Recurrent Context Window Networks for Italian Named Entity Recognizer

Daniele Bonadiman*
DISI, University of Trento

Aliaksei Severyn**
Google Inc. Zurich

Alessandro Moschitti†
Qatar Computing Research Institute,
HBKU

In this paper, we introduce a Deep Neural Network (DNN) for engineering Named Entity Recognizers (NERs) in Italian. Our network uses a sliding window of word contexts to predict tags. It relies on a simple word-level log-likelihood as a cost function and uses a new recurrent feedback mechanism to ensure that the dependencies between the output tags are properly modeled. These choices make our network simple and computationally efficient. Unlike previous best NERs for Italian, our model does not require manual-designed features, external parsers or additional resources. The evaluation on the Evalita 2009 benchmark shows that our DNN performs on par with the best NERs, outperforming the state of the art when gazetteer features are used.

1. Introduction

Named Entity (NE) recognition is the task of detecting *semantically important* noun phrases in text, e.g., proper names, which directly refer to real world entities along with their type, e.g., people, organizations, locations, etc. see, e.g., (Nadeau and Sekine 2007). International challenges on the NER task have been firstly proposed by the Message Understanding Conference (MUC 6, MUC 7), the Conference on Natural Language Learning (CoNLL 2002, 2003) whereas the Automatic Content Extraction program (ACE 2002, 2005) has supported the creation of several datasets based on named entities. More recently, the NER challenges have been extended to other languages, e.g., Italian (Evalita 2007, 2009).

Most NE recognizers (NERs) rely on machine learning models, which require to define a large set of manually engineered features. For example, the state-of-the-art system for English (Ratinov and Roth 2009) uses a simple averaged perceptron and a large set of local and non-local features. Similarly, the best performing system for Italian (Nguyen and Moschitti 2012) combines two learning systems that heavily rely on both local and global manually engineered features. Some of the latter are generated using basic hand-crafted rules (i.e., suffix, prefix) but most of them require huge dictionaries (gazetteers) and external parsers (POS taggers and chunkers). While designing good features for

* DISI, University of Trento, Italy. E-mail: bonadiman.daniele@gmail.com
** Google Inc. Zurich. E-mail: aseveryn@gmail.com - This work was carried out during his PhD in the University of Trento.
† Qatar Computing Research Institute, HBKU, Qatar. E-mail: amoschitti@gmail.com - Professor at DISI, University of Trento.

NERs requires a great deal of expertise and can be labour intensive, it also makes the taggers harder to adapt to new domains and languages since resources and syntactic parsers used to generate the features may not be readily available.

Recently, DNNs have been shown to be very effective for automatic feature engineering, demonstrating state-of-the-art results in many sequence labelling tasks, e.g., (Collobert et al. 2011; dos Santos et al. 2015; Chiu and Nichols 2015), also for Italian language (Attardi 2015).

In this paper, we target NERs for Italian and propose a novel deep learning model that can match the accuracy of the previous best NERs without using manual feature engineering and only requiring a minimal effort for language adaptation. In particular, our model is inspired by the successful neural network architecture presented by (Collobert et al. 2011) to which we propose several innovative and valuable enhancements: (i) a simple recurrent feedback mechanism to model the dependencies between the output tags and (ii) a pre-training process based on two-steps: (a) training the network on a weakly labeled dataset (e.g., automatically annotated) and then (b) refining the weights on the supervised training set. Our final model obtains 82.81 in F1 on the Evalita 2009 Italian dataset (Speranza 2009), which is an improvement of $+0.81$ over the (Zanoli and Pianta 2009) system that won the competition. Our model only uses the words in the sentence, four morphological features and a gazetteer. Interestingly, if the gazetteer is removed from our network, it achieves an F1 of 81.42, which is still on par with the previous best systems yet it is simple and easy to adapt to new domains and languages.

2. Our DNN model for NER

In this section, we first briefly describe the architecture of the Context Window Network (CWN) from (Collobert et al. 2011), pointing out its limitation. We then introduce our Recurrent Context Window Network (RCWN), which extends CWN and aims at solving its drawbacks.

2.1 Context Window Network

We adopt a CWN model that has been successfully applied by (Collobert et al. 2011) for a wide range of sequence labelling NLP tasks. Its architecture is depicted in Fig. 1. It works as follows: given an input sentence $s = [w_1, \ldots, w_n]$, e.g., *Barack Obama è il presidente degli Stati Uniti D'America*[1], for each word w_i, the sequences of word contexts $[w_{i-k/2+1}, .., w_i, .., w_{i+k/2}]$ of size k around the target word w_i ($i = 1, .., n$) are used as input to the network.[2] For example, the Fig. 1 shows a network with $k = 5$ and the input sequence for the target word *è* at position $i = 3$.

The input words w_i from the vocabulary V are mapped to d-dimensional word embedding vectors $\mathbf{w}_i \in \mathbb{R}^d$. Embeddings $\mathbf{w}_i$ for all words in V form an embedding matrix $\mathbf{W} \in \mathbb{R}^{|V| \times d}$, which is learned by the network. An embedding vector $\mathbf{w}_i$ for a word w_i is retrieved by a simple lookup operation in $\mathbf{W}$ (see lookup frame in Fig. 1). After the lookup, the k embedding vectors of the context window are concatenated into a single vector $\mathbf{r_1} \in \mathbb{R}^{kd}$, which is passed to the next hidden layer hl. It applies the following linear transformation: $hl(\mathbf{r_1}) = \mathbf{M_1} \cdot \mathbf{r_1} + b_1$, where the matrix of weights $\mathbf{M_1}$

1 *Barack Obama is the president of the United States of America.*

2 In case the target word i is at the beginning/end of a sentence, up to $(k-1)/2$ placeholders are used in place of the empty input words.

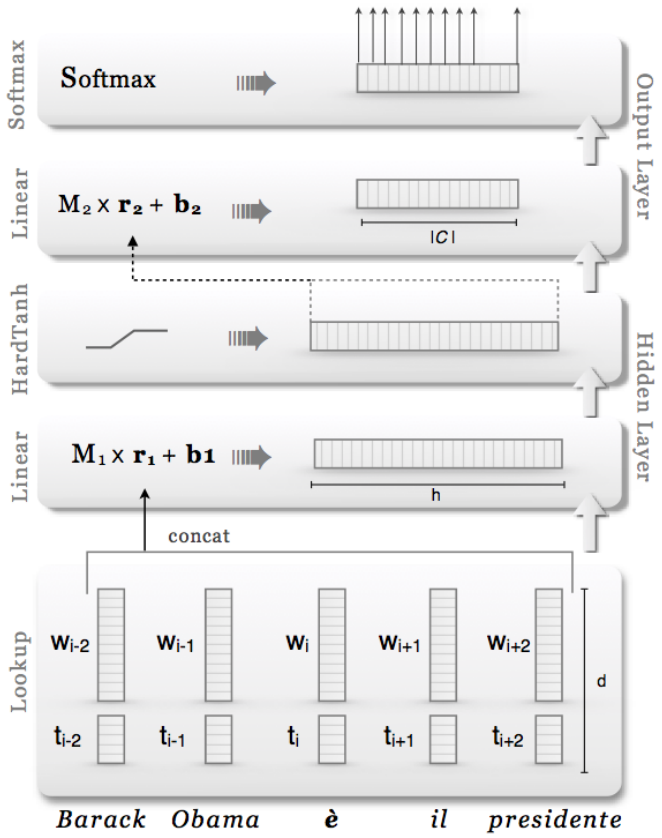

Figure 1
The architecture of Context Window Network (CWN).

and the bias b_1 parametrize the linear transformation and are learned by the network. The goal of the hidden layer is to learn feature combinations from the word embeddings of the context window.

To enable the learning of non-linear discriminative functions, the output of hl is passed through a non-linear transformation also called activation function, i.e., a *HardTanh*() non-linearity, Eq. 1), thus obtaining, $\mathbf{r_2}$.

$$HardTanh(x) = \begin{cases} 1 & x > 1 \\ x & -1 < x < 1 \\ -1 & x < -1 \end{cases} \tag{1}$$

Finally, the output classification layer encoded by the matrix $\mathbf{M_2} \in \mathbb{R}^{|C| \times h}$ and the bias b_2 are used to evaluate the vector $\mathbf{p} = softmax(\mathbf{M_2} \times r_2 + b_2)$ of class conditional probabilities, i.e., $\mathbf{p}_c = p(c|\mathbf{x}), c \in C$, where C is the set of NE tags, h is the dimension of the hl and $\mathbf{x}$ is the input context window.

2.2 Our model

The CWN model described above has several drawbacks: (i) each tag prediction is made by considering only local information, i.e., no dependencies between the output tags are taken into account; (ii) publicly available annotated datasets for NER are usually too small to train neural networks thus often leading to overfitting. We address both problems by proposing: (i) a novel recurrent context window network (RCWN) architecture; (ii) a network pre-training technique using weakly labeled data; and (iii) we also experiment with a set of recent techniques to improve the generalization of our DNN to avoid overfitting, i.e., we use *early stopping* (Prechelt 1998), *weight decay* (Krogh and Hertz 1992), and *Dropout* (Hinton 2014). Dropout prevents feature co-adaptation by setting the output of the hidden units to 0 in the forward pass of the training phase. Furthermore, Dropout acts as an approximate model averaging technique (Hinton 2014).

2.2.1 Recurrent Context Window Network

Generally, the sliding window model cannot capture the dependencies between the output tags since each window of words of the input sequence is processed independently from the others. This results in a performance lower than the one obtained by more traditional methods, which instead models dependencies by considering predicted tags near the target word or also the whole sentence as in the case of Conditional Random Fields (Pereira et al. 2001) or the Sentence Level Log Likelihood (Collobert et al. 2011).

We propose RCWN for modeling dependencies between labels. It extends CWN by using m previously predicted tags as an additional input, i.e., the previously predicted tags at steps $i-m,\ldots,i-1$ are used to predict the tag of the word at position i, where $m < k/2$. Since we proceed from left to right, words in the context window w_j with $j > i-1$, i.e., at the right of the target word, do not have their predicted tags, thus we simply use the special unknown tag, `UNK`, for them.

Since NNs provide us with the possibility to define and train arbitrary embeddings, we associate each predicted tag type with an embedding vector, which can be trained in the same way as word embeddings (see vectors for tags t_i in Fig. 1). More specifically, given k words $w_i \in \mathbb{R}^{d_w}$ in the context window and previously predicted tags $t_i \in \mathbb{R}^{d_t}$ at corresponding positions, we concatenate them together along the embedding dimension obtaining new vectors of dimensionality $d_w + d_t$ Thus, the output of the first input layer becomes a sequence of $k(d_w + d_t)$ vectors.

RCWN is simple to implement and is computationally more efficient than, for example, NNs computing *sentence log-likelihood*, which require Viterbi decoding. RCWN may suffer from an error propagation issue as the network can misclassify the word at position $t-i$, propagating an erroneous feature (the wrong label) to the rest of the sequence. However, the learned tag embeddings seem to be robust to noise[3]. Indeed, the proposed network obtains a significant improvement over the baseline model (see Section 3.2).

3. Experiments

In these experiments, we compare three different enhancements of our DNNs on the data from the Evalita challenge, namely: (i) our RCWN method, (ii) pre-training on weakly supervised data, and (iii) the use of gazetteers.

3 We can use the same intuitive explanation of error correcting output codes.

Dataset	Articles	Sentences	Tokens
Train	525	11,227	212,478
Test	180	4,136	86,419

Table 1
Splits of the Evalita 2009 dataset

3.1 Experimental setup

Dataset. We evaluate our models on the Evalita 2009 Italian dataset for NERs (Speranza 2009) summarized in Tab. 1. There are four types of NEs: person (PER), location (LOC), organization (ORG) and geo-political entity (GPE), (see Tab. 2). Data is annotated using the IOB tagging schema, i.e., for inside, outside and beginning of a entity, respectively. Note that NEs do not overlap and are not nested. For instance, in the case of the entity, *University of Trento*, the entity *Trento* is not labeled as *LOC* since it is nested into an *ORG* entity.

Training and testing the network. We use (i) the Negative Log Likelihood cost function, i.e., $-log(\mathbf{p}_c)$, where c is the correct label for the target word, (ii) stochastic gradient descent (SGD) to learn the parameters of the network and (iii) the backpropogation algorithm to compute the updates. During training, the true labels for all the words but the one to be predicted are used. Conversely, at test time, the predicted tag c, associated with the highest class conditional probability $\mathbf{p}_c$, is selected, i.e., $c = \text{argmax}_{c \in C}\, \mathbf{p}_c$ is used as input for the next iteration.

Features. In addition to words, all our models also use 4 basic morphological features: *all lowercase*, *all uppercase*, *capitalized* and *it contains uppercase character*. These can reduce the size of the word embedding dictionary as showed by (Collobert et al. 2011). In our implementation, these 4 binary features are encoded as one discrete feature associated with an embedding vector of size 5, i.e., similarly to the preceding tags in RCWN. Additionally, we use a similar vector to also encode gazetteer features. Gazetteers are collections of names, locations and organizations extracted from different sources such as the Italian phone book, Wikipedia and stock marked websites. Since we use four different dictionaries one for each NE class, we add four feature vectors to the network.

Word Embeddings. We use a fixed dictionary of size $100K$ and set the size of the word embeddings to 50, hence, the number parameters to be trained is $5M$. Training a model with such a large capacity requires a large amount of labelled data. Unfortunately, the sizes of the supervised datasets available for training NER models are much smaller, thus we mitigate such problem by pre-training the word embeddings on huge unsupervised training datasets. For example, (Collobert et al. 2011) proposed a method to train word embeddings using a sliding window NNs by solving an artificial task on *Wikipedia*. However, this method is rather time consuming, taking up to several months for properly training the embeddings. We use word2vec (Mikolov et al. 2013) skip-gram model (i.e., predicting the context given a word) with negative sampling and a context window of size 5 to pre-train our embeddings on Italian dump of Wikipedia: this only took a few hours. Interestingly, (Qu et al. 2015) report that the Skip-Gram embeddings outperform the ones from (Collobert et al. 2011) in NER systems.

Dataset	PER	ORG	LOC	GPE
Train	4,577	3,658	362	2,813
Test	2,378	1,289	156	1,143

Table 2
Entities distribution in Evalita 2009

Model	F1	Prec.	Rec.
Baseline	78.32	79.45	77.23
RCWN	81.39	82.63	80.23
RCWN+Gazz	83.59	84.85	82.40
RCWN+WLD	81.74	82.93	80.63
RCWN+WLD+Gazz	**83.80**	**85.03**	**82.64**

Table 3
Results on 10-fold cross-validation

Network Hyperparameters. We used $h = 750$ hidden units, a learning rate of 0.05, the word embedding size $d_w = 50$ and a size of 5 for the embeddings of discrete morphological and gazetteer features. Differently, we used a larger embedding, $d_t = 20$ for the NE tags, i.e., the tag predicted by the network for the previous words. We used the same hyperparameters for all the proposed networks and estimated the best ones with a 10-fold cross-validation on the training set.

Pre-training DNNs. Good weight initialization is crucial for training better NN models (Collobert et al. 2011; Bengio 2009). Over the years different ways of pre-training the network have been designed: layer-wise pre-training (Bengio 2009), word embeddings (Collobert et al. 2011) or by relying on distant supervised datasets (Severyn and Moschitti 2015b, 2015a). Here, we propose a pre-training technique using an off-the-shelf NER to generate noisily annotated data, e.g., a sort of distance/weakly supervision or self-training. Our Weakly Labeled Dataset (WLD) is built by automatically annotating articles from the local newspaper "L'Adige", which is the same source of the training and test sets of Evalita challenge. We split the articles in sentences and tokenized them. This unlabeled corpus is composed of 20.000 sentences. We automatically tagged it using EntityPro, which is a NER tagger included in the TextPro suite (Pianta, Girardi, and Zanoli 2008).

3.2 Results

Our models are evaluated on the Evalita 2009 dataset. We applied 10-fold cross-validation to the training set of the challenge[4] for performing parameter tuning and picking the best models.

Table 3 reports performance of our models averaged over 10-folds. We note that (i) modeling the output dependencies with RCWN leads to a considerable improvement in

4 The official evaluation metric for NER is the F1, which is the harmonic mean between Precision and Recall.

Models	F1	Prec.	Rec.
(Gesmundo 2009)	81.46	**86.06**	77.33
(Gesmundo 2009) - Gazz	76.21	83.91	69.79
(Zanoli and Pianta 2009)†	82.00	84.07	80.02
(Nguyen, Moschitti, and Riccardi 2009)	79.77	82.26	77.43
(Nguyen, Moschitti, and Riccardi 2009) - Gazz[5]	73.70	77.05	70.65
(Nguyen and Moschitti 2012) (CRF)	80.34	83.43	77.48
(Nguyen and Moschitti 2012) + RR†	**84.33**	85.99	**82.73**
RCWN	79.59	81.39	77.87
RCWN+WLD	81.42	82.74	80.14
RCWN+Gazz	81.47	83.48	79.56
RCWN+WLD+Gazz	82.81	85.69	80.10

Table 4
Comparison with the best NER systems for Italian. Models below the line were computed after the Evalita challenge. Systems marked with † use combinations of different learning models.

F1 over the CWN model of (Collobert et al. 2011) (our baseline); (ii) adding the gazetteer features leads to an improvement both in Precision and Recall, and therefore in F1; and (iii) pre-training the network on the weakly labeled training set produces improvement (although small), which is due to a better initialization of the network weights.

Table 4 shows the comparative results between our models and the current state of the art for Italian NER on the Evalita 2009 official test set. We used the best parameter values derived when computing the experiments of Table 3. Our model using both gazetteer and pre-training outperforms all the systems participating to the Evalita 2009 (Zanoli and Pianta 2009; Gesmundo 2009). It should be noted that (Nguyen and Moschitti 2012; Nguyen, Moschitti, and Riccardi 2010) obtained better results using a CRF classifier followed by a reranker (RR) based on tree kernels. However, our approach only uses one learning algorithm, which is simpler than models applying multiple learning approaches, such as those in (Nguyen and Moschitti 2012) and (Zanoli and Pianta 2009). Moreover, our model outperforms the (Nguyen and Moschitti 2012) CRF baseline (which is given in input to the tree-kernel based reranker) by ~ 2.5 points in F1. Thus it is likely that applying their reranker on top of our model's output might produce a further improvement over state of the art.

Finally, it is important to note that our model obtains an F1 comparable to the best system in Evalita 2009 without using any extra features (we only use words and 4 morphological features). In fact, when we remove the gazetteer features, our method still obtains the very high F1 of 81.42. Only two systems provided results without external knowledge (Gesmundo 2009) that obtained an F1 of 76.21 and (Nguyen, Moschitti, and Riccardi 2009) that obtained 73.70 on the development dataset. This shows that NNs can obtain high results without manually constructed gazzeteer.

Finally, Table 5 highlights the per-class performance of our models compared to the systems at the Evalita 2009 competitions. Our best system obtains a higher F1 on all the classes. Moreover, our basic model, i.e., RCWN, obtains an absolute improvement of ~ 10 points on the LOC class, which is the rarest entity type in the dataset. This improvement is lower in our best model (~ 4 points), possibly due to errors introduced by the off-the-shelf NER used to produce the weakly labeled dataset.

Models	GPE	LOC	ORG	PER
(Gesmundo 2009)	83.36	50.81	71.08	87.41
(Zanoli and Pianta 2009)†	85.13	51.24	70.56	88.31
(Nguyen, Moschitti, and Riccardi 2009)	82.85	42.34	67.89	86.44
RCWN	82.07	61.54	68.78	85.70
RCWN+WLD	84.01	61.09	71.28	86.95
RCWN+Gazz	85.95	**63.35**	69.23	86.97
RCWN+WLD+Gazz	**86.67**	55.94	**71.65**	**88.23**

Table 5
Per-class F1 comparison with the best NER systems for Italian. Models below the line were computed after the Evalita challenge. Systems marked with † use multiple learned models at test time.

4. Related Work and Discussion

Recently, many different NN models have been successfully applied to the task of Named Entity Recognition, both using Recurrent Neural Networks (Chiu and Nichols 2015) and Convolutional Neural Networks (Collobert et al. 2011). In this work, we opted for a simpler model since the amount of the data available for the Italian language is usually smaller than the data available for English. Moreover, previous approaches require many more parameters than our models, which therefore show faster convergence during training and faster classification at test time.

For the same reason, we did not used character level embeddings to automatically encode morphological features into the model as in (dos Santos et al. 2015; Chiu and Nichols 2015), since it requires an additional Convolutional layer to be applied on every word. Conversely, the neural networks proposed in this work encode features as embeddings and use a feedback loop to model the output dependencies. The feedback loop do not introduce additional complexity to the model compared to the Viterbi decoding described in (Collobert et al. 2011). The model in fact feeds the output of the previous iteration to a multilayer perceptron that slides over the sequence. The proposed model is rather simple but it is efficient and able to obtain the state of the art on the Italian language.

5. Conclusion

In this paper, we have proposed a new DNN for designing NERs in Italian. Its main characteristics are: (i) the RCWN feedback method, which can model dependencies of the output label sequence and (ii) a pre-training technique involving a weakly supervised dataset. Our system is rather simple and efficient as it involves only one model at test time with respect to current state-of-the-art systems that require multiple inference steps (Zanoli and Pianta 2009; Nguyen and Moschitti 2012). Our system achieves results comparable with the state of the art even without using gazetteers. It shows promising results on the LOC class, which is the rarest class and therefore more challenging in the Evalita 2009 dataset. Additionally, it does not require time-consuming feature engineering or extensive data processing for their extraction.

In the future, we would like to apply rerankers to our methods and explore combinations of DNNs with structural kernels.

References

Attardi, Giuseppe. 2015. Deepnl: a deep learning nlp pipeline. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 109–115, Denver, Colorado, June. Association for Computational Linguistics.

Bengio, Yoshua. 2009. Learning Deep Architectures for AI. *Foundations and Trends® in Machine Learning*, 2(1):1–127.

Chiu, Jason PC and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308*.

Collobert, Ronan, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (almost) from Scratch. *The Journal of Machine Learning Research*, 1(12):2493–2537.

dos Santos, Cıcero, Victor Guimaraes, RJ Niterói, and Rio de Janeiro. 2015. Boosting named entity recognition with neural character embeddings. In *Proceedings of NEWS 2015 The Fifth Named Entities Workshop*, page 25.

Gesmundo, Andrea. 2009. Bidirectional Sequence Classification for Named Entities Recognition. *Proceedings of EVALITA*.

Hinton, Geoffrey. 2014. Dropout : A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research (JMLR)*, 15(1):1929–1958.

Krogh, A. and J. Hertz. 1992. A Simple Weight Decay Can Improve Generalization. *Advances in Neural Information Processing Systems*, 4:950–957.

Mikolov, Tomas, Greg Corrado, Kai Chen, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *Proceedings of the International Conference on Learning Representations (ICLR 2013)*, pages 1–12.

Nadeau, David and Satoshi Sekine. 2007. A survey of named entity recognition and classification.

Nguyen, Truc-vien T, Alessandro Moschitti, and Giuseppe Riccardi. 2009. Conditional Random Fields: Discriminative Training over Statistical features for Named Entity Recognition. In *Proceedings of EVALITA*.

Nguyen, Truc-vien T, Alessandro Moschitti, and Giuseppe Riccardi. 2010. Kernel-based Reranking for Named-Entity Extraction. In *COLING '10 Proceedings of the 23rd International Conference on Computational Linguistics: Poster*, number August, pages 901–909. Association for Computational Linguistics.

Nguyen, V. and A. Moschitti. 2012. Structural reranking models for named entity recognition. *INTELLIGENZA ARTIFICIALE*, Volume 6,2.

Pereira, Fernando, John Lafferty, John Lafferty, Andrew Mccallum, Andrew Mccallum, and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In Proceedings of 18th. *International Conference on Machine Learning*, pages:282–289.

Pianta, Emanuele, Christian Girardi, and Roberto Zanoli. 2008. The TextPro tool suite. In *Proceedings of LREC*, pages 2603–2607. Citeseer.

Prechelt, Lutz. 1998. Early stopping-but when? In *Neural Networks: Tricks of the trade*. Springer, pages 55–69.

Qu, Lizhen, Gabriela Ferraro, Liyuan Zhou, Weiwei Hou, Nathan Schneider, and Timothy Baldwin. 2015. Big Data Small Data, In Domain Out-of Domain, Known Word Unknown Word: The Impact of Word Representation on Sequence Labelling Tasks. *arXiv preprint arXiv:1504.05319*.

Ratinov, Lev and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the International Conference On Computational Linguistics*, pages 147–155. Association for Computational Linguistics.

Severyn, Aliaksei and Alessandro Moschitti. 2015a. Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*, pages 959–962.

Severyn, Aliaksei and Alessandro Moschitti. 2015b. UNITN: Training Deep Convolutional Neural Network for Twitter Sentiment Classification. *Proceedings of SEMEVAL*.

Speranza, Manuela. 2009. The named entity recognition task at evalita 2009. In *Proceedings of EVALITA*.
Zanoli, R and E Pianta. 2009. Named Entity Recognition through Redundancy Driven Classi ers. *In: Proceedings of EVALITA 2009. Reggio Emilia, Italy*.