QMUL-SDS @ SardiStance2020: Leveraging Network Interactions to Boost Performance on Stance Detection using Knowledge Graphs

Rabab Alkhalifa[1,2] and **Arkaitz Zubiaga**[1]

[1]Queen Mary University of London, United Kingdom
[2]Imam Abdulrahman bin Faisal University, Saudi Arabia

Abstract

This paper presents our submission to the SardiStance 2020 shared task, describing the architecture used for Task A and Task B. While our submission for Task A did not exceed the baseline, retraining our model using all the training tweets, showed promising results leading to (f-avg 0.601) using bidirectional LSTM with BERT multilingual embedding for Task A. For our submission for Task B, we ranked 6th (f-avg 0.709). With further investigation, our best experimented settings increased performance from (f-avg 0.573) to (f-avg 0.733) with same architecture and parameter settings and after only incorporating social interaction features- highlighting the impact of social interaction on the model's performance.

1 Introduction

Framed as a classification task, the stance detection consists in determining if a textual utterance expresses a supportive, opposing or neutral viewpoint with respect to a target or topic (Küçük and Can, 2020). Research in stance detection has largely been limited to analysis of single utterances in social media. Furthering this research, the SardiStance 2020 shared task (Cignarella et al., 2020) focuses on incorporating contextual knowledge around utterances, including metadata from author profiles and network interactions. The task included two subtasks, one solely focused on the textual content of social media posts for automatically determining their stance, whereas the other allowed incorporating additional features available through profiles and interactions. This paper describes and analyses our participation in the SardiStance 2020 shared task, which was held as part of the EVALITA (Basile et al., 2020) campaign and focused on detecting stance expressed in tweets associated with the Sardines movement.

2 Related Work

In social media, **classical features** can be extracted by using *stylistic signals* from text such as bag of n-grams, char-grams, part-of-speech labels, and lemmas (Sobhani et al., 2019), *structural signals* such as hashtags, mentions, uppercase characters, punctuation marks, and the length of the tweet (Wojatzki et al., 2018; Sun et al., 2016), and *pragmatic signals* related to author's profile (Graells-Garrido et al., 2020). With modern deep learning models, there is shift towards **contextualised representations** using word vector representation algorithms, either by having personalised language models trained on task specific language or as a pre-trained language model offered after training using complex architecture and billions of documents. Using **deep learning layers** as automated feature engineering methods can be implemented to train the model afterwards. In (Augenstein et al., 2016), they utilized Bidirectional Conditional Encoding using LSTM achieving state-of-the-art results on stance detection task. Recently, there is a resurgence of research in incorporating **network homophily** (Lai et al., 2017) to represent social interactions within a network. Moreover, **Knowledge graphs** (Xu et al., 2019) can in turn represent these complex network relationships (e.g. authors friendships) as simple embedded vectors sampled considering the nodes and weighted edges within the network complexity structure.

3 Definition of the Tasks

The stance detection task has been defined in previous work as consisting in determining the

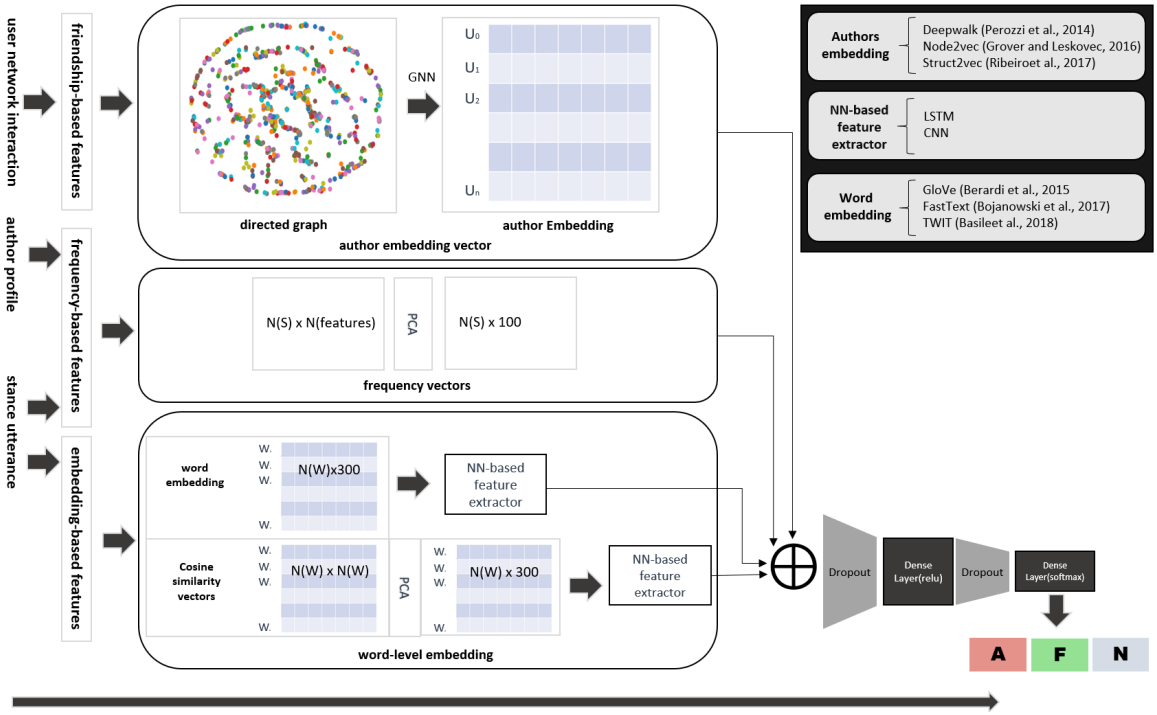

Figure 1: Our framework for investigating different combinations of features. For a network interaction graph, we generate user embeddings, using variations of graph neural network (GNN) embedding methods, namely deep-walk, struct2node and node2vec, and then concatenate author's vector with its corresponding utterance features for each stance. We also extract two types of text embedding representations for each utterance, embedding-based features, namely word embedding vectors and cosine similarity vectors, using different models including variations of CNN and bidirectional models. Further, the results of these two feature extraction methods are concatenated for the final classification step. We also consider the standard methods that extract frequency-based representations from author profiles and stance utterances including unigrams and Tfidf vectors. All these four features where combined and fed into the drop out and dense layers, to finally generate the final label using a softmax activation function. Though, we deactivate some of these four sources of features and alter the frequency-based vector by excluding some features, changing the embedding source and reducing the dimensionality for highly dimensional vectors (e.g. frequency-based features and cosine similarity vectors) using PCA.

viewpoint of an utterance with respect to a target topic (Küçük and Can, 2020), while others define it as that consisting in determining an author's viewpoint with respect to the veracity of a rumour, usually referred to as rumour stance classification (Zubiaga et al., 2018). SardiStance focuses on the former, and is split into two subtasks: Textual Stance Detection (Task A) and Contextual Stance Detection (Task B) (Cignarella et al., 2020). Baselines are provided for Task A using SVM+unigrams as (f-avg. 0.578), and for Task B as (f-avg. 0.628) (Lai et al., 2020).

4 Experimental Settings

Frequency-based features: These represent frequency vectors including unigram, punctuation and hashtags provided by (Cignarella et al., 2020). Further, we include TFiDF vectors.

Embedding-based features: ***word embedding*** Italian Wikipedia Embedding (Berardi et al., 2015) trained using GloVe [1], Fasttext with (Bojanowski et al., 2017) [2] trained using skip-gram model and with 300 dimensions, and TWITA embedding (Basile et al., 2018). For TWITA, two versions of the same tweets were generated. One preprocessing words where each vector has 100 dimensions, provided by (Cignarella et al., 2020)[3] and referred to as TWITA100. The other

[1] https://github.com/MartinoMensio/it_vectors_wiki_spacy

[2] https://fasttext.cc/docs/en/pretrained-vectors.html

[3] https://github.com/mirkolai/

one trained by us without any preprocessing and each vector has 300 dimensions, referred to as TWITA300. We also experimented with multilingual BERT in Task A [4] (Devlin et al., 2019).

Cosine similarity vectors which was introduced previously in (Eger and Mehler, 2016) to encode the word meaning within the embedding space. In our work, we used TWITA300 to train the similarity vectors of all the words in the training set.

Network-based features: ***Encoding users graph.*** To represent user interactions as nodes and edges, we used a counting scalar value and added one if each of the following relationships exists: friendships, retweets, quotes and replies, e.g. if all of them exist then the edge weight between two accounts is four. We calculated all the accounts provided and generate a directed complex graph conditioned by the existence of friendship, resulting in 669,745 nodes, 2,871,791 edges with an average in-degree of 4.2879 and average out-degree of 4.2879.

Generating GNN Embeddings. Taking as input the encoded network relationships, GNN embeddings use different sampling techniques to represent every node as a vector. To extract these vectors, we experiment with different graph neural network models, namely struct2vec (Ribeiro et al., 2017), deepwalk (Perozzi et al., 2014) and node2vec (Grover and Leskovec, 2016).

NeuralNetwork-based features As illustrated in Figure 1, we have different deep learning models to extract features separately for both word embedding and similarity vectors matrices. In our work, we experiment with Convectional Neural Network (CNN) models and Long short-term memory (LSTM) models. Variations of **CNN** models where applied to NLP downstream tasks as feature extraction methods for text classification. In our work, we used two variations of CNN. In one model, we used a CNN as a one-head *1D-CNN* with kernel size of 5 allowing the model to extract features with 5-grams vectors using 32 filters. Followed by a max pooling layer with pool size of 2 then flattened layer. In another model, we used a CNN as a multi-headed *2D-CNN* with 1, 2, 3, 5 grams filter sizes, initialising the kernel weights with a Rectified Linear Unit (ReLU) activation function and normal distribution weights. Followed by a max pooling layer with different pooling sizes taken as one columns pooling filter with the maximum text length excluding few grams sizes. For the **LSTM**, we used two variants. One is a simple *bidirectional LSTM* of 64 units followed by concatenations of max pooling and average pooling layers, and *attention bidirectional LSTM* proposed by (Yang et al., 2016) using 64 units followed by 128 units then attention layers[5].

Feature Reduction. We experiment with different reduction length: 50, 100 and 150. Then. we set our PCA reduction to 100 as it showed best performance on evolution set.

Sentence Cleaning. We set the cleaning function to match the preprocessing function by (Cignarella et al., 2020) to generate TWITA100.

We used four final layers to receive the features and concatenate them (see Figure 1). In all of the experiments, our dropout layer set to 0.2, followed by a dense layer with rule activation function and another dropout layer of 0.2. Finally, a probability vector of the three classes is generated. To determine the correct class, we choose the one class with the highest probability.

5 Results

In this section, we discuss the results of our systems submitted to the two tasks.

For Task A, we used attention Bidirectional LSTM model performance compared to using different word embedding models, also we analysed impact of the preprocessing of the runs. Since there are too many parameters to compare with, we compared the performance of the embedding models. Our submitted models, BERT and TWITA300 illustrated in Table 1 with $*$ showed most promising results using different settings. With only %80 training data, similarity vectors generalised better than all other embedding models. While, when all data are trained, the best model is the multilingual BERT embedding with no pre-processing (f-avg 0.601), followed by similarity vectors using cleaned text (f-avg 589).

For Task B, we used different feature extraction, frequency vectors, word embedding and social interaction embedding models, and monitor their performance while activating the pre-processing step in all experiments. With a diverse range of parameters, we experimented with a total of 3845 random runs. Then, we selected the best mod-

evalita-sardistance/

[4] https://tfhub.dev/tensorflow/bert_multi_cased_L-12_H-768_A-12/2

[5] https://www.kaggle.com/mlwhiz/attention-pytorch-and-keras

Task A				
	Eval.		Tst. f-avg	
Not-preprocessed				
Emd#	**%**	**f-avg**	*T%80*	*T%100*
BERT*	0.480	**0.532**	0.533*	**0.601**
SVs	**0.518**	**0.548**	**0.589**	0.532
TWITA300	**0.482**	0.526	**0.578**	0.551
TWITA100	0.480	0.521	0.494	0.551
Fasttext	0.485	0.521	0.479	0.482
GloVe	0.445	0.308	0.401	0.401
Preprocessed				
SVs	**0.515**	**0.556**	0.524	0.566
TWITA100	**0.513**	**0.543**	0.560*	0.566
FastText	0.485	0.489	0.532	0.528
TWITA300	0.447	0.490	0.541	0.506
GloVe	0.445	0.308	0.401	0.401
BERT	0.475	0.445	0.512	0.213
Baseline			*0.578*	*0.578*

Table 1: Results for Task A. We evaluate all the embeddings using Attention Bidirectional LSTM. Our submissions are the ones represented with *. *Bold fonts show results above baseline*

els considering macro f-score for the two classes under consideration (AGAINST and FAVOR) (f-avg). Results are shown in Table 2. By comparing our runs by adding social interaction features, our models with different settings showed a clear improvement on our models. In 1#M, we utilise Conv2D (see NeuralNetwork-based features) for embedding vectors with TfiDF unigram and tweet length, where the model achieved an increase on performance of (f-avg 0.16) when social interaction vectors incorporated into the model. All other models showed the same improvement with an increase of (f-avg 0.115, 0.118, 0.081, 0.021) for 3#M, 5#M, 7#M and 9#M, respectively.

6 Discussion and main findings

The pipeline depicted in Figure 1 was designed to investigate the impact of multiple features on stance detection using variations of feature extraction methods, which have been experimented in previous work but we adapted them to the Italian language in our settings. The training set contains 2132 instances with no evaluation set. In our work, we create a stratified split of 80-20 to evaluate the model, which leads to a training data with 1705 samples. Further, our investigation attempted to randomise different settings, with the aim of submitting the top two with highest f-avg score on the remaining set (Eval. 426) for both tasks. Consequently, we found that this methodology did not generalise well with the testing results. However, our main findings remain consistent across different settings when compared with our results using the stratified split (T%80) and when the model was retrained using all the data (T%100). While our submission evaluated both tasks separately, we discuss all conclusions jointly in this section.

Having different random settings over all frequency-based features (14, in our case) would be a bad strategy to evaluate the methods and come up with the best approach. To verify if we need to include all of these, we run an experiment by including only one feature from (unigram, Tfidf_unigram, chargrams, network_reply_community, userinfobio). The selection of these features where based on selecting the best runs using only one feature from our randomised parameters. Using all the training set and CONV2D with (fasttext,TWEC300) and reduced SVs with deepwalk user's social interaction vector, (userinfobio,chargrams) achieved (f-avg 0.703 and 0.704), respectively. This is also higher than using AttLSTM for the same settings which achieved (f-avg 0.638 and 0.610). In general, we achieve better performance with CONV2D than AttnLSTM for the same settings on the test data. In another experiment, we reduced all the 14 frequency-based parameters achieving (f-avg 0.714) which performs worse than our best 3#M (see 2). Our main conclusion is that the number of features available is not necessarily correlated with the model's performance boost.

In another experiment, we attempted to compare the performance of TWEC100 with TWEC300 (see Section 4). From Table 1, we observed that lower dimensionality and preprocessing may cause the model to under perform by around (f-avg 0.050), at least. Though, this impact was not significant with T%100. However, matching the processing between the embedding vocabulary and the annotated set yields better performance. For example, TWITA100 was more persistent on performance between T%80 and T%100. This highlights the importance of pre-processing and reducing the differences between the embedding vocabularies and labelled sentences. In general, our embedding experiment for Task A show high sensitivity on model performance with pre-processing settings.

Inspired by previous work on encoding word meanings, we experimented with SVs embedding. Interestingly, these vectors showed high f-avg,

Task B					
	Eval.		Tst. f-avg		
#M	*%*	**f-avg**	*T%80*	*T%100*	**Settings.**
1	0.590	0.651	**0.683**	**0.733**	Conv2D(FastText) + Conv2D(PCA(SVs)) + PCA(unigram + Tfidf_unigram + length) + DeepWalk
2	0.511	0.521	0.605	0.573	Conv2D(FastText) + Conv2D(PCA(SVs)) + PCA(unigram + Tfidf_unigram + length)
3	0.595	0.640	**0.662**	**0.719**	Conv2D(FastText)+ Conv2D(PCA(SVs)) + Conv2D(PCA(Tfidf_unigram + chargrams)) + DeepWalk
4	0.525	0.507	0.608	0.604	Conv2D(FastText)+Conv2D(PCA(SVs))+PCA(Tfidf_unigram + chargrams)
5	0.600	0.645	**0.710**	**0.718**	Conv2D(FastText) + Conv2D(PCA(SVs)) + PCA(unigram + length)+ DeepWalk
6	0.487	0.495	**0.661**	0.600	Conv2D(FastText + Conv2D(PCA(SVs)) + PCA(unigram + length)
7	**0.600**	**0.671**	**0.709***	**0.696**	Conv2D(TWITA300) + Conv2D(PCA(SVs)) + PCA(length + network_quote_community + network_reply_community + network_retweet_community + network_friend_community + userinfobio + tweetinfocreateat) + DeepWalk
9	0.574	0.532	**0.629**	0.615	Conv2D(TWITA300) + Conv2D(PCA(SVs)) + PCA(length + network_quote_community + network_reply_community + network_retweet_community + network_friend_community + userinfobio + tweetinfocreateat)
9	**0.602**	**0.691**	**0.677***	**0.681**	AttLSTM(FastText) + AttLSTM(PCA(SVs)) + PCA(puntuactionmarks + length + network_quote_community + network_retweet_community + network_friend_community + userinfobio) + Node2Vec
10	0.459	0.488	0.456	**0.660**	AttLSTM(FastText) + AttLSTM(PCA(SVs)) + PCA(puntuactionmarks + length + network_quote_community + network_retweet_community + network_friend_community + userinfobio)
Baseline			*0.628*	*0.628*	

Table 2: Top performing settings over all sampled runs using our architecture for Task B. Our submissions are the ones represented with *. *Bold fonts show highest/above baseline results*

better than BERT and TWITA300 with T%80 although it showed a significant drop when the model was trained with T%100. This finding opens an investigation towards the ability of SVs to perform better under different settings. For that, we removed PCA(SVs) and run same settings of #M1, and our model achieved (f-avg 0.678), showing a significant impact of SVs on model's performance. Further, we investigate the robustness of deepwalk modelling over node2vec and struct2vec for the same best settings of #M1, resulting on (f-avg 0.641 and 0.604) for node2vec and struct2vec, respectively. Also, in terms of accuracy, the deepwalk model produces an improved accuracy of (% 0.725) compared to node2vec (% 0.665) and struct2vec (% 0.658). This indicates that deepwalk is more reliable on this testing set than other models.

7 Conclusion

In this work, we described a state-of-the-art stance detection system leveraging different features including author profiling, word meaning context and social interactions. Using different random runs, our best model achieved (f-avg 0.733) leveraging deepwalk-based knowledge graphs embeddings, FastText and similarity feature vectors extracted by two multi-headed convolutional neural networks from auther's utterance. This motivates our future, aiming to reduce the model complexity and automate the feature selection process.

8 Acknowledgments

This research utilised Queen Mary's Apocrita HPC facility, supported by QMUL Research-IT.

References

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance detection with bidirectional conditional encoding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 876–885, Austin, Texas, November.

Valerio Basile, Mirko Lai, and Manuela Sanguinetti. 2018. Long-term social media data collection at the university of turin. In *Fifth Italian Conference on*

Computational Linguistics (CLiC-it 2018), pages 1–6. CEUR-WS.

Valerio Basile, Danilo Croce, Maria Di Maro, and Lucia C. Passaro. 2020. EVALITA 2020: Overview of the 7th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. In Valerio Basile, Danilo Croce, Maria Di Maro, and Lucia C. Passaro, editors, *Proceedings of Seventh Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2020)*. CEUR-WS.org.

Giacomo Berardi, Andrea Esuli, and Diego Marcheggiani. 2015. Word embeddings go to italy: A comparison of models and training datasets. In *IIR*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Alessandra Teresa Cignarella, Mirko Lai, Cristina Bosco, Viviana Patti, and Paolo Rosso. 2020. SardiStance@EVALITA2020: Overview of the Task on Stance Detection in Italian Tweets. In Valerio Basile, Danilo Croce, Maria Di Maro, and Lucia C. Passaro, editors, *Proceedings of the 7th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian (EVALITA 2020)*. CEUR-WS.org.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Steffen Eger and Alexander Mehler. 2016. On the linearity of semantic change: Investigating meaning variation via dynamic graph models. In *Proceedings of ACL (Volume 2: Short Papers)*, pages 52–58, Berlin, Germany, August.

Eduardo Graells-Garrido, Ricardo Baeza-Yates, and Mounia Lalmas. 2020. Every colour you are: Stance prediction and turnaround in controversial issues. In *12th ACM Conference on Web Science*, WebSci '20, page 174–183, New York, NY, USA. Association for Computing Machinery.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of ACM SIGKDD*, pages 855–864.

Dilek Küçük and Fazli Can. 2020. Stance detection: A survey. *ACM Computing Surveys (CSUR)*, 53(1):1–37.

Mirko Lai, Marcella Tambuscio, Viviana Patti, Giancarlo Ruffo, and Paolo Rosso. 2017. Extracting graph topological information and users' opinion. In *Lecture Notes in Computer Science*, volume 10456 LNCS, pages 112–118. Springer Verlag.

Mirko Lai, Alessandra Teresa Cignarella, Delia Irazú Hernández Farías, Cristina Bosco, Viviana Patti, and Paolo Rosso. 2020. Multilingual stance detection in social media political debates. *Computer Speech & Language*, 63:101075.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of ACM SIGKDD*, pages 701–710.

Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. 2017. struc2vec: Learning node representations from structural identity. In *Proceedings of ACM SIGKDD*, pages 385–394.

Parinaz Sobhani, Diana Inkpen, and Xiaodan Zhu. 2019. Exploring deep neural networks for multi-target stance detection. *Computational Intelligence*, 35(1):82–97, feb.

Qingying Sun, Zhongqing Wang, Qiaoming Zhu, and Guodong Zhou. 2016. Exploring various linguistic features for stance detection. In *Natural Language Understanding and Intelligent Applications*, pages 840–847, Cham. Springer International Publishing.

Michael Wojatzki, Torsten Zesch, Saif Mohammad, and Svetlana Kiritchenko. 2018. Agree or Disagree: Predicting Judgments on Nuanced Assertions. In *Proceedings of *SEM*, pages 214–224, Stroudsburg, PA, USA.

Zhenhui Xu, Qiang Li, Wei Chen, Yingbao Cui, Zhen Qiu, and Tengjiao Wang. 2019. Opinion-aware knowledge embedding for stance detection. In Jie Shao, Man Lung Yiu, Masashi Toyoda, Dongxiang Zhang, Wei Wang, and Bin Cui, editors, *Web and Big Data*, pages 337–348, Cham.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of NAACL-HLT*, pages 1480–1489.

Arkaitz Zubiaga, Elena Kochkina, Maria Liakata, Rob Procter, Michal Lukasik, Kalina Bontcheva, Trevor Cohn, and Isabelle Augenstein. 2018. Discourse-aware rumour stance classification in social media using sequential classifiers. *Information Processing & Management*, 54(2):273–290.