LyS at SemEval-2016 Task 4: Exploiting Neural Activation Values for Twitter Sentiment Classification and Quantification[*]

David Vilares[a], Yerai Doval[a,b], Miguel A. Alonso[a] and Carlos Gómez-Rodríguez[a]
[a]Grupo LyS, Departamento de Computación, Universidade da Coruña
Campus de A Coruña s/n, 15071, A Coruña, Spain
[b]Grupo COLE, Departamento de Informática,
E.S. de Enxeñaría Informática, Universidade de Vigo
Campus As Lagoas, 32004, Ourense, Spain
{david.vilares, yerai.doval, miguel.alonso, carlos.gomez}@udc.es

Abstract

In this paper we describe our deep learning approach for solving both two-, three- and five-class tweet polarity classification, and two- and five-class quantification. We first trained a convolutional neural network using pretrained Twitter word embeddings, so that we could extract the hidden activation values from the hidden layers once some input had been fed to the network. These values were then used as features for a support vector machine in both the classification and quantification subtasks, together with additional linguistic information in the former scenario. The results obtained for the classification subtasks show that this approach performs better than a single convolutional network, and for the quantification part it also yields good results. Official rankings locate us: 2nd (practically tied with 1st) for the binary classification task, 2nd for binary quantification and 4th (practically tied with 3rd) for the five-class polarity classification challenge.

1 Introduction

Opinion mining has become an important mechanism to monitor what people are saying about a variety of topics (Cambria et al., 2013). As an example, Thelwall et al. (2011) use SentiStrength to monitor popular events on Twitter (e.g. the Oscars, the SuperBowl), showing how these resonate among the public. In a similar line, Vilares et al. (2015) also use Twitter to measure the level of popularity of the main Spanish political leaders, proving that the results obtained by their systems are similar to the ones obtained by traditional polls.

Opinion mining on Twitter actually involves two different challenges: (1) analyzing the characteristics of individual tweets and (2) quantifying a given set of tweets so that useful statistics can be extracted. This paper describes our different models to overcome such challenges, using the SemEval 2016:Task 4 as the evaluation framework.

2 Sentiment Analysis in Twitter

The SemEval organization proposed two different types of challenges in its 2016 edition: (1) classification into two, three and five classes and (2) quantification into two and five categories. A detailed explanation of the task can be found in the description paper (Nakov et al., 2016). For all subtasks, three official splits are provided: training, development and development test sets. In this paper, we use the training and development sets for training, and the development test set for evaluation.[1]

2.1 Convolutional Neural Network

As a starting point, we train a deep neural network (DNN), in particular a convolutional neural network (CNN), following a similar configuration to the one

[*]This research is supported by the Ministerio de Economía y Competitividad (FFI2014-51978-C2). David Vilares is funded by the Ministerio de Educación, Cultura y Deporte (FPU13/01180). Yerai Doval is funded by the Ministerio de Economía y Competitividad (BES-2015-073768). Carlos Gómez-Rodríguez is funded by an Oportunius program grant (Xunta de Galicia).

[1]For classification into 3 polarities, we include the training set of SemEval 2013 as part of our training set and its development set as a part of our collection for tuning.

used by Severyn and Moschitti (2015). Figure 1 illustrates the topology of the CNN from where we will extract the hidden activation values.

2.1.1 Embeddings layer

Let w be a token of a vocabulary V, a word embedding is a distributed representation of that token as a low dimensional vector $v \in \mathbb{R}^n$. In that way, it is possible to create a matrix of embeddings, $E \in \mathbb{R}^{|V| \times n}$, to act as the input layer to the CNN. Particularly, we rely on a collection of Twitter word embeddings pretrained with Glove[2] (Pennington et al., 2014) with $|V| \approx 10^6$ and n=100.

Thus, given a tweet t=$[w_1, w_2, ..., w_t]$, after running our input layer we will obtain a matrix $T \in \mathbb{R}^{|t| \times n}$ that will serve as the input to the convolutional layer. Since tweets might have variable length, $|t|$ is set to 100, padding with zeros if the tweet is shorter and taking the first 100 words if it is longer. We have realized after the evaluation that this value might be not the best option for short texts, such as tweets, and we plan to optimize this parameter empirically. To avoid overfitting, we first apply dropout (Srivastava et al., 2014), which randomly sets to zero the activation values of x% of the neurons in a given layer (in this paper, $x = 50$).

2.1.2 Convolutional Layer

A convolutional layer exploits local correlations in the input data. In the case of text as input, this translates into extracting correlations between groups of word or character n-grams in a sentence. To do so, each hidden unit of the CNN will only respond (activate) to a specific continuous slice of the input text. This is implemented on `http://keras.io` using convolutional operations with m convolutional filters of width f separately applied to the input, obtaining m representations of this input usually known as feature maps.

Formally, let $T \in \mathbb{R}^{|t| \times n}$ be the matrix embedding for the tweet t and let $F \in \mathbb{R}^{f \times n}$ be a filter, the output of a wide convolution is a matrix $C \in \mathbb{R}^{m \times (|t|+f-1)}$, where each $c_i \in \mathbb{R}^{|t|+f-1}$ is defined as:

$$C_i = \sum_{j,k} T_{[i-f+1:i,:]} \otimes F \qquad (1)$$

[2]http://nlp.stanford.edu/data/glove.twitter.27B.zip

and where $\otimes$ is the element-wise multiplication, $1 < i < m$; and j and k are the rows and columns of the matrix $T_{[i-f+1:i,:]} \otimes F \in \mathbb{R}^{f \times n}$. The non valid rows of T ($T_{(i,:)}$ with $i < 0$) are set to zero.

Following Severyn and Moschitti (2015), in this paper we chose $f = 5$ and $m = 300$. We also rely on $ReLU(x) = max(0, x)$ as the non-linear activation function. To avoid overfitting we incorporate a L2 regularization of 0.0001. After that, a max pooling layer selects $max(ReLU(c_i))$ for each feature map.

2.1.3 Output layer

The output of the pooling layer is then passed to a fully connected layer ($\mathbb{R}^{100}$). We add again dropout (50%) and a ReLU as the activation function. Finally, an additional fully connected layer reduces the dimensionality of the input to fit the output (number of classes) and as the final step we apply a softmax function to make the final prediction.

2.1.4 Current limitations

Obtaining an accurate deep neural network can be a very slow process. Hyper-parameter engineering is often needed, but training a single DNN with its hyper-parameters can be painfully slow without enough computational resources. Additionally, *distant supervision* is also recommended to pretrain the network (Go et al., 2009; Severyn and Moschitti, 2015). These two issues act as limitations that we could not overcome at the moment. We did try pretraining, but at the moment, we did not achieve improvements over the CNN without pretraining. A preliminary analysis suggests that: (1) we need more tweets to exploit distant supervision, (2) fine hyper-parameter engineering needs to be explored to ensure that the fine-tuning on the labeled data does not completely overwrite what the network has already learned and (3), it is easy to collect tweets for analysis into 2 classes, but downloading non-noisy tweets for analysis into 3 and 5 classes is a more challenging issue.

In the following section we show how to exploit the hidden activation values of our deep learning model as part of a supervised system (Poria et al., 2015), when pretraining and fast hyper-parameter engineering are not feasible options.

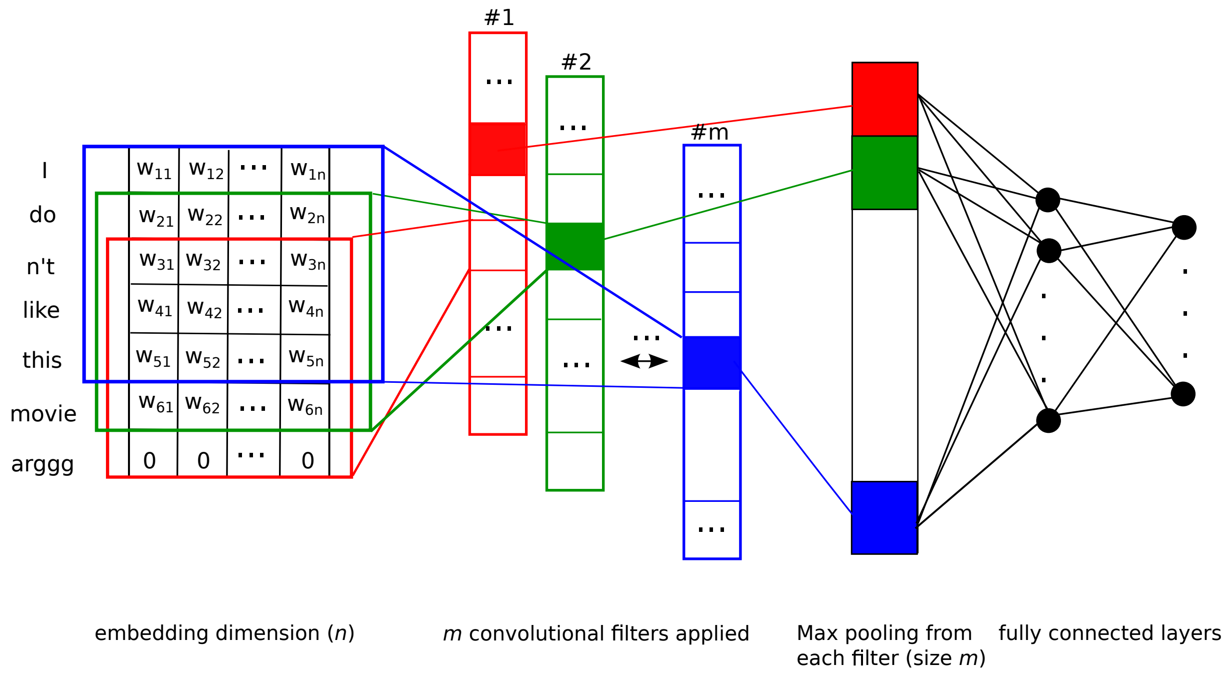

Figure 1: Topology of our CNN from where we will extract the neural activation values

2.2 Classification

Let $S=\{s_1, ..., s_n\}$ be a set of tweets and let $L=\{l_1, ..., l_n\}$ be a set of labels, the classification subtask can be defined as designing a hypothesis function $h_\Theta : S \rightarrow L$, where Θ denotes a set of features representing the texts. We build functions to solve classification into five (strong positive (P+), positive (P), neutral (NEU), negative (N) and strong negative (N+)), three (P, NEU and N) and two (P and N) classes. We rely on a support vector machine (SVM), in particular on a LibLinear (Fan et al., 2008) implementation with L2-regularization, to train our supervised model.[3] As features, we started testing some of those from of our last SemEval system (Vilares et al., 2014), using the total occurrence as the weighting factor. Information gain (IG) is used in all cases. Thus, before training our classifier we run an information gain algorithm to remove all irrelevant features, i.e. those where IG=0:

- *Words* (W): Each single word is considered as a feature to feed the supervised classifier.
- *Psychometric properties* (P): Features extracted from psychological properties coming from LIWC (Pennebaker et al., 2001) that relate terms with psychometric properties (*e.g. anger* or *anxiety*) or topics (*e.g. family* or *religion*).
- *Part-of-speech tags* (T).

Additionally, this year we have included:

- *The last word of the tweet* (LW): The last term of each tweet is used as a separate feature.
- *The psychometric properties of the last word of the tweet* (LP).
- *Hidden activation values from the* CNN (HV): We take the hidden activation values of the last hidden layer.
- *Features extracted from sentiment dictionaries*: We extract the total, maximum, minimum and last sentiment score of a tweet from the Sentiment140 (Mohammad et al., 2013), Hu and Liu (2004) and Taboada et al. (2011) subjective lexica.

2.2.1 Experimental results

Table 1 shows the experimental results for classification into two classes obtained using the SVM with different feature sets and the CNN. The neural network outperforms most of the SVM approaches. Only when we combine a number of linguistic features with the hidden activation values and we

[3]We used Weka (Hall et al., 2009) to build the models.

Features	Recall-P	Recall-N	Macro avg. R
HV.P.D.LW.LP.FT*	0.721	0.803	**0.762**
HV.P.D.LW.LP.FT	0.856	0.581	0.719
HV	0.864	0.560	0.712
P	0.953	0.192	0.573
W	0.969	0.162	0.566
D	0.892	0.249	0.564
CNN	0.802	0.671	0.737

Table 1: Classification into two classes using the development test set 2016. We include feature models that include hidden activation values (HV), words (W), psychometric (P), sentiment dictionaries (D), last word of the tweet (LW) and last psychometric properties (LP). The dot indicates a model that combines those features. * indicates a model where the class weights have been tuned. We compared them against our CNN.

weight the classes, we obtain an improvement over the CNN. We believe that by applying fine hyper-parameter tuning on the CNN we will be able to further improve these results. Similar conclusions can be extracted from the classification into three classes, whose results are shown in Table 2. Finally, Table 3 details the results for the five categories classification subtask. In this case, the neural network does not perform as good as in previous scenarios.

Features	F1-P	F1-Neu	F1-N	Macro avg. F1
HV.P.FT.D.LW.LP*	0.676	0.520	0.538	0.598
HV.P.FT.D.LW.LP	0.664	0.565	0.483	0.576
HV	0.659	0.574	0.469	0.564
P	0.620	0.524	0.353	0.487
W	0.611	0.614	0.327	0.469
D	0.613	0.553	0.302	0.458
CNN	0.674	0.493	0.489	0.582

Table 2: Classification into three classes using both the development test set 2016 and the development set 2013. Macro-averaged F1-measure of positive and negative tweets is used to rank the models.

Features	F1-P+	F1-P	F1-Neu	F1-N	F1-N+	MAE
HV.P.FT.D.LW.LP*	0.277	0.621	0.439	0.296	0.237	**0.83**
HV.P.FT.D.LW.LP	0.098	0.689	0.439	0.304	0.063	0.93
HV	0.000	0.690	0.417	0.277	0.000	0.95
P	0.000	0.676	0.246	0.070	0.000	1.21
W	0.016	0.674	0.227	0.059	0.000	1.28
CNN	0.000	0.703	0.361	0.229	0.000	1.03

Table 3: Classification into five classes using the development test set 2016. Macro-averaged absolute error (MAE) is used to rank the models. F1-measure is used to show the performance over each class.

With respect to SVM-specific parameter optimization, cost parameter (C) and class weigths (w):

- *2 classes*: C=0.005, w_{negative}=2.25 and w_{positive}=0.25.
- *3 classes*: C=0.0001, w_{positive}=0.5, w_{neutral}=0.4 and w_{negative}=2.
- *5 classes*: C=1, $w_{\text{strong negative}}$=5.5, w_{negative}=1, $w_{\text{strong positive}}$=1.5, w_{positive}=0.25 and w_{neutral}=0.5.

2.3 Quantification

For this task we are not interested in predicting the class of each individual instance of the dataset, as in classification tasks, but the relative frequency of each class in whole groups of instances; this is, the *class distribution*. In this context, models trained using loss functions well suited for classification are not necessarily good enough for quantification, as the loss function we need to optimize for has changed just in the same way as the aim of the task, in relation to a classification task (Barranquero et al., 2015).

The most simple approach to tackle this problem would be *Classify and Count* (CC) (Forman, 2008), which estimates the class frequencies counting the positive results of a classifier for each class over the total amount of input instances. Nevertheless, more specialized methods exist, such as the use of an SVM learning algorithm paired with a nonlinear loss function such as the *Kullback-Leibler Divergence* (KLD) (Esuli and Sebastiani, 2015), which we have used in this work thanks to the tool *SVM*$_{perf}$ (Joachims, 2005) patched to work with KLD.[4]

The different feature sets tested for our quantification system were automatically obtained as the activation values from different layers of the convolutional network used in the classification subtasks of this workshop. The SVM model was trained with a linear kernel and no regularization bias, optimizing the KLD over the entire training dataset.

Finally, as our system deals specifically with binary quantification, we took a one-vs-all approach and trained multiple models to generalize the quantification process for n classes rather than just two

[4]http://hlt.isti.cnr.it/quantification/ .

(#) CNN layer	2 classes (KLD)	5 classes (EMD)
(4th) Max pooling	0.07	0.82
(5th) Linear	**0.06**	**0.55**
(6th) Dropout	**0.06**	**0.55**
(7th) ReLU	0.10	0.63
(8th) Linear	0.08	0.65
(9th) SoftMax	**0.06**	1.49
CC	0.26	2.10

Table 4: Quantification into two and five classes using the development test set 2016. Rows: CNN layer from where the activation values were extracted. The last one shows the CC baseline. Columns: system performance measured as KLD for two classes and Earth Mover's Distance (EMD) for five classes.

Test set	Subtask	Score	Ranking
Twitter 2016	A	0.575	(16/34)
Twitter 2016	**B**	**0.791**	**(2/19)**
Twitter 2016	**C**	**0.860**	**(4/11)**
Twitter 2016	**D**	**0.053**	**(2/14)**
Twitter 2016	E	0.360	(7/10)
Twitter 2013	A	0.650	(9/34)
SMS 2013	A	0.579	(13/34)
Twitter 2014	A	0.647	(13/34)
Twitter Sarcasm 2014	A	0.406	(18/34)
Live Journal 2014	A	0.655	(11/34)
Twitter 2015	A	0.603	(12/34)

Table 5: Overall ranking on the test sets following the official metric for each task.

classes. The results obtained later by these models were normalized so that relative frequencies sum up to one.

2.3.1 Experimental results

Results obtained using neural activation values chosen from particular layers of our convolutional network as features for the SVM can be found in Table 4. As our baseline, we performed a CC on the results obtained from the best classifiers from the classification subtasks.

3 Results on the gold test set

Table 5 shows the scores and rankings of our systems for each subtask, according to the official metrics used for each challenge. A detailed report of the results for all participants can be found at Nakov et al. (2016) and the official website.[5]

4 Conclusions

We have described our approach to tackle the classification and quantification challenges proposed at Task 4 of SemEval 2016: Sentiment Analysis in Twitter. Official rankings locate us in top positions for binary classification and quantification and also for the 5-class polarity classification challenge.

We first trained a convolutional neural network to address the classification challenge. Additionally, we used its hidden activation values as features to train support vector machines, both for classification and quantification tasks. In light of the results obtained, we can state that our convolutional network seems to be a good feature extractor for both of these tasks.

As future work, we plan to exploit new distributed representations of the input to improve the performance of our current model. For the quantification task, we are planning on extending our experiments to8 other machine learning arquitectures, such as quantification trees (Milli et al., 2013) and different types of neural networks, and further exploring the feature domain using both handcrafted features and other continuous representation methods such as *doc2vec* (Le and Mikolov, 2014).

[5]http://alt.qcri.org/semeval2016/task4/index.php?id=results

References

J. Barranquero, J. Díez, and J. J. del Coz. 2015. Quantification-oriented learning based on reliable classifiers. *Pattern Recognition*, 48(2):591–604.

E. Cambria, B. Schuller, Y. Xia, and C. Havasi. 2013. New avenues in opinion mining and sentiment analysis. *IEEE Intelligent Systems*, (2):15–21.

A. Esuli and F. Sebastiani. 2015. Optimizing text quantifiers for multivariate loss functions. *ACM Trans. Knowl. Discov. Data*, 9(4):27:1–27:27, June.

R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin. 2008. LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

G. Forman. 2008. Quantifying counts and costs via classification. *Data Min. Knowl. Discov.*, 17(2):164–206, October.

A. Go, R. Bhayani, and L. Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1:12.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. 2009. The WEKA data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, nov.

M. Hu and B. Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM*

SIGKDD international conference on Knowledge discovery and data mining, pages 168–177. ACM.

T. Joachims. 2005. A support vector method for multivariate performance measures. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pages 377–384, New York, NY, USA. ACM.

Q. V. Le and T. Mikolov. 2014. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*.

L. Milli, A. Monreale, G. Rossetti, F. Giannotti, D. Pedreschi, and F. Sebastiani. 2013. Quantification trees. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 528–536. IEEE.

S. M. Mohammad, S. Kiritchenko, and X. Zhu. 2013. NRC-Canada: Building the State-of-the-Art in Sentiment Analysis of Tweets. In *Proceedings of the seventh international workshop on Semantic Evaluation Exercises (SemEval-2013)*, Atlanta, Georgia, USA, June.

P. Nakov, A. Ritter, S. Rosenthal, V. Stoyanov, and F. Sebastiani. 2016. SemEval-2016 task 4: Sentiment analysis in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016)*, San Diego, California, June. Association for Computational Linguistics.

J. W. Pennebaker, M. E. Francis, and R. J. Booth. 2001. Linguistic inquiry and word count: LIWC 2001. *Mahway: Lawrence Erlbaum Associates*, page 71.

J. Pennington, R. Socher, and C. D Manning. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*, volume 14, pages 1532–1543.

S. Poria, E. Cambria, and A. Gelbukh. 2015. Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis. In *Proceedings of EMNLP*, pages 2539–2544.

A Severyn and A Moschitti. 2015. UNITN: Training Deep Convolutional Neural Network for Twitter Sentiment Classification. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 464–469, Denver, Colorado. Association for Computational Linguistics.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational Linguistics*, 37(2):267–307.

M. Thelwall, K. Buckley, and G. Paltoglou. 2011. Sentiment in Twitter events. *J. Am. Soc. Inf. Sci. Technol.*, 62(2):406–418.

D. Vilares, M. Hermo, M. A. Alonso, C. Gómez-Rodríguez, and Y. Doval. 2014. LyS : Porting a Twitter Sentiment Analysis Approach from Spanish to English. In *Proceedings og The 8th InternationalWorkshop on Semantic Evaluation (SemEval 2014)*, number SemEval, pages 411–415.

D. Vilares, M. Thelwall, and M. A Alonso. 2015. The megaphone of the people? Spanish SentiStrength for real-time analysis of political tweets. *Journal of Information Science*, 41(6):799–813.