A contextual word embedding for Arabic sarcasm detection with random forests

Hazem E. Elagbry, Shimaa Attia, Ahmed Abdel-Rahman, Ahmed Abdel-Ate, Sandra Girgis
Faculty of Computer Science and Artificial Intelligence
Helwan University, Egypt
{hazememad, Shimaa_20170268, Ahmed_20170044, Ahmed_20170056, totajo1999}@fci.helwan.edu.eg

Abstract

Sarcasm detection is of great importance in understanding people's true sentiments and opinions. Many online feedbacks, reviews, social media comments, etc. are sarcastic. Several researches have already been done in this field, but most researchers studied the English sarcasm analysis compared to the researches are done in Arabic sarcasm analysis because of the Arabic language challenges. In this paper, we propose a new approach for improving Arabic sarcasm detection. Our approach is using data augmentation, contextual word embedding and random forests model to get the best results. Our accuracy in the shared task on sarcasm and sentiment detection in Arabic was 0.5189 for F1-sarcastic as the official metric using the shared dataset ArSarcasm-V2 (Abu Farha, et al., 2021).

1 Introduction

Sarcasm detection is the process of classifying whether the text (i.e., a full document, a single sentence, a word, etc.) is sarcastic or not. Sarcastic writing is a common way for people to share their sentiment in many online texts (i.e., social media, reviews, posts, comments, etc.). This linguistic phenomenon has much significance for tasks such as sentiment analysis and opinion mining. Sarcasm detection has received much research attention in other languages especially in the English language, but the Arabic language still lags behind. A few efforts have done in Arabic sarcasm detection such as the works of (Al-Ghadhban, et al., 2017, May) and the shared task held by (Ghanem, et al., 2019). There have been some recent efforts to build standard datasets for this task such as (Farha & Magdy, 2020) but still not enough to get good results. A lot of challenges we face when dealing with Arabic sarcasm analysis such as 1) lack of datasets, 2) Imbalanced classes, 3) lack of pre-preprocessing tools (i.e., autocorrect, stemmer, etc.) for Arabic language, 4) diversity of dialects which affects the results. Most recent methods and approaches (i.e., machine-learning or deep learning) use one of the following two intermediate representations[1] for language modeling.

Firstly, and the very common intermediate representation which proved its efficiency in many experiments is TF-IDF (term frequency-inverse document frequency). TF-IDF is a very good way to start your initial representation because it has a lot of advantages such as 1) Ease of computing, 2) extracting the most descriptive terms in a document and 3) you can easily compute the similarity using it. But with all of this, there is a very big disadvantage such as it cannot capture semantic in the given text which affects the algorithms effectiveness.

Secondly, is another very good intermediate representation that proved its effectiveness called word embedding. Fixed word embeddings solve the problem of capturing the semantic of a word that we faced in TF-IDF intermediate representation by providing a fixed embedding to every word which was too useful (examples: word2vec, fastText). The limitations of the fixed word embeddings are that they cannot provide us with the information about words meaning in different

[1] Note that there are more than these ways in the literature but we have mentioned the commonly used techniques.

Proceedings of the Sixth Arabic Natural Language Processing Workshop, pages 340–344
Kyiv, Ukraine (Virtual), April 19, 2021.

contexts when setting a fixed embedding to every word.

Finally, to solve this problem there is a very good language modeling technique which is called contextual word embedding. Contextual word embeddings set a different embedding to the same word according to its meaning in the context which handles the problem of diversity of a single word meaning. In literature, several researches focus on enhancing the Arabic natural language understanding such as (Antoun, et al., 2020; Abdul-Mageed, et al., 2020).

In this paper, we attempt to extract contextual word embedding using Ara-Bert (Antoun, et al., 2020) then feed it to an ensemble learning method called random forests classifier. We provide the git hub link of the code used to extract the contextual embeddings[2].

2 Data

The data we have used to get our results in this shared task are: Ar-Sarcasm-V2 (Abu Farha, et al., 2021), emojis dataset, stop words dataset, and unseen dataset from the shared task on sarcasm and sentiment detection in Arabic as a test set to test our results.

2.1 Data Description

The training dataset was 12548 tweets that is given from the shared task on sarcasm and sentiment detection in Arabic (Abu Farha, et al., 2021) which has the following columns: 1) Sarcasm: this column value is binary (i.e., for every single tweet True if the tweet is sarcastic and False if it isn't sarcastic), 2) Sentiment: this column contains 3 classes POS, NEG, NEU for positive, negative, neutral respectively, 3) Dialect: this column contains 5 different dialects one of these 5 values was assigned to every tweet the values are: MSA, Gulf, Egypt, Levant, Magreb.

Emojis dataset is publicly available from Kaggle[3]. It contains 4159 different emoji with the following columns: 1) Emoji, 2) Name of the emoji in English, 3) Group of the emoji, 4) Sup-group of the emoji, 5) Code-point that represents the emoji. We have used the emoji dataset because they are expressive and all of us use them every day in our tweets, Facebook posts, and here in our dataset.

Stop words is a publicly available dataset which is consisting of two lists of Arabic stop words developed while working on this paper: (El-Khir, 2017). Finally, the Unseen dataset contains 3000 tweets with a column that contains the dialect of each tweet.

We have used an 80/20 split to prepare the training and validation sets in addition to using augmentation techniques which we are going to talk about in section 2.2.

2.2 Data Augmentation

In our dataset, there was a big class imbalance, because the dataset contains 10380 non-sarcastic tweets against 2168 sarcastic tweets with a total number of 12548 tweets. According to this situation we performed different data augmentation techniques inspired by (Liu, et al., 2014, June) to solve this issue such as:

1- Oversampling: we have done this by duplicating a random number of tweets to reach 6018 sarcastic tweets.

2- Undersampling: after this, we removed part of the non-sarcastic class samples to change number of samples from 10380 non-sarcastic to 6018 non-sarcastic tweets.

This made the two classes have the same number of samples with a total number of 12036 tweets.

2.3 Data Preprocessing

We have built a function to detect patterns and clean our text using regular expression package in python. We have removed the following patterns: Users (any name after @ symbol), English letters, Numbers (English or Arabic), Special characters / Punctuations, replace each emoji with its Arabic name without duplicating, Leading spaces (before and after the tweet), dropping empty tweets.

Because our emoji dataset was in English language not in Arabic and our shared task focus in the Arabic language, we have translated the dataset to the Arabic language.

[2] arabert/AraBert_output_Embeddings_PyTorch.ipynb at master · aub-mind/arabert · GitHub

[3] Link of the dataset: Full Emoji Database | Kaggle

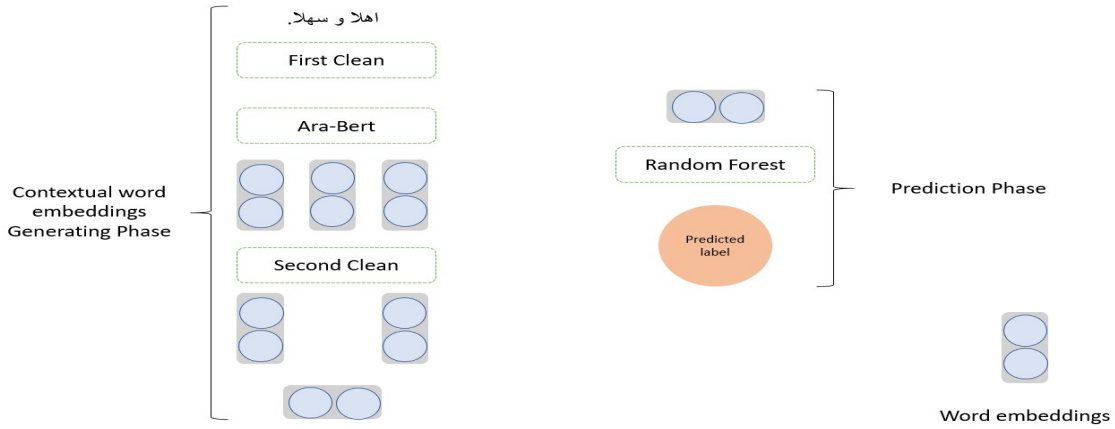

Figure 1: Illustration of our random forest model with the generation of contextual embeddings.

3 System

In our approach, we have used the pre-trained algorithm Ara-Bert for extracting the contextual word embeddings that we have talked about in section 1. BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers (Devlin, et al., 2018). According to BERT's state-of-the-art results, it is a big deal to include it in our classification task, but there is a problem which is BERT alone is not enough and there are limits to it for getting state-of-the-art results. So, and because of Ara-Bert powerful in natural language understanding and its huge corpus which is trained with. We have used Ara-Bert for language modeling (i.e., extracting contextual embeddings) with its base-configuration. $BERT_{BASE}$ consists of (L=12, H=768, A=12, Total Parameters=110M). In this notation, we denote the number of layers (i.e., Transformer blocks) as L, the hidden size as H, and the number of self-attention heads as A.

Till now we did not remove stop words or performing any stemming for our data. We did this removes and stemming after extracting the word embeddings because contextual word embeddings rely on extracting different embeddings for the same word in different contexts, and even ال in Arabic language can affect our understanding of the context. According to this assumption, we have implemented a second text cleaning/preprocessing after extracting our contextual word embeddings.

We have made another interesting mechanism for doing something very close to the stemming which we are going to discuss here. Ara-BERT's input is not a feature but is a sequence of words that is tokenized with a pre-trained Ara-BERT tokenizer. The Ara-BERT tokenizer is created with a word piece model. In this tokenization every word has one of the following forms: 1) the word still the same which means that this word is in BERT's vocabulary, 2) the word will come with a # sign which means that this word is part of a bigger word (i.e., the original word is not in its vocabulary so, BERT split the word into a group of subwords with #), 3) the word with a + sign means that this word is a prefix or suffix to another word (i.e., الكتاب the ال here is a prefix so it will appear like كتاب + ال). According to this form, we have used it to perform stemming especially with dialects which are very hard to stem by removing every word that has the + sign. After this second cleaning, we get the average of word embeddings to get our sentence embeddings with vector size of 768 feature.

After second cleaning and to avoid empty tweets in train or test set, we have removed every empty tweet after saving its index and assign to this tweet a False label which means it is not sarcastic by default.

Finally, we feed this sentence embedding to random forests (which is an ensemble learning method for classification, regression, and other

Hyperparameter	**Value**
Number of estimators	100
Max depth	None
criterion	Gini

Table 1: This table shows us part of random forest hyperparameters

tasks) with its default hyperparameters configuration using scikit-learn package in python see Table [1].

We provide the code [4] of our approach for publicly use to push the Arabic classification tasks to another state-of-the-art accuracy.

4 Results

In this section, we are going to discuss the key results of our approach by sharing with you our evaluation metrics with its key results in section 4.1, then the official metrics of the shared task on sarcasm and sentiment detection in Arabic with its key results in section 4.2.

4.1 Training Results

This section contains our evaluation before submitting the answers and see the final results (i.e., it is not the official metrics). Tables [2] shows our results using contextual word embeddings as our intermediate representation. We have split the Ar-SarcasmV2 (Abu Farha, et al., 2021) data after data augmentation into 80% as a training set and 20% as a test set. This shows us that the random forests using contextual word embeddings scored the with 93% F1-score for each class (i.e., True or False).

	Precision	Recall	F1-Score
False	0.96	0.90	0.93
True	0.90	0.96	0.93

Accuracy			0.93
Macro AVG	0.93	0.93	0.93
Weighted AVG	0.93	0.93	0.93

Table 2: This table shows us the results of using random forests with contextual word embeddings

As we see here the accuracy of 0.93 is a very high accuracy which will affect our judgment of the real performance of our model, and the main reason for this kind of error (i.e., fake high accuracy) is that we have augmented our dataset before splitting it. Like we have said in section 2.2 the oversampling technique that we have used was by duplicating the minor class (i.e., sarcastic class). Thus, the test set was containing data which is most of it duplicated tweets from the training set and this is why these fake results are here (i.e., our model does not perform this well). So, be aware and don't augment the data before splitting it.

4.2 Official Results

This section shares the shared task evaluation metrics using our submitted model which is random forests ensemble learning with contextual word embeddings. Table [3] shows our results for F1-Sarcastic (the official metric), Accuracy, Macro F1, precision, and recall in the unseen test dataset and we have got the following results:

F1-Sarcastic	Accuracy	Macro-F1
0.5189	0.7533	0.6765

Precision	Recall
0.6858	0.6700

Table 3: this table shares our submitted model official metrics in the shared task

For clarification purposes (i.e., showing the big difference between TF-IDF and contextual word embeddings). Table [4] shows the results of using TF-IDF in test data which are very bad in comparison with word embeddings in Table [3].

F1-Sarcastic	Accuracy	Macro-F1
0.41	0.73	0.62

Precision	Recall
0.51	0.34

Table 4: this table shares the results when using the TF-IDF as an intermediate representation

5 Discussion

Our observations and experiments show that in Arabic sarcasm detection 2 main challenges affect the accuracy, validation, and our improvement such as 1) the dataset was too small, 2) dealing with dialects was making this harder (EX. stemming was a very challenging problem when dealing with dialects)

To our knowledge, a very small number of researches are studying the effectiveness of

[4] Link of our colab: https://colab.research.google.com/drive/1GgVpke8wY9aOiiVeYni6VGAcxqFxPEeQ?usp=sharing

Contextual word embeddings in Arabic which are very helpful in dealing with different dialects and unseen data, and the most interesting example for such effectiveness is Ara-Bert which reaches a very good accuracy in dialects while he was trained in MSA corpus.

Dealing with incorrectly entered data especially in social media was incredible and very challenging (i.e., if a user wrote ' لامن حقك ' the tokenization phase will treat this 3 words as a 2 words ' لامن ' and ' حقك ' but if he wrote ' لا من حقك ' with a space between every word the tokenization phase will treat this as 3 words not two 'لا', 'من' and 'حقك') there always a problem when dealing with every step in pre-processing with dialects.

For the above reasons, our next steps are like the following: 1) Focusing more on preprocessing problems (i.e., produce an algorithm for stemming dialects, autocorrect Arabic words), 2) Looking for more data to get better accuracy and validation, 3) we have tried a machine learning approach which is not very good in capturing the amount of information that the contextual word embeddings transfer (i.e., it is like a bottleneck). So, one of our important next steps is dealing with the problem using deep learning models.

6 Conclusion

Our task allowed us to explore a lot of challenges with natural language processing in Arabic. We believe that Arabic natural language processing applications such as sarcasm detection, sentiment analysis, POS, etc. has a lot of areas to be investigated and developed. In our paper, we tried to overcome the problem of dialects and semantic diversity for every word in different contexts by generating a contextual word embedding using Ara-Bert which achieved very good results just using random forests ensemble learning with its initial scikit-learn configuration.

We hope that problems such as POS, Autocorrect, and such important problems take a more interest to solve it because we believe that such problems are a big deal when we talk about enhancing the accuracy of other tasks like: sentiment or sarcasm analysis.

7 References

Liu, P., Chen, W., Ou, G., Wang, T., Yang, D. and Lei, K., 2014, June. Sarcasm detection in social media based on imbalanced classification. In *International Conference on Web-Age Information Management* (pp. 459-471). Springer, Cham.

Elrazzaz, M., Elbassuoni, S., Shaban, K. and Helwe, C., 2017, July. Methodical evaluation of arabic word embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (pp. 454-458).

Farha, I.A. and Magdy, W., 2020, May. From Arabic Sentiment Analysis to Sarcasm Detection: The ArSarcasm Dataset. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection* (pp. 32-39).

Joshi, A., Tripathi, V., Patel, K., Bhattacharyya, P. and Carman, M., 2016. Are word embedding-based features useful for sarcasm detection?. *arXiv preprint arXiv:1610.00883*.

Antoun, W., Baly, F. and Hajj, H., 2020. Arabert: Transformer-based model for arabic language understanding. *arXiv preprint arXiv:2003.00104*.

Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Al-Ghadhban, D., Alnkhilan, E., Tatwany, L. and Alrazgan, M., 2017, May. Arabic sarcasm detection in Twitter. In *2017 International Conference on Engineering & MIS (ICEMIS)* (pp. 1-7). IEEE.

Ghanem, B., Karoui, J., Benamara, F., Moriceau, V. and Rosso, P., 2019, December. Idat at fire2019: Overview of the track on irony detection in arabic tweets. In *Proceedings of the 11th Forum for Information Retrieval Evaluation* (pp. 10-13).

El-Khair, I.A., 2017. Effects of stop words elimination for Arabic information retrieval: a comparative study. *arXiv preprint arXiv:1702.01925*.

Abu Farha, I., Zaghouani, W., and Magdy, W. 2021. Overview of the WANLP 2021 Shared Task on Sarcasm and Sentiment Detection in Arabic. In Proceedings of the Sixth Arabic Natural Language Processing Workshop.

Abdul-Mageed, M., Elmadany, A. and Nagoudi, E.M.B., 2020. ARBERT & MARBERT: Deep Bidirectional Transformers for Arabic. arXiv preprint arXiv:2101.01785.