Aspie96 at IronITA (EVALITA 2018): Irony Detection in Italian Tweets with Character-Level Convolutional RNN

Valentino Giudice
Computer Science Department of the University of Turin
valentino.giudice96@gmail.com

Abstract

English. Irony is characterized by a strong contrast between what is said and what is meant: this makes its detection an important task in sentiment analysis. In recent years, neural networks have given promising results in different areas, including irony detection. In this report, I describe the system used by the Aspie96 team in the IronITA competition (part of EVALITA 2018) for irony and sarcasm detection in Italian tweets.

Italiano. *L'ironia è caratterizzata da un forte contrasto tra ciò che viene detto e ciò che si intende: questo ne rende la rilevazione un importante task nell'analisi del sentimento. In anni recenti, le reti neurali hanno prodotto risultati promettenti in aree diverse, tra cui la rilevazione dell'ironia. In questo report, descrivo il sistema utilizzato dal team Aspie96 nella competizione di IronITA (parte di EVALITA 2018) per la rilevazione dell'ironia e del sarcasmo in tweet italiani.*

1 Introduction

Irony is a rhetorical trope characterized by a strong contrast between what is literally said and what is really meant. Detecting irony trough automatic devices is, therefore, important for other tasks of text analysis too, such as sentiment analysis, as it strongly changes the meaning (and the sentiment) of what is said.

Sarcasm is defined in different ways in different contexts. One such definition is that it is a particular kind of irony: irony with a specific target to attack, more offensive and delivered with a hash tone.

IronITA (Irony Detection in Italian Tweets) (Cignarella et al., 2018), a shared task organized within EVALITA 2018, the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian, has, as a purpose, irony and sarcasm detection in Italian tweets and considers sarcasm as a kind of irony. The competition contained two subtasks: subtask A was about labeling each tweet as either non ironic or ironic, whereas subtask B was about labeling each tweet as non ironic, ironic but not sarcastic or sarcastic. The training dataset was provided by the organizers: within it, the annotations specify, for each tweet, if it is ironic and if it is sarcastic, marking no non-ironic tweet as sarcastic. The non-annotated test dataset was given to the teams to annotate. Each team was allowed a maximum of 4 runs (attempts to annotating the dataset) for either task, each of which was ranked in the leaderboard. Taking part in subtask B implied taking part in subtask A with the same run annotations. Of the maximum of 4 runs for each task, each team was allowed 2 constrained runs (using no other dataset containing irony or sarcasm annotations of tweets or sentences but the provided one) and 2 unconstrained runs (where teams were allowed to use other training data), and taking part in a subtask with an unconstrained run implied taking part in the same subtask with a constrained run as well.

A separate leaderboard was provided for subtask A and subtask B. For each leaderboard, the F1-score of every run for each of the two (for subtask A) or three (for subtask B) classes is shown. The actual score of each run in either leaderboard is the arithmetical average of the F1-scores for all classes in the corresponding subtask.

In recent years, neural network have proven themselves to be a promising approach to various problems of text analysis, including irony detection; the following sections propose, for the task, the model used by the Aspie96 team: a multilayer neural network for binary classification.

The proposed model was used by the team for an individual, constrained, run for the subtask B (therefore also taking part in subtask A with the same annotations). The results of the competition, as well as the details of the model, are described in the following sections.

2 Description of the System

The proposed system is a neural network composed as follows.

It begins with a series of unidimensional convolutional layers, followed by a bidirectional recurrent layer based on GRU units (Cho et al., 2014) (a variation of LSTM (Hochreiter and Schmidhuber, 1997)). The output of the bidirectional layer is then used as the input for a simple feed forward neural network, whose output is just one number between 0 and 1 (low numbers represent the negative class, whereas large numbers represent the positive class): the sigmoid activation function is used to ensure an output within the range. The purpose of the convolutional layers is to convert the input to a form more meaningful for the neural network and to recognize short sequences within the text, whereas the recurrent part of the network has the purpose of converting a sequence of vectors into an individual vector of high-level features.

To better understand the role of the convolutional layers, the first layer should be considered. Its main difference with the following ones is that it also has the purpose of reducing the depth of the input (which can be done without loss of information as the input vectors are rather sparse). Its inputs are in a sequence and every short subsequence is converted into an individual vector. This preserves the information in the order of the sequence since only a short subsequence is used to produce every output vector and, thus, each output vector depends on a specific (short) part of the input sequence. The sequence every output vector depends on is shifted by one for each of them. The reason to use convolutional layers is to provide a context for each character being encoded as the meaning and importance of a character in an alphabetical language depends on its surrounding ones as well. The output of the last convolutional layer, thus, is a sequence of vectors which is just shorter than the original one and still encodes the useful information of each individual character in the sequence, but provides, for each character, a context dependent on the surrounding ones. Each convolutional layer expands the amount of character each vector depends on, while still keeping important information encoded in the sequence (indeed, the context of each character also provides information about which information is most useful and has to be represented). The input produced for the recurrent layer is slightly smaller, preserves temporal information and, for each character, provides information depending on the context, which constitutes a higher level feature than the character itself. The benefit of using convolutional layers is that the kernel doesn't vary throughout the sequence. This is particularly useful because short sequences within the text might have the same or similar meanings regardless of their position.

The convolutional layers do not use any padding and, because of that, they slightly reduce the length of the sequence (the length of the input sequence is slightly larger than the length of the output sequence), but no subsampling layer is used. The width of each kernel is rather small, as is the depth of the output of each layer and they are both hyperparameters that can be decided arbitrarily (except, of course, for the depth of the output layer, which has to be 1). After the recurrent layer, no use was found in adding extra fully connected layers before the output to deal non-linearity, so it is followed only by a fully connected layer whose input is the output of the recurrent layer and whose output is the output of the neural network.

In order to have regularization, dropout layers between the layers of the network of above and a gaussian noise layer applied to the input are used. Their purpose is similar and is to make the neural network better able to generalize, even without a very big training dataset. Particularly, thanks to the gaussian noise applied to the input, during training, the same tweet, read multiple times, does not constitute the same input data and the alteration in the data is propagated through the network.

A visualization of the model is given in fig. 1. The depth of the data at every layer isn't shown for simplicity, thus each box is meant to represent a vector (rather than an individual value). The length of the input sequence would actually be larger, but only a part is shown for simplicity. The regularization layers aren't shown.

The input is represented as an array with length 140, where each element is a vector of flags whose

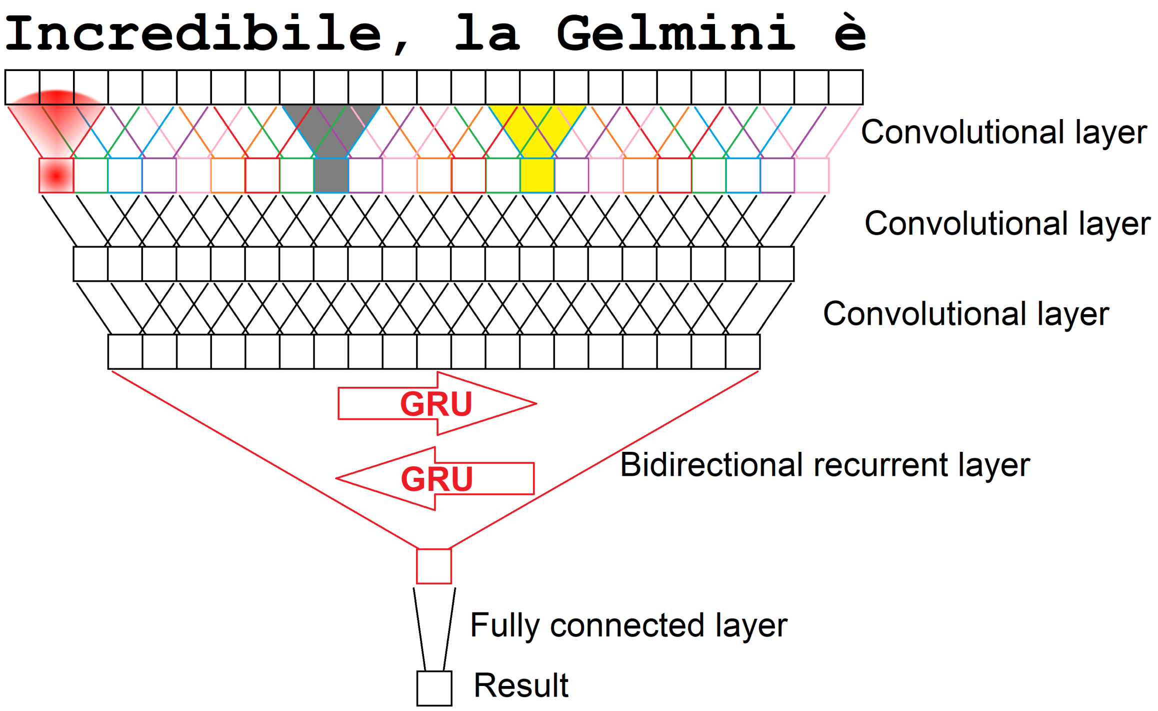

Figure 1: Visualization of the proposed model.

values are either 0 or 1. Each vector represents a character of the tweet: the tweet representation is either padded (by adding vectors with every flag at 0 at the beginning) or truncated (by ignoring its right part) in order to meet the length requirement. Most of the flags (namely 62 of them) are mutually exclusive and each of them represents a known character. Additional flags (namely 6) are used to represent properties of the current character (such as being an uppercase one). The total lenght of each vector is 68. Emojis are represented similarly to their Unicode (English) name, with no delimiters, and flags are used to mark which letters and underscores belong to the name of an emoji and which letter is the start of the name of an emoji. Spaces at the beginning or the end of the tweet and multiple spaces, as unknown characters, are ignored. The full list of known characters (excluding emojis) is as follows:

Space ! ” # $ % & ’ ( ) ∗ + , − . / 0 1 2 3 4 5 6 7
8 9 : ; = ? @ [ ] _ a b c d e f g h i j k l m n o p q
r s t u v w x y z | ˜

The subtask A is of binary classification: the model can be directly applied. The subtask B, however, has three classes. Since taking part in subtask B implied taking part in subtask A as well with the same annotations, the following approach was used: two identical copies of the model were created and trained, the purpose of one was to tell apart non ironic tweets and ironic ones and that of the other was to distinguish, among ironic tweets, which ones were not sarcastic and which ones were. In essence, the subtask B was seen as two separate problems of classification: one identical to subtask A and the second between non sarcastic and sarcastic ironic tweets.

The two identical models were trained individually by using only the data provided by the task organizers for the training phase: all data was used to train the model for the subtask A and only the ironic tweets were used to train the model for the second part of subtask B (detection of sarcasm in ironic examples).

The testing dataset was, therefore, annotated as follows: every tweet which was recognized by the first copy of the model as belonging to the negative class was labeled as non-ironic and non-sarcastic. Every tweet recognized in the positive class was labeled as ironic and also evaluated using the second copy of the model, marking it as non sarcastic when it was recognized in the negative class and as sarcastic otherwise. Therefore, no tweet was

Perceived class	Actual class		
	Non ironic	Ironic (non sarcastic)	Sarcastic
Non ironic	1008	453	295
Ironic (non sarcastic)	184	637	188
Sarcastic	138	449	227

Table 1: Confusion matrix from the 10-fold cross validation on the training dataset.

marked as sarcastic without being also marked as ironic.

3 Results

After training on the dataset provided by the organizers, an individual run was generated from the testing dataset and used for both subtask B and subtask A.

The proposed model ranked 9th among 17 runs (ignoring the baseline ones) in subtask A, as the 5th among 7 teams, considering the best run for each team. The F1-score for the negative class (non-ironic) was 0.668 (precision 0.742 and recall 0.606) and the F1-score for the positive class (ironic) was 0.722 (precision 0.666 and recall 0.789). The score was, thus, 0.695. This is consistent with the results obtained during the testing of the model (before the actual submission of the results), with a relatively low drop in the F1-scores. As a comparison, the first ranked team, with two runs, ranking first and second, got a score of 0.731 and 0.713.

In subtask B, the model ranked 5th among 7 runs (3rd among 4 teams), with the same F1-score for the neutral class (0.668), 0.438 for the ironic (non-sarcastic) class and 0.289 for the sarcastic class, with a score of 0.465.

A 10-fold cross validation using the training data produced the confusion matrix shown in Table 1.

The F1-score of the ironic (sarcastic and non sarcastic combined) class is 0.737. The F1-score for the ironic non-sarcastic class is 0.500 and the F1-score of the sarcastic class is 0.298. The F1-score for the negative class is, instead, 0.653 (for both subtasks). Therefore, using this data to compute the scores for the two subtasks, the score is 0.695 for the subtask A (the same as in the competition) and 0.484 for subtask B.

Table 2 shows, for the copy of the model trained to distinguish between irony and non irony, a few examples of tweets correctly and wrongly classified. These classifications were obtained using cross validation.

Similarly, Table 3 shows a few examples of tweets correctly and wrongly classified by the copy of the model trained to distinguish between ironic non sarcastic and sarcastic tweets, among those correctly detected as ironic.

4 Related work

A roughly similar model based on convolutions for text classification has been presented in 2014 (Kim, 2014) in the context of EMNLP. The model used word-level features (trough a pretrained and then fine-tuned embedding layer) as the input for an individual convolutional layer. The convolutional layer used multiple kernels of different sizes, to detect different high-level feature. To convert the output of the convolutional layer (whose depth was the number of its filters) into an individual vector of high-level features, a timewise max-pooling layer was used, producing a vector whose length was the same as the number of kernels in the convolutional layer (for each element in the resulting vector, its value was the maximum produced by the corresponding kernel along its input). The resulting vector was the input of a fully connected layer producing the output of the neural network. The model produced results better of those of the state of the art at the time on 4 out of 7 tasks.

In 2015 (Zhang et al., 2015), a model more similar to the one proposed was presented. The model used a character-level convolutional neural network for text classification, achieving competitive results. However, it did not use a recurrent layer (and was, in fact, compared with recurrent neural networks) and represented each input character as a one-hot encoded vector (without any aditional flags). The model was trained using very big datasets (hundreds of thousands of instances, whereas only 3977 tweets were provided for IronITA) as this works better for character-level neural networks (because other kind of model often depend on pretrained layers and can, thus, use knowledge, for instance that related to the meaning of words, which couldn't be derived from the training dataset alone). Because of its structure and attributes, the model wasn't much flexible and easily adaptable to different kinds of usage.

5 Discussion

The system is different from others proposed in the past because it strictly works at character-level,

Tweet	Predicted class	Actual class
@matteorenzi ma quale buona scuola con un ministro incompetente? #gianninidimettiti miur ha combinato pasticcio su #concorsonazionale	Non ironic	Non ironic
#sfplm85bis #labuonascuola dopo 5 anni di sfp nel 2017, noi del NO che faremo? Fuori dal concorsone del 2015 e senza supplenze!	Non ironic	Non ironic
#Terremoto #Cile. 3 differenze con l'Italia: magnitudo superiore, rischio #Tsunami e nessuno che nell'emergenza incolpi i #migranti. #Natale	Non ironic	Ironic
Ma in italia non viene Obama a terrorizzarci nel caso di una vittoria del NO? #IoVotoNO #IoDicoNo	Non ironic	Ironic
Mario Monti senatore a vita?	Ironic	Non ironic
l'imbarbarimento è avvenuto perchè ai rom ormai è concesso tutto:rubano,schippano,ti entrano in casa e se ti difendi arrestano te #tagadala7	Ironic	Non ironic
SpecialiTG dopo attacchi terroristici: Sigla A)ISLAM è religione di Pace B)Attentatori eran depressi,omofobi,vittime bullismo o folli Sigla	Ironic	Ironic
Com'è che vince il no, Renzi si dimette ma i migranti arrivano ancora???? Perzone falzeeeee	Ironic	Ironic

Table 2: Examples of ironic and non ironic tweets classified by the proposed model.

Tweet	Predicted class	Actual class
#governo #Monti: ma non c'è nessun indagato fra i #ministri? Nemmeno fra i sottosegretari? E' possibile? In Italia?	Non sarcastic	Non sarcastic
@matteorenzi le risorse della scuola pubblica alle private... Questa è la buona scuola!	Non sarcastic	Non sarcastic
Incredibile, la Gelmini è entusiasta delle linee guida della riforma Renzi - Giannini. Chi lo avrebbe mai detto!? #labuonascuola	Non sarcastic	Sarcastic
#terroristaucciso oltre che nome e cognome dei due agenti,date anche gli indirizzi e i numeri di telefono, così li trovano prima .	Non sarcastic	Sarcastic
Qui nell'hinterland m.se siamo alla follia, posti di blocco dovunque, scene d'apocalisse zombie, rom inseguiti nei parchi #papamilano2017	Sarcastic	Non sarcastic
Passare da Berlusconi a Mario Monti è un salto troppo grosso. Ci vorrebbe almeno un governo di transizione presieduto da Checco Zalone	Sarcastic	Non sarcastic
#tfaordinario = morto che cammina. Anche quest'anno mi fate lavorare 18h...ma sono SENZA futuro grazie a #labuonascuola di @matteorenzi	Sarcastic	Sarcastic
Salvini,oltre a propagandare aggressività,riuscirà a superare il complesso del narciso felpato?Dopo immigrati,rom,si dedicherà ai politici?	Sarcastic	Sarcastic

Table 3: Examples of non sarcastic and sarcastic ironic tweets classified by the proposed model.

without any word features and because it doesn't use any data but what is provided for the specific task during learning, nor is it based on other, simpler, models to extract features. Many other models are instead based on feature-engineering and they, thus, often use layers (such as word embedding) pretrained on data different from that generated for the task. The results of the model in subtask A differ from those of the top ranked run by slightly less than 0.04 and by a maximum of 0.018 from all other runs in between. If other models are based on word-level features or on other simple and pretrained models extracting high level features, this suggests that good and similar results can be obtained by using strictly the data provided and without any word-level feature. The proposed model is not claimed to be the best possible with this properties; rather, it is an extremely simple attempt to the task. Other models may be built in the future which do not use any information outside of that provided in the learning dataset, but obtaining significantly better results than the proposed one. Still, the ranking position of the proposed model suggests that there is value in using knowledge outside of that available for a specific task, giving information about the language (and the meaning and sentiment of words and phrases): further research is needed to understand where the limits of strict character-level text analysis lay and the contexts in which it is a better or a worse solution.

References

Kyunghyun Cho, Bart van Merriënboer, Çağlar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, oct. Association for Computational Linguistics.

Alessandra Cignarella, Simona Frenda, Valerio Basile, Cristina Bosco, Viviana Patti, and Paolo Rosso. 2018. Overview of the evalita 2018 task on irony detection in italian tweets (ironita). In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. 9:1735–80, 12.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao Jake, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *CoRR*, abs/1509.01626.