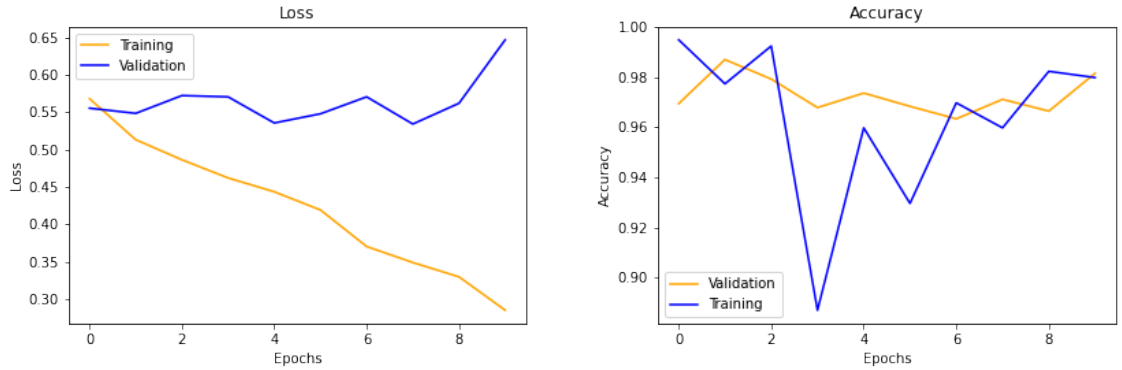


Figure 5. Structure of our best model, based on BERT and GRUs

The model has been trained for 10 epochs, in figure 6 is shown the trend of loss and accuracy with increasing epochs. From figure 6a it can be seen that the training loss decreases with time, as expected, while from figure 6b it can be seen a fluctuating trend of the accuracy. This can be due to the amount of parameters of the model which is big compared with the size of the dataset. This can lead to an early fit of the training set but without generalization capability. In fact, although the accuracy on the training data fluctuates, the one on the validation set has a more stable trend and it is growing in the last epochs.



(a) Epoch x Loss **(b)** Epoch x Accuracy

Figure 6. Statistics of the training of the BERT GRU model

In the table 4 we show some example predictions:

Original Text	Irony		Sarcasm	
	Prediction	Target	Prediction	Target
-Prendere i libri in copisteria-Fare la spesa-Spararmi in bocca-Farmi la doccia	0.956	1	0.45	0
@Alexandra_Voice allontanare dalle istituzioni tutti coloro che negli ultimi anni hanno avuto incarichi elettivi #portaaporta	0.088	0	0.010	0
@flabel82 @PatriziaRametta ecco ora tutto chiaro...Europa unita dal canto,dal calcio,dalle gite pseudo culturali,e da tempo dal terrorismo	0.093	1	0.035	1
@francofontana43 scherzi?Quelli erano italiani!Gli italiani possono emigrare!Ma anche rubare,stuprare,spacciare.Basta che non siano migranti	0.733	1	0.775	1
@AlessiaMorani @19TIZIANO72 Dai, magari è straniero! Cerchiamo di essere inclusivi. Da dove viene, ingegner @19TIZIANO72 ? La stiamo accogliendo bene in Italia?	0.555	1	0.173	1

Table 4. Classification examples

The first row contains an objectively ironic tweet and as it can be seen the model correctly classifies it with an high degree of confidence. The same can be said for the tweet in second position, it is clearly not ironic and the model assigns it values near zero.

The third tweet is a clear example of what we expressed in the introduction. To understand if a sentence is ironic or not, often it is necessary to place it in the context where it was said. In this case the sentence has been labeled in the dataset as ironic and sarcastic, but to understand this you should have to know the latest news at the time it was said. Given the structure of our dataset, composed of individual tweets not placed in any context, this would have been too much to ask to our model.

The last two tweets were chosen to show how less clear sentences, even for a human, are classified with a lower confidence level.

6. Results

In this chapter we compare the results of our best model in the context of the challenge. To compute our score we contacted the organizers of the challenge who gave us the gold-labeled-testing dataset. In particular in table 5 there is the ranking of the task A, while in the table 6 the one regarding task B. In these tables is highlighted in gray the position that we would have reached if we had participated to the challenge, and also the baseline.

Name	F1 Avg
ItaliaNLP	0.731
ItaliaNLP	0.713
BERT-GRU	0.712
UNIBA	0.710
UNIBA	0.710
X2Check	0.704
UNITOR	0.700
UNITOR	0.700
X2Check	0.695
Aspie96	0.695
X2Check	0.693
X2Check	0.683
UOIRO	0.651
UOIRO	0.646
UOIRO	0.629
UOIRO	0.614
baseline-random	0.505
venses-itgetarun	0.470
venses-itgetarun	0.420
baseline-mfc	0.334

Table 5. Ranking task A

Name	F1 Avg
BERT-GRU	0.536
UNITOR	0.520
UNITOR	0.518
ItaliaNLP	0.516
ItaliaNLP	0.503
Aspie96	0.465
baseline-random	0.337
venses-itgetarun	0.236
baseline-mfc	0.223
venses-itgetarun	0.199

Table 6. Ranking task B

From the tables it can be seen that we would have scored third place in the first task, second only to the Italian NLP lab, while we would have been the best scoring team at task B.

7. Conclusion

We faced the challenge IronITA, deploying an effective preprocessing methodology and testing several approaches, very different from each other. Those approaches are about embedding the text and developing the models themselves.
The results of the tests declared as the best model the one based on BERT and GRUs, which is not surprising since the first is obtaining state-of-the-art results on many tasks, while the second is a very efficient implementation of Recurrent Neural Networks.
Regarding the positioning within the two rankings, we can only be satisfied with the results achieved.

Appendix A Code

In this section we describe the code choices and structure. To develop the project we used the following technologies: Python as the programming language of choice due to its flexibility and mature echo system, Tensorflow [9] as the Machine Learning framework and a pre-trained BERT model [3] from huggingface.co. The code was developed and tested on Google Colab using Jupyter Notebooks and then transferred into python modules to be used as a library. As in standard python fashion, all the dependencies required to run the code are in the `requirements.txt` file and can be installed using pip with the following comand `pip install -r requirements.txt`. The use of a virtual environment is encouraged. The main code is split across three folders:

- `preprocessing` - contains utilities for text preprocessing implementing the pipeline described in section 3.
- `models` - contains the models implementations.
- `notebooks` - contains the Jupyter notebooks we used to develop and test the code. The notebooks containing the models' implementations are meant to be run on Colab.

It can be downloaded from GitHub following the following links:

- https://github.com/gerzin/IronySarcasmDetectorIT.
- https://github.com/MatteoMass/IronySarcasmDetectorIT.

From there it is also possible to download the weights of the trained BERT-GRU model with detailed instructions on how to load and use the model.

References

1. Cignarella, A.T.; Frenda, S.; Basile, V.; Bosco, C.; Patti, V.; Rosso, P.; others. Overview of the EVALITA 2018 task on Irony Detection in Italian tweets (IronITA). Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian (EVALITA 2018). CEUR-WS, 2018, Vol. 2263, pp. 1–6.
2. Cimino, A.; De Mattei, L.; Dell'Orletta, F. Multi-task learning in deep neural networks at evalita 2018. *Proceedings of the Wvaluation Campaign of Natural Language Processing and Speech tools for Italian* **2018**, pp. 86–95.
3. Italian BERT model. https://huggingface.co/dbmdz/bert-base-italian-xxl-cased. Accessed: 09-30-2021.
4. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019, [arXiv:cs.CL/1810.04805].
5. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need, 2017, [arXiv:cs.CL/1706.03762].
6. OPUS corpora collection. https://opus.nlpl.eu/.
7. LSTM and GRU. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21. Accessed: 09-30-2021.
8. Albawi, S.; Mohammed, T.A.; Al-Zawi, S. Understanding of a convolutional neural network. 2017 International Conference on Engineering and Technology (ICET), 2017, pp. 1–6. doi:10.1109/ICEngTechnol.2017.8308186.
9. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Goodfellow, I.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Jozefowicz, R.; Kaiser, L.; Kudlur, M.; Levenberg, J.; Mané, D.; Monga, R.; Moore, S.; Murray, D.; Olah, C.; Schuster, M.; Shlens, J.; Steiner, B.; Sutskever, I.; Talwar, K.; Tucker, P.; Vanhoucke, V.; Vasudevan, V.; Viégas, F.; Vinyals, O.; Warden, P.; Wattenberg, M.; Wicke, M.; Yu, Y.; Zheng, X. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Software available from tensorflow.org.