bot.zen @ EVALITA 2016 - A minimally-deep learning PoS-tagger (trained for Italian Tweets)

Egon W. Stemle
Institute for Specialised Communication and Multilingualism
EURAC Research
Bolzano/Bozen, Italy
egon.stemle@eurac.edu

Abstract

English. This article describes the system that participated in the *POS tagging for Italian Social Media Texts* (PoSTWITA) task of the 5th periodic evaluation campaign of Natural Language Processing (NLP) and speech tools for the Italian language EVALITA 2016.

The work is a continuation of Stemle (2016) with minor modifications to the system and different data sets. It combines a small assertion of trending techniques, which implement matured methods, from NLP and ML to achieve competitive results on PoS tagging of Italian Twitter texts; in particular, the system uses word embeddings and character-level representations of word beginnings and endings in a LSTM RNN architecture. Labelled data (Italian UD corpus, DiDi and PoSTWITA) and unlabbelled data (Italian C4Corpus and PAISÀ) were used for training.

The system is available under the APLv2 open-source license.

Italiano. *Questo articolo descrive il sistema che ha partecipato al task* POS tagging for Italian Social Media Texts (PoSTWita) *nell'ambito di EVALITA 2016, la 5° campagna di valutazione periodica del Natural Language Processing (NLP) e delle tecnologie del linguaggio.*

Il lavoro è un proseguimento di quanto descritto in Stemle (2016), con modifiche minime al sistema e insiemi di dati differenti. Il lavoro combina alcune tecniche correnti che implementano metodi comprovati dell'NLP e del Machine Learning, per raggiungere risultati competitivi nel PoS tagging dei testi italiani di Twitter. In particolare il sistema utilizza strategie di word embedding e di rappresentazione character-level di inizio e fine parola, in un'architettura LSTM RNN. Dati etichettati (Italian UD corpus, DiDi e PoSTWITA) e dati non etichettati (Italian C4Corpus e PAISÀ) sono stati utilizzati in fase di training.

Il sistema è disponibile sotto licenza open source APLv2.

1 Introduction

Part-of-speech (PoS) tagging is an essential processing stage for virtually all NLP applications. Subsequent tasks, like parsing, named-entity recognition, event detection, and machine translation, often utilise PoS tags, and benefit (directly or indirectly) from accurate tag sequences.

Actual work on PoS tagging, meanwhile, mainly concentrated on standardized texts for many years, and frequent phenomena in computer-mediated communication (CMC) and Web corpora such as emoticons, acronyms, interaction words, iteration of letters, graphostylistics, shortenings, addressing terms, spelling variations, and boilerplate (Androutsopoulos, 2007; Bernardini et al., 2008; Beißwenger, 2013) still deteriorate the performance of PoS-taggers (Giesbrecht and Evert, 2009; Baldwin et al., 2013).

On the other hand, the interest in automatic evaluation of social media texts, in particular for microblogging texts such as tweets, has been growing considerably, and specialised tools for

Twitter data have become available for different languages. But Italian completely lacks such resources, both regarding annotated corpora and specific PoS-tagging tools.[1] To this end, the *POS tagging for Italian Social Media Texts* (PoSTWITA) task was proposed for EVALITA 2016 concerning the domain adaptation of PoS-taggers to Twitter texts.

Our system combined `word2vec` (`w2v`) word embeddings (WEs) with a single-layer Long Short Term Memory (LSTM) recurrent neural network (RNN) architecture. The sequence of unlabelled `w2v` representations of words is accompanied by the sequence of n-grams of the word beginnings and endings, and is fed into the RNN which in turn predicts PoS labels.

The paper is organised as follows: We present our system design in Section 2, the implementation in Section 3, and its evaluation in Section 4. Section 5 concludes with an outlook on possible implementation improvements.

2 Design

Overall, our design takes inspiration from as far back as Benello et al. (1989) who used four preceding words and one following word in a feed-forward neural network with backpropagation for PoS tagging, builds upon the strong foundation laid down by Collobert et al. (2011) for a neural network (NN) architecture and learning algorithm that can be applied to various natural language processing tasks, and ultimately is a variation of Nogueira dos Santos and Zadrozny (2014) who trained a NN for PoS tagging, with character-level and WE representations of words.

Also note that an earlier version of the system was used in Stemle (2016) to participate in the *EmpiriST 2015 shared task on automatic linguistic annotation of computer-mediated communication / social media* (Beißwenger et al., 2016).

2.1 Word Embeddings

Recently, state-of-the-art results on various linguistic tasks were accomplished by architectures using neural-network based WEs. Baroni et al. (2014) conducted a set of experiments comparing the popular `w2v` (Mikolov et al., 2013a; Mikolov et al., 2013b) implementation for creating WEs to other distributional methods with state-of-the-art results across various (semantic) tasks. These results suggest that the word embeddings substantially outperform the other architectures on semantic similarity and analogy detection tasks. Subsequently, Levy et al. (2015) conducted a comprehensive set of experiments and comparisons that suggest that much of the improved results are due to the system design and parameter optimizations, rather than the selected method. They conclude that "there does not seem to be a consistent significant advantage to one approach over the other".

Word embeddings provide high-quality low dimensional vector representations of words from large corpora of unlabelled data, and the representations, typically computed using NNs, encode many linguistic regularities and patterns (Mikolov et al., 2013b).

2.2 Character-Level Sub-Word Information

The morphology of a word is opaque to WEs, and the relatedness of the meaning of a lemma's different word forms, i.e. its different string representations, is *not* systematically encoded. This means that in morphologically rich languages with long-tailed frequency distributions, even some WE representations for word forms of common lemmata may become very poor (Kim et al., 2015).

We agree with Nogueira dos Santos and Zadrozny (2014) and Kim et al. (2015) that sub-word information is very important for PoS tagging, and therefore we augment the WE representations with character-level representations of the word beginnings and endings; thereby, we also stay language agnostic—at least, as much as possible—by avoiding the need for, often language specific, morphological pre-processing.

2.3 Recurrent Neural Network Layer

Language Models are a central part of NLP. They are used to place distributions over word sequences that encode systematic structural properties of the sample of linguistic content they are built from, and can then be used on novel content, e.g. to rank it or predict some feature on it. For a detailed overview on language modelling research see Mikolov (2012).

A straight-forward approach to incorporate WEs into feature-based language models is to use the embeddings' vector representations as features.[2] Having said that, WEs are also used in NN

[1] http://www.evalita.it/2016/tasks/postwita

[2] For an overview see, e.g. Turian et al. (2010).

architectures, where they constitute (part of) the input to the network.

Neural networks consist of a large number of simple, highly interconnected processing nodes in an architecture loosely inspired by the structure of the cerebral cortex of the brain (O'Reilly and Munakata, 2000). The nodes receive weighted inputs through these connections and *fire* according to their individual thresholds of their shared activation function. A firing node passes on an activation to all successive connected nodes. During learning the input is propagated through the network and the output is compared to the desired output. Then, the weights of the connections (and the thresholds) are adjusted step-wise so as to more closely resemble a configuration that would produce the desired output. After all input cases have been presented, the process typically starts over again, and the output values will usually be closer to the correct values.

RNNs are NNs where the connections between the elements are directed cycles, i.e. the networks have loops, and this enables them to model sequential dependencies of the input. However, regular RNNs have fundamental difficulties learning long-term dependencies, and special kinds of RNNs need to be used (Hochreiter, 1991); a very popular kind is the so called long short-term memory (LSTM) network proposed by Hochreiter and Schmidhuber (1997).

Overall, with this design we not only benefit from available labelled data but also from available general or domain-specific unlabelled data.

3 Implementation

We maintain the implementation in a source code repository at `https://github.com/bot-zen/`. The version tagged as `1.1` comprises the version that was used to generate the results submitted to the shared task (ST).

Our system feeds WEs and character-level subword information into a single-layer RNN with a LSTM architecture.

3.1 Word Embeddings

When computing WEs we take into consideration Levy et al. (2015): they observed that one specific configuration of `w2v`, namely the skip-gram model with negative sampling (SGNS) "is a robust baseline. While it might not be the best method for every task, it does not significantly underperform in any scenario. Moreover, SGNS is the fastest method to train, and cheapest (by far) in terms of disk space and memory consumption". Coincidentally, Mikolov et al. (2013b) also suggest to use SGNS. We incorporate `w2v`'s original C implementation for learning WEs[3] in an independent pre-processing step, i.e. we pre-compute the WEs. Then, we use gensim[4], a Python tool for unsupervised semantic modelling from plain text, to load the pre-computed data, and to compute the vector representations of input words for our NN.

3.2 Character-Level Sub-Word Information

Our implementation uses a *one-hot encoding* with a few additional features for representing subword information. The one-hot encoding transforms a categorical feature into a vector where the categories are represented by equally many dimensions with binary values. We convert a letter to lower-case and use the sets of ASCII characters, digits, and punctuation marks as categories for the encoding. Then, we add dimensions to represent more binary features like *'uppercase'* (was uppercase prior to conversion), *'digit'* (is digit), *'punctuation'* (is punctuation mark), *whitespace* (is white space, except the new line character; note that this category is usually empty, because we expect our tokens to *not* include white space characters), and *unknown* (other characters, e.g. diacritics). This results in vectors with more than a single *one-hot* dimension.

3.3 Recurrent Neural Network Layer

Our implementation uses Keras, a high-level NNs library, written in Python and capable of running on top of either TensorFlow or Theano (Chollet, 2015). In our case it runs on top of Theano, a Python library that allows to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently (The Theano Development Team et al., 2016).

The input to our network are sequences of the same length as the sentences we process. During training, we group sentences of the same length into batches and process the batches according to sentence length in increasing order. Each single word in the sequence is represented by its subword information and two WEs that come from two sources (see Section 4). For unknown words,

[3] `https://code.google.com/archive/p/word2vec/`

[4] `https://radimrehurek.com/gensim/`

i.e. words without a pre-computed WE, we first try to find the most similar WE considering 10 surrounding words. If this fails, the unknown word is mapped to a randomly generated vector representation. In Total, each word is represented by $2,280$ features: two times 500 (WEs), and sixteen times 80 for two 8-grams (word beginning and ending). If words are shorter than 8 characters their 8-grams are zero-padded.

This sequential input is fed into a LSTM layer that, in turn, projects to a fully connected output layer with softmax activation function. During training we use dropout for the projection into the output layer, i.e. we set a fraction (0.5) of the input units to 0 at each update, which helps prevent overfitting (Srivastava et al., 2014). We use categorical cross-entropy as loss function and backpropagation in conjunction with the RMSprop optimization for learning. At the time of writing, this was the Keras default—or the explicitly documented option to be used—for our type of architecture.

4 Results

We used our slightly modified implementation to participate in the *POS tagging for Italian Social Media Texts* (PoSTWITA) shared task (ST) of the 5th periodic evaluation campaign of Natural Language Processing (NLP) and speech tools for the Italian language EVALITA 2016. First, we describe the corpora used for training, and then the specific system configuration(s) for the ST.

4.1 Training Data for `w2v` and PoS Tagging

4.1.1 DiDi-IT (PoS, `w2v`)

didi-it (Frey et al., 2016) (version September 2016) is the Italian sub-part of the DiDi corpus, a corpus of South Tyrolean German and Italian from Facebook (FB) users' wall posts, comments on wall posts and private messages.

The Italian part consists of around 100,000 tokens collected from 20 profiles of Facebook users residing in South Tyrol. This version has about 20,000 PoS tags semi-automatically corrected by a single annotator.

The anonymised corpus is freely available for research purposes.

4.1.2 Italian UD (PoS, `w2v`)

Universal Dependencies (UD) is a project that is developing cross-linguistically consistent treebank annotation for many languages.[5]

italian-UD[6] (version from January 2015) corpus was originally obtained by conversion from ISDT (Italian Stanford Dependency Treebank) and released for the dependency parsing ST of EVALITA 2014 (Bosco et al., 2014). The corpus has semi-automatically converted PoS tags from the original two Italian treebanks, differing both in corpus composition and adopted annotation schemes.

The corpus contains around 317,000 tokens in around 13,000 sentences from different sources and genres. It is available under the CC BY-NC-SA 3.0[7] license.

4.1.3 PoSTWITA (PoS and `w2v`)

postwita is the Twitter data made available by the organizers of the ST. It contains Twitter tweets from the EVALITA2014 SENTIPLOC corpus: the development and test set and additional tweets from the same period of time were manually annotated for a global amount of 6438 tweets (114,967 tokens) and were distributed as the development set. The data is PoS tagged according to UD but with the additional insertion of seven Twitter-specific tags. All the annotations were carried out by three different annotators. The data was only distributed to the task participants.

4.1.4 C4Corpus (`w2v`)

c4corpus[8] is a full documents Italian Web corpus that has been extracted from CommonCrawl, the largest publicly available general Web crawl to date. See Habernal (2016) for details about the corpus construction pipeline, and other information about the corpus.

The corpus contains about $670m$ tokens in $22m$ sentences. The data is available under the CreativeCommons license family.

4.1.5 PAISÀ (`w2v`)

paisa (Lyding et al., 2014) is a corpus of authentic contemporary Italian texts from the web (harvested in September/October 2010). It was created

[5] `http://universaldependencies.org/`

[6] `http://universaldependencies.org/it/overview/introduction.html`

[7] Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported, i.e. the data can be copied and redistributed, and adapted for purposes other than commercial ones. See `https://creativecommons.org/licenses/by-nc-sa/3.0/` for more details.

[8] `https://github.com/dkpro/dkpro-c4corpus`

in the context of the project PAISÀ (Píattaforma per l'Apprendimento dell'Italiano Su corpora Annotati) with the aim to provide a large resource of freely available Italian texts for language learning by studying authentic text materials.

The corpus contains about $270m$ tokens in about $8m$ sentences. The data is available under the CC BY-NC-SA 3.0[9] license.

4.2 PoSTWITA shared task

For the ST we used one overall configuration for the system but three different corpus configurations for training. However, only one corpus configuration was entered into the ST: we used PoS tags from *didi-it + postwita* (run 1), from *italian-UD* (run 2), and from both (run 3). For `w2v` we trained a 500-dimensional skip-gram model on *didi-it + italian-UD + postwita* that ignored all words with less than 2 occurrences within a window size of 10; it was trained with negative sampling (value 15). We also trained a 500-dimensional skip-gram model on *c4corpus + paisa* that ignored all words with less than 33 occurrences within a window size of 10; it was trained with negative sampling (value 15).

The other `w2v` parameters were left at their default settings[10].

The evaluation of the systems was done by the organisers on unlabelled but pre-tokenised data (4759 tokens in 301 tweets), and was based on a token-by-token comparison. The considered metric was accuracy, i.e. the number of correctly assigned PoS tags divided by the total number of tokens.

(1) *didi-it + postwita*	**76.00**
(2) *italian-UD*	80.54
(3) *didi-it + postwita + italian-UD*	81.61
Winning Team	93.19

Table 1: Official result(s) of our PoS tagger for the three runs on the PoSTWITA ST data.

We believe, the unexpectedly little performance gain from utilizing the much larger *italian-UD* data over the rather small *didi-it + postwita* data may be rooted in the insertion of Twitter-specific tags into the data (see 4.1.3), something we did not account for, i.e. $18,213$ of $289,416$ and more importantly $7,778$ of $12,677$ sentences had imperfect information during training.

5 Conclusion & Outlook

We presented our submission to the PoSTWITA task of EVALITA 2016, where we participated with moderate results. In the future, we will try to rerun the experiment with training data that takes into consideration the Twitter-specific tags of the task.

Acknowledgments

The computational results presented have been achieved in part using the Vienna Scientific Cluster (VSC).

References

Jannis K. Androutsopoulos. 2007. Neue Medien – neue Schriftlichkeit? *Mitteilungen des Deutschen Germanistenverbandes*, 1:72–97.

Timothy Baldwin, Paul Cook, Marco Lui, Andrew MacKinlay, and Li Wang. 2013. How noisy social media text, how diffrnt social media sources? In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 356–364, Nagoya, Japan, October. Asian Federation of Natural Language Processing. `http://aclweb.org/anthology/I13-1041`.

Marco Baroni, Georgiana Dinu, and German Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247. Association for Computational Linguistics. `http://www.aclweb.org/anthology/P14-1023`.

Michael Beißwenger, Sabine Bartsch, Stefan Evert, and Kay-Michael Würzner. 2016. EmpiriST 2015: A Shared Task on the Automatic Linguistic Annotation of Computer-Mediated Communication, Social Media and Web Corpora. In *Proceedings of the 10th Web as Corpus Workshop (WAC-X) and the EmpiriST Shared Task*, pages 78–90, Berlin, Germany. Association for Computational Linguistics.

Michael Beißwenger. 2013. Das Dortmunder Chat-Korpus: ein annotiertes Korpus zur Sprachverwendung und sprachlichen Variation in der deutschsprachigen Chat-Kommunikation. *LINSE - Linguistik Server Essen*, pages 1–13.

Julian Benello, Andrew W. Mackie, and James A. Anderson. 1989. Syntactic category disambiguation with neural networks. *Computer Speech & Language*, 3(3):203–217, July. `http:`

[9] `https://creativecommons.org/licenses/by-nc-sa/3.0/`

[10] `-sample 1e-3 -iter 5 -alpha 0.025`

//www.sciencedirect.com/science/article/pii/0885230889900181.

Silvia Bernardini, Marco Baroni, and Stefan Evert. 2008. A WaCky Introduction. In *Wacky! Working papers on the Web as Corpus*, pages 9–40. GEDIT, Bologna, Italy. http://wackybook.sslmit.unibo.it/pdfs/bernardini.pdf.

Cristina Bosco, Felice Dell'Orletta, Simonetta Montemagni, Manuela Sanguinetti, and Maria Simi. 2014. The EVALITA 2014 Dependency Parsing Task. In *Proceedings of CLiC-it 2014 and EVALITA 2014*, pages 1–8. Pisa University Press.

Franois Chollet. 2015. Keras: Deep Learning library for Theano and TensorFlow. https://github.com/fchollet/keras.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537. https://arxiv.org/abs/1103.0398.

Jennifer-Carmen Frey, Aivars Glaznieks, and Egon W. Stemle. 2016. The DiDi Corpus of South Tyrolean CMC Data: A multilingual corpus of Facebook texts. Upcoming.

Eugenie Giesbrecht and Stefan Evert. 2009. Is Part-of-Speech Tagging a Solved Task? An Evaluation of POS Taggers for the German Web as Corpus. *Web as Corpus Workshop (WAC5)*. http://sigwac.org.uk/raw-attachment/wiki/WAC5/WAC5_proceedings.pdf#page=27.

Ivan Habernal, Omnia Zayed, and Iryna Gurevych. 2016. C4Corpus: Multilingual Web-size corpus with free license. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, page (to appear), Portorož, Slovenia, May. European Language Resources Association (ELRA).

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November. http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Sepp Hochreiter. 1991. *Untersuchungen zu dynamischen neuronalen Netzen*. diploma thesis, TU München.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2015. Character-Aware Neural Language Models. *CoRR*, abs/1508.0. http://arxiv.org/abs/1508.06615.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225. https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/570.

Verena Lyding, Egon Stemle, Claudia Borghetti, Marco Brunello, Sara Castagnoli, Felice Dell Orletta, Henrik Dittmann, Alessandro Lenci, and Vito Pirrelli. 2014. The PAISÀ Corpus of Italian Web Texts. In *Proceedings of the 9th Web as Corpus Workshop (WaC-9)*, pages 36–43, Gothenburg, Sweden. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. *CoRR*, abs/1301.3781. http://arxiv.org/abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. *CoRR*, abs/1310.4546, October. http://arxiv.org/abs/1310.4546.

Tomáš Mikolov. 2012. *Statistical Language Models Based on Neural Networks*. Ph.D. thesis, Brno University of Technology. http://www.fit.vutbr.cz/~imikolov/rnnlm/thesis.pdf.

Cícero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning Character-level Representations for Part-of-Speech Tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1818–1826. http://jmlr.org/proceedings/papers/v32/santos14.pdf.

Randall C. O'Reilly and Yuko Munakata. 2000. *Computational Explorations in Cognitive Neuroscience Understanding the Mind by Simulating the Brain*. MIT Press. http://books.google.com/books?id=BLf34BFTaIUC{\&}pgis=1.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout : A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research (JMLR)*, 15:1929–1958.

Egon W. Stemle. 2016. bot.zen @ EmpiriST 2015 - A minimally-deep learning PoS-tagger (trained for German CMC and Web data). In *Proceedings of the 10th Web as Corpus Workshop (WAC-X) and the EmpiriST Shared Task*, pages 115–119. Association for Computational Linguistics.

The Theano Development Team, Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, and et al. 2016. Theano: A Python framework for fast computation of mathematical expressions. *CoRR*, abs/1605.02688. http://arxiv.org/abs/1605.02688.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394, Stroudsburg, PA, USA. Association for Computational Linguistics. http://dl.acm.org/citation.cfm?id=1858681.1858721.