Several approaches for tweet topic classification in COSET – IberEval 2017

Carlos Villar Lafuente and Gonçal Garcés Díaz-Munío

Universitat Politècnica de València, Spain
carlosbetera3dx@gmail.com, ggarces@dsic.upv.es

Abstract. These working notes summarize the different approaches we have explored in order to classify a corpus of tweets related to the 2015 Spanish General Election (COSET 2017 task from IberEval 2017). Two approaches were tested during the COSET 2017 evaluations: Neural Networks with Sentence Embeddings (based on TensorFlow) and N-gram Language Models (based on SRILM). Our results with these approaches were modest: both ranked above the "Most frequent baseline", but below the "Bag-of-words + SVM" baseline. A third approach was tried after the COSET 2017 evaluation phase was over: Advanced Linear Models (based on fastText). Results measured over the COSET 2017 Dev and Test show that this approach is well above the "TF-IDF+RF" baseline.

Keywords: COSET2017, text classification, neural networks, sentence embeddings, language models, linear models

1 Introduction

In this work we have explored several approaches for text classification on Twitter messages, tackling the COSET 2017 task from IberEval 2017 [1][2].

The aim of the task was to classify a corpus of tweets in Spanish about the 2015 Spanish General Election into five categories or topics, namely:

1. Political issues (related to the most abstract electoral confrontation)
2. Policy issues (about sectorial policies)
3. Campaign issues (related with the evolution of the campaign)
4. Personal issues (on the life and activities of the candidates)
5. Other issues

2 Data

The COSET 2017 organizers provided a corpus split in three sets of tweets, two of them labelled for training (Train) and development (Dev), and the other one unlabelled as a Test (which we had to classify, submitting the results for evaluation). The tweets were provided (password protected) in CSV files containing the tweet ID, the tweet itself and (only in the Train and Dev files) the label [1].

Table 1. COSET 2017 corpus: Number of tweets per partition set (Train, Dev & Test)

Set	# of tweets	% of tweets
Train	2242	72%
Dev	250	8%
Test	624	20%
Total	3116	100%

Table 2. COSET 2017 Train & Dev sets: Topic distribution of the tweets

	Train		Dev	
Topic	# of tweets	% of tweets	# of tweets	% of tweets
Political issues	530	23%	57	23%
Policy issues	786	35%	88	35%
Campaign issues	511	23%	71	28%
Personal issues	152	7%	9	4%
Other issues	263	12%	25	10%
Total	2242	100%	250	100%

Table 1 shows the number of tweets for each set in the COSET 2017 corpus.

Table 2 reflects the topic distributions for the Train and Dev sets.

The output expected for submissions was a file where each line is a tuple of tweet ID and a label, for each tweet in the test file. Up to 5 submissions were allowed (of which we used 2 before the deadline).

After the COSET 2017 competition was over, a new, much larger tweet dataset was provided by the organizers for a second phase test on robustness in which each participant used their best approach [2]. This additional corpus was made up of a total of 15,806,057 tweets, of which 10,417,058 tweets were labelled by the organizers and used as a hidden test set on which to compute the results. We report results on this test as well in Section 4.

3 Systems description

3.1 Preprocessing

We carried out the preprocessing of the Train, Dev and Test sets with our own Python script based on regular expressions (regex library). The aim was to leave the tweets in the best condition possible in order to make it easier later to extract meaningful features for the classification models that we used in our approaches.

First of all, as many of the the tweets in the CSV files came divided in several lines (we can detect the end of a tweet because the next line begins with double quotation marks + an 18-digit tweet ID), our script converted each tweet to one string without line breaks (to make it easier to process the files for the rest of the modelling and classification steps).

Other modifications were the conversion of hashtags to lower case (as, for Twitter, the hashtags #COSET2017, #CoSeT2017 and #Coset2017 are the same), removing redundant quotation marks (when a tweet contained quoted text, the API used for tweet retrieval had added redundant quotation marks), removing hyperlinks (URLs are shortened by Twitter and sometimes cut by the API, and this provides zero information), shortening citations (Twitter added a lot of text in order to indicate that a tweet cites another tweet), and basic tokenization (separating punctuation from words).

Apart from the initial preprocessing we just described, in the case of some of the approaches described below, we also tried converting the full text of the tweets to lowercase, by using the lowercase.perl script that comes with the Moses toolkit for statistical machine translation [3].

3.2 Classification approaches

We faced the task by exploring different techniques and solutions instead of focusing on a single approach, with the aim of achieving better results.

Two approaches were tested during the COSET 2017 evaluation phase: Neural Networks with Sentence Embeddings (based on TensorFlow) and N-gram Language Models (based on SRILM). A third approach was tried after the COSET 2017 evaluation phase: Advanced Linear Models (based on fastText).

N-gram Language Models (SRILM). The most basic approach we tried was to train an n-gram language model for each topic. Then, new tweets (the tweets in the Test set) were compared to these 5 topic n-gram models based on perplexity. Each tweet was assigned the topic with which it showed less perplexity.

The well-known n-gram language modelling toolkit SRILM [4] was used for this approach. N-gram models were trained with SRILM's default smoothing method: Good-Turing smoothing. Different n-gram orders were tried (2, 3, 4, 5), with the best F1-macro result on Dev having been obtained with 3-grams.

SRILM was also used to compute the perplexity of new tweets against each one of the 5 topic n-gram models trained. As explained before, each tweet was classified as pertaining to the topic with which it showed less perplexity.

In the end, we sent one submission for evaluation in COSET 2017 based on this approach. The models chosen were 3-grams with Good-Turing smoothing.

Neural Networks with Sentence Embeddings (TensorFlow). In this case, the aim was to use a TensorFlow-based neural network [5] for tweet classification.

In order to extract the features of each tweet to train the neural network (NN) models, we decided to use the Sentence Embeddings approach, by which each text document (in this case, each tweet) is represented as a feature vector. To this end, we used the software Doc2vec from the gensim Python library [6]. Doc2vec is an NLP tool that modifies the Word2vec word embedding algorithm [9] to carry out an unsupervised learning of continuous representations for larger blocks of text, such as sentences, paragraphs or entire documents.

After sentence embeddings were computed with Doc2vec for all of the tweets, these feature vectors (one per tweet) were used as input to train with TensorFlow a NN model for tweet classification into the task's 5 topics.

The Doc2vec model for converting tweets into vectors of fifty features (a compromise between having enough representation power and having too many parameters to train) was built from the train data. The training method of the model was called ten times, each training step taking ten iterations. While each training step had a fixed learning rate, between each call, the learning rate was decreased (learning rate annealing, starting at 0.025, decreasing 0.02 per step).

After the Doc2vec model construction, all tweet data was converted into feature vectors using it and saved in new files. Each line contained the tweet id (in the case of test data), the feature vector and the label (in the case of Train or Dev data), all separated by spaces.

Finally, we fed the data into a NN with TensorFlow. A traditional feed-forward NN topology was used, with a hidden layer of 1024 nodes (with ReLU as the activation function), an input layer of 50 nodes (feature vector size), and an output layer of 5 nodes (number of classes). The labels were converted from a numerical value to a vector with a component for each class (0,1,2,3,4, following the order from Section 1), with the vector size being determined by the highest label's value plus one (in this case, highest value 4, vector size 5).

The NN was fed with batches of 100 tweets during training. After training, the NN was fed with the test data and the output vector was converted into a class label depending on which position of the vector had the highest value.

We sent one submission with this approach for evaluation in COSET 2017.

Advanced Linear Models (fastText). Linear models with bag of n-grams representation and hierarchical classifiers have been explored by Facebook Research as a way to develop text classifiers on par with deep learning classifiers in terms of accuracy and more efficient for training and evaluation [7]. Here, we have worked with Facebook Research's fastText[1] tool for word embedding and text classification [8] so as to optimize results over the COSET 2017 Dev set.

In order to predict the labels for the COSET 2017 Test set, a text classification model was trained with fastText from the COSET 2017 Train and Dev sets. Then, the Test set was classified using fastText with this model.

There are several fastText parameters we can adjust for model training. The best results on Dev were observed with fastText's defaults for word vector size (100), word n-gram order (1), loss function (5-negative sampling), and learning rate (0.05), but increasing the number of epochs (from the default of 5 to 25).

Finally, regarding preprocessing, with this approach we tried converting to lowercase the results of our preprocessing described in Section 3.1. Our measurements showed that using lowercase tweets for system training and for classification improved the results on Dev (from F-1 macro 0.5494 with truecase to F-1 macro 0.5982 with lowercase); thus, the result for this approach reported in Section 4 below is based on converting tweets to lowercase during preprocessing.

[1] https://github.com/facebookresearch/fastText

4 Evaluation and discussion

4.1 Evaluation metric

The metric designated for evaluating the participating systems in COSET 2017 [2] was F-1 macro (which considers the precision and the recall of the system's predictions, using the macro-averaging method for preventing systems biased towards the most populated classes):

$$F_1 - macro = \frac{1}{|L|} \sum_{l \in L} F_1(y_l, \hat{y}_l) \tag{1}$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{2}$$

$$precision = \frac{1}{|L|} \sum_{l \in L} Pr(y_l, \hat{y}_l) \tag{3}$$

$$recall = \frac{1}{|L|} \sum_{l \in L} R(y_l, \hat{y}_l) \tag{4}$$

where L is the set of labels, y_l the ground truth, and $\hat{y}_l$ the predicted labels.

4.2 Results and discussion

As mentioned before, we only were able to submit results before the COSET 2017 competition deadline using the Language Model approach and the Neural Network approach. After the COSET 2017 submission deadline had passed, we carried out additional experiments using the Linear Model approach.

In Table 3 we show our results measured in F-1 macro over the Dev and Test sets of COSET 2017. The COSET 2017 baselines and winning system have also been included for reference (see [2] for the complete results).

Table 3. Results on COSET 2017 corpus: F-1 macro over COSET 2017 Dev & Test sets for each approach tested (COSET 2017 *baselines* and *winner* in italics for ref.)

System	Dev	Test
Winner – ELiRF-UPV run 1	–	*0.6482*
Adv. Linear Models (unsubmitted)	0.5982	0.5783
Baseline – TF-IDF + RF	–	*0.4236*
Baseline – BOW+SVM	–	*0.2644*
N-gram Language Models (run 2)	0.2738	0.2446
Neural Networks with S. E. (run 1)	0.2615	0.2410
Baseline – Most frequent	*0.1041*	*0.1070*

As we can see, our results with the N-gram Language Model and Neural Networks with Sentence Embeddings approaches were modest: both ranked above the "Most frequent baseline", but below the "Bag-of-words + SVM" baseline. However, this does not mean that it would not be possible to obtain better results with these approaches, if more time had been available for experimentation. For starters, a different preprocessing, such as converting all tweets to lowercase, could improve the results. Then, in the case of N-gram Language Models, trying n-gram character models instead of n-gram word models could be effective. As to the case of Neural Networks, it could be useful to try Word2vec [9] for word embedding (instead of sentence embedding with Doc2vec), or skipping word embedding computation altogether, and instead feeding the tweet texts directly into TensorFlow so that neural networks are used to compute the most appropriate features for each tweet for the training of the neural network model; additionally, different neural network topologies could be explored to obtain better results.

As to the Linear Model approach, results on the COSET 2017 Dev & Test sets show that this approach ranks well above the "TF-IDF+RF" baseline, being closer to the winning system [2]. In our tests, and with this result, fastText has proven to be a powerful, efficient, easy-to-use tool for text classification.

As explained in Section 2, after the COSET 2017 competition was over, a second phase test on robustness was proposed by the organizers, in which we ran our best approach on a new, much larger tweet set. As we can see in Table 4, by running the Advanced Linear Models system described in Section 3.2 (trained this time on the original COSET 2017 Train, Dev and Test sets), our results came in second in the list of participants, close to the winning systems [2].

Table 4. Results on COSET 2017 second phase evaluation (F-1 macro)

Team	Test (COSET2)
ELiRF-UPV.1	*0.9586*
ELiRF-UPV.2	*0.9523*
Team 17 (Adv. Linear Models)	0.9482
atoppe	*0.8960*
LTRC_IIITH	*0.8509*

As future work, we can recapitulate here some of the possible ways for improvement: for preprocessing, trying other techniques on the tweets, such as lemmatization, or smiley and emoji preprocessing [10]; in the N-gram Language Model approach, trying n-gram character models instead of word models; and in the Neural Network approach, trying Word2vec or fastText for word embedding (instead of sentence embedding with Doc2vec), or skipping word embedding precomputation and instead feeding the tweet texts directly into TensorFlow.

References

1. COSET (IberEval 2017): Classification of Spanish Election Tweets (2017). Retrieved from http://mediaflows.es/coset/
2. 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
3. Moses Tokenizer scripts. Retrieved from https://github.com/moses-smt/mosesdecoder/tree/master/scripts/tokenizer
4. SRILM: The SRI Language Modeling Toolkit. Retrieved from http://www.speech.sri.com/projects/srilm/
5. TensorFlow: An open-source software library for Machine Intelligence. Retrieved from https://www.tensorflow.org/
6. gensim doc2vec: Deep learning with paragraph2vec. Retrieved from https://radimrehurek.com/gensim/models/doc2vec.html
7. Joulin, P., Grave, E., Bojanowski, P., Mikolov, T.: "Bag of Tricks for Efficient Text Classification". CoRR abs/1607.01759 (2016)
8. fastText: Library for efficient learning of word representations and sentence classification. Retrieved from https://github.com/facebookresearch/fastText
9. Word2vec: Tool for computing continuous distributed representations of words. Retrieved from https://code.google.com/archive/p/word2vec/
10. Potts, C. Sentiment Symposium Tutorial (2011). Retrieved from http://sentiment.christopherpotts.net/