Ensemble Learning for Irony Detection in Arabic Tweets

Muhammad Khalifa[1], Noura Hussein[2]
[1] Computer Science Department, Cairo University, Egypt
{muhammad.e.khalifa@gmail.com}
[2] Computer Science Department, Benha University, Egypt
{nourahussein193@gmail.com}

ABSTRACT: *In this paper, we describe and show the results of our 3 systems submitted for the Irony Detection in Arabic Tweets Shared Task at the Forum for Information Retrieval (FIRE 2019). We employ ensemble learning for this task through 3 different types of ensemble models, namely classical, deep and hybrid (that combines both). We extract types of features from the tweets including TF-IDF word n-gram features, topic modeling features, bag-of-words and sentiment features. Our submitted systems scored the top 3 places with our best system achieving 84.4 F1 points on the test set.*

Keywords: Irony Detection, Ensemble Learning, Text Classification

Received: 18 November 2019, Revised 29 February 2020, Accepted 8 March 2020

DOI: 10.6025/jdp/2020/10/2/57-61

1. Introduction

Irony is defined as a trope whose meaning is different from what is literally enunciated [10]. Failing to detect irony leads to the misinterpretation of the message intended causing degradation of the performance of Natural Language Understanding (NLU) systems. However, irony detection can be challenging since it requires world knowledge and a more complex understanding of the context [10]. Detecting irony efficiently can help with various Natural Language Processing (NLP) tasks such as sentiment analysis, hate speech detection, fake news detection, and online harassment detection.

Taking sentiment analysis as an example, [8] shows how the presence of irony can negatively impact sentiment classification performance. Remarkably, while the sentiment classification performance in regular tweets could reach up to an F1 score of 71, the performance on ironic tweets reached only a maximum of 57. Thus, accurate sentiment classification requires accurate irony detection so that the sentiment classifier can acknowledge that the intended sentiment is contrary to the literal one.

Text classification of Arabic tweets is typically faced by a few challenges. First, there is the difficulty of dealing with Arabic itself, which is a morphologically rich language with characteristics that make dealing with it a challenge [3]. Second, Arabic tweets are

usually replete with unstandardized, dialectal and transliterated words (ex. "hello" becomes "هالو"). This leads to what is known as the out-of-vocabulary (OOV) problem where the learning system may fail to generalize due to a large number of unseen words during training. Moreover and in addition to dialectal Arabic, tweets can include code-switching between Arabic and other languages such as English or French. This contributes more to the difficulty of the task by introducing more unknown words or phrases whose understanding could be essential to the task of irony detection.

In this paper, we describe our submitted systems to the shared task of Irony Detection in Arabic Tweets. Given a tweet, our system should automatically decide whether it is ironic or not. We employ ensemble learning using both classical and deep models and our results show that classical ensembles outperform deep ensembles on this task. Moreover and prior to classification, we extract various features from tweets including Term Frequency-Inverse Document Frequency (TF-IDF) word n-gram features, topic modeling features and sentiment-based features. We conduct experiments to assess the importance of each category of features and our results show that TF-IDF features, Bag-of-words representation, and count-based features are most significant for irony detection.

2. Systems Description

2.1 Preprocessing

Before feature extraction, we apply various preprocessing to the tweets in the dataset. Our preprocessing stage comprises mainly of text normalization such as replacing all instances of 'ى' with 'ي' and 'ة' with 'ه'. Besides, all instances of *Hamza* are replaced with 'ء' to account for incorrect word spellings. We also normalize all instances of repeated characters such that "لووووول", for instance, becomes "لول" and strip all diacritics (if any).

2.2 Feature Extraction

Given a tweet, we extract five different types of features:

• **Word n-gram TF-IDF:** We extract TF-IDF-weighted features of word *n* grams where $n \in [1, 6]$. We use only the top frequent 50K *n*- grams.

• **Topic Modeling Features:** We run Latent Dirichlet Allocation (LDA) [1] on the training set setting the number of topics $k = 20$ and using both unigrams and bigrams. Then, each tweet *t* is represented using a *k*- dimensional vector V^t such that $V^t_d = P(topic = d \mid text = t)$.

• **Sentiment Features:** Given a tweet, we average the sentiment scores of its constituent words. The sentiment scores used are extracted from the Arabic sentiment lexicon proposed in [11].

• **Pretrained Word Vectors** We compute a Bag-of-words (BOW) representation of each tweet by averaging the word vectors of its constituent words. We use the pretrained 300-dimensional Twitter-CBOW word vectors provided by [9].

• **Count-based Features:** These features include word and character counts, word density (number of characters per word), punctuation count, stopwords count and the standard deviation of the word length per tweet.

Table 1 shows F1 obtained using each of the performance of XGBoost using different categories of the features. Apparently, TF-IDF and word vectors give the best performance on the development set.

Features	**F1**
Sentiment	59.0
Topic Modeling	60.0
Count Features	67.0
TF-IDF word *n*-gram	81.1
Word2vec BOW	83.6
All	**85.6**

Table 1. F1 score on the development set using each features category. Model used is XGBoost

2.3 Classical Ensemble

For our first submission, we use an ensemble of 3 models. Namely, Gradient Boosting [4], Random Forest [2] and Multilayer Perceptron (MLP) [5]. This ensemble is trained on all of the previously discussed features. To compute the final predictions from all the ensembles, we use Soft Voting where we sum the probabilities of each class across models and the class with the highest probability sum is chosen.

2.4 Word-level Bi-LSTM Ensemble

Our second submission is an ensemble model based on a word-level bidirectional LSTM (bi-LSTM) network [7]. We augment the bi-LSTM with a subset of the aforementioned features. These features are processed through a feed-forward network and the output is concatenated with the output of last hidden state of the bi-LSTM. This is passed through another feed-forward network and then projected into a Sigmoid unit for classification. See Figure 1. The subset of the additional features used includes TF-IDF, topic modeling and count-based features. We use an ensemble of 8 models for our final submission.

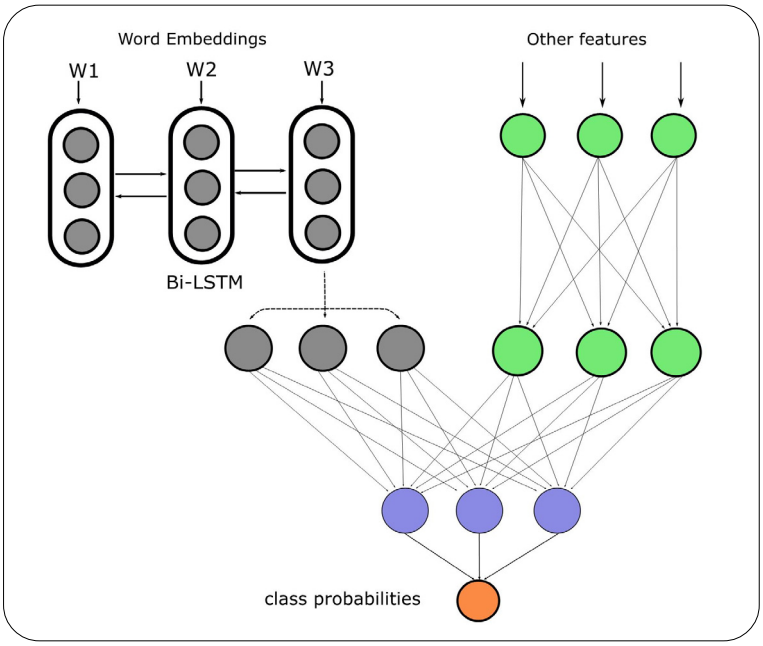

Figure 1. The Bidirectional LSTM-based model used in our second submission

2.5 Hybrid Ensemble

We combine both our first and second systems into our third submission which is an ensemble of Gradient Boosting, Random Forest, Multi-Layer Perceptron and 8 bi-LSTMs.

3. Experiments and Results

3.1 Dataset

We use the training dataset provided by the Irony Detection in Arabic Tweets shared task [6]. The dataset is a collection of 4024 tweets with only two classes: ironic and non-ironic. Both classes contain 2091 and 1933 samples, respectively. We do not use any additional training data. The test set, on the other hand, contains 1006 tweets.

3.2 Experimental Setup

For hyperparameter selection, we use a randomly sampled 20% of the training set as a development set. However, before final submission, we train each system on the whole training set. Table 2 shows the hyperparameter settings for all models used.

3.3 Results

Table 3 shows the results on both development and test sets using both single and ensemble models. Noticeably, the single XGBoost model performs best on the development set compared to all other single models with an F1 score of 85.6.

Model	Hyperparameters
Random Forest MLP	n_trees=60 n_layers=3
XGBoost	layer_sizes=(128, 64, 32) n_trees=200 max_depth=10 gamma=0.5 embeddings_dim=300 embeddings_init=random_normal
Bi-LSTM	lstm_n_layers=1 lstm_hidden_units=128 feedforward1_units=64 feedforward2_units=128 feedforward_activation='relu' dropout=0.6

Table 2. Hyperparameters for all models used

By combining XGBoost with Random Forest and MLP, the classical ensemble achieves the best F1 scores of 86.5 and 84.4 on both development and test sets, respectively. The hybrid ensemble achieved the next best scores of 86.2 and 83.3 and the Bi-LSTM ensemble comes last with 84.6 and 82.2. Since the dataset size is relatively small, it makes sense for classical models to outperform deep models.

Model	Dev	Test
Random Forest -	80.4	-
XGBoost	85.6	-
MLP	80.8	-
RF + MLP + XGBoost (ensemble)	**86.5**	**84.4**
Bi-LSTM	82.6	-
8 Bi-LSTMs (ensemble)	84.6	82.8
Hybrid ensemble	86.2	83.3

Table 3. Results of our three systems on development and test sets

4. Conclusion

In this paper, we described our three submitted systems to the Irony Detection in Arabic Tweets Shared task at the Forum for Information Retrieval Evaluation (FIRE 2019). Our submitted systems are classical, deep and hybrid ensembles that operate of a set of features extracted from each tweet. The extracted features include TF-IDF word n-gram features, bag-of-words representation, sentiment based features and topic modeling features. Our results show the classical ensemble outperforming both deep and hybrid ensembles with 84.4 F1 points on the test set and achieving the first place on the task leaderboard.

References

[1] Blei, D. M., Ng, A. Y., Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3 (January),

993–1022.

[2] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1) 5–32.

[3] Farghaly, A., Shaalan, K. (2009). Arabic natural language processing: Challenges and solutions. ACM Transactions on Asian Language Information Processing (TALIP), 8 (4), 14.

[4] Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, p. 1189–1232.

[5] Gardner, M. W., Dorling, S. (1998). Artificial neural networks (the multilayer perceptron)— a review of applications in the atmospheric sciences. *Atmospheric Environment*, 32 (14-15), 2627–2636.

[6] Ghanem, B., Karoui, J., Benamara, F., Moriceau, V., Rosso, P. (2019). Idat@fire2019: 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.

[7] Hochreiter, S., Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9 (8), 1735–1780.

[8] Nakov, P., Ritter, A., Rosenthal, S., Sebastiani, F., Stoyanov, V. (2016). Semeval-2016 task 4: Sentiment analysis in twitter. *In*: Proceedings of the 10th international workshop on semantic evaluation (semeval-2016). p. 1–18.

[9] Soliman, A. B., Eissa, K., El-Beltagy, S. R. (2017). Aravec: A set of arabic word embedding models for use in arabic nlp. *Procedia Computer Science*, 117, 256–265.

[10] Van Hee, C., Lefever, E., Hoste, V. (2018). Semeval-2018 task 3: Irony detection in English tweets. *In*: Proceedings of The 12th International Workshop on Semantic Evaluation. p. 39–50.

[11] Vo, D.T., Zhang, Y. (2016). Don't count, predict! an automatic approach to learning sentiment lexicons for short text. *In*: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). p. 219–224.