Models of Irony detection in Natural Language Processing

Tharindu Ranasinghe, Hadeel Saadany, Alistair Plum, Salim Mandhari, Emad Mohamed
Constantin Orasan, Ruslan Mitkov
Research Group in Computational Linguistics
University of Wolverhampton, UK
{ftharindu.ranasinghe, h.a.saadany, a.j.plum, s.m.almandhari, e.mohamed2, c.orasan, r.mitkovg}@wlv.ac.uk

ABSTRACT: *Using specific deep learning models, we have introduced irony detection in Arabic language with the help of the IDAT 2019 Shared Task. We have tested a few available models and understand how the document content cleaning and pre-processing work. In the trials we have conducted we found that a higher F1 score is achieved and the RGGL ranks in a top level. We finally found that the introduced system can able to get competitive results.*

Keywords: Irony Detection, Deep Learning

Received: 19 October 2019, Revised 11 March 2020, Accepted 24 April 2020

DOI: 10.6025/jet/2020/11/3/83-90

1. Introduction

According to philosophers such as Grice, irony could be defined as an utterance which violates a conversational maxim [6]. The user of irony intentionally breaks the norms by an unexpected play on words. The wordplay in irony is often based on common knowledge shared by the speaker and the listener which can relate to their specific culture or social background. This unexpectedness factor cuts through different linguistic levels.

On one level unexpectedness is achieved by what can be dubbed as \sentiment imbalance". It refers to the use of a negative or positive word where the opposite polarity is expected from the context. The ironic tweet, "... الاقتصاد المصري ينزف لكنه صامد" (The Egyptian economy is crumbling yet it is unfaltering) is a good example of this. The use of the word 'صامد' ('unfaltering or rm') with its positive polarity where a negative item is expected creates an ironic effect.

Moreover, the out-of-context ironic element can be pointed out by a typo-graphical element such as quotes, bold typing, ellipsis and emoticons. Other observed features of the ironic data are the frequent use of parallel syntactic structures, repetition of one or more lexical items and the use of polysemy or multi-sense words.

Recently, a lot of research has been carried out in the field of natural language processing in order to detect irony. This is evidenced by an increase in irony detection shared tasks. SemEval-2018 Task 3 focused on irony detection in English tweets [7]. Also, there were shared tasks for irony detection in French [1] and Italian [3].

Another such task, which is at the FIRE 2019 conference, is Irony Detection in Arabic Tweets (IDAT). Given a tweet, systems have to classify it as either ironic or not ironic. This paper describes our submission to the IDAT shared task in irony detection. We propose a simple, low-effort approach, with minimal data processing. We employ six different neural network architectures in order to detect irony in tweets, evaluate each network and select the three best performing architectures for our final submissions.

The paper is structured as follows. Section 2 describes the system developed for this shared task and the dataset used to train and test it. Section 3 presents an analysis of the results of our evaluation of the five different architectures (Section 3.1), as well as of the final submission (Section 3.2). Section 4 describes the error analysis we performed for the test set. The paper is concluded by Section 5 offering some final remarks.

2. System Description

This section describes the shared task data, as well as the system that was used to classify the data. We use minimal preprocessing in order to use the data. For classification, we used and compared six different neural network architectures suited to this task. Our implementation has been made available on Github.[1]

2.1 Dataset

The dataset contains tweets related to different political issues and events related to the Middle East that hold during the years 2011 to 2018 [5]. The data provided by the task organisers was split into training and evaluation sets. The complete dataset is comprised of 4024 instances. We used 20% of the available data for evaluation and the rest of the data for training, resulting in 805 instances for evaluation and 3219 instances for training. The tweets were labelled with 1 to indicate irony and with 0 for non-irony. Duplicates, retweets and tweets containing pictures which would need to be interpreted to understand the ironic content have been removed by the organisers. The tweets are written using standard Arabic (formal) and different Arabic language varieties: Egypt, Gulf, Levantine, and Maghrebi dialects.

2.2 Text Processing

The main objective of the classification task is to capture the irony-defining features in the Arabic tweets, and hence the data has been cleaned accordingly. The team has chosen to delete a number of textual features that do not contribute to the classification and maintain others which may be significant in spotting an ironic tone in the tweet.

Twitter users writing in Arabic can either write in Modern Standard Arabisc (MSA) or in their particular dialects. If the tweet is written in MSA, it may or may not include diacritics whereas dialectical Arabic does not include any. Ironic tweets can be formulated in any of these different versions of the Arabic transcript. Thus, in order to avoid false classifications due to a non-defining feature, regular expressions were used to delete diacritics.

Moreover, native Arabic tweeters can either include punctuation or ignore it completely. Again, regular expressions were used to delete a number of punctuation markers, and multiple spaces were reduced to a single space. Some punctuation markers were not deleted if it was assumed to be significant for capturing irony. For example, it has been observed that the ellipsis is often used in ironic tweets to indicate a pause, a hesitation or a trailing-o in thought that marks a shift to an ironic tone.

Similarly, emojis were not omitted as they were deemed important for the classification. Finally, English characters which are either used to refer to the user's account or to a word in Arabizi (the Arabic Chat Alphabet) were also deleted. Arabizi characters were removed as they are not written in an Arabic script, but rather a mixture of English characters and symbols which would not

[1] https://github.com/TharinduDR/Irony-Detection

be captured by the Arabic word-embedding model employed for classification.

2.3 System Architecture

After data processing each text is encoded using Arabic fast text [13] embeddings.[2] The encoded tweets are then classified by one of the neural network architectures. We evaluated six different neural network architectures for the classification tasks: pooled Gated Recurrent Unit (GRU) (Section 2.3.1), Long Short-Term Memory (LSTM) and GRU with Attention (Section 2.3.3), 2D Convolution with Pooling (Section 2.3.4), GRU with Capsule (Section 2.3.5) and LSTM with Capsule and Attention (Section 2.3.6).

The parameters of each architecture were optimised using 10-fold cross validation considering a binary cross entropy loss function and using adam optimiser [11] which provided the best results of all tested optimisers. We also used the reduced-learning rate on plateau technique when a deep learning architecture stopped improving. Deep learning architectures often benefit from reducing the learning rate by a factor once learning stagnates [18]. We monitored validation loss and if no improvement was seen for 2 epochs, the learning rate was reduced by a factor of 0.6. These architectures were successfully applied to a number of classification tasks such as aggression detection [8,15], toponym detection [16,17] and their success in these tasks inspired us to use them for this task.

2.3.1 Pooled GRU

In this architecture, after the embedding layer, embedding vectors are fed to the bi-directional GRU [2] at their respective timestep. The bi-directional GRU-layer has 80 units. The final timestep output is fed into a max pooling layer and an average pooling layer in parallel [19]. After this, the outputs of the two pooling layers are concatenated and connected to a dense layer [10] activated with a sigmoid function. Additionally, there is a spatial dropout [22] between the embedding layer and the bi-directional GRU layer to avoid over-fitting. This architecture has been discussed in [12] as a common architecture to perform text classification tasks.

2.3.2 Stacked LSTM with Attention

In this architecture, each of the embedding vectors is fed into a bi-directional LSTM-layer [20]. The output of this layer is again fed into a bi-directional LSTM-layer [20] with self attention [23]. Each of the bi-directional LSTM-layers has 64 units. Finally, the output is connected to two dense layers that are [10] activated first with a relu function, and then with a sigmoid function. We adopted this architecture from the Toxic Comment Classification Challenge in Kaggle.[3]

2.3.3 LSTM and GRU with Attention

With this architecture, the output of the embedding layer goes through a spatial dropout [22] and is then fed in parallel to a bi-directional LSTM-layer [20] with self attention and a bi-directional GRU-layer [2] with self attention [23]. Both the bi-directional LSTM-layer and the bi-directional GRU-layer have 40 units. The output from the bi-directional GRU-layer is fed into an average pooling layer and a max pooling layer. The output from these layers and the output of the bi-directional LSTM-layer are concatenated and connected to a dense layer with ReLU activation. After that, a dropout [21] is applied to the output and connected to a dense layer activated with a sigmoid function.

2.3.4 2D Convolution with Pooling

The fourth architecture takes a different approach than the previous architectures by using 2D convolution layers [25], rather than LSTM or GRU layers. The outputs of the embedding layers are connected to four 2D convolution layers [25], each with max pooling layers. All the 2D convolution layers were initialised with a normal kernel initialiser. The outputs of these are concatenated and connected to a dense layer activated with a sigmoid function after applying a dropout [21]. This architecture has been used in the Quora Insincere Questions Classification Kaggle competition.[4]

2.3.5 GRU with Capsule

Most of the previous architectures rely on a pooling layer. However, this architecture uses a capsule layer [9] rather than pooling layers. After applying a spatial dropout [22] the output of the embedding layer is fed into a bi-directional GRU-layer [2]. The

[2] https://dl.fbaipublicfiles.com/fasttext/vectors-wiki/wiki.ar.vec

[3] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

[4] https://www.kaggle.com/c/quora-insincere-questions-classification

bi-directional GRU-layer has 100 units and was initialised with the Glorot normal kernel initialiser and orthogonal recurrent initialiser with 1.0 gain. The output is then connected to a capsule layer [9]. The output of the capsule layer is attened and connected to a dense layer with ReLU activation, a dropout [21] and batch normalisation applied, and re-connected to a dense layer with sigmoid activation. This architecture has been used to detect aggression in tweets [8].

2.3.6 LSTM with Capsule and Attention

The final architecture uses combination of a capsule layer [9] and a self attention layer [23]. After the embedding layer a spatial dropout [22] is applied to the output, which is then fed into a bi-directional LSTM-layer [20] with 80 units. The layer is initialised with the Glorot normal kernel initialiser and orthogonal recurrent initialiser with 1.0 gain. The output of the bi-directional LSTM-layer is fed into a capsule layer and to a self attention layer in parallel. Then each output of both capsule layers and the self attention layer goes through a DropConnect [24]. They are concatenated before connecting to a dense layer with sigmoid activation. This architecture has been used in the Jigsaw Unintended Bias in Toxicity Classification competition.[5]

3. Results

This section presents the results of the evaluation of the six architectures, as well as the evaluation of the final submissions. The competitors were allowed to submit three runs of their system to the evaluations. Therefore, we compare the performance of six different neural network architectures in order to select three submissions for the task.

3.1 Architecture Evaluation

This section describes how we selected the architectures for the final submission. To evaluate the architectures, we used 20% of the available training data, and used the rest of the data for training. Table 1 shows the evaluation results of each architecture. We used three evaluation metrics: Precision (P), Recall (R) and F1 score as denoted in Table 1. As shown, 2D Convolution with Pooling, GRU with Capsule and Pooled GRU architectures had the best F1 scores of the six experimented architectures. Therefore, we submitted outputs from those three architectures as our final submissions.

Architecture	P	R	F1
Pooled GRU	0.800	0.789	0.785
Stacked LSTM with Attention	0.788	0.766	0.760
LSTM and GRU with Attention	0.783	0.768	0.762
2D Convolution with Pooling	0.806	0.801	0.800
GRU with Capsule	0.807	0.800	0.798
LSTM with Capsule and Attention	0.776	0.768	0.764

Table 1. Results of the architectures

3.2 Submission Results

This section presents the results of the evaluation of our submission. The evaluation was carried out by the task organisers, and at the time of writing the paper the GOLD standard for the test set is not available. Therefore, we report only the evaluation provided to us by the task organisers which is solely based on F1 score. Our selected architectures 2D Convolution with Pooling, Pooled GRU and GRU with Capsule had F1 scores of 0.818, 0.816 and 0.804, respectively. Our best result 2D Convolution with Pooling ranked seventh in the final results.

4. Error Analysis

The deep learning architecture used in our models is fundamentally based on word-embeddings and their sequential relations.

[5] http://bit.ly/32toTbN

Accordingly, the first step for error analysis was to explore the most informative features on the word-level which were crucial in classification. In order to achieve this, the top 30 vectors for ironic and non-ironic tweets were extracted (see Figure (1) for the most ironic vectors and Figure (2) for the most non-ironic).

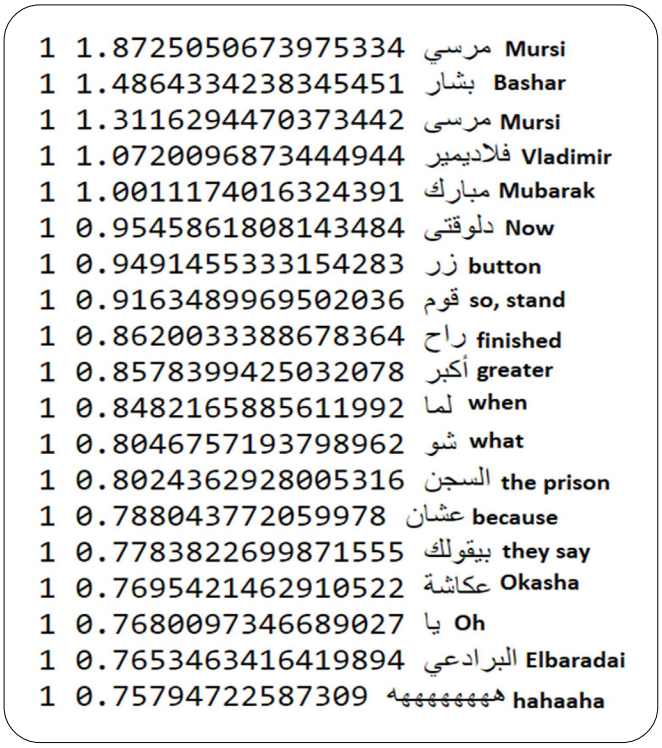

Figure 1. Most Ironic Features Produced by Baseline Model

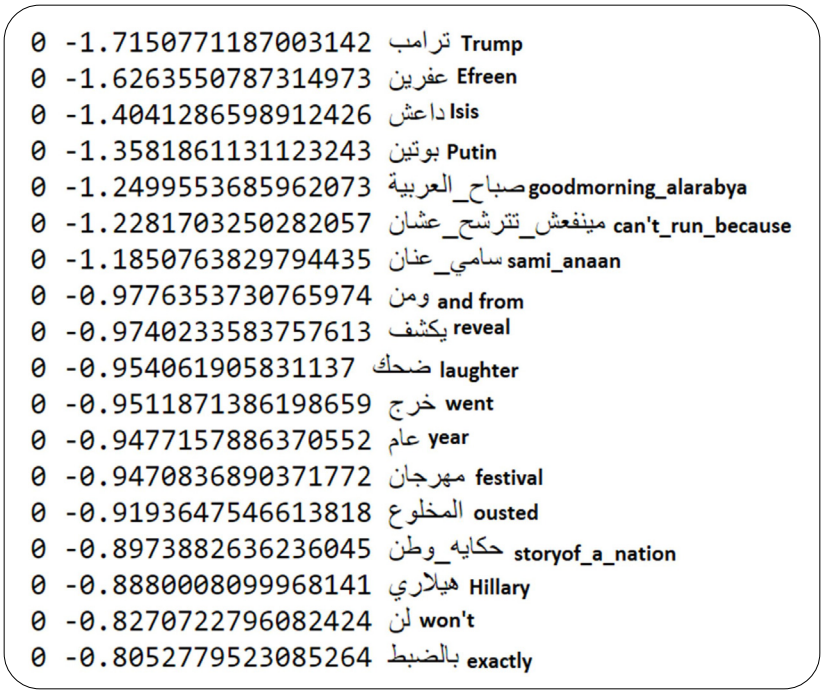

Figure 2. Most Non-Ironic Features Produced by Baseline Model

A number of observations were made on the above lists. First, the most informative tokens for ironic tweets revolve around Middle East political figures who were usually the target of jokes in social platform discussions during the time frame of the compiled data (e.g 'Mursi', former president of Egypt, 'Bashar', president of Syria, 'Mubarak', ousted president of Egypt, 'Okasha', an Egyptian TV anchor famous for propagating hoax news). These are essentially divisive political figures who typically provoke ironic comments on their actions or political stance. Second, elongated words such as 'هههههه' (hahaha) referring to an exaggerated sentiment is irony defining. Third, the most non-ironic features similarly include words referring to political figures or entities such as 'Trump', 'Isis' and 'Putin'. However, it is noticeable that a less formal reference to the political figure is included in the ironic tweet. For example, in the most informative ironic features, reference to 'Vladimir Putin' is informally given by his first name 'Vladimir' whereas his surname 'Putin' becomes the most informative feature in non-ironic tweets. Fourth, the most informative features for non-ironic tweets seem to include particular hashtags which may have been retweeted in the corpus. This is concluded from the use of the underscore, '_', in more than one non-ironic vector (e.g. 'سامي عنان' (Sami Anaan), seems to be a retweeted hashtag concerned with the imprisonment of the Egyptian chief of staff who was imprisoned after announcing his run for the Egyptian presidential elections). These hashtags refer to current news which may be more factual than sarcastic. Finally, 50% of the most ironic tokens belong to different Arabic dialects whereas only 10% of the non-ironic features was dialectical.

The second step in the error analysis was the investigation of the misclassified instances in the test set. Given that access to the official annotation for the test set was not available, one of the authors of the paper, who is an Egyptian Arabic native speaker analysed the automatic classification. The total misclassified tweets was 188 tweets: 133 belong to the false positives, where the tweet is wrongly classified as ironic, and 55 false negatives, where the tweet is mistakenly considered non-ironic. Generally speaking, the false positive instances typically include reference to one of the political figures contained in the most informative ironic features shown in Figure (1) above. For example, the following tweet:

أسماء زوجة الرئيس السوري بشار الأسد حامل بشهرها الخامس

(Asmaa, wife of the Syrian president Bashar, is pregnant in her fifth month) is misclassified as ironic because of the presence of the word 'بشار' (Bashar) which is one of the most informative features for the positive class. Thus, tweets reporting news about one of the gures typically mentioned in ironic tweets are misclassified. It was also noticed that a number of the false positives were wrongly annotated tweets. For example:

يامكشوف عنك الحجاب . . يامكشوف عنك البنطلون # باسم _ يوسف _ ستايل

(Oh you who see through the future .. oh you who see through pants #bassem_yussif_style) is an ironic tweet mistakenly annotated as non-ironic. Interestingly, our model correctly classified it as ironic. Moreover, a number of tweets that were misclassified included slang or dialectical words. For example, is part of a tweet written in dialectical Egyptian Arabic that was misclassified as ironic.

بس بس هتضيعنا انت رايح فين

As for false negatives, it was observed that the ironic tweet typically included an emoticon indicating a positive sentiment (such as laughter or a smile). The preprocessing stage for our model deleted any non-characters and hence emoticons were not captured by the classier. In the future, we plan to investigate whether the use of other features such as emoticons can improve the classification accuracy.

5. Conclusions

In this paper, we have presented our system for identifying irony in Arabic tweets. The system uses minimal preprocessing and relies on word embeddings. We experimented with different neural network architectures in order to determine the most suitable for this task. Going by our evaluation, and the results provided by the task organisers, it is clear that 2D Convolution with Pooling scores highest overall.

While the word embeddings performed well in the task there were a lot of out of vocabulary words. In the future, we would like

to experiment with the performance of contextualised word embeddings, such as ELMo [14] and BERT [4] in irony detection. We have experienced these architectures in the recent Germeval Task 2, 2019 | Shared Task on the Identification of Oensive Language [15]. Therefore, we would like to take the system presented here further, in order to see how it may perform in other languages on similar tasks.

References

[1] Benamara, F., Grouin, C., Karoui, J., Moriceau, V., Robba, I. (2017). Analyse d'opinion et langage guratif dans des tweets : pr–esentation et r–esultats du d–efouille de textes deft2017.

[2] Chung, J., Caglar Gulcehre, Cho, K., Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. CoRR.

[3] Cignarella, A. T., Frenda, S., Basile, V., Bosco, C., Patti, V., Rosso, P. (2018). Overview of the evalita 2018 task on irony detection in italian tweets (ironita). In: EVALITA@CLiC-it.

[4] Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *In*: NAACL-HLT.

[5] 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.

[6] Grice, H. P. (1989). Studies in the Way of Words. Harvard University Press.

[7] Hee, C. V., Lefever, E., Hoste, V. (2018). Semeval-2018 task 3: Irony detection in english tweets. *In*: Proceedings of SemEval@NAACL-HLT.

[8] Hettiarachchi, H., Ranasinghe, T. (2019). Emoji powered capsule network to detect type and target of oensive posts in social media. In: Proceedings of RANLP 2019.

[9] Hinton, G. E., Sabour, S., Frosst, N. (2018). Matrix capsules with EM routing. *In*: Proceedings of ICLR 2018.

[10] Huang, G., Liu, Z.,Weinberger, K. Q. (2017). Densely Connected Convolutional Networks. Proceedings of IEEE CVPR 2017.

[11] Kingma, D. P., Ba, J. (2015). Adam: A Method for Stochastic Optimization. Proceedings of CoRR 2015 (2015) 10 T. Ranasinghe et al.

[12] Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L. E., Brown, D. E. (2019). Text Classification Algorithms: A Survey. *Information* 10 (4).

[13] Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., Joulin, A. (2018). Advances in PreTraining Distributed Word Representations. *In*: Proceedings of LREC 2018.

[14] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L. (2018). Deep contextualized word representations.

[15] Plum, A., Ranashinghe, T., Orasan, C., Mitkov, R. (2019). Rgcl at germeval 2019: Offensive language detection with deep learning. *In*: Proceedings of GermEval at KONVENS 2019.

[16] Plum, A., Ranasinghe, T., Calleja, P., Orasan, C., Mitkov, R. (2019). Rgcl-wlv at semeval- 2019 task 12: Toponym detection. In: Proceedings of SemEval 2019. p. 1297-1301.

[17] Plum, A., Ranasinghe, T., Orasan, C. (2019). Toponym detection in the bio-medical domain: A hybrid approach with deep learning. *In*: Proceedings of RANLP 2019.

[18] Ravaut, M., Gorti, S. (2018). Gradient descent revisited via an adaptive online learning rate. arXiv preprint arXiv:1801.09136.

[19] Scherer, D., Mnuller, A. C., Behnke, S. (2010). Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. *In*: Proceedings of ICANN 2010.

[20] Schuster, M., Paliwal, K. K. (1997). Bidirectional recurrent neural networks. IEEE Trans. *Signal Processing* 45, 2673-2681.

[21] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15, 1929-1958.

[22] Tompson, J., Goroshin, R., Jain, A., LeCun, Y., Bregler, C. Efficient object localization using Convolutional Networks. Proceedings of IEEE CVPR 2015.

[23] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I. (2017). Attention Is All You Need. *In*: Proceedings of NIPS.

[24] Wan, L., Zeiler, M.D., Zhang, S., LeCun, Y., Fergus, R. (2013). Regularization of Neural Networks using DropConnect. *In*: ICML.

[25] Wu, Y., Yang, F., Liu, Y., Zha, X., Yuan, S. (2018). A Comparison of 1-D and 2-D Deep Convolutional Neural Networks in ECG Classification. *In*: Proceedings of IEEE Engineering in Medicine and Biology Society.