Detecting Irony in Arabic Microblogs using Deep Convolutional Neural Networks

Linah Alhaidari, Khaled Alyoubi, Fahd Alotaibi
Department of Computing and Information Technology,
King Abdulaziz University,
Jeddah, 21589, Saudi Arabia

***Abstract*—A considerable amount of research has been developed lately to analyze social media with the intention of understanding and exploiting the available information. Recently, irony has took a significant role in human communication as it has been increasingly used in many social media platforms. In Natural Language Processing (NLP), irony recognition is an important yet difficult problem to solve. It is considered to be a complex linguistic phenomenon in which people means the opposite of what they literally say. Due to its significance, it becomes essential to analyze and detect irony in subjective texts to improve the analysis tools to classify people opinion automatically. This paper explores how deep learning methods can be employed to the detection of irony in Arabic language with the help of Word2vec term representations that converts words to vectors. We applied two different deep learning models; Convolutional Neural Network (CNN) and Bidirectional Long Short-Term Memory (BiLSTM). We tested our frameworks with a manually annotated datasets that was collected using Tweet Scraper. The best result was achieved by the CNN model with an F1 score of 0.87.**

***Keywords*—Verbal irony; natural language processing; machine learning; automatic irony detection**

I. Introduction

Sentiment analysis is known as the extraction and interpretation of opinions expressed in a text written in a natural language on a certain subject [1]. Recently, sentiment analysis and opinion mining became extremely popular due to the increase of social network usage, which led to produce a huge number of texts. Researchers and many organizations became more interested in analyzing such a text type in order to understand human communication better. Hence, irony is a sophisticated form of sentiment expression where people express their opinions in a certain way [2]. Therefore, it has become an important topic in NLP as it flips the polarity of the posts.

According to Cambridge dictionary[1], irony is described as a figure of speech that means the opposite of what people really say in a way of being humorous. For example, " I love going to the dentist!!". Irony can be either situational or verbal. Situational event occurs when naturally expecting something to happen but the opposite take place [3]. While verbal irony is when individuals express words that represent the opposite of what they actually feel and that is the focus of the existing studies in irony detection. Sarcasm is another term that often occur along with Irony. There exists a lack of conformity in the relationships between irony and sarcasm. Some researchers [4] [5] define sarcasm as a type of irony in which it is directed at an individual, with the purpose to mock. While others [6], [7] see them as a distinct phenomenon and consider sarcasm differ from irony in which sarcasm include an element of ridicule that irony has not.

In our regular everyday communication, we meet people who like to use irony in the conversation. In most cases, we can detect irony depending on the tone, context, facial expressions, and the person's character. However, when ironic posts in social media are in question, recognizing the irony becomes more challenging and complex due to its ambiguous nature, the missing intonation of the person who writes the message, restricted length of the words, the informal language, the use of hashtags, and the context is not always clear.

Irony has been studied by many research fields such as psychology [8], linguistics education [9], and computational science. It is used widely as an indirect negation in order to achieve different communication goals in many situations such as criticizing, make fun of people, and manipulate answers to upsetting questions.

In recent years, social media have become a part of people's everyday modern life. It enables them to share their opinions on different topics along with other matters. Therefore, the appearance of irony in social networks such as microblogs has greatly increased. For this reason, one of the primary motivations behind this research is detecting real intention behind posts accurately and understanding how people feel regarding specific matters can be useful for many applications. It can help in correctly identifying security issues such as threatening posts by verifying whether the threat words are literal or not. Also, it can be helpful in distinguishing figurative language devices that are used in the different social media platforms, product reviews, feedback, etc., and recognizing the ironic negative reviews/post that being misinterpreted as positive. In general, any tool that aims to extract the meaning of a post effectively can benefit from such a property.

Existing work on detecting irony in Arabic language have mainly focused on classical machine learning [10], [11], [12]. While recent research on detecting irony in English language, such as [13], [14], and [15], applied neural network approaches. Hence, the importance of this research lies on adopting neural network techniques to improve recognizing irony in Arabic texts. To the best of our knowledge, there is only one experiment [12] on detecting irony in Arabic using a neural model, namely BiLSTM. Therefore, we propose a framework that learns irony using a convolutional neural network (CNN).

[1]https://dictionary.cambridge.org/

In addition, we experimented with Bidirectional Long Short-Term Memory (BiLSTM) as well. The approach we applied outperformed the state of the art. The main contributions of this research can be summarized as follows:

1) A manually annotated ironic Arabic dataset.
2) We believe that this is the first work on using CNN for irony detection in Arabic texts.

This paper is structured as follows: Section 2 presents a brief literature review on irony detection; Section 3 discusses the approach including data generation process, feature extraction, and the proposed method; Section 4 dedicated to the results; lastly, Section 5 concludes the paper.

II. Related Work

The interest in adopting neural networks to detect irony on social media has been increased. Classical methods, e.g. SVM, depend on feature engineering and manually convert texts into feature vectors prior to the classification task. In contrast, neural networks approaches can automatically grasp the representations of input texts with different levels of abstraction and then use the gained knowledge to perform the classification task. Several deep learning-based methods have been reported for the field of automatic irony detection. One of the early work was by Poria et. al. [16] who proposed an architecture based on a pre-trained convolutional neural network to detect sarcasm on balanced and unbalanced datasets. Sentiment, emotion and personality features were extracted and applied to the system. The balanced datasets achieved the highest f1 score of 0.97. Ghosh et al. [17] applied a semantic neural network model to detect sarcasm over social media content. The architectures are composed of two CNN layers followed by two long short-term memory (LSTM) layers and a deep neural network (DNN) layer. The evaluation of the model achieved an F-score of .92. Ilić et. al. [18] proposed a deep neural network model that depends on character-level word representations extracted using the Embeddings from Language Models (ELMo). ELMo is a contextualized representation method that uses vectors extracted from BiLSTM. They tested their system on seven datasets obtained from three different data sources. The results yield 0.87 F-score using Twitter dataset. Authors noted that annotating data manually is necessary in order to improve the performance results. Furthermore, transfer learning approaches, i.e., applying the knowledge of an already trained model to a new task, became very popular in many problems including irony detection in recent research. Potamias et al. [14] proposed a transformer based architecture that builds on the pre-trained RoBERTa model and integrated with a recurrent convolutional neural network (RCNN) that uses non-hand crafted features as they argue that overly trained deep learning approach does not need engineered feature step. They used several benchmark datasets that contain ironic, sarcastic, and figurative expressions. The highest performance of the hybrid neural system achieved f1 score of 0.90 by the sarcastic Riloff's dataset [19]. As for Zhang et al. [20], they focused on finding implicit incongruity without depending on explicit incongruity expressions. They used three transfer learning-based techniques to enhance the attention mechanism of RNNs. The applied attention-based Bi-LSTM achieved higher outcomes on the hashtag-labeled corpus compared to the human-labeled one. The authors discuss that manually-labeled dataset is considerably more difficult than the hashtag-based dataset and that human annotated dataset results in a more accurate prediction for irony in real applications. Gonzale et al. [21] used a Transformer Encoder (TE) architecture on two corpora; English and Spanish languages. They applied two TE models with and without the sine-cosine positional information: TE-Pos and TE-NoPos. As a result, TE-NoPos outperformed the TE-Pos system. They also studied the affect of the transformer architecture's multi-head self-attention processes on the irony detection topic. Wu et al. [13] proposed a framework based on four layers of BiLSTM with three dense layers. they combined three tasks which includes finding the missing irony hashtags, classifying ironic or non-ironic and detecting the irony types. The system is concatenated with the sentiment and sentence embedding features that improved the performance by achieving an f1 score of 0.70 and 0.49. Huang et al. [22] considered three deep learning models; Convolutions Neural Network (CNN), Recurrent Neural Network (RNN), and Attentive RNN. The Attentive RNN outperformed other models by achieving F1 score of 0.89. They discussed how attention mechanism improved the irony detection performance. Khalifa and Hussein [12] implemented an ensemble of 8 models based on biLSTM network with TF-IDF, topic modeling and word and character counts features on an Arabic tweets. The ensemble achieved the F1 score of 0.82. Golazizian et al [15] employed a bidirectional LSTM (BiLSTM) network to detect irony in Persian language. They used emoji prediction to construct a pre-trained model that include an attention layer that improved the performance. They reached an accuracy of 83.1. Ren et al. [23] employed two context-augmented neural network methods on Twitter dataset to recognize sarcastic signs from contextual data. Baruah et al. [24] used BiLSTM and BERT transformer based architecture to detect sarcasm in Twitter texts. They applied historical conversational features such as response only and response with varied number of utterances from the dialogue. The F-score of 0.74 was achieved using the BERT classifier. A recent study by Razali et al. [25] which used CNN to extract lexical and contextual features. They chose FastText as word embedding technique. Authors claim that manually extracted contextual features improves the overall accuracy.

III. Approach

A. Data Generation

A critical challenge for automated irony detection is the availability and quality of a set of ironic examples in order to train a model. This step is performed in three stages shown in Fig. 1.

B. Data Collection

For the ironic data, we initially collected 12700 Arabic tweets using Tweet Scraper [26]. We gathered the data using the following hashtags: #irony, #Sarcasm,
#استهزاء, #سخرية, #أيروني, #تريقه, #تهكم, #بتريق.

As for the non-ironic datasets, we decided to use a random sample of an existed Arabic sentiment corpus [27].

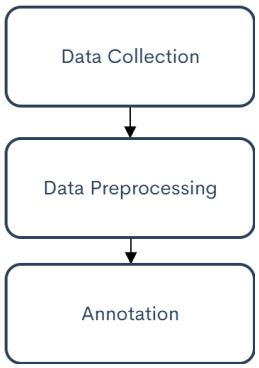

Fig. 1. Data Generation Stages.

C. Preprocessing of Data

This step is considered as an essential task in sentiment analysis. The goal of preprocessing is to remove corrupt and irrelevant information from raw text. The stream of data we gathered from Twitter is noisy as it has lots of retweets, social interaction, etc. Hence, tweet preprocessing is essential in order to eliminate noisy data that are not useful to the task. We performed some basic text preprocessing listed below:

1) Remove metadata: time and ID.
2) Remove usernames, mentions, and RTT.
3) Remove all numbers.
4) Remove redundant texts.
5) Remove punctuation.
6) Remove foreign characters.
7) Remove emojis.
8) Remove hashtags.
9) Remove repeated characters.
10) Remove diacritics (shaddah, fatha, tanwin, damma, kasra, and sukoon)
11) Remove tweets that contain URLs or images as their existence could be required to identify any figurative language present in the texts.
12) Replace emoticon with its corresponding meaning.
13) Use NLTK stopwords
14) Normalize some Arabic letters:

- إأآا to ا
- ي to ى
- ؤ ئ to ء
- گ to ك
- ة to ه

D. Data Annotation

To deal with the problem of [28] that tweets include hashtags are biased and noisy, we manually labelled Both datasets by two Arabic speakers. However, it is considered a time-consuming process. Three main guidelines were given to the annotators:

1- The tweet is ironic if the literal word is opposite to the intended.

2-The tweet is ironic if it was written in a context other than the common context of communication with the aim of negatively mocking sayings, ideas, beliefs or objects.

3- Otherwise, the text is not ironic.

After including the tweets that both annotators agreed on as ironic and non-ironic, the total amount of data are 5620 ironic and 5620 non ironic tweets (Table I). This qualitative analysis revealed that despite having irony-related hashtags, many tweets turned to be not ironic, which shows the significance of manual corpus annotations.

TABLE I. DATASET SUMMARY

Label	Number of Tweets
Ironic	5620
Non-ironic	5620
	Total: 11240

E. Features Extraction

We adopted a simple feature extraction technique, which is (pre-trained) word embedding. Word embedding is a form of terms representation for text analysis in which When two words have the same meaning, they are represented in a vector space by similar vectors that are near together. Otherwise, if the terms have different meanings, then the real-valued vectors are far from each other.

To construct such an embedding, Word2Vec [29] is one of the popular techniques. There exist two different methods to learn the embedding: Skip Gram and Continuous Bag of Words (CBOW) (Fig. 2). The CBOW model takes in context words as an input and try to predict the target word equivalent to the context. on the contrary, skip gram tries to predict the surrounding words given a target word which is the opposite of what the CBOW model does. For the purpose of this work, we applied the pre-trained Arabic word embedding model AraVec 3.0 [30], which provides various pre-trained Arabic words. It has a total of 12 distinct word embedding models extracted from various Arabic content domains, which are Twitter, World Wide Web (WWW) pages, and Wikipedia. Moreover, we used Twitter SkipGram 300D-embeddings as according to [29], Skip Gram works good with small dataset and is able to represent uncommon terms. For our neural network model, the embedding vectors are utilized to instate the weights of the embedding layer. Then, it is linked with the remainder of the layers in the system.

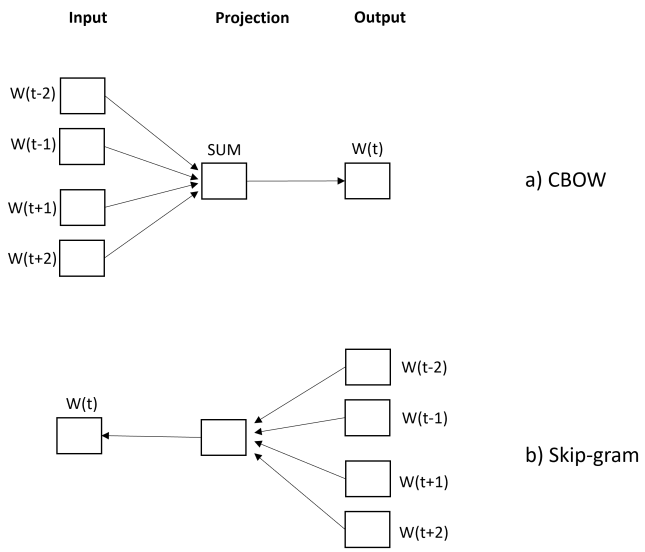

Fig. 2. CBOW and Skip-gram.

F. Methods

1) Convolutional Neural Network (CNN): We adopted a basic CNN architecture, similar to [31], to extract local features from each utterance found in the training dataset by using word vectors that are captured from the pre-trained Word2Vec model. Fig. 3 represents the various layers that are applied to perform the convolution function on the dataset. The Keras library was used to implement the embedding layer. We configured three parameters, which are:

- input_dim: describes the size of the vocabulary in the data text; it is composed as 30462.
- output_dim: represents the dimension of the dense embedding; it is configured as 300.
- input_length: identifies the length of the maximum document; it is composed as 41.

Once we obtained the suitable word vectors from the skip-gram model, the neural network takes the output features as inputs and applies two convolutional layers to the features to learn context information of the words. The convolution operation is described as formula (1).

$$C_i = f(W.X + b) \tag{1}$$

where C is the convolution output that is created from a window of words X, W is the convolution matrix, f is the activation function and b is a bias term.

The two convolution layers have kernel sizes of 4 and 5 to look at sequences of the word embeddings. Each layer consists of 100 filtered outputs. Furthermore, a ReLu activation was applied to the outputs of the convolutional layers followed by a maxpooling layer that takes the highest element from the rectified feature. It minimizes the dimensionality of the feature map and helps capture the key feature. The maxpooling operation (P), as expressed in Equation 3, is done for feature selection that nominates the significant features suitable to various hidden layers.

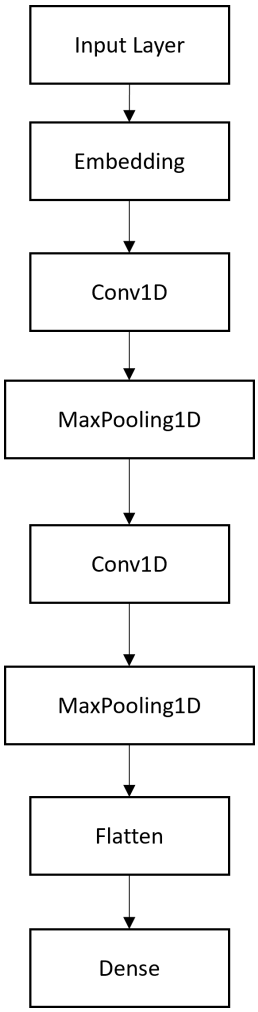

Fig. 3. The Architecture of our CNN.

$$C = (c_1, c_2 ... c_n) \tag{2}$$

$$P_i = max(C) \tag{3}$$

$$S(x) = \frac{1}{1 + e^{-x}} \tag{4}$$

Next, the output is sent to a layer that flattens the matrices. Finally, the result is a fully connected layer having the outputs of the sigmoid function (Equation 4), that determines whether a sentence is ironic or not, along with the binary cross entropy loss function, which is a good option for binary classification. The model was trained using "Adam" learning rate method.

2) Bidirectional Long Short-Term Memory (BiLSTM): RNN is excellent for sequence learning, but it struggles with long-range dependency due to exploding gradient. On the other hand, LSTM has the capability to learn those long-range dependencies. Bi-LSTM is a variant of LSTM that contains two LSTMs to capture the input information in a forward and backwards directions. In other words, it allows in any point in time to preserve information from both past and future. Given a document,

$$D = (x_1, x_2, .., x_n)$$

Bi-LSTM model results with a set of hidden state vectors h_t for the document sequence. Furthermore, Bi-LSTM combines the forward LSTM (Equation 5) and the backward LSTM (Equation 6).

$$\overrightarrow{h}_t = LSTM(x_t, \overrightarrow{h}_{t-1}) \quad (5)$$

$$\overleftarrow{h}_t = LSTM(x_t, \overleftarrow{h}_{t+1}) \quad (6)$$

The architecture of our model consists of one Bi-LSTM layer followed by a dropout layer to occupy most of the parameters by ignoring neurons during the training of particular set of units, which is chosen randomly. Next, the model flattens the input data into a 1-dimensional array for inputting to the dense layers. Lastly, two fully connected layers were applied where the first layer contains 32 neurons and the second fully connected layer, which also represents the output layer, contains 2 neurons with the ReLU and Sigmoid activation functions used respectively.

G. Hyperparameters Setting

There are various important hyperparameters in our systems, and we tune their values using the development corpus. The tuned values of our models hyperparameters are summarized in Table II and Table III.

TABLE II. TUNED VALUES OF THE CNN HYPERPARAMETERS

Hyperparameter	Value
Kernel size	4 and 5
Number of filters	100
Maxpooling size	2
Learning rate (adam optimizer)	0.0001
Loss function	Binary crossentropy
Batch size	100

TABLE III. TUNED VALUES OF THE BILSTM HYPERPARAMETERS

Hyperparameter	Value
BiLSTM number of layers	1
Number of hidden units	128
Dropout rate	0.2
Optimizer (adam)	0.0001
Loss function	Binary crossentropy

Using the proper learning rate and setting the correct learning value is critical for enhancing the weights and offsets of the neural model. The low value may cause long training time while its high value could lead to network instability. After several experiments, we set the learning value to 0.0001.

Batch size is another key hyperparameter in neural networks. It defines the number of training examples a neural network can process before resetting the model internal parameters. If the value of batch size too low, then it could slow down the training process. On the contrary, if the value is too high, it may needs more memory and decrease the generalization capability. Therefore, we started with a small batch value and then increased the value to 64 and 100 in order to use less memory and achieve a durable system.

Neural networks model has the ability of learning complex connections between their inputs and outputs. Yet, some of these relations could be affected by the sampling noise. Hence, the connections will be shown during the training phase but will not occur in the real test data. This problem may cause overfitting and that decreases the classifier's accurate prediction and lead to a poor performance. Therefore, we applied two methods for the purpose of avoiding overfitting in our proposed model. First, we employed early stopping function which refer to stopping the training iterations before the learner passes the point where the model's capability to generalize can decrease as it starts to over-train, thus overfit the training data. Second, we used dropout which is a regularization method of neural networks developed by [32]. It refers to randomly ignore units along with their relationships from the neural network during training process. This stops units from co-adapting too much.

IV. RESULTS

In this section, we explore the performances of our deep learning models trained with word2vec (skipgram). Table IV shows the performance results of our proposed models. We used the accuracy, recall, precision, and F1 score as performance metrics. These scores are defined as follows [33]:

$$Precision = \frac{TP}{TP + FP} \quad (7)$$

$$Recall = \frac{TP}{TP + FN} \quad (8)$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \quad (9)$$

$$Accuracy = \frac{TP + TF}{TP + TF + FP + FN} \quad (10)$$

Where TP referred to True Positives, FP is the number of False Positives, FN is False Negatives, and TN is True Negatives. The predicted values are described as positive and negative, while the real values are described as true and false. Recall measure how much out of all the positives were predicted correctly, whereas precision measures how many are actually positive. F-score helps in measuring both recall and precision at the same time. Finally, accuracy measures the overall classifier correctness.

At first, we tried to apply one CNN layer which resulted with 0.86 F1 score. However, the two-layer CNNs slightly improved the performance of the classifier and achieved F1 score of 0.87. We also experimented with BiLSTM and reached 0.86 F1 score, while the results achieved by [12] using BiLSTM model is 0.83. Our scores outperformed the classical and neural approaches used in irony detection in the Arabic language shared task [34]. In spite of the fact that performance of various work existed vary since different datasets have

TABLE IV. RESULTS OF OUR CNN MODELS AGAINST OTHER MODELS

Model	Accuracy	Precision	Recall	F1
CNN- One Layer	0.86	0.90	0.83	0.86
CNN- Two Layers	0.87	0.90	0.84	**0.87**
BiLSTM	0.87	0.88	0.85	0.86
BiLSTM [12]	-	-	-	0.83

different data distributions, our experiment suggests that the manual annotation improves the learning and prediction tasks. Also, fine tuning the pre-trained vectors and the model play a role in improving the overall results.

Based on the literature [14] [22] [17] [24] [25], CNN's are useful at finding local and position-invariant features whereas BiLSTM are good when classification is specified by a long range semantic relationships and dependency. In our experiment, CNN worked better than BiLSTM in detecting irony since such a sentiment is commonly determined by some key phrases. The output of each convolution layer will let off when a pattern is detected regardless of their position. Changing the size of the kernels and concatenating the received outputs allowed to discover patterns of multiples sizes (4 and 5).

V. CONCLUSION AND FUTURE WORK

Irony detection research has grown remarkably in recent years. In this paper, we presented an approach based on pre-trained word embedding called AraVec, to address the problem of detecting irony in Arabic tweets. We adopted a basic CNN architecture which found to be very effective for irony detection in Arabic language. The experiment reveals that the method we used performs well with two convolutional layers and even outperform an existing technique. The CNN model achieved F1 score of 0.87. We also experimented with BiLSTM and the results reached 0.86 F1 score. For future work, we plan to extend the extraction of meaningful features, such as sentiment and contextual clues, in order to find the optimal features. Additionally, we can experiment with combined CNNs and RNNs to gain the characteristics of both methods for enhancing the classification performance.

REFERENCES

[1] K. S. Sabra, R. Zantout, M. A. E. Abed, and L. Hamandi, "Sentiment analysis: Arabic sentiment lexicons," *Sensors Networks Smart and Emerging Technologies (SENSET)*, pp. 1–4, 2017.

[2] P. Rosso, F. Rangel, I. H. Farías, L. Cagnina, and W. Zaghouani, "A survey on author profiling, deception, and irony detection for the arabic language," *wiley*, 2018.

[3] J. Lucariello, "Situational irony: A concept of events gone awry," *Journal of Experimental Psychology*, vol. 2, pp. 129–145, 1994.

[4] R. Filik, A. Turcan, C. RalphNearman, and A. Pitiot, "What is the difference between irony and sarcasm? an fmri study," *cortex*, vol. 115, pp. 112–122, 2019.

[5] A. Bowes and A. Katz, "When sarcasm stings," *Discourse Processes*, p. 215 — 236, 2011.

[6] C. Lee and A. Katz, "The differential role of ridicule in sarcasm and irony." *Metaphor and Symbol*, vol. 13, pp. 1–15, 1998.

[7] E. Sulis, D. Farías, P. Rosso, V. Patti, and G. Ruffo, "Figurative messages and affect in twitter: Differences between irony, sarcasm, and not," *Knowl.-based Syst*, vol. 108, pp. 132–143, 2016.

[8] R. Filik, E. Brightman, C. Gathercole, and H. Leuthold, "The emotional impact of verbal irony: Eye-tracking evidence for a two-stage process," *Journal of Memory and Language*, vol. 93, pp. 193–202, 2017.

[9] N. Banasik-Jemielniak, A. Bosacki, S. Mitrowska, D. Wyrebek, K. Wisiecka, N. Copeland, L. Wieland, L. Popovic, J. Piper, and A. Siemieniuk, ""wonderful! we've just missed the bus." – parental use of irony and children's irony comprehension." *PLOS ONE*, vol. 15(2), 2020.

[10] J. Karoui, F. Zitoune, and V. Moriceau, "Soukhria:towards an irony detection system for arabic in social media," *3rd International Conference on Arabic Computational Linguistics*, 2017.

[11] H. A. Nayel, W. Medhat, and M. Rashad, "Benha@idat: Improving irony detection in arabic tweets using ensemble approach," *Proceedings of the IDAT@FIRE2019*, 2019.

[12] M. Khalifa and N. Hussein, ""ensemble learning for irony detection in arabic tweets," *Proceedings of the IDAT@FIRE2019*, 2019.

[13] C. Wu, F. Wu, S. Wu, J. Liu, Z. Yuan, and Y. Huang, "Irony detection with densely connected lstm and multi-task learning," *Proceedings of The 12th International Workshop on Semantic Evaluation*, 2018.

[14] R. A. Potamias, G. Siolas, and A. Stafylopatis, "A transformer-based approach to irony and sarcasm detection," *Neural Computing and Applications*, vol. 32, no. 23, p. 17309–17320, 2020.

[15] P. Golazizian, S. Behnam, A. Seyed, A. Zahra, O. Momenzadeh, and R. Fahmi, "Irony detection in persian language: A transfer learning approach using emoji prediction," *LREC*, 2020.

[16] S. Poria, E. Cambria, and A. Gelbukh, "Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis," *Proceedings conference on empirical methods in naturallanguage processing*, p. 2539–2544, 2015.

[17] A. Ghosh and T. Veale, "Fracking sarcasm using neural network," *NAACL-HLT*, 2016.

[18] S. Ilic, E. Marrese-Taylor, J. A. Balazs, and Y. Matsuo, "Deep contextualized word representations for detecting sarcasm and irony," *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, 2018.

[19] E. Riloff, A. Qadir, P. Surve, L. DeSilva, N. Gilbert, and R. Huang, "Sarcasm as contrast between a positive sentiment and negative situation," *EMNLP 2013 conference on empirical methods in natural language processing*, p. 704–714, 2013.

[20] S. Zhang, X. Zhang, J. Chan, and P. Rosso, "Irony detection via sentiment-based transfer learning," *Inform. Proc. and Manag*, vol. 56, p. 1633–1644, 2019.

[21] J. Gonzalez, L. Hurtado, and F. Pla, "Transformer based contextualization of pre-trained word embeddings for irony detection in twitter," *Information Processing and Management*, vol. 57, 2020.

[22] Y. Huang, H. Huang, and H. Chen, "Irony detection with attentive recurrent neural networks," *Advances in Information Retrieval, Springer*, p. 534–540, 2017.

[23] Y. Ren, D. Ji, and H. Ren, "Context-augmented convolutional neural networks for twitter sarcasm detection," *Neurocomputing*, vol. 308, pp. 1–7, 2018.

[24] A. Baruah, K. B. Das, and K. F. nd Dey, "Context-aware sarcasm detection using bert," *Proceeding 2nd Workshop Figurative Language Process*, pp. 83–87, 2020.

[25] M. S. Razali, A. Halin, L. Ye, S. Doraisamy, and N. M. Norowi, "Sarcasm detection using deep learning with contextual features," *IEEE*, vol. 9, pp. 68 609–68 618, 2021.

[26] TweetScraper, "Jonbakerfish/tweetscraper: Tweetscraper is a simple crawler/spider for twitter search without using api," 2020. [Online]. Available: https://github.com/jonbakerfish/TweetScraper

[27] M. Saad, "Arabic sentiment twitter corpus," 2019. [Online]. Available: https://www.kaggle.com/mksaad/arabic-sentiment-twitter-corpus/data?select=test_Arabic_tweets_positive_20190413.tsv

[28] D. Davidov, O. Tsur, and A. Rappoport, "Semi-supervised recognition of sarcastic sentences in twitter and amazon," *In Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pp. 107–116, 2010.

[29] T. Mikolov, S. I., K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[30] A. Soliman, K. Eissa, and S. El-Beltagy, "Aravec: A set of arabic word embedding models for use in arabic nlp," *Procedia Computer Science*, vol. 117, pp. 256–265, 2017.

[31] Y. kim, "Convolutional neural networks for sentence classification," *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2014.

[32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Machine Learning Research*, vol. 15, pp. 1958–7929, 2014.

[33] M. Hossin and M. Sulaiman, "A review on evaluation mertrics for data classification evaluations," *International Journal of Data Mining and Knowledge Management Process (IJDKP)*, vol. 5, 2015.

[34] B. Ghanem, J. Karoui, F. Benamara, V. Moriceau, and P. Rosso, "Overview of the track on irony detection in arabic tweets," *the 11th Forum for Information Retrieval Evaluation, FIRE '19*, pp. 10–13, 2019.