Convolutional Ladder Networks for Legal NERC and the Impact of Unsupervised Data in Better Generalizations

Cristian Cardellino, Laura Alonso Alemany, Milagro Teruel, Serena Villata, Santiago Marro

Convolutional Ladder Networks for Legal NERC and the Impact of Unsupervised Data in Better Generalizations

Cristian Cardellino,[1] **Laura Alonso Alemany,**[1] **Milagro Teruel,**[1]
Serena Villata,[2] **Santiago Marro**[1]
[1]Natural Language Processing Group, FAMAFyC-UNC, Córdoba, Argentina.
[2]Université Côte d'Azur. CNRS, Inria, I3S, France.
ccardellino@unc.edu.ar, lauraalonsoalemany@unc.edu.ar,
mteruel@unc.edu.ar, villata@i3s.unice.fr, smarro@gmail.com

Abstract

In this paper we adapt the semi-supervised deep learning architecture known as "Convolutional Ladder Networks", from the domain of computer vision, and explore how well it works for a semi-supervised Named Entity Recognition and Classification task with legal data. The idea of exploring a semi-supervised technique is to assess the impact of large amounts of unsupervised data (cheap to obtain) in specific tasks that have little annotated data, in order to develop robust models that are less prone to overfitting. In order to achieve this, first we must check the impact on a task that is easier to measure. We are presenting some preliminary experiments, however, the results obtained foster further research in the topic.

Introduction

In recent years, deep learning methods have provided very powerful models for different kind of tasks, like computer vision and natural language processing. An important asset for many of these deep learning models is the existence of vasts amounts of labeled data to train them. Having the power of a neural network with small amounts of data comes with the burden of the model memorizing the inputs and overfitting the data, which renders useless models.

Obtaining labeled data to improve these kind of deep learning models is expensive and sometimes very difficult. Depending on the task, it may require domain experts for the annotation. Legal Named Entity Recognition and Classification (NERC) is an example of this case, where lawyers are needed to, if not annotate, at least supervise the process. On the other hand, however, we have large amounts of unlabeled data available that is cheap and fast to obtain in large pools. However, unlabeled data cannot be used directly to train a regular deep learning model. In this scenario, a semi-supervised deep learning model like "Convolutional Ladder Networks" (Rasmus et al. 2015) is of high value.

In this work, by adapting Convolutional Ladder Networks (CLadder) from image to text, we explore how the information given by unlabeled data sources improves on the generalization of the model, making it less prone to overfitting. We compare the use of unlabeled data by testing: (i) a purely supervised Convolutional Neural Network, (ii) the CLadder, and comparing it with (iii) the Stanford CRF-NER as a reference tool for the NERC task. Even if CLadder does not reach the same performance as the reference tool, results show how unlabeled data contributes to reduce overfitting, by not drastically dropping performance when changing from training to test data.

The rest of the paper is organized as follows. First we lay down previous work and compare it with our approach. Then we highlight the main idea behind Ladder Networks and explain how we adapted CLadder for text data. We continue with a brief description of the dataset used for the experimentation and the experimental work (with its corresponding evaluation) done in this paper. Next, we analyze the results and end with some conclusions and future work.

Related work

Named entity recognition and classification (NERC) in the legal domain have been recently explored in the works of (Cardellino et al. 2017a) and (Cardellino et al. 2017b). In their works, they explore Curriculum Learning (Bengio et al. 2009) to train a supervised classifier for different hierarchies of the ontology. This work is based on their dataset, but as we are only in early stages of research we limited ourselves to the most abstract layer (i.e. the one with less classes).

Although convolutional neural networks (CNN) were originally associated to computer vision tasks, in recent years there have been a lot of different applications of convolutional networks in natural language processing. In particular, the network we base our architecture on is the one of (Kim 2014). This work evaluates a CNN architecture on various classification datasets, mostly comprised of Sentiment Analysis and Topic Categorization tasks, achieving good performance for the different datasets. The network is very simple, yet powerful, where the input is the sentence comprised of concatenated word2vec (Mikolov et al. 2013) embeddings, followed by a convolutional layer with multiple filters, a max-pooling layer, and finally a SoftMax classifier.

In the area of NERC, one of the latest neural networks architectures being used, which has reached state-of-the-art performances, is the Bidirectional LSTM-CNN network (Chiu and Nichols 2016). In this architecture, CNNs are used to extract character based features for the recurrent network. Unlike them, we explore the CNNs on a word level approach.

Ladder networks

This work adapts a semi-supervised deep learning technique from the domain of computer vision. Ladder networks were introduced in the work of (Rasmus et al. 2015), that extends the original work by Valpola (Valpola 2015) which introduces the concept of ladder networks for unsupervised learning.

The idea of using unsupervised learning to help training a neural network was proposed by Suddarth and Kergosien (Suddarth and Kergosien 1990). Most of the methods that use an auxiliary task to help the supervised learning are only applied at pre-training, followed by normal supervised learning (Hinton and Salakhutdinov 2006) (this is sometimes known in the literature as "disjoint semi-supervised learning"). In contrast, with ladder networks representations are learned jointly (this is sometimes called "joint semi-supervised learning" in the relevant literature). Unsupervised learning is implemented through an auxiliary task, for example, reconstructing the input. In learning, the hidden representations among supervised and unsupervised tasks are shared, and thus the network generalizes better.

The key idea of ladder networks is to simultaneously train a feed-forward neural network (whether it is fully connected or convolutional) alongside an autoencoder, with shared weights. The network is trained by learning two different objectives: a supervised one, given by the prediction error of the labeled data; and an unsupervised one, given by the reconstruction error of the unlabeled data.

The model structure is an autoencoder with skip connections from the encoder to decoder and the learning task is similar to that in denoising autoencoders but applied to every layer, not just the inputs. The skip connections relieve the pressure to represent details in the higher layers of the model because, through the skip connections, the decoder can recover any details discarded by the encoder. For a more detailed description of the ladder network we refer the reader to (Rasmus et al. 2015).

Convolutional ladder networks for text data

Convolutional ladder networks (CLadder) are a variation of ladder networks that were also proposed by (Rasmus et al. 2015). In their work the convolutional layers and the max pooling layers are stacked forming a deep convolutional network. Each convolution window can have different sizes of width and length, this also applies for the max pooling layers.

As we follow the approach by (Kim 2014), our Convolutional Neural Network (CNN) network is not "deep" but "wide". It has a single convolution layer (that can be considered a wide convolution because of the different sizes), a layer of global max pooling per convolution filter, and finally one fully connected layer with a SoftMax classifier.

For text, the convolution moves through one dimension only, which represents the number of words that the window

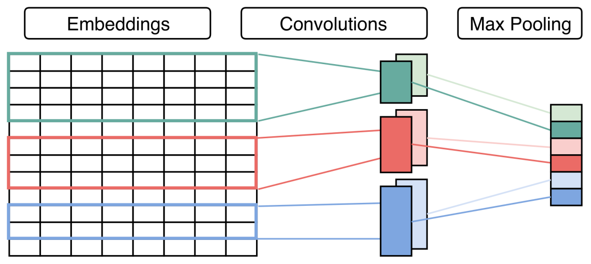

Figure 1: Convolutional network with text input. There are multiple windows sizes (2, 3 and 4 words for the colors blue, red and green respectively) and for each there are multiple number of filters. Each filter is then max pooled to get a convolved feature. The convolved features are concatenated to form a feature vector that is later fed to the SoftMax classifier.

will take into account to apply the filters. Sometimes this is referred as "temporal convolution" in the literature. The max pooling layer, on the other hand, is global to the feature map (that is the convolved features obtained by the convolution operation), thus giving only one feature per filter instead of multiple features per filter region as is the case for CNNs used in computer vision.

In this architecture there are different window sizes (i.e. number of words covered by the convolution) applied directly to the word representation input (generally a word embedding). This gives multiple feature maps, one per each feature, which gives one convolved feature after applying global max pooling. This is visualized in Fig. 1, where there are three sizes for the sliding windows (taking 2, 3 and 4 words, represented by colors blue, red and green respectively), as well as two filters per window (represented by different color intensity), we get a total of 6 convolved features map (the layers overlapped in the middle of the Figure. Finally, the layer with global max pooling operation gives a total of 6 features (3 sliding window with 2 filters each), which is the "convolved features vector" that is then fed to the SoftMax classifier.

Following this structure we defined the CLadder represented in Fig. 2. The ladder network has two encoder paths, one corrupted (the one on the left, marked by the red arrows) and one clean (the one on the right, marked by the green arrows), and one decoder path (in the center, with the blue arrows going from top to bottom). The corrupted encoder adds Gaussian noise in each layer as a method of regularization. The decoder, which works as an unsupervised learner, inverts the mappings of each layer of the encoder. It uses a denoising function to reconstruct the activations of each layer given the corrupted version of the mirror layer (denoted by the purple arrow going from left to right), and the previous layer output (the blue arrow). The target at each layer is the clean version of the activation (the dotted arrow going from right to left) and the difference between the reconstruction and the clean version serves as the denoising cost of that layer. The denoising function used in these experiments is

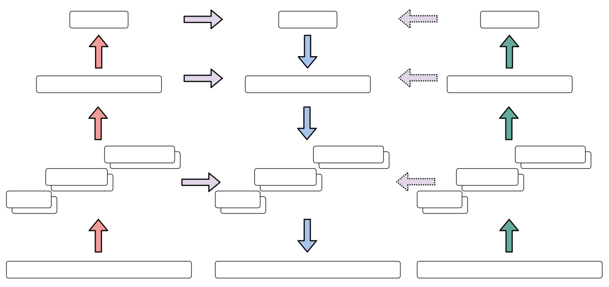

Figure 2: Convolutional ladder network: the corrupted encoder is the one on the left, connecting the layers by red arrows. The clean encoder is the one on the right, connecting the layers with the green arrows. The decoder path is the one in the middle, connecting the layers with the blue arrows going from top to bottom. Each layer in the decoder is fed with information of the previous layer (blue arrow) and the same level layer of the corrupted encoder path (purple arrows on the left). It uses a denoising function with that information to try and reconstruct the input of each layer given by the clean encoder (purple dotted arrows on the right).

the same one defined in the original paper of CLadders. In the encoder paths, the bottom layer represents the inputs and the top layer the outputs (predictions). In the decoder path, the bottom layer is the output of the autoencoder (the reconstruction of the original input).

The supervised cost between the predicted label and the ground truth label is calculated from the output of the corrupted encoder and the target label. The unsupervised cost is the sum of the denoising cost of all layers scaled by a hyperparameter that denotes the importance of each layer. For example, the first layers are more important than the last to reconstruct the input. The final cost is the sum of the supervised and the unsupervised cost. Batch normalization (Ioffe and Szegedy 2015) is applied to each preactivation including the topmost layer to improve convergence (due to reduced covariate shift) and to prevent the denoising cost from encouraging the trivial solution (encoder outputs constant values as these are the easiest to denoise). Beside this we also decided to add regularization by L2 norm of the weights in the encoder and decoder path, which proved to be useful for better generalizations (we also tried dropout but without any improvements on the task).

In the encoder paths, the first layer (from bottom to top) represents the convolutions over the words, the different sizes of the layers mean different sizes of the sliding window, and the multiple filters are represented by the rectangles overlapped (in Fig. 2 there are 3 different sizes for convolution and 2 filters per convolution size, giving a total of 6 feature maps). The second layer represents the pooled features of each filter and each convolution size (by global max pooling). The final layer is a fully connected layer with a SoftMax classifier.

In the decoder path, the first layer (from top to bottom) is a "transpose" of the same layer of the encoder path (the fully connected layer). The second layer (that corresponds to the max pooling layer in the encoder) is an "upsampling" layer that simply takes the input and repeats it to map the dimensions of the convolutional layers. Finally, the last layer is a "transposed convolutional" layer (also know as "deconvolutional layer") that maps the convolutions to the original input matrix (the one in Fig. 1). It is important to note that the mappings (and reconstruction objectives) of the decoder path to the encoder, in the case of the convolutional layer, are for each of the different window slide sizes (that is there is mapping between the convolution layer with 2-word windows, 3-word windows, etc.), not between convolutions of different window sizes.

Training dataset

As labeled data, we exploited Wikipedia links. To build our corpus, we downloaded a XML dump of the English Wikipedia[1] from March 2016, and we processed it via the WikiExtractor (of Pisa 2015) to remove all the XML tags and Wikipedia markdown tags, but leaving the links. We extracted all those articles that contained a link to an entity of the WordNet- and Wikipedia-based YAGO ontology[2] (Suchanek, Kasneci, and Weikum 2007) that has been mapped to the legal ontology presented in (Cardellino et al. 2017b). We considered as tagged entities the spans of text that are an anchor for a hyperlink whose URI is one of the mapped entities. We obtained a total of 4,5 million mentions, corresponding to 102,000 unique entities. Then, we extracted sentences that contained at least one mention of a named entity.

We consider the problem of Named Entity Recognition and Classification as a word-based representation, i.e., each word represents a training instance and is represented, in the network, by the concatenation of all the words that precede it. Then, words within the anchor span belong to one of the NE classes, others to the O class (Outside a Named Entity). The O class made more than 90% of the instances. This imbalance in the classes results largely biased the classifiers, so we randomly downsampled non-named entity words to make them at most 50% of the corpus. The resulting corpus consists of 10 million words, with words belonging to the O-class already downsampled. The data was split in three datasets: 80% for training, 10% for validation and 10% for test. The test dataset is the one used for evaluation and is held out until the final models were ready. The validation dataset was the one used to tune the hyperparameters of the different models.

As said before, Ladder Networks rely on a big quantity of unlabeled data to obtain a good representation of the universe. For these experiments, we use the same Wikipedia corpus as unlabeled data, because it is big enough. However, the autoencoder part of the ladder network captures other aspects of the same instances, obtaining a different representation that improves the representation obtained in the supervised task. In some of the experiments, the labeled data is limited to a percentage of the total labeled data, however the

[1]https://dumps.wikimedia.org/

[2]https://www.yago-knowledge.org/

unlabeled data is still the same amount. That is a more realistic scenario, because CLadder can take advantage of much unlabeled data for its training objective.

Experimental setting

In this paper we want to assess how the unsupervised objective of the ladder network helps improve the generalization of the model. We measure this by as the reduction in overfitting of the CLadder as compared to supervised methods. Moreover, we want to explore how the amount of labeled training data affects the overfitting tendency of a model. In order to do that, we carry out two sets of experiments.

First we compare: (i) a purely supervised Convolutional Neural Network (CNN) for NERC, (ii) the same architecture in a semi-supervised environment given by the ladder networks, and, to have a point of comparison with a more standard algorithm, (iii) we use the reference given by the Stanford CRF-NER tool (Finkel, Grenager, and Manning 2005).

As we want to assess the impact provided by unlabeled data in the final task, we train two sets of classifiers, one using the whole corpus and one using only 10% of the labeled corpus.

Finally, for the case of the CNN and the CLadder, we inspect how the model's supervised cost progresses as different epochs occur while augmenting the number of unlabeled examples (one epoch is one pass of the mini-batch stochastic gradient descent algorithm through the whole corpus).

The neural networks were trained using backpropagation and the ADAM optimization (Kingma and Ba 2014). The sizes of the sliding windows were 2 to 5 words, with 32 filters each (which gives a total of 448 convolved features). The hyperparameters (learning rate, regularization rate, noise of the corrupted encoder, weights of the reconstruction costs, number of filter, sizes of the windows, etc.) were chosen by random hyperparameter optimization and evaluated on the validation data.

Each instance is the sequence of words up to the word that is being classified. The words are represented by a concatenation of their dense vector and the dense vector of the part-of-speech tag of that word. Both the word and part-of-speech tags vectors were calculated previously with word2vec on the legal Wikipedia corpus. Originally we only used the word vector alone, but after some experiments we found out that the use of a PoS tag embedding improved the final overall performance. The PoS tags were obtained through the Stanford Parser (Klein and Manning 2003).

Evaluation

The evaluation of the experiments that compare the different networks and the Stanford NER were done on the training dataset, after the model finished training, as well as the held-out test set described previously.

Our idea is to measure the overfitting of different models when comparing the results of the training and test sets. An indicator of model overfitting is the difference in performance between training and test. The metrics we measure are three: accuracy, F1-score macro average (i.e. unweighted mean) and F1-score weighted average. The idea of showing both these averages is to also see how biased the models are toward the most frequent class, as a higher difference between these results (i.e. a very low value in macro average and very high value in weighted average) shows the algorithm is more biased towards the most frequent class.

In order to assess the evolution of the supervised cost (i.e. the prediction error) for training and validation data, we show the learning curves of 5 different experiments: (i) using only supervised CNN (i.e. no unlabeled data), and using CLadder with (ii) 25%, (iii) 50%, (iv) 75%, and (v) 100% of the unlabeled data (in all but (v), the unlabeled data is randomly sampled from the unannotated corpus). For these different experiments we split the supervised training data into 10-folds, train a model with 9 folds and evaluate it on the remaining fold (i.e. the "validation data", in this scenario, is really part of the training data). This is done for each fold.

We record the training and validation supervised cost on each epoch on a run of 10 epochs and use the information to calculate the mean and the standard deviation of the loss. The idea is to see how the proportion of unlabeled data in the ladder network affects the model generalization as well as the "error due to high variance" (Manning, Raghavan, and Schütze 2008) the models have. This type of error is defined as the variation of the prediction of learned classifiers: it measures how inconsistent the predictions are from one another, over different datasets, not whether they are accurate or not.

Analysis of results

Evaluation of models' overfitting tendency

For the first set of experiments, described in the previous section, we show the results in Table 1. The table displays Accuracy, F1-score macro average, and F1-score weighted average on training and test data.

There are two sets of experiments, one with the full labeled dataset and one with only 10% (in the case of the CLadder, the unlabeled data remains total since it is a real case scenario with the availability of unlabeled information).

When using the full labeled dataset we see that, on training data, the two supervised approaches overcome the CLadder in all three metrics. However, on test data, accuracy and F1-score drop drastically for both supervised approaches, while performance is roughly maintained at the same level for CLadder. Moreover, CLadder's performance is much better than the performance of the supervised CNN. This tendency is even more visible if only 10% of the corpus is used for training, as in that case, the supervised CNN overfitting is even bigger, since the difference between training and test data performance increases.

Compared with the Stanford CRF, CLadder is not so far behind in performance on the test data. In future work, we will explore the impact of adding a Recurrent Layer or CRF to the top of the CLadder, as this kind of architecture has been successful for a variety of sequential NLP tasks, including NERC.

Regarding the F1-score averages, it is noticeable that, while in all cases drops significantly, for the macro average

Training Size	Algorithm	Training Data			Test Data		
		Acc	F1 M	F1 W	Acc	F1 M	F1 W
Full	Stanford CRF	0.99	0.98	0.99	0.83	0.51	0.82
	Supervised CNN	0.80	0.60	0.79	0.72	0.45	0.71
	CLadder	0.78	0.58	0.74	0.77	0.48	0.76
10%	Stanford CRF	0.99	0.99	0.99	0.75	0.44	0.73
	Supervised CNN	0.84	0.63	0.82	0.61	0.40	0.60
	CLadder	0.79	0.58	0.73	0.72	0.43	0.7

Table 1: Table of results for Stanford CRF-NER, supervised CNN and ladder CNN. It shows the Accuracy (Acc), F1-score Macro Average (F1 M) and F1-score Weighted Average (F1 W) for both training and test data and for training data using the full training dataset and only 10% of the training dataset

(the one that is the most susceptible to changes in classes with less instances), it is for CLadder that also drops the least, while maintaining the weighted average. This means that in general, the unlabeled data, besides helping in the model generalization, it makes it more robust to the less frequent classes, something that is very problematic in unbalanced datasets (very common in natural language processing tasks).

From these results it is clear that unsupervised data is indeed helpful to obtain a better generalization in the model, as the fact that the model cannot fit so well the training data (as the supervised models do) doesn't affect how well the model works with unseen examples of the test data.

Learning curves

Figure 3 shows the results of the second experiment we described in the previous section. The plot consists of 5 different plots, one for each experiment using different sizes of the unlabeled corpus to train the CLadder. The leftmost graphic shows the case for the supervised CNN, and as we move to the right we have more unlabeled data available for the CLadder. It is interesting to see how using only 25% of the unlabeled data to train the network affects the learning curve as a whole.

In all cases, for the CLadder, the training cost in the first epoch begins higher than for Supervised data (in which practically stays the same) and decreases after some epochs. Specially, the more unlabeled data there is for each epoch, the more the training cost decreases. This is a symptom of the unlabeled data affecting the weights of the encoder path (by trying to optimize the reconstruction) and thus making it more difficult the model overfitting training data. However, the most important result in this graphic is shown in the case of the validation data. For supervised CNN, there is a high gap between training and validation data throughout all the iterations, but for the case of CLadder this gap decreases as the model fits the training data better. Moreover, the more unlabeled data there is available, the more similar are the training and validation learning curves and the better results we obtain for validation data. Also, there is the error due to high variance, shown by the width of the shade (i.e. the standard deviation for the folds), which is higher for Supervised CNN than for CLadders. This means fully supervised models suffer from higher variance, thus making them more prone to overfitting when there is no presence of unlabeled data that helps the model generalize.

Conclusions and future work

In this paper we adapted a semi-supervised deep learning technique from computer vision into a natural language processing task of named entity recognition and classification in the legal domain. We wanted to assess how unlabeled data affects a model. By doing some experiments with this adapted technique we have shown the importance of unlabeled data and the impact it can have on better model generalization. The results we achieved, even if not state-of-the-art, are promising in that they generalize well to unseen data, and we will continue to explore them.

We are currently working on applying this approach to smaller datasets, where the impact of a mechanism to counterbalance the overfitting tendency will is more valuable.

Further areas of study are to adapt different convolutional neural networks architectures used on other natural language processing tasks and see if that helps improving the performance even more. For example, something in the line of (Chiu and Nichols 2016), where the character level convolution could be included in the CLadder. Even more, there are some research for Recurrent Ladder Networks applied in image data which could also be adapted for natural language processing tasks.

Acknowledgements

The authors have received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skodowska-Curie grant agreement No 690974 for the project MIREL: MIning and REasoning with Legal texts.

References

Bengio, Y.; Louradour, J.; Collobert, R.; and Weston, J. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, 41–48. New York, NY, USA: ACM.

Cardellino, C.; Teruel, M.; Alemany, L. A.; and Villata, S. 2017a. Learning Slowly To Learn Better: Curriculum Learning for Legal Ontology Population. In *Thirtieth International Florida Artificial Intelligence Research Society Conference (FLAIRS 2017).*

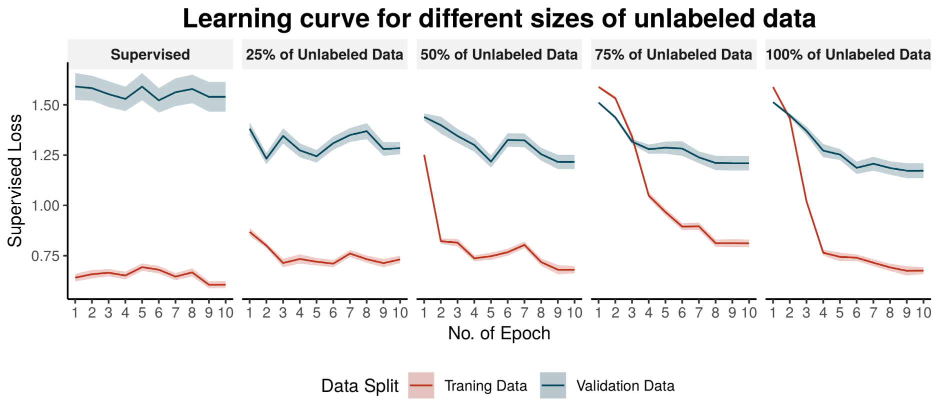

Figure 3: Learning curve: mean and standard deviation of the supervised cost function for training (red) and validation (indigo) datasets across the successive epochs of the algorithm. The line represents the mean of the loss and the shaded area represents the standard deviation (error due to high variance) for the supervised cost for the different folds of training data. The left-most graphics represents the case of purely supervised CNN classification (no unsupervised data). The other represent the models using different portions of the unlabeled corpus in the ladder network.

Cardellino, C.; Teruel, M.; Alemany, L. A.; and Villata, S. 2017b. A low-cost, high-coverage legal named entity recognizer, classifier and linker. In *Proceedings of the 16th Edition of the International Conference on Articial Intelligence and Law*, ICAIL '17, 9–18. New York, NY, USA: ACM.

Chiu, J., and Nichols, E. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics* 4:357–370.

Finkel, J. R.; Grenager, T.; and Manning, C. D. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In for Computer Linguistics, T. A., ed., *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, 363–370.

Hinton, G. E., and Salakhutdinov, R. R. 2006. Reducing the dimensionality of data with neural networks. *Science* 313(5786):504–507.

Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, 448–456. JMLR.org.

Kim, Y. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1746–1751. Association for Computational Linguistics.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Klein, D., and Manning, C. D. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, 423–430. Stroudsburg, PA, USA: Association for Computational Linguistics.

Manning, C. D.; Raghavan, P.; and Schütze, H. 2008. *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, 3111–3119. USA: Curran Associates Inc.

of Pisa, M. U. 2015. Wikiextractor. http://medialab.di.unipi.it/wiki/Wikipedia_Extractor.

Rasmus, A.; Valpola, H.; Honkala, M.; Berglund, M.; and Raiko, T. 2015. Semi-supervised learning with ladder networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, 3546–3554. Cambridge, MA, USA: MIT Press.

Suchanek, F. M.; Kasneci, G.; and Weikum, G. 2007. Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, 697–706. New York, NY, USA: ACM.

Suddarth, S. C., and Kergosien, Y. L. 1990. Rule-injection hints as a means of improving network performance and learning time. In Almeida, L. B., and Wellekens, C. J., eds., *Neural Networks*, 120–129. Berlin, Heidelberg: Springer Berlin Heidelberg.

Valpola, H. 2015. Chapter 8 - from neural pca to deep unsupervised learning. In Bingham, E.; Kaski, S.; Laaksonen, J.; and Lampinen, J., eds., *Advances in Independent Component Analysis and Learning Machines*. Academic Press. 143 – 171.