Methods, Resources and Tools for Irony Detection in NLP

Leila Moudjari[1], Karima Akli-Astouati[1,2]
[1]University of Science and Technology Houari Boumediene, Research in Intelligent Computing, Mathematics and Applications (RIIMA laboratory)
Algeria
{moudj11@gmail.com} {kakli@usthb.dz}

ABSTRACT: *Irony detection is considered as an important system in opinion communication for a specific task in social networks. Thus, we studied and documented the results of IDAT 2019 Task for irony detection in the tweets of texts where we took Arabic to study. In this work we initially used the labelled data from the selected tweets. Finally we are able to present the system, resources, and tools that we have deployed to achieve better results.*

Keywords: Irony Detection, Machine Learning, Feature Selection, Classification, CNN

Received: 28 November 2019, Revised 23 February 2020, Accepted 6 March 2020

DOI: 10.6025/jet/2020/11/3/91-96

1. Introduction

The advent of the social web produced a crescent interest for opinion analysis. When it comes to Natural Language Processing (NLP), the use of sarcasm and irony arose new issues. The detection of irony is a complex linguistic task -especially for languages such as Arabic. Although, research is this area made progress, irony detection is still a hot topic in the research community and it is widely studied in philosophy and linguistics.

The IDAT task: Irony Detection in Arabic Texts, focuses on irony detection over a collection of Arabic tweets collected and annotated. The challenge is to build a binary classier that determines whether an Arabic tweet is ironic or not.

For this, we propose a word embedding based model and a hybridisation between sub-words embedding based and feature based model by relying on deep neural model.

The remaining paper is structured as follows. Section 2 reviews some existing works related to irony detection in social media

and the use of embedding for classification tasks. Section 3 describes the proposed models. The evaluation performance is discussed in Section 4. Conclusion is outlined in section 5 with some perspectives for a future work.

2. Related Work

In order to capture the irony expressed in a text we explore the embedding. A semantic layer which can provide more information about the context. Therefore, in this section we present some related work to irony detection and models of embedding.

2.1 Irony Detection

In recent years, several approaches have been proposed to deal with irony detection in social media ([4], [2], [5]).

Irony detection has been treated as a problem of classification, where classifiers such as decision trees and support vector machine (SVM) give the best results ([3]).

Regarding Arabic, the attempts in which irony has been addressed in literature are fewer. According to [11], there are no automatic approaches to detect irony.

In [12], the authors manually analyzed the similarities and differences between ironic expressions in English and Arabic. They used data from books, articles and Internet.

Karoui and al. in [6] used multiple features such as surface, sentiment, false assertion and exaggeration to infer the context needed to detect irony in Arabic social media texts.

Such works encourage the idea to explore sentiment analysis approaches and apply them to irony detection.

2.2 Embedding

Embedding provide a dense representation of the entity (words, characters, subwords) and their relative meanings.

Embeddings can be learned from text data and reused among machine learning algorithms. They can also be learned as part of fitting a neural network on text data.

While the point of network training is to learn good parameters, word vector representations follow the notion that similar words are closer to each other [9].

There are many models offered for learning word embedding from raw text. Among these are GloVe [10] and dependency-based word embedding [7]. The well-known and widely used word2vec model introduced by Mikolov in 2013 at Google ([8], [9]) was chosen for use.

Word2vec describes two architectures for computing continuous vectors representations, the skip-gram (SG) and Continuous Bag-Of-Words (CBOW). The former predicts the context-words from a given source word, while the latter performs the opposite and predicts a word according to its context window.

Embedding has shown their strengths in other classification tasks such as sentiment analysis. Therefore, we decided to apply it to the irony detection task, since we need to classify a text as ironic or non-ironic.

Our test showed that the CBOW model performs a little better compared to the SG model.

In the next section, we describe our models and the data used.

3. Methodology

The detection of irony is a difficult task that is complicated by the complexity of the Arabic text. In this section we describe the methodology we followed to resolve the problem that is raised. We first start by describing the data used to train and test the models.

3.1 Data

The IDAT shared dataset [1] includes tweets related to different political issues and events related to the Middle East that was held during the years 2011 to 2018. Tweets are written in a formal language (standard Arabic) and in dialects of some Arab countries: Egypt, Gulf, Levantine (dialect of Syria, Lebanon and Palestine), and Maghrebi dialects. Table 1 shows the distribution of the tweets for the shared task of irony detection.

Label	Total
Ironic (1)	2087
Non ironic (0)	1929

Table 1. Tweets distribution over classes

For the present task, several models have been built and tested. In the next sections, we detail our models and provide our results.

3.2 Models

Baseline: In every experiment, we used frequency Bag of Words (BoW) model as lexical features. We also performed standard text pre-processing by deleting user mentions, URLs, stop words and the words containing less than 3 characters.

To set a baseline for our models we used the SVM model which is widely used for classification tasks in general and the irony detection in particular.

Convolutional neural network (CNN): Our both proposed models implement CNN. However, each one uses different parameters. We used one convolution (conv1D) layer, followed by a max pooling layer and a dropout of 20%.

Before the dense layer we used a atten layer.

In Figure 1 we give the general architecture of our CNN model.

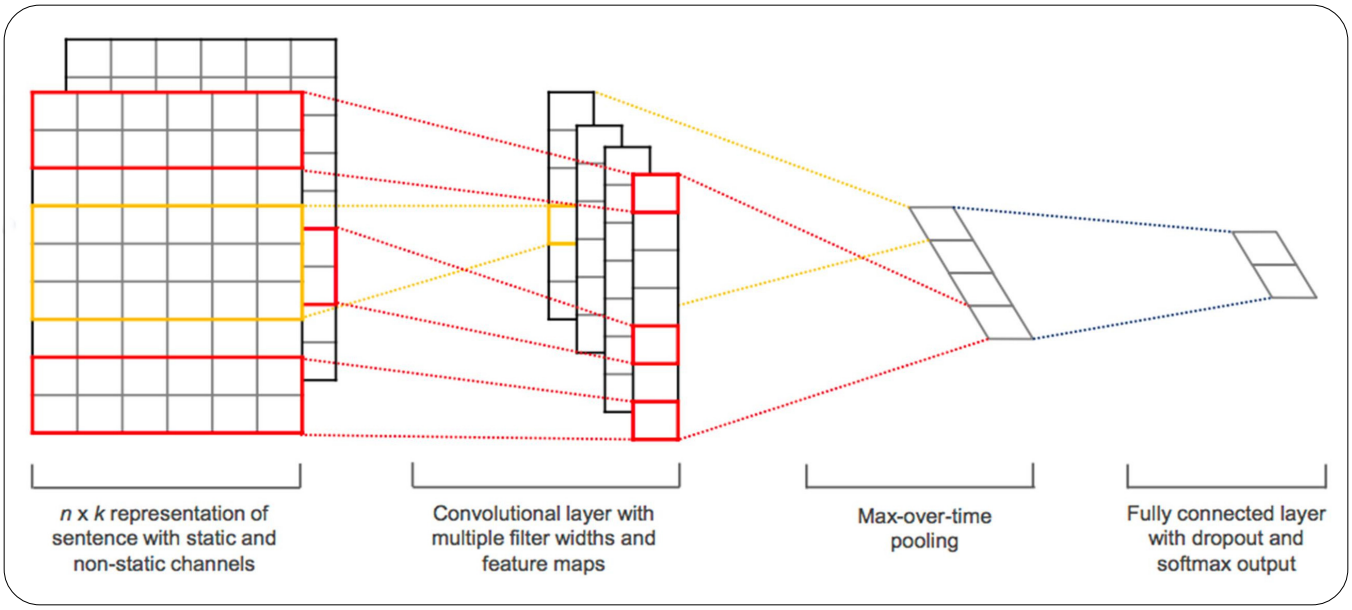

Figure 1. The architecture of the Convolutional neural network

Word embedding model (M1): We used keras Embedding layer [3]. It requires that the input data is digitally encoded. The Embedding layer is initialized with random weights and will learn an embedding for all of the words in the training dataset.

For this first proposed model we tried creating pre-trained vectors, but during the tests the use of the keras Embedding layer helped improve the results.

Sub-word embedding and feature selection (M2): The sub-word embedding model relies on the best features that yield the best results. Therefore, we followed some steps to help us achieve this.

First, we relied on chi2 [4] to extract relative features (url existence, word count, negative and positive word count, punctuation count, tag existence). In order to extract the positive and negative words, we relied on publicly available lexicons [5].

The following graph (Figure 2) shows the features that gave the best scores.

Second, we created our sub-words in a BOW format.

Third, we used keras layer to create the embedding vectors.

Lastly, we merged the features array with the sub-words embedding array for each tweet.

The following table (2) gives the detailed parameters for each model:

The tests concluded that these two models give the best results compared to the others (long short term memory, ...). The following section details these results.

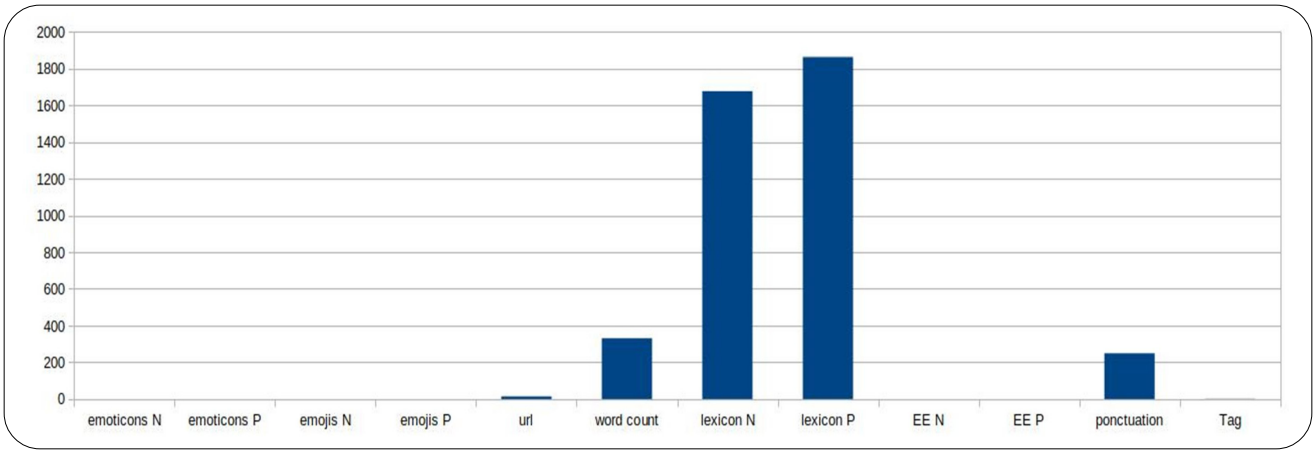

Figure 2. The graph representing Chi2 scores

4. Results

We tested several machine learning algorithms in order to evaluate and select the best performing one. Table 3 presents the results of the tests during the testing phase for each class and the official results. In terms of accuracy (A), macro-averaged F-score (F), precision (P) and recall (R).

As we can see the M2 model gave the best performance. These results were obtained by using a train-test configuration of a

[3] https://keras.io/layers/embeddings/#embedding

[4] https : ==scikit–learn:org=stable=modules=generated=sklearn:featureselection:chi2:html

[5] https://saifmohammad.com/WebPages/ArabicSA.html

random test, not by using cross validation. These results are comparable to the ones obtained during the testing phase. Although we encountered a significant decrease in the systems performance in the official test, we believe that our system can be used for tasks such as irony or sarcasm detection.

Parameter	**M1**	**M2**
Embedding size	300	100
validation split	0:1	0:1
batch	200	300
inner layer neurones	50	70
epoch	5	8
activation	relu	relu
optimizer	adam	adam
loss	mse	mse
classification	sigmoid	sigmoid

Table 2. CNN parameters for both models

Testing phase							**Official results**
metrics	F(0)	F(1)	A	P	R	F	F
SVM	0.76	0.78	0:76	0:77	0:77	0:77	0:689
M1	0.68	0.79	0:75	0:77	0:74	0:73	0:687
M2	**0.82**	**0.80**	**0.81**	**0.81**	**0.81**	**0.81**	**0.695**

Table 3. Irony detection of Arabic tweets results

5. Conclusion

In this article, we have proposed two models that can be used to identify ironic messages, incorporating various features to capture ironic statements. Our results revealed good classification performance on the training dataset, but a lower performance on the evaluation data, with a notable decrease in the F-score. We found that a majority of the systems we tested consistently provided slightly higher scores for the ironic texts. We believe that this is due to the fact that the dataset has more texts labeled as ironic.

In our future work we plan to study ways to improve our system for classification tasks. We also want to test the part-of-speech tagging to see whether it affects the results. What are the words that are more ironic adjectives or adverbs? Such features and more are very interesting to explore.

References

[1] Ghanem, B., Karoui, J., Benamara, F., Moriceau, V., Rosso, P. (2019). Idat@re2019: Overview of the track on irony detection in arabic tweets. *In*: Mehta P., Rosso P., Majumder P., Mitra M. (Eds.) Working Notes of the Forum for Information Retrieval Evaluation (FIRE 2019). CEUR Workshop Proceedings. *In*: CEUR-WS.org, Kolkata, India, December 12-15 (2019).

[2] Ghosh, A., Veale, T. (2018). Ironymagnet at semeval-2018 task 3: A siamese network for irony detection in social media. *In*: Proceedings of The 12th International Work-shop on Semantic Evaluation. p. 570-575.

[3] Hern–andez-Far–as, I., Bened–, J. M., Rosso, P. (2015). Applying basic features from sen- timent analysis for automatic irony detection. *In*: Iberian Conference on Pattern Recognition and Image Analysis. p. 337-344. Springer.

[4] Jia, X., Deng, Z., Min, F., Liu, D. (2019). Three-way decisions based feature fusion for chinese irony detection. *International Journal of Approximate Reasoning*.

[5] Karoui, J., Farah, B., Moriceau, V., Patti, V., Bosco, C., Aussenac-Gilles, N. (2017). Exploring the impact of pragmatic phenomena on irony detection in tweets: A multilingual corpus study. *In*: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: 1, p. 262-272.

[6] Karoui, J., Zitoune, F. B., Moriceau, V. (2017). Soukhria: Towards an irony detection system for arabic in social media. *Procedia Computer Science* 117, 161-168.

[7] Levy, O., Goldberg, Y. (2014). Dependency-based word embeddings. *In*: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, 2, p. 302-308.

[8] Mikolov, T., Chen, K., Corrado, G., Dean, J. (2013). Ecient estimation of word repre- sentations in vector space. arXiv preprint arXiv:1301.3781.

[9] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *In*: Advances in neural information processing systems. p. 3111-3119.

[10] Pennington, J., Socher, R., Manning, C. (2014). Glove: Global vectors for word representation. *In*: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). p. 1532-1543.

[11] Rosso, P., Rangel, F., Far–as, I. H., Cagnina, L., Zaghouani, W., Char, A. (2018). A survey on author proling, deception, and irony detection for the arabic language. *Language and Linguistics Compass* 12 (4) e12275.

[12] Sigar, A. H., Taha, Z. S. (2012). A contrastive study of ironic expressions in english and arabic. *College of Basic Education Researches Journal* 12 (2) 795-815.