Neural models for StanceCat shared task at IberEval 2017

Luca Ambrosini[1] and Giancarlo Nicolò[2]

[1] Scuola Universitaria Professionale della Svizzera Italiana
[2] Univesitat Politècnica De València
luca.ambrosini@supsi.ch
giani1@inf.upv.es

Abstract. This paper describes our participation in the *Stance and Gender Detection in Tweets on Catalan Independence (StanceCat)* task at IberEval 2017. Our approach was focused on neural models, firstly using classical and specific model from state of the art, then we introduce a new topology of convolutional network for text classification.

1 Introduction

Nowadays the pervasive use of social network as a mean of communication helps researchers to found useful insight over open problems in the field of Natural Language Processing. In this context, the *Twitter* social network has a huge role in text classification problems, because thanks to its *API* is possible to retrieve specific formatted text (i.e., a sentence of maximum 140 characters called tweet) from a huge real-time text database, where different users publish their daily statements.
This huge availability of data gives raise to the investigation of new text classification problems, with special interest in prediction problems related to temporal events that can influence statements published by social network users. An example of this problem category is the stance detection related to political events, where the *Stance and Gender Detection in Tweets on Catalan Independence (StanceCat)* task at IberEval 2017 is a concrete example.
In StanceCat, the principal aim is to automatically detect if the text's author is in favor of, against, or neutral towards the Catalan Independence. Moreover, as a secondary aim, participants are asked to infer the author's gender.
To tackle the above problem we built a classification system that can be decomposed in three main modules, each representing a specific approach widely used in the NLP literature: text pre-processing, text representation and classification model. During the modules design, we explore different design combinations leading the system development to a comparative study over the possible modules interactions. Analysing the produced study interesting insight can be drawn to create a system baseline for the tweet classification problem.
In the following sections we firstly describe the StanceCat task (Section 2), then we report the development process of the classification system and its module design (Section 3), after that, the evaluation of deployed systems over the provided corpus is analysed (Section 4), finally, conclusion over the whole work are outlined (Section 5).

2 Task definition

The StanceCat shared task aim was to detect the author's gender and stance with respect to the target *independence of Catalonia* in tweets written in Spanish and/or Catalan, where participants is allowed in the detection of both stance and gender or only in stance detection.
Participants had access to a labelled corpus for each languages composed of 4319 tweets. We analysed it and find the following statistical informations presented in tables 1 and 2.

Label	Favor	Neutral	Against	Total
ES	335	2538	1446	4319
CA	2648	1540	131	4319

Table 1: Statistical analysis of given corpus' tweets.

Tweets	Average	Deviation	Max
ES	14	3	23
CA	13	4	20

Table 2: Statistical analysis of given corpus' tweets.

3 Systems description

In this section we describe the stance&gender detection systems. Organizing the system by modules, it is organized in two blocks: text pre-preprocessing (Section 3.1) and classification model (Section 3.3).

3.1 Text pre-processing

Regarding the text pre-preprocessing, has to be mentioned that the corpus under observation can not be treated as proper written language, because computer-mediated communication (CMC) is highly informal, affecting diamesic[3] variation with creation of new items supposed to pertain lexicon and graphematic domains [6,7]. Therefore, our pre-processing follows two approaches: classic and microblogging related. As classic aproach we used stemming (i.e., ST), stopwords (i.e., SW) and punctuation removal (i.e., PR). For microblogging approach we focus our attention over the following items: (i) mentions (i.e., MT), (ii) smiley (i.e., SM), (iii) emoji (i.e., EM), (iv) hashtags (i.e., HT), (v) numbers (i.e., NUM), (vi) URL (i.e., URL) (vii) and Tweeter reserve-word as RT and FAV (i.e., RW). For each of these items we leave the possibility to be removed or substituted by constant string.
In relation to above approaches we implement them using the following tools: (i) NLTK [3] and (ii) Preprocessor[4].

3.2 Text representation

To represent the text we used word embeddings as described by [4], where tweet elements like *words* and *word n-grams* are represented as vectors of real number with fixed dimension $|v|$. In this way a whole sentence s, with length $|s|$ its number of word, is represented as a *sentence-matrix* M of dimension $|M| = |s| \times |v|$. $|M|$ has to be fixed a priori, therefore $|s|$ and $|v|$ have to be estimated. $|v|$ was fixed to 300 following [4]. $|s|$ was estimated analyzing table 2, in details we decided to fix it as the sum of average length plus the standard deviation (i.e. $|s| = 17$ for both language), with this choice input sentences longer than $|s|$ are truncated, while shorter ones are padded with null vectors (i.e., a vector of all zeros). Choosing words as elements to be mapped by the embedding function, raise some challenge over the function estimation related to data availability. In our case the available corpus is very small and estimated embeddings could lead to low performance. To solve this problem, we decided to used a pre-trained embeddings estimated over Wikipedia using a particular approach called *fastText* [4].

[3] The variation in a language across medium of communication (e.g. Spanish over the phone versus Spanish over email)

[4] Preprocessor is a preprocessing library for tweet data written in Python, https://github.com/s/preprocessor

3.3 Classification models

Following, we describe the neural models used for the classification module, where for each of them the input layer uses text representations described in Section 3.2 (i.e., sentence-matrix).

Convolutional Neural Network. Convolutional Neural Networks (CNN) are considered state of the art in many text classification problem. Therefore, we decide to use them in a simple architecture composed by a convolutional layer, followed by a *Global Max Pooling* layer and two dense layers.

Dilated KIM. This model is our new topology of CNN. It can be seen as an extension of Kim's model [1] using the dilation ideas from computer graphics field [13]. The original Kim's model is a particular CNN where the convolutional layer has multiple filter widths and feature maps. The complete architecture is illustrated in Figure 1, here the input layer (i.e., sentence-matrix) is processed in a convolutional layer of multiple filters with different width, each of these results are fed into *Max Pooling* layers and finally the concatenation of them (previously flatten to be dimensional coherent) is projected into a dense layer. Our extension is to use a dilated filters in combination with normal ones, the intuition is that normal filter capture *adjacent words* features, while dilated one are able to capture relations *non adjacent words*. This behaviour can't be achieved by the original Kim's model, because, even though the filters size can be changed, they will capture only features from adjacent words.

Recurrent neural network. LSTM is a type of Recurrent Neural Network (RNN) that is relatively insensitive to gap length. Thanks to this behaviour, they are considered state of the art in some NLP problems. Our architecture was made of an embedded input layer followed by an LSTM layer of 128 units, terminated by a dense layer. Moreover, to avoid overfitting we used dropout and recurrent dropout.
Similar to the previous model, bidirectional LSTM is a variation of LSTM where the two RNN receive different inputs, the original and its reverse order, and their results are connected through the recurrent layers. Our architecture follow the previous one with an LSTM layer of 128 units terminating with a dense layer, where all the layer used dropout and recurrent dropout.

4 Evaluation

In this section we are going to illustrate results from the comparative study elaborated during the system development. First we illustrate the metric used to evaluate the system (Section 4.1) and then we report results produced by a 10-fold cross validation over the given data set (Section 4.2).

4.1 Metrics

System evaluation metrics were given by the organizers and reported here in the following equations (1) to (4). Their choice was to use an $F_{1-macro}$ measure due to class unbalance in the corpus.

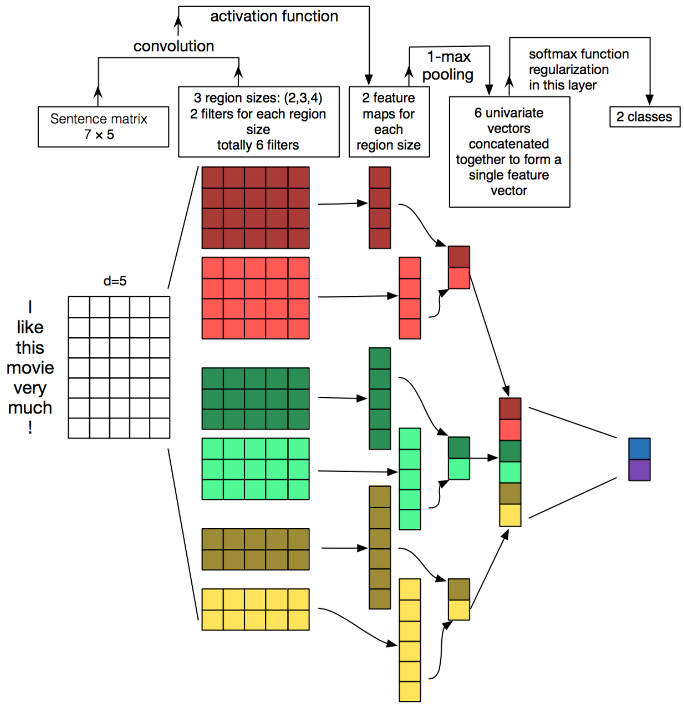

Fig. 1: [8] Illustration of a Convolutional Neural Network (CNN) architecture for sentence classification

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

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

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (2)$$

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

where L is the set of classes, y_l is the set of correct label and $\hat{y}_l$ is the set of predicted labels.

4.2 Results

Following, we present a comparative study over possible combinations of pre-processing (Table 3) and word embeddings (Table 4), in both cases results are calculated from averaging three runs of a 10-fold cross validation over the complete data set. Notations used in Table 3 refer to the one introduced in Section 3.1, where the listing of a notation means its use for the reported result. Regarding the tweet specific pre-processing, all the items have been substituted, with the exception for URL and RW that have been removed. We report the contribution of each analysed pre-processing alone.
From the analysis of Table 3 no absolute conclusion can be drawn, meaning that it wasn't possible to find a combination of pre-processing that gives the best performance for all the model, meaning that each model is highly sensible to the performed combination. Nevertheless, some relative observation can be made:

Table 3: Pre-processing study comparing 10-fold cross validation results over the development set in terms of percentuage of $F_{1-macro}$ score. For each model processing tecnique that brought an improvement has its result in bold.

Models	Pre-processing									
	Nothing	ST	SW	URL	RW	MT	HT	NUM	EM	SM
Kim	0.543	0.528	**0.557**	**0.571**	0.533	**0.558**	0.540	**0.554**	0.537	0.539
FastText	0.546	0.533	**0.550**	0.534	**0.553**	0.519	0.538	**0.558**	**0.552**	**0.566**

- SW (i.e., removing spanish stop words) and NUM (i.e., substitute numbers with a constant string) leads to performance improvement to all the model respect to no pre-processing at all,
- ST (i.e., stemming) and HT (i.e., substitute hashtags with a constant string) decrease the performance of both models respect to no-preprocessing at all,

Table 4: Pre-processing study comparing 10-fold cross validation results over the development set in terms of percentuage of $F_{1-macro}$ score. For each model the best performing word embeddings configuration has its result in bold.

Models	Text representation				
	Non-static	CA static	ES static	CA non-static	ES non-static
Kim	0.541	0.345	0.550	0.555	**0.589**
FastText	0.556	0.351	0.450	0.509	**0.579**

Analysing results in Table 4, here the used notation refers to the one introduced in Section 3.2, where the listing of a notation means its use as embedded input layer for the reported result. From its analysis the following interpretation can be drawn:

- Setting as *static* the sentence matrix weights has the worst performance (independently of the used language)
- As opposite to the previous point, setting as *non-static* leads to better performance, where this insight can be deduced by corpus characteristic (i.e., a good example of Computer Mediated Communication)
- The use of pre-trained embedding is useful in combination with *non-static* weights (i.e., best performances with ES non-static)
- Even if is not available a pre-trained embedding for the task language, the use of a similar language with non-static weight (i.e., CA non-static) can increase the performance respect only to non-static. This can be interpreted as a case of transfer learning.

Overview some text here

5 Conclusions

In this paper we have presented our participation in the IberEval2017 Classification Of Spanish Election Tweets (COSET) shared task. Five distinct neural models were explored, in combination with different types of preprocessing and text representation. From the systems evaluation it wasn't possible to find a combination of pre-processing that gives the best performance for all the models, meaning that each model is highly sensible to the pipeline combination. Regarding the analysed text representation, the setting of sentence matrix to non-static always leads to good performance as a result of the specific text

Table 5: Pre-processing study comparing 10-fold cross validation results over the development set in terms of percentuage of $F_{1-macro}$ score.

System	$F_{1-macro}$
LSTM	0.556 ($\pm$ 0.012)
Bi-LSTM	0.555 ($\pm$ 0.035)
CNN	0.571 ($\pm$ 0.030)
FastText	**0.589** ($\pm$ 0.018)
Kim	0.579 ($\pm$ 0.009)

under observation (i.e., a CMC corpus). Moreover, the use of pre-trained word embedding is always suggested even when not available of the language under observation but of a similar language (i.e., is possible to take advantage of transfer learning between similar languages). Moreover, we outline a not so promising performance of the recurrent model, meaning that for this task the word order (a feature well captured by LSTM family model) seams not so prominent as other tasks.

References

1. Kim, Yoon. "Convolutional neural networks for sentence classification." arXiv preprint arXiv:1408.5882 (2014).
2. Joulin, Armand, et al. "Bag of tricks for efficient text classification." arXiv preprint arXiv:1607.01759 (2016).
3. Edward Loper and Steven Bird. 2002. NLTK: the Natural Language Toolkit. In Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics - Volume 1 (ETMTNLP '02), Vol. 1. Association for Computational Linguistics, Stroudsburg, PA, USA, 63-70. DOI=http://dx.doi.org/10.3115/1118108.1118117
4. Bojanowski, Piotr and Grave, Edouard and Joulin, Armand and Mikolov, Tomas. "Enriching Word Vectors with Subword Information" arXiv preprint arXiv:1607.04606 (2016).
5. Harris, Zellig S. "Distributional structure." Word 10.2-3 (1954): 146-162.
6. Bazzanella, Carla. "Oscillazioni di informalit e formalit: scritto, parlato e rete." Formale e informale. La variazione di registro nella comunicazione elettronica. Roma: Carocci (2011): 68-83.
7. Cerruti, Massimo, and Cristina Onesti. "Netspeak: a language variety? Some remarks from an Italian sociolinguistic perspective." Languages go web: Standard and non-standard languages on the Internet (2013): 23-39.
8. Zhang, Ye, and Byron Wallace. "A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification." arXiv preprint arXiv:1510.03820 (2015).
9. C. Bosco, M. Lai, V. Patti, F. Rangel, P. Rosso (2016) Tweeting in the Debate about Catalan Elections. In: Proc. LREC workshop on Emotion and Sentiment Analysis Workshop (ESA), LREC-2016, Portoro, Slovenia, May 23-28, pp. 67-70.
10. F. Rangel, P. Rosso, B. Verhoeven, W. Daelemans, M. Potthast, B. Stein (2016) Overview of the 4th Author Profiling Task at PAN 2016: Cross-Genre Evaluations. In: Balog K., Cappellato L., Ferro N., Macdonald C. (Eds.) CLEF 2016 Labs and Workshops, Notebook Papers. CEUR Workshop Proceedings. CEUR-WS.org, vol. 1609, pp. 750-784.
11. Mohammad, Saif M., Parinaz Sobhani, and Svetlana Kiritchenko. "Stance and sentiment in tweets." arXiv preprint arXiv:1605.01655 (2016).

12. Mohammad, Saif M., et al. "Semeval-2016 task 6: Detecting stance in tweets." Proceedings of SemEval 16 (2016).
13. Fisher Yu,Vladlen Koltun , Multi-Scale Context Aggregation by Dilated Convolutions