ALC

Luca Ambrosini and Giancarlo Nicolò

Univesitat Politècnica De València

Abstract. This paper describes a methodology to create from scratch a set of baselines for tackling text 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

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

2 Task definition

TODO
problem over the tweet (chiedi alla tipa di torino i tweet challenges)
First draft
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 Methods

To address this text classification task we first tried the most widely used text representations and classifiers. We tried representations based on lexical features as Bag Of Words [**?**],Bag Of N-Grams (bigrams and trigrams), both with and without with TF-IDF normalization. As classifiers we tried Random Forest, Decision Trees, Support Vector Machines and MultiLayer Perceptron, but since that results obtaines with the combination of those tecniques were outperformed by neural models, we decided not to report those in the paper. Bag of words

4 Representation

vedi citazioni su word embedding di mark e bag of word da maite
si ottimizzato il parametro del numero di parole da usare come input alla rete neurale, arrivando alla misura di 30
numeri superiore risultano in un eccesso di padding e perdita di informazione della rete. Numeri inferiori perdono troppa informazione

4.1 Word embedding

Word embedding is a tecnique where elements (in out case words and n-grams) are mapped into a vector of real numbers. The sentence is mapped into a matrix with dimension sentence lenght x 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. We tried static pretrained vectors [**?**],learning them during training starting from a random matrix or from the pretrained embeddings. GIANCARLO DEVO DIRE CHE ESPLORIAMO TUTTE LE COMBINAZIONI?

4.2 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 available. Bigrams brought a small improvement, bigger n-grams result in performance decrease.

5 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].

6 Models

6.1 Neural models

Start whith the meaning reason behind this model (e.g. LSTM because in translation is a state of the art).
Long short term memory. Bidirectional long short term memory.
Convolutional neural network. Hybrid convolutional neural network.

Fast text. same and equal of original but changing the number of layer (worstening in increasing the number of layer) + adding of the gaussian noise and batch normalization

KIM. Pippo [1]
from kim model we optimize ttill the following net configuration/topology: description of the net

7 Evaluation

7.1 Metrics

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

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

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

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

7.2 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.
–

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

8 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

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