Building the state-of-the-art in POS tagging of Italian Tweets

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 POS tagging for Italian Social Media Texts (PoSTWITA). We developed a two-branch bidirectional Long Short Term Memory recurrent neural network, where the first bi-LSTM uses a typical vector representation for the input words, while the second one uses a newly introduced word-vector representation able to encode information about the characters in the words avoiding the increasing of computational costs due to the hierarchical LSTM introduced by the character–based LSTM architectures. The vector representations calculated by the two LSTM are then merged by the sum operation. Even if participants were allowed to use other annotated resources in their systems, we used only the distributed data set to train our system. When evaluated on the official test set, our system outperformed all the other systems achieving the highest accuracy score in EVALITA 2016 PoSTWITA, with a tagging accuracy of 93.19%. Further experiments carried out after the official evaluation period allowed us to develop a system able to achieve a higher accuracy. These experiments showed the central role played by the handcrafted features even when machine learning algorithms based on neural networks are used.

Italiano. *In questo articolo descriviamo il sistema che abbiamo utilizzato per partecipare al task POS tagging for Italian Social Media Texts (PoSTWITA) della conferenza EVALITA 2016. Per questa partecipazione abbiamo sviluppato un sistema basato su due reti neurali parallele entrambi bidirezionali e ricorrenti di tipo Long Short Term Memory (LSTM). Mentre la prima rete neurale è una LSTM bidirezionale che prende in input vettori che rappresentano le parole in maniera tipica rispetto a precedenti lavori, la seconda prende in input una nuova rappresentazione vettoriale delle parole che contiene informazioni sui caratteri contenuti evitando un incremento del costo computazionale del sistema rispetto a LSTM che prendono in input rappresentazioni vettoriali delle sequenze di caratteri. Le rappresentazioni vettoriali ottenute dalle due LSTM vengono in fine combinate attraverso l'operatore di somma. Il nostro sistema, utilizzando come dati annotati solo quelli distribuiti dagli organizzatori del task, quando valutato sul test set uffciale ha ottenuto il miglior risultato nella competizione EVALITA 2016 PoSTWITA, riportando una accuratezza di 93.19%. Ulteriori esperimenti condotti dopo il periodo ufficiale di valutazione ci hanno permesso di sviluppare un sistema capace di raggiungre una accuratezza ancora maggiore, mostrandoci l'importanza dell'ingegnerizzazione manuale delle features anche quando vengono utilizzati algoritmi di apprendimento basati su reti neurali.*

1 Description of the system

Our approach to EVALITA 2016 PoSTWITA (Bosco et al., 2016) task was implemented in a software prototype operating on tokenized sentences which assigns to each token a score expressing its probability of belonging to a given part-of-speech class. The highest score represents the most probable class.

Differently from the previous EVALITA part of speech tagging tasks (Tamburini (2007), Attardi and Simi (2009)), in EVALITA 2016 PoSTWITA the participants must tackle the problem of analyzing text with low conformance to common writing practices. For example, capitalization rules may be ignored; excessive punctuation, particularly repeated ellipsis and question marks may be used, or spacing may be irregular (Agichtein et al., 2008). Our development system strategy took into account this issue. In particular, we implemented a multiple input bidirectional Long Short Term Memory recurrent neural network (LSTM) model. We developed a two-branched bidirectional LSTM (bi-LSTM) where the first bi-LSTM uses a typical vector representation of the input words commonly used for different classification tasks, while the second one uses a newly introduced word-vector representation specifically designed to handle peculiarities of ill-formed or not standard texts typical of social media texts.

To create the input vectors for the two branches we use a combination of different components extracted from three different word embedding lexicons, from a manually created morpho-syntactic lexicon and from handcrafted features specifically defined to improve the accuracy of the system when tested on social media texts.

In this work we used Keras (Chollet, 2016) deep learning framework to generate the neural network models.

1.1 Lexicons

In order to improve the overall accuracy of our system, we developed three word embedding lexicons[1] and we used a manually created morpho-syntactic lexicon.

1.1.1 Word Embedding lexicons

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

For this purpose, we trained two predict models using the word2vec[2] 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[3]. 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.

In addition to these two lexicons, we built another word embedding lexicon based on fastText (Bojanowski et al., 2016), a library for efficient learning of word representations and sentence classification. FastText allows to overcome the problem of out-of-vocabulary words which affects the relying methodology of word2vec. Generating out-of-vocabulary word embeddings is a typical issue for morphologically rich languages with large vocabularies and many rare words. FastText overcomes this limitation by representing each word as a bag of character n-grams. A vector representation is associated to each character n-gram and the word is represented as the sum of these character n-gram representations. To build the lexicon based on fastText, we adopted as learning corpus the same set of tokenized tweets used to build the word2vec based lexicon.

1.1.2 Morpho-syntactic lexicon

We used a large Italian lexicon of about 1,300,000 forms, developed as part of the SemaWiki project[4]. The full-form lexicon was generated from a base lexicon of 65,500 lemmas, initially inspired by the Zanichelli dictionary[5], and updated along several years and cross-checked with other online dictionaries[6]. For each form the lexicon

[1] The three word embedding lexicons are freely available at the following website: *http://www.italianlp.it/*.

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

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

[4] http://medialab.di.unipi.it/wiki/SemaWiki

[5] Zingarelli: Il nuovo Zingarelli minore, 2008.

[6] Aldo Gabrielli: Il Grande Dizionario di Italiano; Tullio De Mauro: Il Dizionario della lingua italiana.

contains all the possible parts-of-speech and provides information on morpho-syntactic features, but using a different tagset (*ISST-TANL Tagsets*[7]) with respect to the one used for PoSTWITA.

1.2 The POS tagger 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. 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. Similar big improvements have not been obtained in tagging tasks, such as Part-Of-Speech tagging. This is most due to the fact that state-of-the art systems for part of speech tagging exploit strong performing learning algorithms and hard feature engineering. In addition, a little knowledge of the surrounding context is enough to reach very high tagging performance. On the contrary, LSTM networks perform very well with respect to other learning algorithms when word dependencies are long. Although without a big improvement, POS tagging systems which exploit LSTM as learning algorithm have been proven to reach state-of-the-art performances both when analyzing text at character level (Ling et al., 2015) and at word level (Wang et al., 2016). More specifically they used a bidirectional LSTM allows to capture long-range dependencies from both directions of a sentence by constructing bidirectional links in the network (Schuster et al., 1997). In addition, (Plank et al., 2016) have proposed a model which takes into account at the same time both word level and character level information, showing very good results for many languages. As proposed by these systems, we employed a bidirectional LSTM architecture. We implemented a 2-branch bidirectional LSTM but instead of using the character based branch we introduced another specific word level branch in order to reduce the computational cost of the hierarchical LSTM introduced by the character based LSTM. This branch encodes information about the characters in each word of a sentence. The vector representations calculated by the two LSTM are then merged by the sum operation. For what concerns the optimization process, categorical cross-entropy is used as a loss function and the optimization process is performed by the rmsprop optimizer (Tieleman and Hinton, 2012). Each bidirectional LSTM branch is configured to have 24 units. 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 of 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 0.35 for each branch.

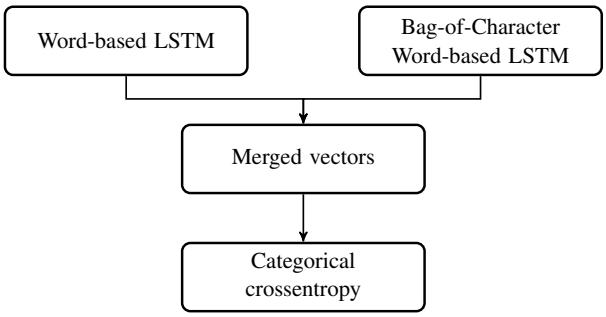

Figure 1: Diagram of the two-branched bi-LSTM architecture.

1.2.1 Word-based bi-LSTM

In this part, we describe the Word-based bidirectional LSTM branch of the proposed neural network architecture and the word level information given in input to this layer. 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. To train this LSTM branch, each input word in the tweet is represented by a 979-dimensional vector which is composed by:

Word2vec word embeddings: the concatenation of the two word embeddings extracted by the two available *word2vec* Word Embedding Lexicons (128 components for each word embedding, thus resulting in a total of 256 components), and for each word embedding an extra component was added in order to handle the "unknown word" (2 components).

FastText word embeddings: the word embeddings extracted by the *fastText* Word Embedding Lexicon (128 components).

[7] http://www.italianlp.it/docs/ISST-TANL-POStagset.pdf

Morpho-syntactic category: the parts-of-speech and the corresponding morpho-syntactic features obtained by exploiting the Morpho-syntactic lexicon, resulting in 293 components.
Spell checker: the parts-of-speech and the corresponding morpho-syntactic features of the word obtained by analyzing the current word using a spell checker (*pyenchant*[8]) and exploiting the Morpho-syntactic lexicon, resulting in 295 components.
Word length: a component representing the length of the analyzed word.
Is URL: a component indicating whether the "http" substring is contained in the analyzed word.
Is uppercase: a component indicating if the analyzed word is uppercase.
Is capitalized: a component indicating if the analyzed word is capitalized.
End of sentence: a component indicating whether or not the sentence was totally read.

1.2.2 Bag-of-Character Word-based bi-LSTM

In this part, we describe the Bag-of-Character Word-based bidirectional LSTM branch of the proposed neural network architecture and the word level information given in input to this layer. Differently from the Word-based LSTM branch, in this branch we did not use pretrained vectors. To train this LSTM branch, each input word in the tweet is represented by a 316-dimensional vector which is composed by:
Characters: a vector representing the set of characters which compose the current word. Since our considered alphabet is composed by 173 different characters, the resulting in a 173-dimensional vector.
Lowercased characters: 134 components representing the set of lowercased characters which compose the current word.
Has numbers: a component indicating whether or not the current word contains a number.
Contains not numbers: a component indicating whether or not the current word contains non numbers.
Contains lowercased: a component indicating whether or not the current word contains lowercase characters.
Contains uppercased: a component indicating whether or not the current word contains uppercase characters.
Contains alphanumeric: a component indicating whether or not the current word contains alphanumeric characters
Contains not alphanumeric: a component indicating whether or not the current word contains non alphanumeric characters
Contains alphabetics: a component indicating whether or not the current word contains alphabetic characters.
Contains not alphabetics: a component indicating whether or not the current word contains non alphabetic characters.
End of sentence: a component indicating whether the sentence was totally read.

[8] http://pythonhosted.org/pyenchant/

2 Results and Discussion

To develop our system, we created an internal development set of 368 tweets randomly selected from the training set distributed by the task organizers. The first row in Table 1 reports the accuracy achieved by our final system on the internal development set and on the official test set (row *Two-branch bi-LSTM*).

Configuration	Devel	Test
Two-branch bi-LSTM	**96.55**	93.19
Word bi-LSTM	96.03	92.35
Bag-of-Char. Word bi-LSTM	84.47	80.77
No Morpho-syntactic lexicon	96.48	**93.54**
No spell checker	96.49	93.31
No word2vec lexicons	93.23	89.87
No fastText lexicon	95.85	92.43
No feature engineering	96.39	93.06

Table 1: Tagging accuracy (in percentage) of the different learning models on our development set and the official test set.

We tested different configurations of our system in order to evaluate the contribution on the tagging accuracy of: *i)* each branch in the proposed architecture, *ii)* the different word embedding and morpho-syntactic lexicons and *iii)* the handcrafted features. We carried out different experiments that reflect the questions we wanted to answer, more specifically the questions are:

- (a) what are the contributions of the *Word-based bi-LSTM* and of the *Bag-of-Character Word-based bi-LSTM*?

- (b) what is the contribution of the *Morpho-syntactic lexicon*?
- (c) what is the contribution of the spell checker?
- (d) what is the contribution of fastText with respect to word2vec *Word Embedding lexicons*?

In order to answer to the question (a), first we run the Word-based LSTM excluding the Bag-of-Character Word-based bi-LSTM branch, then we excluded the Word-based bi-LSTM to verify the Bag-of-Character Word based bi-LSTM contribution. The results of these experiments are reported in *Word bi-LSTM* and *Bag-of-Char. Word bi-LSTM* rows in Table 1. The Word-based bi-LSTM is clearly the best performer with respect to the Bag-of-Character one, but remarkable is that our proposed two-branch architecture shows an improvement of about 0.5 points in the development set with respect to the best single bi-LSTM. The same behaviour is shown in the test set, where the combined system achieves an improvement of 0.84 points with respect to the single Word-based bi-LSTM.

In order to answer to the question (b), we excluded from the input vectors of the Word-based bi-LSTM branch the morpho-syntactic category components extracted from Morpho-syntactic lexicon. Row *No Morpho-syntactic lexicon* reports the results and shows that this information gives a negligible improvement on the development set and unexpectedly a slight drop on the test set.

For what concerns the question (c), we excluded the morpho-syntactic category components of the word obtained using the spell checker. The results are reported in the *No spell checker* row. Similarly to what happened in the (b) experiment, also such information do not contribute in increasing the tagging performances.

In order to compare the contributions of fastText and word2vec lexicons (question (d)), we considered two different system configurations: one removing the two word2vec lexicons (*No word2vec lexicons* row) and one removing fastText and itWac word2vec lexicons (*No fastText lexicon* row). In this second configuration, we removed also the itWac word2vec lexicon to compare fastText and word2vec using the same learning corpus (the twitter corpus described in section 1.1.1). In both configurations we excluded the other Word-based LSTM components, while we left all the components of the Bag-of-Character Word-based LSTM. The results show that word2vec seems to be a better choice with respect to fastText, both in development and in test sets. This is in contrast with what we would have expected considering that fastText learns the word embedding representation using subword information that should be particularly useful for the analysis of non standard text such as social media ones.

2.1 Single bi-LSTM and Handcrafted features

After the submission of the final system results, we devised two further experiments. The first one was devoted to testing the tagging performances of a single word-based bi-LSTM architecture with respect to the presented Two-branch bi-LSTM. The second experiment was aimed to study the effect of handcrafted features combined with the learning ones. To this aim, we developed a Part-of-Speech tagger based on a single word-based bi-LSTM, where each input word vector is the concatenation of the two input word representations of the bi-LSTMs presented in Section 1.2.1 and Section 1.2.2.

Table 2 reports the results of these experiments. As shown in the *Single bi-LSTM* row, the use of the single architecture instead of the two-branch one does not affect tagging results, actually the single bi-LSTM slightly outperforms the two-branch architecture when tested on the test set (+0.48%).

In order to evaluate the effect of handcrafted features, we conducted a last experiment where we removed all the components from the input vectors of the single Word-based bi-LSTM with the exceptions of word2vec and fastText word embeddings. *No handcrafted features* row shows the relevance of the handcrafted features that yield an improvement of 1.34% and 1.68% on the development and the test sets respectively. These results show the important role of *feature engineering* even when neural networks learning algorithms are used.

3 Conclusion

In this paper we reported the results of our participation to the EVALITA 2016 POS tagging for Italian Social Media Texts (PoSTWITA). By resorting to a two-branch bidirectional LSTM, word em-

Configuration	Devel	Test
Single bi-LSTM	**96.39**	**93.67**
No handcrafted features	95.22	91.99

Table 2: Tagging accuracy of the single word-based bi-LSTM on our development set and the official test set.

beddings and morpho-syntactic lexicons and hand crafted features we achieved the best score. In particular, we showed the relevance of handcrafted features that allowed an improvement of more than one percentage point in terms of tagging accuracy both in development and test sets when combined with learned features such as word embedding lexicons. As future research direction we will test the contribution of a pure character based LSTM with respect to character handcrafted features.

References

Eugene Agichtein and Carlos Castillo and Debora Donato and Aristides Gionis and Gilad Mishne. 2008. Finding High-quality Content in Social Media. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*. New York, USA.

Giuseppe Attardi and Maria Simi. 2009. Overview of the EVALITA 2009 Part-of-Speech Tagging Task. In *Proceedings of Evalita '09, Evaluation of NLP and Speech Tools for Italian*. December, Reggio Emilia, Italy.

Piotr Bojanowski, Edouard Grave, Armand Joulin and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:607.04606*.

Cristina Bosco and Fabio Tamburini and Andrea Bolioli and Alessandro Mazzei. 2016. Overview of the EVALITA 2016 Part Of Speech on TWitter for ITAlian Task. In *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)*. Associazione Italiana di Linguistica Computazionale (AILC).

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

Cìcero Nogueira dos Santos and Bianca Zadrozny. 2013. Learning Character-level Representations for Part-of-Speech Tagging. In *Proceedings of the 31st International 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*.

Wang Ling, Chris Dyer, Alan W. Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo and Luis Tiago 2016. Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 1520–1530 , Lisbon, Portugal. ACL.

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

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)*.

Barbara Plank, Anders Søgaard and Yoav Goldberg. 2016. Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models an Auxiliary Loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. August, Berlin, Germany.

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

Fabio Tamburini. 2007. Evalita 2007: The Part-of-Speech Tagging Task. In *Proceedings of Evalita '07, Evaluation of NLP and Speech Tools for Italian*. September, Rome, Italy.

Duyu Tang, Bing Qin and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* 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*.

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)*.

Peilu Wang, Yao Qian, Frank K. Soong, Lei He, and Hai Zhao . 2016. Learning Distributed Word Representations For Bidirectional LSTM Recurrent Neural Network. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* 527–533, San Diego, CA, USA. ACL.