Tandem LSTM-SVM Approach for Sentiment Analysis

Andrea Cimino and **Felice Dell'Orletta**
Istituto di Linguistica Computazionale "Antonio Zampolli" (ILC–CNR)
ItaliaNLP Lab - *www.italianlp.it*
{andrea.cimino, felice.dellorletta}@ilc.cnr.it

Abstract

English. In this paper we describe our approach to EVALITA 2016 SENTIPOLC task. We participated in all the subtasks with constrained setting: Subjectivity Classification, Polarity Classification and Irony Detection. We developed a tandem architecture where Long Short Term Memory recurrent neural network is used to learn the feature space and to capture temporal dependencies, while the Support Vector Machines is used for classification. SVMs combine the document embedding produced by the LSTM with a wide set of general–purpose features qualifying the lexical and grammatical structure of the text. We achieved the second best accuracy in Subjectivity Classification, the third position in Polarity Classification, the sixth position in Irony Detection.

Italiano. *In questo articolo descriviamo il sistema che abbiamo utilizzato per affrontare i diversi compiti del task SENTIPOLC della conferenza EVALITA 2016. In questa edizione abbiamo partecipato a tutti i sotto compiti nella configurazione vincolata, cioè senza utilizzare risorse annotate a mano diverse rispetto a quelle distribuite dagli organizzatori. Per questa partecipazione abbiamo sviluppato un metodo che combina una rete neurale ricorrente di tipo Long Short Term Memory, utilizzate per apprendere lo spazio delle feature e per catturare dipendenze temporali, e Support Vector Machine per la classificazione. Le SVM combinano la rappresentazione del documento prodotta da LSTM con un ampio insieme di features che descrivono la struttura lessicale e grammaticale del testo. Attraverso questo sistema abbiamo ottenuto la seconda posizione nella classificazione della Soggettività, la terza posizione nella classificazione della Polarità e la sesta nella identificazione dell'Ironia.*

1 Description of the system

We addressed the EVALITA 2016 SENTIPOLC task (Barbieri et al., 2016) as a three-classification problem: two binary classification tasks (Subjectivity Classification and Irony Detection) and a four-class classification task (Polarity Classification).

We implemented a tandem LSTM-SVM classifier operating on morpho-syntactically tagged texts. We used this architecture since similar systems were successfully employed to tackle different classification problems such keyword spotting (Wöllmer et al., 2009) or the automatic estimation of human affect from speech signal (Wöllmer et al., 2010), showing that tandem architectures outperform the performances of the single classifiers.

In this work we used Keras (Chollet, 2016) deep learning framework and LIBSVM (Chang et al., 2001) to generate respectively the LSTM and the SVMs statistical models.

Since our approach relies on morpho-syntactically tagged texts, both training and test data were automatically morpho-syntactically tagged by the POS tagger described in (Dell'Orletta, 2009). In addition, in order to improve the overall accuracy of our system (described in 1.2), we developed sentiment polarity and word embedding lexicons[1] described below.

[1] All the created lexicons are made freely available at the following website: *http://www.italianlp.it/*.

1.1 Lexical resources

1.1.1 Sentiment Polarity Lexicons

Sentiment polarity lexicons provide mappings between a word and its sentiment polarity (positive, negative, neutral). For our experiments, we used a publicly available lexicons for Italian and two English lexicons that we automatically translated. In addition, we adopted an unsupervised method to automatically create a lexicon specific for the Italian twitter language.

Existing Sentiment Polarity Lexicons

We used the Italian sentiment polarity lexicon (hereafter referred to as $OPENER$) (Maks et al., 2013) developed within the OpeNER European project[2]. This is a freely available lexicon for the Italian language[3] and includes 24,000 Italian word entries. It was automatically created using a propagation algorithm and the most frequent words were manually reviewed.

Automatically translated Sentiment Polarity Lexicons

- The Multi–Perspective Question Answering (hereafter referred to as $MPQA$) Subjectivity Lexicon (Wilson et al., 2005). This lexicon consists of approximately 8,200 English words with their associated polarity. In order to use this resource for the Italian language, we translated all the entries through the Yandex translation service[4].

- The Bing Liu Lexicon (hereafter referred to as BL) (Hu et al., 2004). This lexicon includes approximately 6,000 English words with their associated polarity. This resource was automatically translated by the Yandex translation service.

Automatically created Sentiment Polarity Lexicons

We built a corpus of positive and negative tweets following the Mohammad et al. (2013) approach adopted in the Semeval 2013 sentiment polarity detection task. For this purpose we queried the Twitter API with a set of hashtag seeds that indicate positive and negative sentiment polarity. We selected 200 positive word seeds (e.g. "vincere" *to win*, "splendido" *splendid*, "affascinante" *fascinating*), and 200 negative word seeds (e.g., "tradire" *betray*, "morire" *die*). These terms were chosen from the OPENER lexicon. The resulting corpus is made up of 683,811 tweets extracted with positive seeds and 1,079,070 tweets extracted with negative seeds.

The main purpose of this procedure was to assign a polarity score to each n-gram occurring in the corpus. For each n-gram (we considered up to five n-grams) we calculated the corresponding sentiment polarity score with the following scoring function: $score(ng) = PMI(ng, pos) - PMI(ng, neg)$, where PMI stands for pointwise mutual information.

1.1.2 Word Embedding Lexicons

Since the lexical information in tweets can be very sparse, to overcame this problem we built two word embedding lexicons.

For this purpose, we trained two predict models using the word2vec[5] toolkit (Mikolov et al., 2013). As recommended in (Mikolov et al., 2013), we used the CBOW model that learns to predict the word in the middle of a symmetric window based on the sum of the vector representations of the words in the window. For our experiments, we considered a context window of 5 words. These models learn lower-dimensional word embeddings. Embeddings are represented by a set of latent (hidden) variables, and each word is a multidimensional vector that represent a specific instantiation of these variables. We built two Word Embedding Lexicons starting from the following corpora:

- The first lexicon was built using a tokenized version of the itWaC corpus[6]. The itWaC corpus is a 2 billion word corpus constructed from the Web limiting the crawl to the .it domain and using medium-frequency words from the Repubblica corpus and basic Italian vocabulary lists as seeds.

- The second lexicon was built from a tokenized corpus of tweets. This corpus was collected using the Twitter APIs and is made up of 10,700,781 italian tweets.

1.2 The LSTM-SVM tandem system

SVM is an extremely efficient learning algorithm and hardly to outperform, unfortunately these type

[2] http://www.opener-project.eu/

[3] https://github.com/opener-project/public-sentiment-lexicons

[4] http://api.yandex.com/translate/

[5] http://code.google.com/p/word2vec/

[6] http://wacky.sslmit.unibo.it/doku.php?id=corpora

of algorithms capture "sparse" and "discrete" features in document classification tasks, making really hard the detection of relations in sentences, which is often the key factor in detecting the overall sentiment polarity in documents (Tang et al., 2015). On the contrary, Long Short Term Memory (LSTM) networks are a specialization of Recurrent Neural Networks (RNN) which are able to capture long-term dependencies in a sentence. This type of neural network was recently tested on Sentiment Analysis tasks (Tang et al., 2015), (Xu et al., 2016) where it has been proven to outperform classification performance in several sentiment analysis task (Nakov et al., 2016) with respect to commonly used learning algorithms, showing a 3-4 points of improvements. For this work, we implemented a tandem LSTM-SVM to take advantage from the two classification strategies.

Figure 1 shows a graphical representation of the proposed tandem architecture. This architecture is composed of 2 sequential machine learning steps both involved in training and classification phases. In the training phase, the LSTM network is trained considering the training documents and the corresponding gold labels. Once the statistical model of the LSTM neural network is computed, for each document of the training set a document vector (document embedding) is computed exploiting the weights that can be obtained from the penultimate network layer (the layer before the SoftMax classifier) by giving in input the considered document to the LSTM network. The document embeddings are used as features during the training phase of the SVM classifier in conjunction with a set of widely used document classification features. Once the training phase of the SVM classifier is completed the tandem architecture is considered trained. The same stages are involved in the classification phase: for each document that must be classified, an embedding vector is obtained exploiting the previously trained LSTM network. Finally the embedding is used jointly with other document classification features by the SVM classifier which outputs the predicted class.

1.2.1 The LSTM network

In this part, we describe the LSTM model employed in the tandem architecture. The LSTM unit was initially proposed by Hochreiter and Schmidhuber (Hochreiter et al., 1997). LSTM units are able to propagate an important feature that came early in the input sequence over a long distance, thus capturing potential long-distance dependencies.

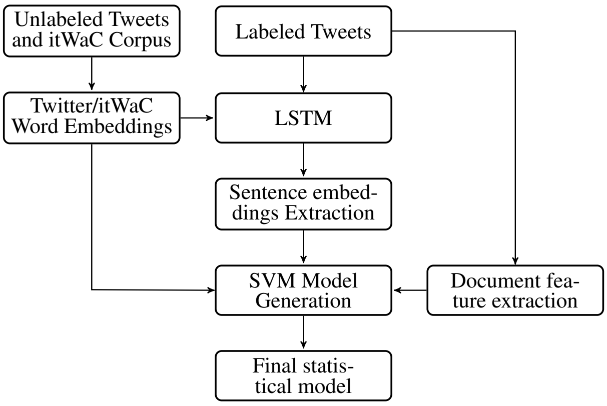
Unlabeled Tweets and itWaC Corpus
Labeled Tweets
Twitter/itWaC Word Embeddings
LSTM
Sentence embeddings Extraction
SVM Model Generation
Document feature extraction
Final statistical model

Figure 1: The LSTM-SVM architecture

LSTM is a state-of-the-art learning algorithm for semantic composition and allows to compute representation of a document from the representation of its words with multiple abstraction levels. Each word is represented by a low dimensional, continuous and real-valued vector, also known as word embedding and all the word vectors are stacked in a word embedding matrix.

We employed a bidirectional LSTM architecture since these kind of architecture allows to capture long-range dependencies from both directions of a document by constructing bidirectional links in the network (Schuster et al., 1997). In addition, we applied a dropout factor to both input gates and to the recurrent connections in order to prevent overfitting which is a typical issue in neural networks (Galp and Ghahramani , 2015). As suggested in (Galp and Ghahramani , 2015) we have chosen a dropout factor value in the optimum range $[0.3, 0.5]$, more specifically 0.45 for this work. For what concerns the optimization process, categorical cross-entropy is used as a loss function and optimization is performed by the rmsprop optimizer (Tieleman and Hinton, 2012).

Each input word to the LSTM architecture is represented by a 262-dimensional vector which is composed by:

Word embeddings: the concatenation of the two word embeddings extracted by the two available Word Embedding Lexicons (128 dimensions for each word embedding, a total of 256 dimensions), and for each word embedding an extra component was added in order to handle the "unknown word"

(2 dimensions).
Word polarity: the corresponding word polarity obtained by exploiting the Sentiment Polarity Lexicons. This results in 3 components, one for each possible lexicon outcome (negative, neutral, positive) (3 dimensions). We assumed that a word not found in the lexicons has a neutral polarity.
End of Sentence: a component (1 dimension) indicating whether or not the sentence was totally read.

1.2.2 The SVM classifier

The SVM classifier exploits a wide set of features ranging across different levels of linguistic description. With the exception of the *word embedding combination*, these features were already tested in our previous participation at the EVALITA 2014 SENTIPOLC edition (Cimino et al., 2014). The features are organised into three main categories: *raw and lexical text features*, *morpho-syntactic features* and *lexicon features*.

Raw and Lexical Text Features
Topic: the manually annotated class of topic provided by the task organizers for each tweet.
Number of tokens: number of tokens occurring in the analyzed tweet.
Character n-grams: presence or absence of contiguous sequences of characters in the analyzed tweet.
Word n-grams: presence or absence of contiguous sequences of tokens in the analyzed tweet.
Lemma n-grams: presence or absence of contiguous sequences of lemma occurring in the analyzed tweet.
Repetition of n-grams chars: presence or absence of contiguous repetition of characters in the analyzed tweet.
Number of mentions: number of mentions (@) occurring in the analyzed tweet.
Number of hashtags: number of hashtags occurring in the analyzed tweet.
Punctuation: checks whether the analyzed tweet finishes with one of the following punctuation characters: "?", "!".

Morpho-syntactic Features
Coarse grained Part-Of-Speech n-grams: presence or absence of contiguous sequences of coarse–grained PoS, corresponding to the main grammatical categories (noun, verb, adjective).
Fine grained Part-Of-Speech n-grams: presence or absence of contiguous sequences of fine-grained PoS, which represent subdivisions of the coarse-grained tags (e.g. the class of nouns is subdivided into proper vs common nouns, verbs into main verbs, gerund forms, past particles).
Coarse grained Part-Of-Speech distribution: the distribution of nouns, adjectives, adverbs, numbers in the tweet.

Lexicon features
Emoticons: presence or absence of positive or negative emoticons in the analyzed tweet. The lexicon of emoticons was extracted from the site *http://it.wikipedia.org/wiki/Emoticon* and manually classified.
Lemma sentiment polarity n-grams: for each n-gram of lemmas extracted from the analyzed tweet, the feature checks the polarity of each component lemma in the existing sentiment polarity lexicons. Lemma that are not present are marked with the *ABSENT* tag. This is for example the case of the trigram "tutto molto bello" (*all very nice*) that is marked as "*ABSENT-POS-POS*" because *molto* and *bello* are marked as positive in the considered polarity lexicon and *tutto* is absent. The feature is computed for each existing sentiment polarity lexicons.
Polarity modifier: for each lemma in the tweet occurring in the existing sentiment polarity lexicons, the feature checks the presence of adjectives or adverbs in a left context window of size 2. If this is the case, the polarity of the lemma is assigned to the modifier. This is for example the case of the bigram "non interessante" (*not interesting*), where "interessante" is a positive word, and "non" is an adverb. Accordingly, the feature "non_POS" is created. The feature is computed 3 times, checking all the existing sentiment polarity lexicons.
PMI score: for each set of unigrams, bigrams, trigrams, four-grams and five-grams that occur in the analyzed tweet, the feature computes the score given by $\sum_{\mathrm{i}-gram \in tweet} score(\mathrm{i}-gram)$ and returns the minimum and the maximum values of the five values (approximated to the nearest integer).
Distribution of sentiment polarity: this feature computes the percentage of positive, negative and neutral lemmas that occur in the tweet. To overcome the sparsity problem, the percentages are rounded to the nearest multiple of 5. The feature is computed for each existing lexicon.
Most frequent sentiment polarity: the feature returns the most frequent sentiment polarity of the lemmas in the analyzed tweet. The feature is com-

puted for each existing lexicon.

Sentiment polarity in tweet sections: the feature first splits the tweet in three equal sections. For each section the most frequent polarity is computed using the available sentiment polarity lexicons. The purpose of this feature is aimed at identifying change of polarity within the same tweet.

Word embeddings combination: the feature returns the vectors obtained by computing separately the average of the word embeddings of the nouns, adjectives and verbs of the tweet. It computed once for each word embedding lexicon, obtaining a total of 6 vectors for each tweet.

2 Results and Discussion

We tested five different learning configurations of our system: linear and quadratic support vector machines (linear SVM, quadratic SVM) using the features described in section 1.2.2, with the exception of the document embeddings generated by the LSTM; LSTM using the word embeddings described in 1.2.2; A tandem SVM-LSTM combination with linear and quadratic SVM kernels (linear Tandem, quadratic Tandem) using the features described in section 1.2.2 and the document embeddings generated by the LSTM. To test the proposed classification models, we created an internal development set randomly selected from the training set distributed by the task organizers. The resulting development set is composed by the 10% (740 tweets) of the whole training set.

Configuration	Subject.	Polarity	Irony
linear SVM	0.725	0.713	0.636
quadratic SVM	0.740	0.730	0.595
LSTM	0.777	0.747	0.646
linear Tandem	0.764	0.743	0.662
quadratic Tandem	**0.783**	**0.754**	**0.675**

Table 1: Classification results of the different learning models on our development set.

Table 1 reports the overall accuracies achieved by the classifiers on our internal development set for all the tasks. The accuracy is calculated as the F–score obtained using the evaluation tool provided by the organizers. It is worth noting that there are similar trends for what concerns the accuracies of the proposed learning models for all the three tasks. In particular, LSTM outperforms SVM models while the Tandem systems clearly outperform the SVM and LSTM ones. In addition, the quadratic models perform better than the linear ones. These results lead us to choose the linear and quadratic tandem models as the final systems to be used on the official test set.

Configuration	Subject.	Polarity	Irony
best official Runs	**0.718**	0.664	**0.548**
quadratic SVM	0.704	0.646	0.477
linear SVM	0.661	0.631	0.495
LSTM	0.716	**0.674**	0.468
linear Tandem*	0.676	0.650	0.499
quadratic Tandem*	0.713	0.643	0.472

Table 2: Classification results of the different learning models on the official test set.

Table 2 reports the overall accuracies achieved by all our classifier configurations on the official test set, the official submitted runs are starred in the table. The *best official Runs* row reports, for each task, the best official results in EVALITA 2016 SENTIPOLC. As can be seen, the accuracies of different learning models reveal a different trend when tested on the development and the test sets. Differently from what observed in the development experiments, the best system results to be the LSTM one and the gap in terms of accuracy between the linear and quadratic models is lower or does not occur. In addition, the accuracies of all the systems are definitely lower than the ones obtained in our development experiments. In our opinion, such results may depend on the occurrence of out domain tweets in the test set with respect to the ones contained in the training set. Different groups of annotators could be a further motivation for these different results and trends.

3 Conclusion

In this paper, we reported the results of our participation to the EVALITA 2016 SENTIPOLC tasks. By resorting to a tandem LSTM-SVM system we achieved the second place at the Subjectivity Classification task, the third place at the Sentiment Polarity Classification task and the sixth place at the Irony Detection task. This tandem system combines the ability of the bidirectional LSTM to capture long-range dependencies between words from both directions of a tweet with SVMs which are able to exploit document embeddings produced by LSTM in conjunction with a wide set of general-purpose features qualifying the lexical and grammatical structure of a text. Current direction of research is introducing a character based LSTM (dos Santos and Zadrozny, 2013) in the tandem system. Character based LSTM proven to be particularly suitable when analyzing social media texts (Dhingra et al., 2016).

References

Francesco Barbieri, Valerio Basile, Danilo Croce, Malvina Nissim, Nicole Novielli, Viviana Patti. 2016. Overview of the EVALITA 2016 SENTiment POLarity Classification Task. In Pierpaolo Basile, Anna Corazza, Franco Cutugno, Simonetta Montemagni, Malvina Nissim, Viviana Patti, Giovanni Semeraro and Rachele Sprugnoli, editors, *Proceedings of Third Italian Conference on Computational Linguistics (CLiC-it 2016) & Fifth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2016)*. December, Naples, Italy.

Chih-Chung Chang and Chih-Jen Lin. 2001. LIBSVM: a library for support vector machines. *Software available at http://www.csie.ntu.edu.tw/cjlin/libsvm*.

François Chollet. 2016. Keras. *Software available at https://github.com/fchollet/keras/tree/master/keras*.

Andre Cimino, Stefano Cresci, Felice Dell'Orletta, Maurizio Tesconi. 2014. Linguistically-motivated and Lexicon Features for Sentiment Analysis of Italian Tweets. In *Proceedings of EVALITA '14, Evaluation of NLP and Speech Tools for Italian*. December, Pisa, Italy.

Felice Dell'Orletta. 2009. Ensemble system for Part-of-Speech tagging. In *Proceedings of EVALITA '09, Evaluation of NLP and Speech Tools for Italian*. December, Reggio Emilia, Italy.

Bhuwan Dhingra, Zhong Zhou, Dylan Fitzpatrick, Michael Muehl, William Cohen. 2016. Tweet2Vec: Character-Based Distributed Representations for Social Media. In *Proceedings of the 54th Annual Meeting of the ACL*. Berlin, German.

Cìcero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning Character-level Representations for Part-of-Speech Tagging. In *Proc. of the 31st Inter. Conference on Machine Learning (ICML 2014)*.

Yarin Gal and Zoubin Ghahramani. 2015. A theoretically grounded application of dropout in recurrent neural networks. *arXiv preprint arXiv:1512.05287*

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. *Neural computation*

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '04*. 368-177, New York, NY, USA. ACM.

Isa Maks, Ruben Izquierdo, Francesca Frontini, Montse Cuadros, Rodrigo Agerri and Piek Vossen. 2014. Generating Polarity Lexicons with WordNet propagation in 5 languages. *9th LREC, Language Resources and Evaluation Conference*. Reykjavik, Iceland.

Tomas Mikolov, Kai Chen, Greg Corrado and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv1:1301.3781.

Saif Mohammad, Svetlana Kiritchenko and Xiaodan 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*. 321-327, Atlanta, Georgia, USA.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani and Veselin Stoyanov. 2016. SemEval-2016 task 4: Sentiment analysis in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45(11):2673–2681

Duyu Tang, Bing Qin and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of EMNLP 2015*. 1422-1432, Lisbon, Portugal.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-RmsProp: Divide the gradient by a running average of its recent magnitude. In *COURSERA: Neural Networks for Machine Learning.*

Theresa Wilson, Zornitsa Kozareva, Preslav Nakov, Sara Rosenthal, Veselin Stoyanov and Alan Ritter. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of HLT-EMNLP 2005*. 347-354, Stroudsburg, PA, USA. ACL.

Martin Wöllmer, Florian Eyben, Alex Graves, Björn Schuller and Gerhard Rigoll. 2009. Tandem BLSTM-DBN architecture for keyword spotting with enhanced context modeling *Proc. of NOLISP*.

Martin Wöllmer, Björn Schuller, Florian Eyben and Gerhard Rigoll. 2010. Combining Long Short-Term Memory and Dynamic Bayesian Networks for Incremental Emotion-Sensitive Artificial Listening *IEEE Journal of Selected Topics in Signal Processing*

XingYi Xu, HuiZhi Liang and Timothy Baldwin. 2016. UNIMELB at SemEval-2016 Tasks 4A and 4B: An Ensemble of Neural Networks and a Word2Vec Based Model for Sentiment Classification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*