Graph Convolutional Networks for Named Entity Recognition

Cetoli, Alberto **Bragaglia, Stefano** **O'Harney, Andrew Daniel** **Sloan, Marc**
Context Scout
{alberto, stefano, andy, marc}@contextscout.com

Abstract

In this paper we investigate the role of the dependency tree in a named entity recognizer upon using a set of Graph Convolutional Networks (GCNs). We perform a comparison among different Named Entity Recognition (NER) architectures and show that the grammar of a sentence positively influences the results. Experiments on the *OntoNotes 5.0* dataset demonstrate consistent performance improvements, without requiring heavy feature engineering nor additional language-specific knowledge.[1]

1 Introduction and Motivations

The recent article by Marcheggiani and Titov (Marcheggiani and Titov, 2017) opened the way for a novel method in Natural Language Processing (NLP). In their work, they adopt a GCN (Kipf and Welling, 2016) approach to perform semantic role labeling, improving upon previous architectures. While their article is specific to recognizing the predicate-argument structure of a sentence, their method can be applied to other areas of NLP. One example is NER.

High performing statistical approaches have been used in the past for entity recognition, notably *Markov* models (McCallum et al., 2000), Conditional Random Fields (CRFs) (Lafferty et al., 2001), and Support Vector Machines (SVMs) (Takeuchi and Collier, 2002). More recently, the use of neural networks has become common in NER.

The method proposed by Collobert et al. (Collobert et al., 2011) suggests that a simple feed--forward network can produce competitive results with respect to other approaches. Shortly thereafter, Chiu and Nichols (Chiu and Nichols, 2015) employed Recurrent Neural Networks (RNNs) to address the problem of entity recognition, thus achieving state-of-the-art results. Their key improvements were twofold: using a bi-directional Long Short-Term Memory (LSTM) in place of a feed-forward network and concatenating morphological information to the input vectors.

Subsequently, various improvements appeared: using a CRF as a last layer (Huang et al., 2015) in place of a `softmax` function, a gated approach to concatenating morphology (Cao and Rei, 2016) and predicting nearby words (Rei, 2017). All such methods, however, understand text as a one dimensional collection of input vectors; any syntactic information – namely the parse tree of the sentence – is ignored.

We believe that dependency trees and other linguistic features play a key role on the accuracy of NER and that GCNs can grant the flexibility and convenience of use that we desire. In this paper our contribution is twofold: on one hand, we introduce a methodology for tackling entity recognition with GCNs; on the other hand we measure the impact of using dependency trees for entity classification upon comparing the results with prior solutions. At this stage our goal is not to beat the state-of-the-art but rather to quantify the effect of our novel architecture.

[1] A version of this system can be found at `https://github.com/contextscout/gcn_ner`.

Proceedings of the 16th International Workshop on Treebanks and Linguistic Theories (TLT16), pages 37–45,
Prague, Czech Republic, January 23–24, 2018.

Keywords: NER, dependency tree, graph, convolution

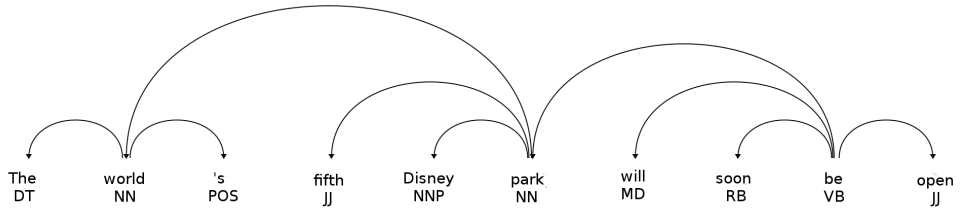


Figure 1: An example sentence along with its dependency graph. GCNs propagate the information of a node to its nearest neighbours.

As a final note, we notice that treebanks offer more information than a one dimensional sequence of words. This information is not used in conventional RNNs systems. Our paper opens the way for exploiting the syntax and dependency structures available in a treebank.

The remainder of the article is organized as follows: in Section 2 we introduce the theoretical framework for our methodology, then the features considered in our model and eventually the training details. Section 3 describes the experiments and presents the results. We discuss relevant works in Section 4 and draw the conclusions in Section 5.

2 Methods and Materials

2.1 Theoretical Aspects

Graph Convolutional Networks (Kipf and Welling, 2016) operate on graphs by convolving the features of neighbouring nodes. A GCN layer propagates the information of a node onto its nearest neighbours. By stacking together N layers, the network can propagate the features of nodes that are at most N hops away.

While the original formulation did not include directed graphs, they were further extended in Marcheggiani and Titov to be used on directed syntactic/dependency trees. In the following we rely on their work to assemble our network.

Each GCN layer creates new node embeddings by using neighbouring nodes and these layers can be stacked upon each other. In the undirected graph case, the information at the k^{st} layer is propagated to the next one according to the equation

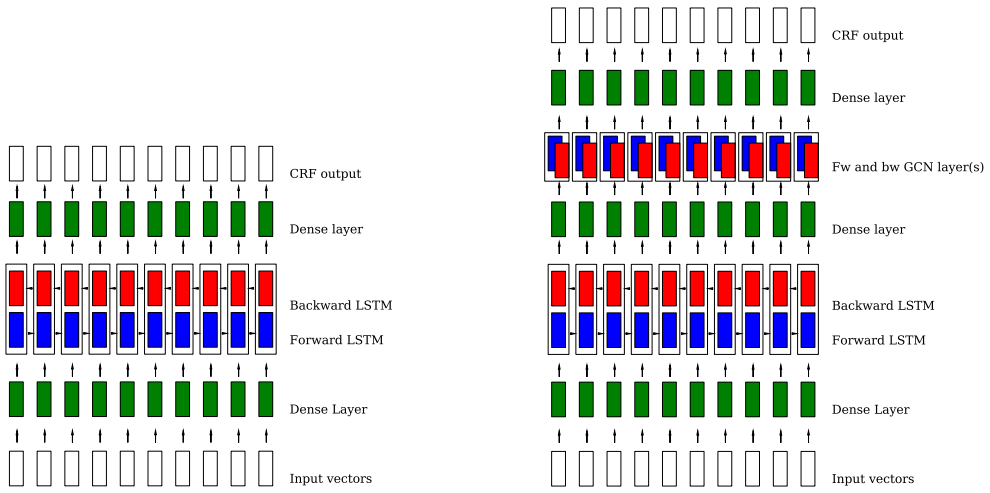
$$h_v^{k+1} = \mathrm{ReLU}\left(\sum_{u \in \mathcal{N}(v)} \left(W^k h_u^k + b^k\right)\right), \tag{1}$$

where u and v are nodes in the graph. $\mathcal{N}$ is the set of nearest neighbours of node v, plus the node v itself. The vector h_u^k represents node u's embeddings at the k^{st} layer, while W and b are a weight matrix and a bias – learned during training – that map the embeddings of node u onto the adjacent nodes in the graph; h_u belongs to $\mathbb{R}^m$, $W \in \mathbb{R}^{m \times m}$ and $b \in \mathbb{R}^m$.

Following the example in Marcheggiani and Titov, we prefer to exploit the directness of the graph in our system. Our inspiration comes from the bi-directional architecture of stacked RNNs, where two different neural networks operate forward and backward respectively. Eventually the output of the RNNs is concatenated and passed to further layers.

In our architecture we employ two stacked GCNs: One that only considers the incoming edges for each node

$$\overleftarrow{h}_v^{k+1} = \mathrm{ReLU}\left(\sum_{u \in \overleftarrow{N}(v)} \left(\overleftarrow{W}^k h_u^k + \overleftarrow{b}^k\right)\right), \tag{2}$$



(a) (b)

Figure 2: bi-directional architectures: (a) LSTM; and, (b) GCN layers.

and one that considers only the outgoing edges from each node

$$\overrightarrow{h}_v^{k+1} = \text{ReLU}\left(\sum_{u \in \overrightarrow{N}(v)} \left(\overrightarrow{W}^k h_u^k + \overrightarrow{b}^k\right)\right). \tag{3}$$

After N layers the final output of the two GCNs is the concatenation of the two separated layers

$$h_v^N = \overrightarrow{h}_v^N \oplus \overleftarrow{h}_v^N. \tag{4}$$

In the following, we refer to the architecture expressed by Equation 4 as a ***bi-directional GCN***.

2.2 Implementations Details

2.2.1 Using the dataset

We employ the *OntoNotes 5.0* dataset (Weischedel, 2013) for training and testing. This dataset annotates various genres of text for the purpose of entity recognition and co-reference resolution. The annotated sentences are provided with Part-of-Speech (PoS) tags and syntactic information. While we include the PoS tags in our tests, the Phrase Structure Grammar (PSG) structures in the *OntoNotes 5.0* are not used. The dependency graphs that are fed to the graph convolutional network are instead computed by an external parser, *Spacy v1.8.2* (Honnibal and Johnson, 2015).

In principle we could have translated the syntactic trees in the dataset to dependency graphs using - for example - the CCGBank manual (Hockenmaier and Steedman, 2007). We will investigate this approach in future works, while this paper lays down the technique for boosting entity recognition using GCNs.

2.2.2 Models

Our architecture is inspired by the work of Chiu and Nichols (Chiu and Nichols, 2015), Huang et al. (Huang et al., 2015), and Marcheggiani and Titov (Marcheggiani and Titov, 2017). We aim to combine a Bi-directional Long Short-Term Memory (Bi-LSTM) model with GCNs, using CRF as the last layer in place of a `softmax` function.

We employ seven different configurations by selecting from two sets of PoS tags and two sets of word embedding vectors. All the models share a bi-directional LSTM which acts as the foundation upon which we apply our GCN. The different combinations are built using the following elements:

Bi-LSTM We use a bi-directional LSTM structured as in Figure 2(a). The output is mediated by two fully connected layers ending in a CRF (Huang et al., 2015), modelled as a Viterbi sequence. The best results in the *dev set* of *OntoNotes 5.0* were obtained upon staking two LSTM layers, both for the forward and backward configuration. This is the number of layers we keep in the rest of our work. This configuration – when used alone – is a consistency test with respect to the previous works. As seen in Table 1, our findings are compatible with the results in (Chiu and Nichols, 2015).

Bi-GCN In this model, we use the architecture created in (Marcheggiani and Titov, 2017) where a GCN is applied on top of a Bi-LSTM. This system is shown in Figure 2(b) (right side). The best results in the *dev set* were obtained upon using only one GCN layer, and we use this configuration through our models. We employ two different embedding vectors for this configuration: one in which only word embeddings are fed as an input, the other one where PoS tag embeddings are concatenated to the word vectors.

Input vectors We use three sets of input vectors. First, we simply employ the word embeddings found in the *Glove* vectors (Pennington et al., 2014):

$$x_{\text{input}} = x_{\text{glove}}. \tag{5}$$

In the following, we employ the 300 dimensional vector from two different distributions: one with 1M words and another one with 2.2M words. Whenever a word is not present in the *Glove* vocabulary we use the vector corresponding to the word "entity" instead.

The second type of vector embeddings concatenates the *Glove* word vectors with PoS tags embeddings. We use randomly initialized Part-of-Speech embeddings that are allowed to fine-tune during training:

$$x_{\text{input}} = x_{\text{glove}} \oplus x_{\text{PoS}}. \tag{6}$$

The final quality of our results correlates to the quality of our Part-of-Speech tagging. In one batch we use the manually curated PoS tags included in the *OntoNotes 5.0* dataset (Weischedel, 2013) (*PoS (gold)*). These tags have the highest quality.

In another batch, we use the PoS tagging inferred from the parser (*PoS (inferred)*) instead of using the manually tagged ones. These PoS tags are of lower quality. An external tagger might provide a different number of tokens compared to the ones present in the training and evaluation datasets. This presents a challenge. We skip these sentences during training (1602 sentences out of 112300), while considering the entities in such sentences as incorrectly tagged during evaluation.

Finally, we add the morphological information to the feature vector for the third type of word embeddings. The reason – explained in (Cao and Rei, 2016) – is that out-of-vocabulary words are handled badly whilst using only word embeddings:

$$x_{\text{input}} = x_{\text{glove}} \oplus x_{\text{PoS}} \oplus x_{\text{morphology}}. \tag{7}$$

We employ a bi-directional RNN to encode character information. The end nodes of the RNN are concatenated and passed to a dense layer, which is integrated to the feature vector along with the embeddings and PoS information. In order to speed up the computation, we truncate the words by keeping only the first 12 characters. This operation is only done when computing the morphology vector, the word embeddings still refer to the full word. Truncation is not commonly done, as it hinders the network's performance; we leave further analysis to following works.

Dropout In order to tackle over-fitting, we apply dropout to all the layers on top of the LSTM. The probability to drop a node is set at 20% for all the configurations. The layers that are used as input to the LSTM do not use dropout.

Network output At inference time, the output of the network is a 19-dimensional vector for each input word. This dimensionality comes from the 18 tags used in *OntoNotes 5.0*, with an additional dimension which expresses the absence of a named entity. No Begin, Inside, Outside, End, Single (BIOES) markings are applied; at evaluation time we simply consider a *name chunk* as a contiguous sequence of words belonging to the same category.

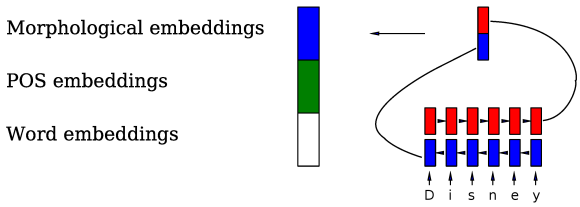


Figure 3: Feature vector components. Our input vectors have up to three components: the word embeddings, the PoS embeddings, and a morphological embedding obtained through feeding each word to a Bi-LSTM and then concatenating the first and last hidden state.

2.2.3 Training

We use *TensorFlow* (Abadi et al., 2015) to implement our neural network. Training and inference is done at the sentence level. The weights are initialized randomly from the uniform distribution and the initial state of the LSTMs are set to zero. The system uses the configuration in Appendix A.

The training function is the CRF loss function as explained in (Huang et al., 2015). Following their notation, we define $[f]_{i,t}$ as the matrix that represents the score of the network for the t^{th} word to have the i^{th} tag. We also introduce A_{ij} as the transition matrix which stores the probability of going from tag i to tag j. The transition matrix is usually trained along with the other network weights. In our work we preferred instead to set it as constant and equal to the transition frequencies as found in the training dataset.

The function f is an argument of the network's parameters θ and the input sentence $[x]_1^T$ (the list of embeddings with length T). Let the list of T training labels be written as $[i]_1^T$, then our loss function is written as

$$\mathcal{S}\left([x]_1^T,[i]_1^T,\theta,A_{ij}\right)-\sum_{[j]_1^T}\exp\left([x]_1^T,[j]_1^T,\theta,A_{ij}+[f]_{[i]_t,t}\right), \tag{8}$$

where

$$\mathcal{S}\left([x]_1^T,[i]_1^T,\theta,A_{ij}\right)=\sum_1^T\left(A_{[i]_{t-1},[i]_t}+f(\theta,A_{ij})\right). \tag{9}$$

At inference time, we rely on the Viterbi algorithm to find the sequence of tokens that maximizes $\mathcal{S}\left([x]_1^T,[i]_1^T,\theta,A_{ij}\right)$. We apply mini-batch stochastic gradient descent with the *Adam* optimiser (Kingma and Ba, 2014), using a learning rate fixed to 10^{-4}.

3 Experimental Results

In this section, we compare the different methods applied and discuss the results. The scores in Table 1[2] are presented as an average of 6 runs with the error being the standard deviation; we keep only the first significant digit of the errors, approximating to the nearest number.

The results show an improvement of $2.2\pm0.5\%$ upon using a GCN, compared to the baseline result of a bi-directional LSTM alone (1^{st} row). When concatenating the gold PoS tag embedding in the input vectors, this improvement raises to $4.6\pm0.6\%$. However, the gold tags in the *OntoNotes 5.0* only refer to the sentences within the dataset. Therefore, the performance of the system on new sentences must rely on inferred PoS tags.

The F_1 score improvement for the system while using inferred tags (from the parser) is lower: $3.2\pm0.6\%$.

[2]The results from Ratinov and Roth and Finkel and Manning are taken from Chiu and Nichols.

Description	DEV prec	DEV rec	DEV F_1	TEST prec	TEST rec	TEST F_1
Bi-LSTM + 1M *Glove* + CRF	80.9	78.2	79.5±0.3	79.1	75.9	77.5±0.4
Bi-LSTM + 1M *Glove* + CRF + GCN	82.2	79.5	80.8±0.3	82.0	77.5	79.7±0.3
Bi-LSTM + 1M *Glove* + CRF + GCN + PoS (gold)	82.1	83.7	82.9±0.3	82.4	81.8	82.1±0.4
Bi-LSTM + 2.2M *Glove* + CRF + GCN + PoS (gold)	83.3	84.1	83.7±0.4	83.6	82.1	82.8±0.3
Bi-LSTM + 2.2M *Glove* + CRF + GCN + PoS (inferred)	83.8	82.9	83.4±0.4	82.2	80.5	81.4±0.3
Bi-LSTM + 2.2M *Glove* + CRF + GCN + PoS (gold) + Morphology	86.6	82.7	84.6±0.4	86.7	80.7	83.6±0.4
Bi-LSTM + 2.2M *Glove* + CRF + GCN + PoS (inferred) + Morphology	85.3	82.3	83.8±0.4	84.3	80.1	82.0±0.4
Chiu and Nichols			**84.6**±0.3	86.0	86.5	**86.3**±0.3
Ratinov and Roth				82.0	84.9	83.4
Finkel and Manning				84.0	80.9	82.4
Durrett and Klein				85.2	82.9	84.0

Table 1: Results of our architecture compared to previous findings.

For comparison, increasing the size of the *Glove* vector from 1M to 2.2M gave an improvement of $0.7 \pm 0.5\%$. Adding the morphological information of the words, albeit truncated at 12 characters, improves the F_1 score by $2.2 \pm 0.5\%$.

Our results strongly suggest that syntactic information is relevant in capturing the role of a word in a sentence, and understanding sentences as one-dimensional lists of words appears as a partial approach. Sentences embed meaning through internal graph structures: the graph convolutional method approach – used in conjunction with a parser (or a *treebank*) – seems to provide a lightweight architecture that incorporates grammar while extracting named entities.

Our results – while competitive – fall short of achieving the state-of-the-art. We believe this to be the result of a few factors: we do not employ BIOES annotations for our tags, lexicon and capitalisation features are ignored, and we truncate words when encoding the morphological vectors.

Another improvement could come from converting the manually parsed trees in the *OntoNotes 5.0* dataset into dependency graphs. Using these graphs during training would eliminate any possible erroneous contributions coming from the external parser.

Our main claim is nonetheless clear: grammatical information positively boosts the performance of recognizing entities, leaving further improvements to be explored.

4 Related Works

There is a large corpus of work on named entity recognition, with few studies using explicitly non-local information for the task. One early work by Finkel et al. (Finkel et al., 2005) uses Gibbs sampling to capture long distance structures that are common in language use. Another article by the same authors uses a joint representation for constituency parsing and NER, improving both techniques. In addition, dependency structures have also been used to boost the recognition of bio-medical events (McClosky et al., 2011) and for automatic content extraction (Li et al., 2013).

Recently, there has been a significant effort to improve the accuracy of classifiers by going beyond vector representation for sentences. Notably the work of Peng et al. (Peng et al., 2017) introduces *graph LSTMs* to encode the meaning of a sentence by using dependency graphs. Similarly Dhingra et al. (Dhingra et al., 2017) employ *Gated Recurrent Units (GRUs)* that encode the information of acyclic graphs to achieve state-of-the-art results in co-reference resolution.

5 Concluding Remarks

We showed that dependency trees play a positive role for entity recognition by using a GCN to boost the results of a bidirectional LSTM. In addition, we modified the standard convolutional network architecture and introduced a bidirectional mechanism for convolving directed graphs. This model is able to improve

upon the LSTM baseline: Our best result yielded an improvement of $4.6 \pm 0.6\%$ in the F_1 score, using a combination of both GCN and PoS tag embeddings.

Finally, we prove that GCNs can be used in conjunction with different techniques. We have shown that morphological information in the input vectors does not conflict with graph convolutions. Additional techniques, such as the gating of the components of input vectors (Rei et al., 2016) or neighbouring word prediction (Rei, 2017) should be tested together with GCNs. We will investigate those results in future works.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. http://tensorflow.org/.

Kris Cao and Marek Rei. 2016. A joint model for word embedding and word morphology. *CoRR* abs/1606.02601. http://arxiv.org/abs/1606.02601.

Jason P. C. Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional LSTM-CNNs. *CoRR* abs/1511.08308. http://arxiv.org/abs/1511.08308.

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(Aug):2493–2537.

Bhuwan Dhingra, Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2017. Linguistic knowledge as memory for recurrent neural networks. *CoRR* abs/1703.02620. http://arxiv.org/abs/1703.02620.

Greg Durrett and Dan Klein. 2015. Neural CRF parsing. *CoRR* abs/1507.03641. http://arxiv.org/abs/1507.03641.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs Sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '05, pages 363–370. https://doi.org/10.3115/1219840.1219885.

Jenny Rose Finkel and Christopher D. Manning. 2009. Joint parsing and named entity recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, NAACL '09, pages 326–334. http://dl.acm.org/citation.cfm?id=1620754.1620802.

Julia Hockenmaier and Mark Steedman. 2007. CCGBank: A corpus of CCG derivations and dependency structures. *Comput. Linguist.* 33(3):355–396. https://doi.org/10.1162/coli.2007.33.3.355.

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1373–1378. https://aclweb.org/anthology/D/D15/D15-1162.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR* abs/1508.01991. http://arxiv.org/abs/1508.01991.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980. http://arxiv.org/abs/1412.6980.

Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with Graph Convolutional Networks. *CoRR* abs/1609.02907. http://arxiv.org/abs/1609.02907.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ICML '01, pages 282–289. http://dl.acm.org/citation.cfm?id=645530.655813.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 73–82. http://aclanthology.coli.uni-saarland.de/pdf/P/P13/P13-1008.pdf.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with GCN for semantic role labeling. *CoRR* abs/1703.04826. http://arxiv.org/abs/1703.04826.

Andrew McCallum, Dayne Freitag, and Fernando C. N. Pereira. 2000. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ICML '00, pages 591–598. http://dl.acm.org/citation.cfm?id=645529.658277.

David McClosky, Mihai Surdeanu, and Christopher D. Manning. 2011. Event extraction as dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT '11, pages 1626–1635. http://dl.acm.org/citation.cfm?id=2002472.2002667.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence N-ary relation extraction with Graph LSTMs. *Transactions of the Association of Computational Linguistics* 5:101–115. http://aclanthology.coli.uni-saarland.de/pdf/Q/Q17/Q17-1008.pdf.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

L. Ratinov and D. Roth. 2009. Design challenges and misconceptions in named entity recognition. In *CoNLL*. http://cogcomp.org/papers/RatinovRo09.pdf.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. *CoRR* abs/1704.07156. http://arxiv.org/abs/1704.07156.

Marek Rei, Gamal Crichton, and Sampo Pyysalo. 2016. Attending to characters in neural sequence labeling models. *CoRR* abs/1611.04361. http://arxiv.org/abs/1611.04361.

Koichi Takeuchi and Nigel Collier. 2002. Use of support vector machines in extended named entity recognition. In *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*. Association for Computational Linguistics, Stroudsburg, PA, USA, COLING-02, pages 1–7. https://doi.org/10.3115/1118853.1118882.

Ralph et al. Weischedel. 2013. Ontonotes release 5.0. *Linguistic Data Consortium, Philadelphia, PA* LDC2013T19. https://catalog.ldc.upenn.edu/docs/LDC2013T19/OntoNotes-Release-5.0.pdf.

A Configuration

Parameter	Value
Glove word embeddings	300 dim
PoS embedding	15 dim
Morphological embedding	20 dim
First dense layer	40 dim
LSTM memory	(2×) 160 dim
Second dense layer	160 dim
GCN layer	(2×) 160 dim
Final dense layer	160 dim
Output layer	16 dim
Dropout	0.8 (*keep probability*)

Table 2: Summary of the configuration used for training the network.