Sarcasm and Sentiment Detection in Arabic: Investigating the Interest of Character-level Features

Dhaou Ghoul
STIH Lab, Sorbonne University, Paris
dhaou.ghoul@gmail.com

Gaël Lejeune
STIH Lab, Sorbonne University, Paris
gael.lejeune@paris-sorbonne.fr

Abstract

We present three methods developed for the Shared Task on Sarcasm and Sentiment Detection in Arabic. We present a baseline that uses character n-gram features. We also propose two more sophisticated methods: a recurrent neural network with a word level representation and an ensemble classifier relying on word and character-level features. We chose to present results from an ensemble classifier but it was not very successful as compared to the best systems : 22th/37 on sarcasm detection and 15th/22 on sentiment detection. It finally appeared that our baseline could have easily been tuned and achieve much better results.

1 Introduction

Arabic language is one of the most widely spoken language in the world, currently considered as the fifth language (Chung, 2008) with more than 330 million speakers. It is the official language in more than 22 countries and is therefore an important language to handle for NLP systems. Its written form is commonly referred as Literary Arabic and divided into two categories: Classical Arabic and Modern Standard Arabic (MSA). But Arabic has in fact many variants and research effort have been made on tasks like Arabic Dialects Identification(Abdul-Mageed et al., 2020). More fine-grained tasks have been tackled by the NLP community and this paper describes our participation to such a task : the Shared Task on Sarcasm and Sentiment Detection in Arabic(Abu Farha et al., 2021). This shared task focuses on analysing tweets and identifying the sentiment (negative, positive or neutral) and whether a tweet is sarcastic or not. More precisely, the task are defined as follows:

- **Subtask 1:** (Sarcasm Detection): Identifying whether a tweet is sarcastic or not, this is a binary classification task.
- **Subtask 2:** (Sentiment Analysis): Identifying the sentiment of a tweet and assigning one of three labels (Positive, Negative, Neutral), multiclass classification task.

The paper is organized as follows: Section 2, describes the dataset and Section 3 is devoted to the methods we developed. Section 4 details our results, future directions are given in Section 5.

2 Data

The dataset for both tasks is a combination of the ArSarcasm (Abu Farha and Magdy, 2020) and DAICT (Abbes et al., 2020), we used a 30% subset of the training data to get a dev set. Statistics on the resulting data are shown in Table 1.

Datasets	Train	Dev	Test
# lines	9,549	3,000	3,000
# words	144,158	45,047	51,478
# characters	1,822,547	570,592	552,871

Table 1: Size of the Train, Dev and Test sets

The distribution of labels (sarcasm and sentiment label) is rather imbalanced. In subtask-1, sarcastic tweets represent only 17% of the data (1666 sarcastic VS 7883 non-sarcastic). In subtask-2, positive tweets only represent 18% of the data and negative tweets represent no more than 37%. Hereafter we present some examples of labelled tweets.

Sarcastic : بيقولك أخر مره فاز فيها الزمالك علي الأهلي كانت أيام ما كان الحكم بيصفر بصباعه (They say that Zamalak's last victory against Al-Ahly happened when the referees had to use their fingers to whistle).

Positive : طلعنا من حلب الشرقية بسلام والحمدلله (We came out of East Aleppo in peace, thank God)

Proceedings of the Sixth Arabic Natural Language Processing Workshop, pages 329–333
Kyiv, Ukraine (Virtual), April 19, 2021.

Negative:لبنان: ''سخرية'' من تعيين رجل وزيرا لشؤون المرأة. (Lebanon: "mocking" a male nominated as women's affairs minister)

Neutral :في حدا نازل على حلب في ورق ضروري لازم ابعته و شكرا. (Is there anyone going to Aleppo, I have important papers to send there. Thank you)

3 Methods developed for both tasks

We did not developed a specific method for each task, we rather tried to find features that could be useful for both tasks. We first present a simple baseline (Section 3.1) that shows the interest of character-level features. Afterwards we develop more sophisticated methods: a simple Deep Learning approach (Section 3.2) and an ensemble classifier taking advantage of both character-level and word-level features (Section 3.3).

3.1 Word-based or character-based models ?

The purpose of this method is to see how much simple baselines can be competitive for such tasks. In particular, it seemed important to see how much important is tokenization in NLP pipelines. We compared word-level representations and character-level representations with various classifiers (Multinomial Naive Bayes, Decision trees, Random Forest, Logistic regressions and SVM). It appeared that, except for the MNB classifier, word-based models only outperforms character-based models when the n-gram size is too small (using only 1-grams for instance) or way too big (using only N-grams with $N > 8$ for instance). It is not surprising since, without lemmatization, words are just a subset of all the character n-grams of a text. This observation is coherent with previous work showing the interest of character-level models for various NLP tasks (Nakov and Tiedemann, 2012; Kuru et al., 2016; Buscaldi et al., 2018).

Due to the restricted space we will just exhibit the results for the best classifier which was Logistic Regression. Figure 1 exhibits the influence of N-gram size on the results for both tasks. We can make two comments out of this figure : (i) unigrams are useful for classification even if they are not the best features by themselves and (ii) there is a plateau when the maximal size is set to 5, afterwards the results do not improve much (and even drop for the sarcasm detection task). This result leaded us to choose n-grams from 1 to 5 for our ensemble method (Section 3.3). This method was supposed to act as a baseline but finally, using tf-idf and weighting the classes made this baseline even better that our two other methods.

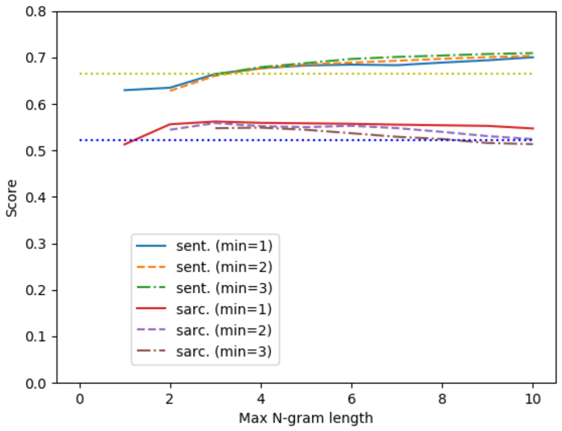

Figure 1: Results on subtask 1 (sarc., lower curves) and subtask2 (sent., upper curves) wrt. minimal size (min) and maximal size (abscissa) of character n-grams, the horizontal lines represent the best baseline with word uni-grams

3.2 A deep Learning approach

To build this system we used the ARAVEC pretrained embeddings proposed by *Soliman et al.* ARAVEC is composed of three models trained trained with Word2Vec skip-gram and CBOW (Mikolov et al., 2013) on three different types of textual data in Arabic: tweets, web pages and Wikipedia articles (Soliman et al., 2017). We used the 100-dimensional Twitter N-Grams model since it seemed to better fit to this task.

The network is composed as follows: the input layer, an embedding layer, two LSTM layers and two Dense layers. To prevent over-fitting we add a dropout layer after each LSTM layer and the first dense layer. We use the pre-mentioned ARAVEC embeddings to enrich the representation. The final output is passed into one hidden layer and followed by a softmax output layer. There have been 10 epochs for the training.

3.3 Voting Ensemble Classifier with word and character-level features

We use the method developed for the 2020 NADI challenge (Ghoul and Lejeune, 2020). To build and train the model, we used the FEATUREUNION in SCIKIT-LEARN (Pedregosa et al., 2011) which allows to combine easily different n-gram representations at the word level and the character level as shown in Figure 2. To train this model,we concatenate three vectors with the following features (weigthed with TF-IDF): word n-grams (1 to 5-grams), character n-grams (1 to 4-grams) and character n-grams inside words (1 to 5-grams).

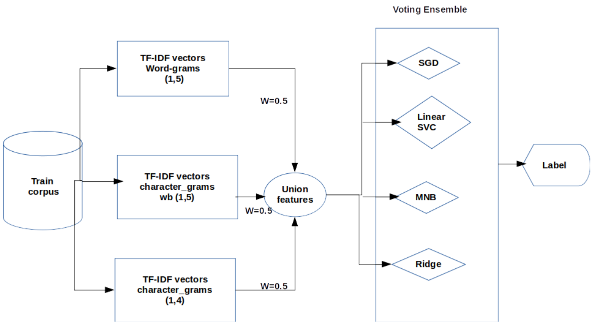
Voting Ensemble
TF-IDF vectors Word-grams (1,5)
W=0.5
Train corpus
TF-IDF vectors character_grams wb (1,5)
W=0.5
Union features
TF-IDF vectors character_grams (1,4)
W=0.5
SGD
Linear SVC
MNB
Ridge
Label

Figure 2: Model of the Ensemble Classifier

In this method, we use a set of simple classifiers to build an ensemble voting classifier that uses predicted class labels for majority rule voting. This ensemble is a combination of the following classifiers (`pen` stands for `penalty`):

- SGDClassifier ($\alpha = 10^{-5}$, `pen`='l2')
- LinearSVC (`pen`='l2', Tolerance =10^{-3}
- Multinomial Naive Bayes ($\alpha = 10^{-2}$)
- Ridge Classifier ($\alpha = 1$)

4 Results and Discussion

4.1 Subtask1: Sarcasm Detection

The results obtained by our models on the dev and test set are presented in Table 2. We chose to focus on the ensemble classifier since it showed a better accuracy on both subtask-1 and subtask-2. Figure 3 shows the result of this ensemble classifier on the dev set as a confusion matrix. We can see that this system has not been able to handle the imbalance in the dataset and had a strong tendency to yield False negatives (363). The number of False Positives (96) is much lower, that can lead to the conclusion that this system has been unable to find appropriate features for lowering silence. Combining multiple classifiers is promising but it seems that weighting the classes, as we finally did for the character-level method would have been more efficient.

It seems interesting to check for discrepancies in annotations. An important number of tweets contains words that belong to the sarcastic lexicon, for instance : هههههههههههه (laughing). The train/dev data contains 135 tweets with this word. Among these tweets, there are 32 tweets were not annotated as sarcastic. Some of them seem to have a wrong annotation, see for example:

- هههههههههههههههههههههههههههههههلا بس اضحك ع الي يقول ديبالا خليفة ميسي (hahaha. Non, I'm laughing on the guy who said Dybala will be the next Messi).
- سامي عنان والا كوفي عنان ؟؟ أعتقد إن مرتضي لــه يـــد في الموضوع ده هههههههههههههههه. (Sami Anan or Kofi Annan ?? I think Morteza is involved in this matter, Hahahaha)

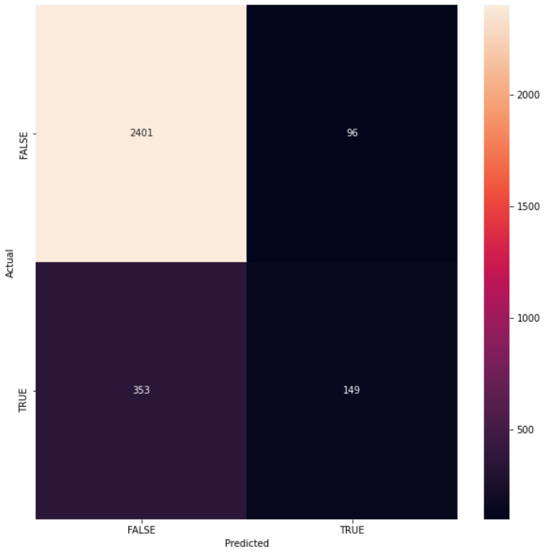

Figure 3: Confusion Matrix on the sarcasm dev set for the ensemble method

4.2 Subtask2: Sentiment Analysis

For this subtask, we use the exact same three methods. Table 3 shows that according to macro-F the best model is the voting Ensemble Classifier with an F-score (POS and NEG) of 65.06%. Surprisingly, the character-level method performed much better on the test than on the dev set and achieved better resultts than the ensemble method on the test set.

Method	**Trained on**	**Tested on**	**Macro P.**	**Macro R.**	**Macro F.**	**Acc.**	**F1 Sarc.**
Voting Ensemble	Train Set	Dev Set	**73.65**	62.78	65.48	**84.93**	39.60
	Train+Dev	Test Set	**71.16**	**61.94**	**63.13**	**76.30**	**41.09**
Deep Learning	Train Set	Dev Set	68.50	62.93	64.76	83.19	27.49
	Train+Dev	Test Set	72.41	58.80	58.87	75.77	32.50
Character Level	Train Set	Dev Set	66.39	**72.93**	**68.19**	78.80	49.81
	Train+Dev	Test Set	66.39	69.76	66.73	70.30	**55.83**

Table 2: Results of our three methods for Subtask1 (Sarcasm) in different training configurations (in blue the best scores, in bold the results of the submitted system, the main metric is in red)

Method	**Trained on**	**Tested on**	**Macro P.**	**Macro R.**	**Macro F.**	**Acc.**	**F. PN**
Voting Ensemble	Train Set	Dev Set	**71.27**	**65.79**	**67.26**	**71.19**	63.37
	Train+Dev	Test Set	**58.99**	**57.10**	**57.84**	**64.73**	**65.06**
Deep Learning	Train Set	Dev Set	68.40	60.54	62.23	67.06	58.15
	Train+Dev	Test Set	55.72	54.09	54.45	59.77	59.27
Character Level	Train Set	Dev Set	64.39	64.91	64.63	67.65	63.78
	Train+Dev	Test Set	59.40	55.60	56.85	62.80	**71.47**

Table 3: Results of our three methods for Subtask2 (Sentiment) in different training configurations (in blue the best scores, in bold the results of the submitted system, the main metric is in red)

The confusion matrix on the dev set is presented in Figure 4 shows that positive tweets were harder to predict with only 230 True Positives, 90 False Positives and 306 False Negatives. Recall has been issue in all the experiments we performed.

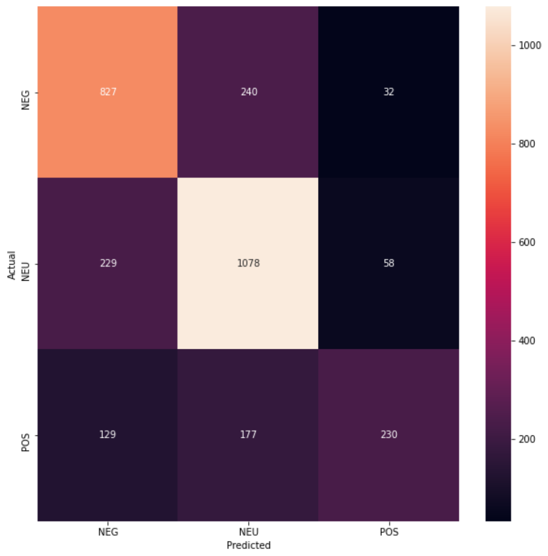

Figure 4: Confusion Matrix on the Sentiment dev set for the ensemble method

	Precision	Recall	F1-score
NEG	69.95	74.98	72.38
NEU	72.03	79.11	75.40
POS	71.83	43.28	54.02
Macro	71.27	65.79	67.26
Micro	71.23	71.19	70.47

Table 4: Classification report for the Voting Classifier on the Dev set for Subtask2 (acc. : 71.19%)

5 Conclusion

In this paper, we described three systems developed for the Shared Task on Sarcasm and Sentiment Detection in Arabic. We tried to show the interest of combining word-level and character-level features. In order to ease the choice of the type of character n-grams used as features we developed a baseline with a Logistic Regression classifier using character n-grams. It helped us to find an appropriate combination of features to produce the ensemble method whose results were submitted to this shared task. It appeared that adding tf-idf and class weighting made this baseline better than the ensemble method, showing maybe the interest of digging into character-level features.

References

Ines Abbes, Wajdi Zaghouani, Omaima El-Hardlo, and Faten Ashour. 2020. DAICT: A dialectal Arabic irony corpus extracted from Twitter. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 6265–6271, Marseille, France. European Language Resources Association.

Muhammad Abdul-Mageed, Chiyu Zhang, Houda Bouamor, and Nizar Habash. 2020. NADI 2020: The first nuanced Arabic dialect identification shared task. In *Proceedings of the Fifth Arabic Natural Language Processing Workshop*, pages 97–110, Barcelona, Spain (Online). Association for Computational Linguistics.

Ibrahim Abu Farha and Walid Magdy. 2020. 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*, pages 32–39, Marseille, France. European Language Resource Association.

Ibrahim Abu Farha, Wajdi Zaghouani, and Walid Magdy. 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*.

Davide Buscaldi, Joseph Le Roux, and Gaël Lejeune. 2018. Modèles en Caractères pour la Détection de Polarité dans les Tweets. In *Atelier DEFT 2018*, Rennes, France.

Wingyan Chung. 2008. Web searching in a multilingual world. *Commun. ACM*, 51(5):32–40.

Dhaou Ghoul and Gaël Lejeune. 2020. Voting Classifier vs Deep learning method in Arabic Dialect Identification. In *: Proceedings of the Fifth Arabic Natural Language Processing Workshop, COLING 2020*, Barcelone, Spain.

Onur Kuru, Ozan Arkan Can, and Deniz Yuret. 2016. CharNER: Character-level named entity recognition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 911–921, Osaka, Japan. The COLING 2016 Organizing Committee.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

Preslav Nakov and Jörg Tiedemann. 2012. Combining word-level and character-level models for machine translation between closely-related languages. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 301–305, Jeju Island, Korea. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Abu Bakr Soliman, Kareem Eissa, and Samhaa R. El-Beltagy. 2017. Aravec: A set of arabic word embedding models for use in arabic nlp. *Procedia Computer Science*, 117:256 – 265. Arabic Computational Linguistics.