ALC

Luca Ambrosini and Giancarlo Nicolò

Univesitat Politècnica De València

Abstract. Version one
This paper describes an explorative approach to create from scratch a set of baselines for tackling tweet classification problems using natural language processing and machine learning techniques. Our approach focuses on neural network models taken from state of the art that exploit different corpus preprocessing, tweets representations and features extraction methods. Finally, we will discuss how we applied this methodology to the *Classification Of Spanish Election Tweets (COSET)* task at IberEval 2017 and present the results we obtained
Version two
This paper describes our participation in the *Classification Of Spanish Election Tweets (COSET)* task at IberEval 2017. During the searching process for the best classification system to send for the competition, we developed a comparative study over possible combination of corpus preprocessing, text representations and classification models. After an initial models exploration, we focus our attention over specific neural models. Interesting insight can be drawn from the comparative study helping future practitioner tackling tweets classification problems to create system baseline for their work.
Possible add for the report in general and
handcrafted features
explain that we haven't put a proper module for handcrafted features, but our modular approach to solve the classification problem lead to an easy customization and integration of a proper module for these handcrafted features.

1 Introduction

Intro over nlp and text classification: motivation why this field is important and the actual challenge and open problems (will be top just to list some problem and their way to be solved, for example citing the different workshop that we had during the course)
Text classification is an important task in Natural Language Processing with many applications, such as web search, information retrieval, ranking and document classification (Deerwester et al., 1990; Pang and Lee, 2008). Recently, models based on neural networks have become increasingly popular (Kim, 2014; Zhang and LeCun, 2015; Conneau et al., 2016). While these models achieve very good performance in practice, they tend to be relatively slow both at train and test time, limiting their use on very large datasets.
The
vedi citazioni su word embedding di mark e bag of word da maite
Introduce informally our task and list some way of tackling this problem from the different perspectives, maybe say that we de-construct tthe actual approach and cathegorized their inner process in: representation, preprocessing, model and post processing (that we haven't done).
Finally explain how we structured this report
In the following we firstly describe the
In this paper we describe our participation for addressing the PR-SOCO task. The rest of the paper is organized as follows. Next section is devoted to de
ne the Personality Recognition task. In Section 4 the model proposed is described. Following, in Section 5, the results achieved are presented. Finally, in Section 6 our results are discussed, and future work is proposed.

2 Task definition

The COSET shared task aim was to classify Spanish written tweets talking about the 2015 Spanish General Election, where each tweet had to be classified into one of five different categories: (i) political issues, related to the most abstract electoral confrontation; (ii) policy issues, about sectorial policies; (iii) personal issues, on the life and activities of the candidates; (iv) campaign issues, related with the evolution of the campaign; (v) and other issues.
We analysed the given trainig data and find the following statistical information (that guided us in the baselines building):

- Average train sequence length:
  - 135 chars
  - 24 words
- Max train sequence length:
  - 140 chars
  - 49 words

3 Systems description

giancarlo modifications: I've changed a little bit the structure to create a kind of flux over the contents. The main idea is to switch the mind set from METHOD → (CLASSIFICATION) SYSTEM, in this way we can report our work from a software engineer point of view, analysing each part/process as a module. It seams stupid, but in this way I have a name fro each step of the METHOD/SYSTEM and I suppose can help the readers to better understand what we have done.
In this section we are going to describe the tweets classification systems built, from a module perspective we can describe our systems as composed of three main blocks: text pre-preprocessor (section 3.2), text representation (section 3.3) and classification model (section 3.4).

3.1 Initial investigation

To address the tweets classification problem we began our investigation analysing some of the most widely used text representations and classifiers. We began analysing possible text representations focusing our attention on lexical features based: *Bag Of Words* [6],*Bag Of N-Grams* (bigrams and trigrams), both with and without *term frequency-inverse document frequency* normalization (i.e. TF-IDF norm). Regarding possible classifiers exploiting above representations, we analysed *Random Forest, Decision Trees, Support Vector Machines* and *Multi Layer Perceptron*, but since the results obtained with the combination of those *model + representation* were outperformed by neural network based models, we decided not to report their analysis in this paper, but rather focus the modules description in relation to neural models.

3.2 Text preprocessing

We explored different combinations of twitter pre-processing, as converting some elements such mentions, emoji, smiley, hashtags into constant string (i.e. Tokenize @Ambros and #atoppe :) → Tokenize $MENTION and $HASHTAG $SMILEY); removeing elements as URLs, reserved words and numbers. We also measured the contribution of stemming, stopwords and punctuation removal. To address those pre-processing we used the following [3] [4].

3.3 Text representation

As a tweet representation used as first layer
To represent to tweet fed into the classifier we decided to used word embeddings,which is a tecnique where elements (in our case words and n-grams) are mapped into a vector of real numbers. The whole sentence is mapped into a matrix with dimension *sentence lenght* × *embedding dimension*. We left the sentence length as a parameter and the best results were obtained with length=30, that's reasonable since the average tweet length in words in 24. If the sentence dimension is bigger than the selected window we used the first length words, if the sentence is smaller than the windwos we used 0-right padding.

Words vectors Since the number of tweets is considered small to learn vector representation of words, we tried to initialize the embedding matrix with pretrained word vectors. In particular we used vectors trained on wikipedia using fastText [5]. We tried static pretrained vectors, learning them during training starting from a random matrix or from the pretrained embeddings.

N-gram embedding We also tried to learn an embedding for n-grams (bigrams and trigrams), but, since the corpus is small, n-grams frequencies are very low and the algorithm is not able to learn a valid embedding. Also there are no pre-trained n-grams embeddings available. Bigrams brought a small improvement, bigger n-grams result in performance decrease.

3.4 Classification models

In the following section we present different neural network models, each model has as first layer an embedding layer that maps words to the corresponding word vectors, as described in the previous section.

Convolutional Neural Network Convolutional Neural Networks are considered state of the art in many text classification problem. (GIANCARLO, citazioni?) This model is composed by a convolutional layer, followed by a global max pooling layer and two fully connected layers.

Long short-term memory LSTM is a type of Recurrent Neural Network that is relatively insensitive to gap length, compared to others RNN, they are considered state of the art in many NLP topics, like machine translation. In this model the classical embedding layer is followed by an LSTM layer with 128 units, terminated by a fully connected layer. We also tried Bidirectional LSTM, since they are bringing an improvement in different tasks, but they are performing worst on our use case.

Fast text This model is based on this paper [2]. Differently to the other models where just words are used, here Bag of n-grams are added to the input as additional features to capture some partial information about the local word order. It this model the embedding output is directly fed into a *GlobalAveragePooling* layer, that transofrms the whole sentece in a single vector computed as the average of the word vectors. This vector is then projected into 2 fully connected layers. In the paper the number of hidden layers is fixed to 10, but we measured better performance using just 2 layers. **@ Luca: Source:**

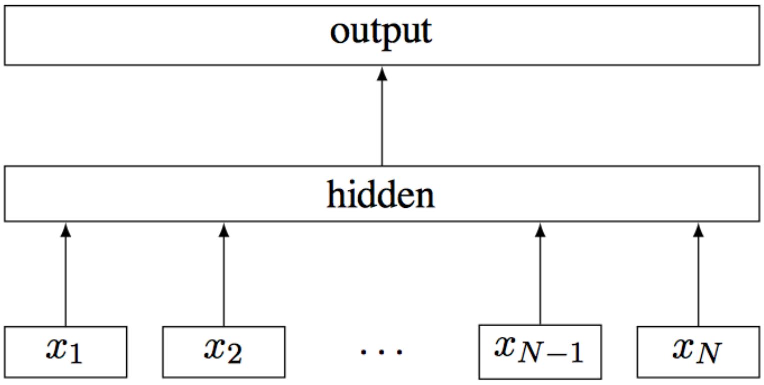

Fig. 1: [2] Model architecture of *fastText* for a sentence with N n-gram features $x_1, \ldots, x_N$. The features are embedded and averaged to form the hidden variable.

KIM. @ Luca: Source: Zhang, Y., & Wallace, B. (2015). A Sensitivity Analysis of (and Practitioners Guide to) Convolutional Neural Networks for Sentence Classification. http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/
@giancarlo: Non penso che ants_example vada esattamente qui :D
This model is based on this paper [1]. It is based on different filter region size convolutions, followed by maxpooling layers. All those outputs are concatenated to build sentence representation that is finally projected into a dense layer for the classification. The intuition behind this model is that smaller filter region size should be able to capture short sentence patterns (similar to ngrams), while bigger sizes should capture sentence level features. We reached the best performance using [2, 3, 5, 7] as filter sizes.

4 Evaluation

4.1 Metrics

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

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

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

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

4.2 Results

Put an intro over the results that will be analyzed and then comments over them
table of only the 5 models main models and 6 kind of preprocessing only the mean and say that is 10 fold CV
table of word representation+
HERE legend of notation in bf the winner about the preprocessing configuration and in shared grey box the best preprocessing configuration for a model.

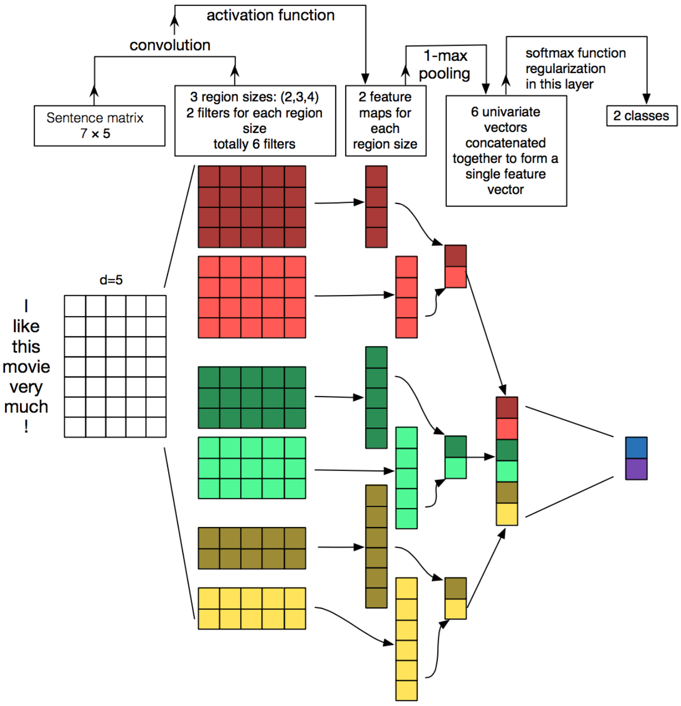

Fig. 2: Illustration of a Convolutional Neural Network (CNN) architecture for sentence classification

–

we don't add an average column
Eleven teams have presented their respective systems. In total, 48 systems were submitted for evaluation. All the systems we have submitted have performed better than the mean of the systems proposed using the RMSE

5 Conclusions

Transfer learning lstm
stemming online learning
expert modelling kim model
leave trainable the vector embedding is always positive even though you already have a trained vector embedding representation.
transfer learning
In this paper we have presented our participation in the PAN@FIRE Personality Recognition in Source Code 2016 shared task. Two approaches were proposed an Author-Based approach and a Code-Based approach. The AB approach performed better for all the traits. This could be explained because the samples we used to train the systems that followed the Code-Based approach were not independent. Therefore, the results we obtained in the development

Table 1: Available test set result over the $F_{1-macro}$ score [%].

Preprocessing	Models				
	CNN	LSTM	B-LSTM	FAST-TEXT	KIM
Nothing	50,5	55,6	51,1	53,0	**55,8**
ST	49,6	49,9	47,5	**53,1**	**53,1**
ST+SW	47,6	**55,3**	47,6	52,9	51,1
ST+SW+CL	51,9	**56,8**	52,2	56,0	50,8
ST+SW+CL+MT	54,6	**56,1**	47,7	54,6	53,6
ST+SW+CL+MT+NUM	53,2	**55,5**	51,0	53,4	51,5
ST+SW+CL+MT+NUM+EM	52,7	**56,4**	53,7	54,2	51,8
ST+SW+CL+MT+NUM+EM+HT	55,1	54,0	52,9	**56,7**	51,1
SW+CL+MT+NUM+EM	54,5	54,8	53,9	**57,0**	54,8
CL	55,9	43,4	48,5	56,5	**57,7**
CL+EM	57,1	52,1	49,9	54,8	**58,9**
CL+MT+NUM+EM	54,3	54,6	55,5	**56,8**	54,8

Table 2: Available test set result over the $F_{1-macro}$ score [%].

Embedding	Models				
	CNN	LSTM	B-LSTM	FAST-TEXT	KIM
ES	**48,10**	36,13	38,90	36,37	43,57
ES+Online	57,10	52,10	49,93	54,80	**58,87**
CA	**45,07**	30,57	38,53	30,13	39,60
CA+Online	53,57	53,63	46,27	55,13	**56,23**
Online	52,63	47,33	51,80	53,30	**54,70**

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. Preprocessor is a preprocessing library for tweet data written in Python, https://github.com/s/preprocessor
5. Bojanowski, Piotr and Grave, Edouard and Joulin, Armand and Mikolov, Tomas. ”Enriching Word Vectors with Subword Information” arXiv preprint arXiv:1607.04606 (2016).
6. Harris, Zellig S. ”Distributional structure.” Word 10.2-3 (1954): 146-162.