IberEval 2017, *COSET* task: a basic approach

Carlos Diez Alba, Jesús Vieco Pérez

Departament de Sistemes Informàtics i Computació
Universitat Politècnica de València
{cardieal,jeviepe}@dsic.upv.es

Abstract. This paper discusses the IberEval 2017 shared task on Classification Of Spanish Election Tweets (COSET) [5]. This task has the goal to analyze tweets that talk about Spanish General Election of 2015 and classify them in one of these 5 categories: political issues, policy issues, personal issues, campaign issues and other issues.

Keywords: UPV, bag-of-words, embeddings, neural networks, support vector machine, random forest

1 Introduction

Election dates increase the political conversations in Twitter. We can extract several useful information from all the conversations if we are able to segment them in different categories. COSET [5] has the aim to classify a simited corpus of Spanish tweets which are related with the 2015 Spanish General Election in one of these categories:

- Political issues: related to the electoral confrontation
- Policy issues: related to the political sectors
- Personal issues: related to the life and activities of the candidates
- Campaign issues: related to the hustings
- Other issues: remaining ones that do not fit in the previous categories

The aim of this paper is to perform a study of different feature extraction methods and machine learning algorithms for classifying this set of tweets into their categories.

2 Machine learning algorithms

Before explaining the feature extraction we need some algorithms that can classify the tweets in the different categories. To solve this problem we tried different machine learning algorithms:

- Support vector machines (SVM): probably the most used algorithm in tweets classification, the SVM is one of the state-of-the-art algorithm for tasks related with text.

– Random forest (RF): this algorithm is not usually employed for a text task but we decided to use it because we have possibilities of overfitting with the size of the corpus and the random forest prevents that.
– Neural network (NN): currently the most used machine learning algorithm due to its flexibility. We used both multilayer perceptron and long short term memory (LSTM) [6] networks.

We used support vector and random forest from scikit learn [14]. Neural networks are programmed using Keras [3] and Theano [20].

3 Feature extraction

For this task we have a list of characters for every tweet but we can not feed a machine learning algorithm with a list of characters: before it we have to extract features for every tweet. We try three different methods of feature extraction. All of them need a previous preprocessing:

– Tokenization: we split every tweet in a list of tokens like words, URL links, punctuation marks, etc.
– Cleaning: we remove all URL links, emoticons and some other special characters.

3.1 N-gram

The n-gram idea is based on grouping the tokens in groups of n. We create a model based on the idea of bag of words but using the n-gram themselves [13], so we get a vector with the number of different n-grams as size and the number of times a n-gram appears in the tweets as a feature.

3.2 Tf-idf

Another approach is using the Term Frequency Inverse Document Frequency (tf-idf) to generate vectors for every tweet with the tweets tokenized and clean, despite of getting the n-gram model. It is similar to the previous one but we have a probability of a word in every feature related with the importance of a word in a tweet instead of getting a count in every feature. This method is used in some experiments like [10] and [7].

3.3 Vector representations of words: Embeddings

The third feature extraction method is the proposed one [12]. The features extracted by this method are called embeddings and we can found several recent papers where the embeddings are combined whit neural networks and get state-of-the-art results [19]. The main idea is to obtain a vector representation for every word learning it in an unsupervised way of a set of sentences.
To train this embedding we will use the Gensim [15] software. Due to this model

needs many tweets to train and we have less than 3000, we used a bigger set of tweets provided by the UPV researcher Javier Palanca Cámara, used in other fields [21] [22], in addition of what we already have. These tweets have political correlation and are linked with the manifestation for the independence of Catalonia in 2016 called Diada.
This feature extraction gets a vector for every word, so we got different numbers of vectors for every tweet. To train a neural network we need a recurrent layer to deal with different size of vectors. In our case we used Long short term memory (LSTM) layers [6] and we compare it with the bidirectional [17] long short term memory (BLSTM), which has some improvements in this field [11] [2].

3.4 Combining methods

We used both feature extraction methods tf-idf and word embeddings in order to improve our past model. We need a neural network which can join both. In Figure 1 we can see our base neural network to work with.

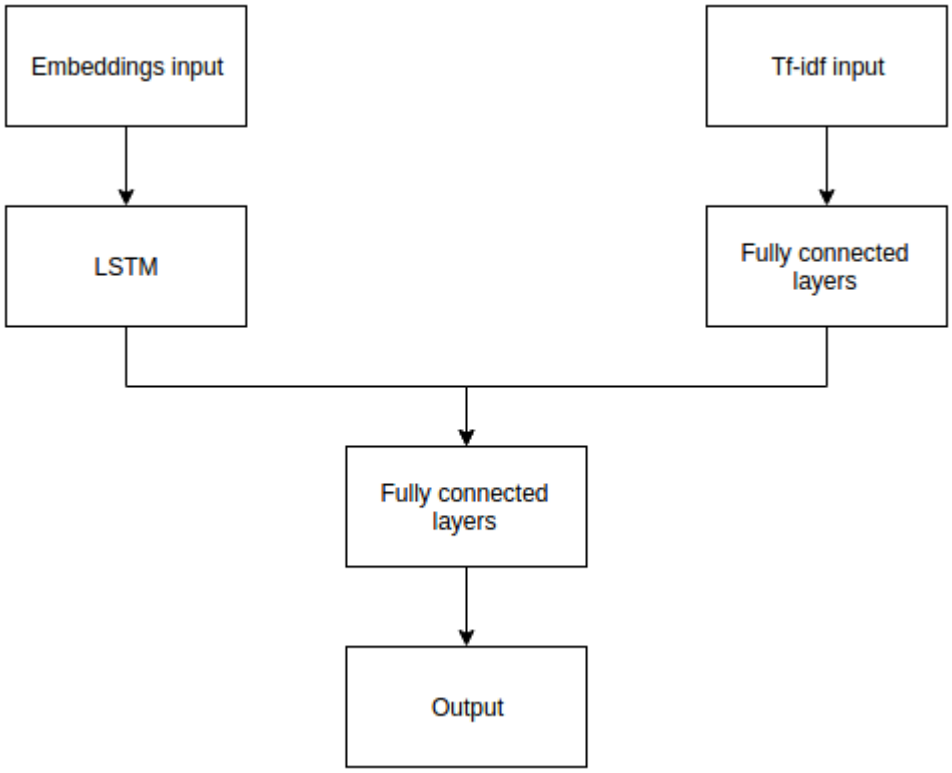

Fig. 1: Network schema

We tried some different models, replacing the LSTM by a BLSTM, adding dropout, batch normalization and more fully connected layers. To improve the LSTM learned, we added another output layer just before the LSTM.

4 Results and Discussions

We are going to show the experiments done. We will divide these sections in two parts:
The first is for the experiments with the first feature extraction method, which was the n-grams approximation with the tf-idf approach. Next we proceed to do the same with the embeddings method. Finally we join both methods in one. We show the best results with f1 macro score obtained with every model and feature extraction because the task will be evaluate with it due to the unbalanced corpus. In addition we show the average precision for every result. The f1 score considers both the precision and recall. The formula for the f1 score is:

$$f1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{1}$$

To calculate the f1 macro score we only need to calculate the f1 score for every class and find their unweighted mean.

4.1 N-grams and tf-idf

In Table 1 we can see our best result in terms of f1 macro score and its correspondent accuracy obtained in every machine learning algorithm. First of all, we can see the support vector machine column. In the second column we have the experiments with random forest and the third column is for the test with neural networks.

Table 1: N-grams and tf-idf experiments using svm, rf and nn

	SVM		**RF**		**NN**	
	accuracy	**f1_score**	**accuracy**	**f1_score**	**accuracy**	**f1_score**
1-gram	60.80	56.70	51.20	43.77	64.00	58.74
2-gram	60.40	55.92	50.80	41.19	62.80	59.09
3-gram	62.80	56.73	50.08	41.77	62.40	59.05
tf-idf	63.60	57.24	52.40	47.63	**64.80**	**59.15**

The random forest results are quite far from the others as it is shown in the table. As we said before the random forest is not so common as svm for task related to text but we want to make a comparison because we saw some studies where obtain very good marks [4] [1].
For the random forest, the best results are obtained with the tf-idf feature extraction and a random forest form by ten decision trees forced to have only one

sample in every leaf.
In the case of the support vector machine, we get the best with the same feature extraction and a group of SVM (one for every pair of classes) with linear kernel faced each other to decide the predicted label.
To obtain the best result on this table we use a multilayer perceptron with 3 hidden layers of size 1024, 512, 256 for each layer respectively. Due to the size of the corpus we need to avoid the overfitting. In this case we use dropout [18] along with batch normalization [8]. The curious thing about this model is that avoiding cleaning the tweets (remove all URL links, emoticons and some other special characters) we improve our results in 3 points but we did not see this improvement in the other algorithms.

4.2 Embeddings

In this section we trained two models of embeddings. The first one was trained with only the training tweets provided for the competition and the second one with these tweets along with the tweets provided by Javier Palanca Cámara [22] and [21]. We obtained better results with the second model in all the experiments with embeddings. Therefore, to avoid redundant results we will not show the ones of the first model.

Table 2: Embeddings experiments

	accuracy	**f1_score**
lstm	46.80	42.12
blstm	50.00	47.92
svm	57.24	63.60
rf	47.63	52.40
nn	59.15	64.80

In Table 2 are shown the best results obtained with LSTM and BLSTM along with the best results obtained in the previous section with svm, rf and nn. As we say before BLSTM have some improvements in LSTM but both have worse results compared to the approach of n-grams and tf-idf. We can conclude that embeddings and recurrent neural networks in this task need much more samples that other approximations to be competitive.

4.3 Combining methods

Best results we obtained are shown in Table 3. As we said before, we got better result with the BLSTM than with the LSTM. We have improved the results of the LSTM and the BLSTM with embeddings but can not improve the results with the neural network and td-idf. One of the biggest problems for this method is the number of parameters we need to estimate. Furthermore, more data to train the model is needed.

Table 3: Combining experiments

	accuracy	**f1_score**
lstm	54.22	50.95
blstm	61.85	56.34

5 Conclusions and future work

We can conclude that traditional models like support vector machines and multilayer perceptron have worked better than fashionable approaches such as LSTM and BLSTM. We saw the same results comparing n-grams and tf-idf against embeddings. These results are due to the size of the training set, so we might obtain better results using data augmentation techniques.
In order to improve our previous models, we can try some new approaches like fastText [9]. We can also try adding other feature extraction like part-of-speech tags as we can see in [16].

References

1. Aramaki, E., Maskawa, S., Morita, M.: Twitter catches the flu: detecting influenza epidemics using twitter. In: Proceedings of the conference on empirical methods in natural language processing. pp. 1568–1576. Association for Computational Linguistics (2011)
2. Augenstein, I., Rocktäschel, T., Vlachos, A., Bontcheva, K.: Stance detection with bidirectional conditional encoding. arXiv preprint arXiv:1606.05464 (2016)
3. Chollet, F., et al.: Keras. `https://github.com/fchollet/keras` (2015)
4. Fernández Anta, A., Núñez Chiroque, L., Morere, P., Santos Méndez, A.: Sentiment analysis and topic detection of spanish tweets: a comparative study of nlp techniques (2013-03)
5. Giménez, M., Baviera, T., Llorca, G., Gámir, J., Calvo, D., Rosso, P., Rangel, F.: Overview of the 1st classification of spanish election tweets task at ibereval 2017. In: Proceedings of the Second Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2017), Murcia, Spain, September 19, CEUR Workshop Proceedings. CEUR-WS.org, 2017 (sep 2017)
6. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation 9(8), 1735–1780 (1997)
7. Hong, L., Dan, O., Davison, B.D.: Predicting popular messages in twitter. In: Proceedings of the 20th international conference companion on World wide web. pp. 57–58. ACM (2011)
8. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
9. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759 (2016)
10. Lee, K., Palsetia, D., Narayanan, R., Patwary, M.M.A., Agrawal, A., Choudhary, A.: Twitter trending topic classification. In: Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on. pp. 251–258. IEEE (2011)
11. Limsopatham, N., Collier, N.: Bidirectional lstm for named entity recognition in twitter messages. WNUT 2016 p. 145 (2016)

12. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
13. Pak, A., Paroubek, P.: Twitter as a corpus for sentiment analysis and opinion mining. In: LREC. vol. 10 (2010)
14. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830 (2011)
15. Řehůřek, R., Sojka, P.: Software Framework for Topic Modelling with Large Corpora. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. pp. 45–50. ELRA, Valletta, Malta (May 2010), http://is.muni.cz/publication/884893/en
16. Robinson, T.: Disaster tweet classification using parts-of-speech tags: a domain adaptation approach. Ph.D. thesis, Kansas State University (2016)
17. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing 45(11), 2673–2681 (1997)
18. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15(1), 1929–1958 (2014)
19. Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., Qin, B.: Learning sentiment-specific word embedding for twitter sentiment classification. In: ACL (1). pp. 1555–1565 (2014)
20. Theano Development Team: Theano: A Python framework for fast computation of mathematical expressions. arXiv e-prints abs/1605.02688 (May 2016), http://arxiv.org/abs/1605.02688
21. Val, E.D., Palanca, J., Rebollo, M.: U-Tool: A Urban-Toolkit for enhancing city maps through citizens' activity. In: 14th International Conference on Practical Applications of Agents and Multi-Agent Systems. pp. 243–246 (2016)
22. Vivanco, E., Palanca, J., Val, E.D., Rebollo, M., Botti, V.: Using geo-tagged sentiment to better understand social interactions. In: 15th International Conference on Practical Applications of Agents and Multi-Agent Systems (2017)