A Deep Learning Solution to Named Entity Recognition

Rudra Murthy V and Pushpak Bhattacharyya

Indian Institute Of Technology Bombay, India
{rudra,pb}@cse.iitb.ac.in

Abstract. Identifying Named Entities is vital for many Natural Language Processing (NLP) applications. Much of the earlier work for identifying named entities focused on using handcrafted features and knowledge resources (feature engineering). This is a barrier for resource-scarce languages as many resources are not readily available. Recently, Deep Learning techniques have been proposed for various NLP tasks requiring little/no hand-crafted features and knowledge resources, instead the features are learned from the data. Many proposed deep learning solutions for Named Entity Recognition (NER) still rely on feature engineering as opposed to feature learning. However, it is not clear whether the deep learning system or the engineered features are responsible for the positive results reported. This is in contrast with the goal of deep learning systems *i.e.,* to learn the features from the data itself. In this study, we show that a feature learned deep learning system is a viable solution to NER task. We test our deep learning systems on CoNLL English and Spanish NER datasets. Our system is able to give comparable results with the existing state-of-the-art feature engineered systems for English. We report the best performance of 89.27 F-Score for English when comparing with systems which do not use any handcrafted features or knowledge resources. Evaluation of our trained system on out-of-domain data indicate that the results are promising with the reported results. Our system when tested on Spanish NER achieves the best reported F-Score of 82.59 indicating its applicability to other languages.

Keywords: deep learning, named entity recognition, recurrent neural network, convolutional neural network

1 Introduction

Named Entity Recognition has been an important task in NLP. Various problems like Information Extraction, Question Answering, Machine Translation requires identifying named entities. Existing Supervised NER systems typically require large amounts of training data. Success of many of these systems depend on handcrafted features and knowledge resources in the form of gazetteers, part-of-speech taggers etc. This poses a major constraint for resource poor languages.

A diverse set of machine learning algorithms have been applied for tackling NER as part of CoNLL 2003 NER Shared Task Challenge [20]. The shared task saw use of many statistical models like Hidden Markov Models, Maximum Entropy Models, Conditional Random Fields (CRF), Voted Perceptrons, Recurrent Neural Networks. Many of the participants reported results on NER using system combination. Much of the focus was on using handcrafted features in the form of gazetteers, part-of-speech tags, affixes, capitalization features, chunk tags. The inclusion of these features and also system combination achieved best results on the shared task challenge.

Deep learning systems, which use multiple layers of neural network have shown promising results in various applications. The advantage offered by the deep learning technique is the requirement for little/no handcrafted features for these tasks. This is opposed to non-deep learning systems which typically rely on knowledge resources and lots of handcrafted features. The deep learning systems typically learn useful representations (feature learning) in an unsupervised way on a large unlabeled data which are then used as features in the supervised task.

Deep Learning techniques have also been successfully applied in various NLP tasks [4, 17, 16, 3]. Word embeddings [10, 11] are continuous low-dimensional dense vector representations of words learned in an unsupervised way. These word embeddings have been shown to capture various syntactic and semantic information about the word. Such representations when used in mainstream NLP applications have shown to perform better or achieve comparable performance compared to existing systems.

In this paper, we explore the use of a complete deep learning based approach for NER. We use Long Short Term Memory (LSTM) variant of Recurrent Neural Network along with Convolutional Neural Network (CNN) to train a NER system. The system uses only pre-trained word embeddings as input and and the decoder is much simpler compared to the decoder used in Senna[4]. Our experiments indicate that a feature learned deep learning approach is able to achieve closer to state-of-the-art results.

2 Related Work

System based on the combination of various machine learning algorithms was the best performing system[8] in CoNLL Shared task for English. This system used various features like part-of-speech tags, affixes, orthographic information, gazetteers, chunk information etc. Later a system based on multi-task approach [1] further improved the performance. The approach used a handful of features like part-of-speech tags, affixes, tokens in a syntactic window chunk etc.

A complete deep learning system, Senna[4] was proposed to solve various NLP problems. The model proposed to use a time-delay neural network which ran over words and a decoder layer calculating sentence-level likelihood using transition matrix at the top to find the best tag sequence at the sentence level. The model used pre-trained word embeddings and during training of the NER system the word embeddings were updated. The only features used by the system were pre-trained word embeddings, uppercase information and gazetteer list.

A new way to train phrase embeddings was proposed in the context of NER [12]. These phrase embeddings are then used with other features to train an NER system. The system used two CRFs, where the output from first CRF is given as input to the second CRF. This system achieves the best reported results on CoNLL Shared task data.

Bidirectional LSTMs[9] were also tested for NER task. They observe that vanilla Bidirectional LSTMs have the disadvantage of not being able to capture tag dependencies. They use a decoder layer similar to [4] at the top to capture the tag dependencies.

Another deep learning solution, CharWNN[14], to automatically retrieve relevant features from the character sequence forming a word in the context of part-of-speech tagging was proposed. They send the character sequence through a CNN and obtain character level features. These character level features are then augmented to the word embeddings. The model was later applied to Portuguese and Spanish NER [5].

A common theme emerging from most of the systems is feature engineering. Existing deep learning systems proposed also rely on feature engineering. Deep Learning approach with no feature engineering for NER as a viable solution needs to be explored. This is important because for many resource poor languages, there is unavailability of resources required and the need for creating handcrafted features. In this paper we study the feasibility of such a complete deep learning approach on English and Spanish NER. We also compare our approach with CharWNN[5] for Spanish NER as theirs is a complete deep learning approach.

3 Deep Learning NER

Given a sequence of word-entity label pairs i.e, $D = (X, Y)$ where $X = (x_1, \ldots, x_n)$ is the sequence of words in a sentence and $Y = (y_1, \ldots, y_n)$ is the corresponding tags. The task is to find the best possible named entity tag sequence t^* for a given sequence of words as in equation 1.

$$\operatorname*{argmax}_{t^*} P(t|X) \tag{1}$$

This involves estimating the parameters of the conditional probability $P(Y|X)$. The conditional probability can be decomposed as in equation 2.

$$P(y_1, \ldots, y_n|x_1, \ldots, x_n) = \prod_{i=1}^{N} P(y_i|x_1, \ldots, x_n, y_{i-1}) \tag{2}$$

LSTMs have traditionally been favored for problems with sequential nature. For NER task the modeling using LSTM is given in equation 3,

$$P(y_1, \ldots, y_n|x_1, \ldots, x_n) = \prod_{i=1}^{N} P(y_i|g(x_1, \ldots, x_i)) \tag{3}$$

where g is a LSTM which extracts relevant features by looking at current word and all the previous words.

Since the above modeling does not take into account the information from the right context Bidirectional LSTM [15] is preferred as it is able to capture information from both the directions.

$$P(y_1, \ldots, y_n | x_1, \ldots, x_n) = \prod_{i=1}^{N} P(y_i | g(x_1, \ldots, x_i), h(x_i, \ldots, x_n)) \tag{4}$$

Here both g and h are LSTMs and we use the same LSTM for both forward direction as well as backward direction. We call this architecture Bi-LSTM.

This is the simplest LSTM model usually used for Sequence Labeling tasks. The major disadvantage with the above approach is the independence assumption between successive tags i.e, the model does not account for $P(y_i|y_{i-1})$. For example in NER, modeling that *I-PER* tag always follows *B-PER* tag is important. Modeling of this dependence is crucial for successful application of LSTMs for sequence labeling task.

Later BI-LSTM-CRF[9] was proposed which used a CRF like layer which was added on top of LSTM layer to obtain the best tag sequence. They show that Bidirectional LSTM performs relatively poorly and the accuracy increases with the addition of CRF like top layer.

In our work, we instead use a *teacher training* model to capture the tag dependencies. We use Feedforward Neural network as the decoder. This decoder takes in representation from the Bidirectional LSTMs as well as the correct previous tag as input. This kind of architecture is similar to the decoder in Neural Machine Translation system. During testing, Viterbi decoding is used to find the best possible tag sequence.

$$P(y_1, \ldots, y_n | x_1, \ldots, x_n) = \prod_{i=1}^{N} P(y_i | g(x_1, \ldots, x_i), h(x_i, \ldots, x_n), y_{i-1}) \tag{5}$$

The architecture of the model is as shown in figure 1. The input to the system is pre-trained word embeddings. Forward LSTM reads the entire source sequence one word at a time left-to-right. Similarly backward LSTM reads the words from right-to-left. The hidden state of both the forward and backward LSTMs for a particular word is concatenated. Additionally true previous tag is concatenated and given to the decoder for predicting the NER tag.

3.1 Character nGram Features

The bidirectional LSTM model described above relies heavily on pre-trained word embeddings. It is clear that various character-level features like suffixes, uppercase information, presence of non-alphanumeric characters help in NER task. Unlike existing systems which augments these handcrafted features into the word embedding we follow the path of learning these features from the data

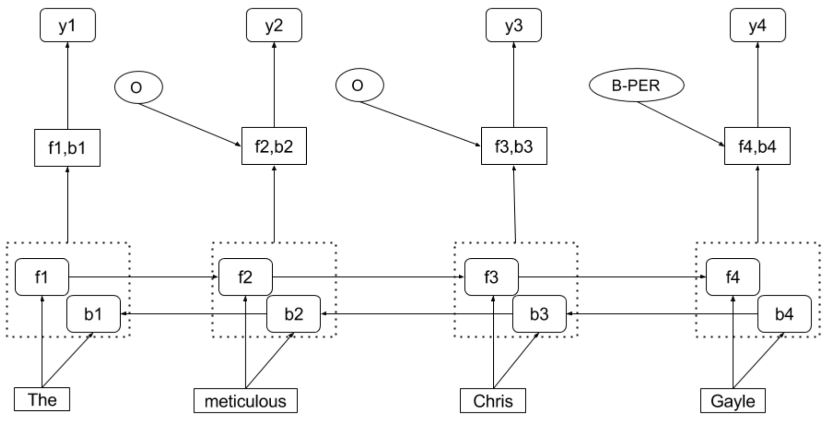

Fig. 1: Architecture of Bidirectional LSTM for NER

[14, 5]. The major difference from our approach and theirs is the use of multiple region sizes. Unlike CharWNN[14, 5] where region size is kept to 5 characters, we extract features from convolutional layers for each region size from 1 to 4. The convolutional layer is followed by a max-pooling layer. These character-level features bring in additional information along with the word embeddings. The intuition for having parallel convolutional layers of varying region size is to extract relevant nGram character features. For example, convolutional layer looking at unigram character tries looks for presence of uppercase characters and presence of non-alphanumeric characters. Convolutional layer looking at trigram characters tries to extract relevant trigram character features and need not worry about presence/absence of uppercase characters.

The architecture of CNN layer which runs over a unigram and extract multiple features is as shown in figure 2. Unlike CharWNN[14, 5] we do not have a common character lookup table before the convolutional layer. The convolutional layer which runs over unigrams extracts character embedding but they serve a different purpose. By not having a common character lookup table we directly look for relevant ngram character sequences. The features extracted are augmented into the pre-trained word embeddings. We call this system *Feature Learned NER*.

3.2 Sparse Word Embeddings

The current neural embeddings are dense and uninterpretable. These dense embeddings capture many modalities of the word. Extracting relevant features from this dense representation for a particular task may require a complex model. To

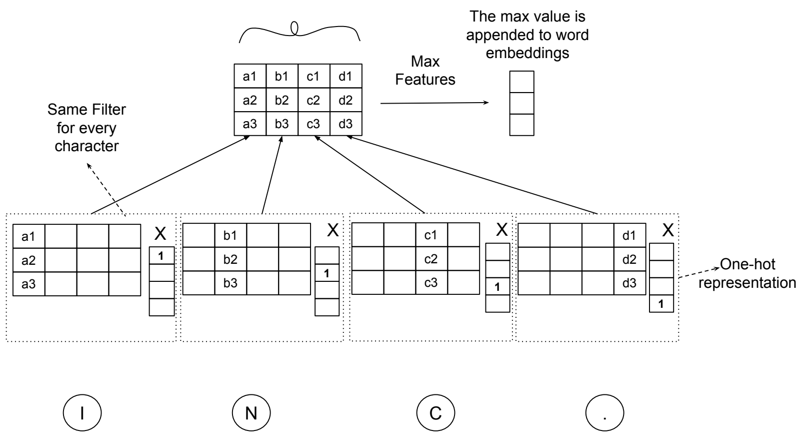
The max value is appended to word embeddings
Max Features
Same Filter for every character
a1 b1 c1 d1
a2 b2 c2 d2
a3 b3 c3 d3
X
1
One-hot representation
I
N
C
.

Fig. 2: Character Level CNN for NER

tackle this we would like to have a relatively sparse representation for the word but capturing much of the information from the dense representation.

There has been some work to obtain interpretable sparse representations from the word embeddings [7]. The approach uses sparse coding to obtain sparse interpretable word representations. Unlike the previous work which looks for interpretable word representations we use a relatively sparse word representations on NER task to evaluate its usefulness.

We learn a sparse word representation by sending the pre-trained word embedding through an autoencoder layer. We use Rectified Linear Units (ReLU) at the hidden layer followed by dropout layer [18]. By using ReLUs we achieve some kind of sparsity in the hidden layer representation and dropout layer bring in regularization into the model. The sparse hidden layer representation obtained replaces the word embeddings in our model.

4 Experimental Setup

We begin the section by describing the datasets used in our experiments and the description of various hyper-parameters used in our experiments. The hyper-parameters are chosen by doing a random grid search and evaluation on the development set. In all our experiments the pre-trained word embeddings are not updated.

4.1 Word Embeddings

We have used the pre-trained Glove word embeddings [13]. For our experiments we choose the 50 dimensional word embeddings trained on Wikipedia and Gigaword. The word embeddings are available for around 400K words. We also test using word embeddings from Senna [4] in our experiemnts. Since, senna word embeddings are fine-tuned for the NER task, we believe that the results would be better than glove word embeddings. Interesting observation would be performance of senna word embeddings when used on out-of-domain tasks. For Spanish NER, we use the publicly available Spanish word embeddings[1] [2]. The word embeddings are of 300 dimensional and trained using word2vec [10, 11] tool.

4.2 Corpus

We evaluate our system on the standard CoNLL 2003 English NER Shared Task data [20]. The data is distributed in the IOB format with B-XXX tag used when two mentions of the same entity appear next to each other. The data identifies 4 types of tags, Person, Location, Organization and Miscellaneous. This data is converted to standard IOB format where every entity begins with the tag B-XXX followed by I-XXX if it is a multiword named entity.

We test our trained model on out-of-domain datasets. MUC-7 dataset is the common choice for reporting out-of-domain performance. It is annotated for named entities like Person, Location and temporal entities like Date, Time and Number expressions like monetary units. We choose MUC7 Formal run for comparison with existing systems. Since the dataset has no Miscellaneous tags as in CoNLL 2003 data, we follow the same procedure as [21] where we keep only *Person, location, Organization* tags and all other tags as non-named entities (*O* tag). Similarly after prediction is done, we replace all predicted *Misc* tags by *O*.

We also test our Deep Learning model on Wiki50 Dataset [22]. The Wiki50 Dataset contains annotations for both Named Entities and Multiwords. The dataset was created from fifty wikipedia articles from different domains. We convert all non-named entity tags to *Others*. The Named Entity tags are converted into IOB Format. We consider the entire Wiki50 dataset for testing.

We compare the performance of our model with various existing systems. We compare the results of our model on out-of-domain datasets and the reported performance by other systems. For Wiki50 Dataset and MUC7 formal run, we run Senna [4] without tokenization and report the results.

For Spanish NER, we use the CoNLL 2002 Spanish corpus which was used for CoNLL 2002 Shared Task [19]. The data is similar to the English CoNLL 2003 data. The official splits are used as training, development and test files.

4.3 Parameter Setup

The number of hidden layer neurons were set to 200 in all our models. For Convolution layer at the character layer we considered ngrams of 1,2,3 and 4. Each convolutional layer extracted 15 sets of features from the character layer. The features extracted from convolution layers which run over ngram characters of 1,2,3 and 4 are concatenated and augmented to word embeddings. If any

[1] http://crscardellino.me/SBWCE/

of the word had less than 4 characters we pad it with a special symbol. Pre-trained word embeddings are not updated in our model. For word with no word embeddings we kept it's corresponding word embedding as all zeroes. We used Adagrad [6] in all our experiments for optimization. After every epoch the cost on development set was monitored and when the cost increased, the learning rate was halved. We used Backpropagation Through Time algorithm without truncation for training. We observed that batch size of 1 gave the best results. We used dropout layer after the bidirectional LSTM layer in all our models.

For getting sparse word representation, the Glove word embeddings are sent through an autoencoder layer. We keep the number of hidden layer neurons to 500. The hyper-parameter space was between 100 to 600 dimensions. Our goal was not to learn an interpretable sparse representation so we did not try using large number of hidden neurons. The training was stopped for 4 epochs to prevent the model from learning an identity function. This learned representation gave the best performance on the CoNLL English Shared task development set. The obtained representation was also evaluated on syntactic and semantic analogy tasks [11] and the results were lower than reported by [13].

For Spanish NER, we set the number of hidden layer neurons is kept to 150. There are 4 parallel convolutional layers running over character ngrams of 1,2,3 and 4 and extracting 20 features each.

5 Results

In this section we analyze the obtained results. The table 1 discusses the results obtained from Knowledge Lean Deep Learning models along with baseline systems and Deep Learning based state-of-the-art systems.

Table 1: CoNLL English NER Shared Task Test Results

System	**F1 (%)**
Senna [4] (no Gazetteers)	88.67
Huang [9] (no Gazetteers)	88.83
Passos [12]	**90.90**
Bi-LSTM	81.10
Bi-LSTM + Transition	83.30
Bi-LSTM + Character Features	85.87
Feature Learned NER (CNN Single Width)	88.19
Feature Learned NER	88.90
Feature Learned NER (Senna)	89.20
Feature Learned NER + PreTrained	89.27

The results obtained from the Vanilla Bi-LSTM with only pre-trained word embeddings are much lower compared to existing systems. The major source for these errors were the confusion with non-named entities and named entities. Another major source was missing word embeddings for many words forcing the network to predict correct tag only looking at the context. Adding transition features (previous tag information) into the system improves the results but the improvements in the results were not satisfactory.

Adding a Convolutional layer over the character sequence and using features obtained after max pooling along with word embeddings boosted the results for the Vanilla Bi-LSTM. The convolutional layer looked at ngrams of 3 characters.

Combining previous tag information with character level features improved the results further and brought into the regime of existing systems. Now instead of running a single convolutional layer over 3 characters having parallel convolutional layers each running over n characters (with n =1,2,3,4) gave a F-Score of 88.9. Using Senna embeddings as input we see an increase in F-Score to 89.2. The results indicate that our system is able to give comparable results to feature engineered systems.

Learning a sparse representation for words over using pre-trained word embeddings gave the best performance of our system. The results obtained are comparable with state-of-the-art systems. When we consider systems which does not use any handcrafted features or knowledge resources, our reported results are the best with a F-Score of 89.27. The results indicate that a complete deep learning approach is a viable solution for resource scarce languages.

The following table reports the results for Spanish NER. We report the results only for Feature Learned NER system which uses multiple character features along with transition features from *teacher training.* The results indicate that our feature learned deep learning model is able to achieve state-of-the-art performance for Spanish NER.

Table 2: Results on Spanish NER: CoNLL 2002 Test Data

System	**F1 (%)**
CharWNN[5]	82.21
Feature Learned NER	**82.59**

Out-of-domain Results

Here we report the results of our model on two out-of-domain datasets. The results are provided in tables 3 and 4. Here we compare only those systems which were trained on CoNLL Shared Task dataset and the trained model was tested on MUC7 datasets. For Wiki50 dataset, we report the results only for Senna.

On MUC7 and Wiki50 datasets, the results are lower compared to existing systems. On analysis on Wiki50 data, we find that a significant number of named

Table 3: Wiki50: Out-of-Domain Results

System	**F1 (%)**
Senna (with Gazetteers)	54.26
Feature Learned NER	54.53
Feature Learned NER + PreTrained	52.34
Feature Learned NER (Senna)	**55.51**

entities do not have any corresponding word vectors. The statistics of unknown words present in different datasets for *Feature Learned NER* model is given in table 5. This makes our system to rely heavily on learned character features. As our model tries to learn the character level features in a supervised way it might not generalize well.

Table 4: MUC7 Formal Run: Out-of-Domain Results

System	**F1 (%)**
CRF + Glove [13]	82.2
Senna (with Gazetteers)	79.66
CRF [21]	**82.71**
Feature Learned NER	80.53
Feature Learned NER + PreTrained	79.60
Feature Learned NER (Senna)	79.49

Sparse word representations lost some of the information present in the original word embedding. This was also evident when these representations were tested for syntactic and similarity analogy tasks. This could be the reason for the poor performance on out-of-domain task. Our model outperforms Senna system on both MUC7 dataset and Wiki50 dataset.

Table 5: Statistics on Known and Unknown words in Test Set

Dataset	CoNLL	MUC7	Wiki50
Present + Tagged Correct	37909	50145	70096
Present + Incorrect Tagged	5430	4909	9345
Absent + Correct	3051	4227	16089
Absent + Incorrect	276	145	4838

6 Conclusion

In this paper, we showed that a feature learned deep learning system is a viable approach for NER. Our experiments show that a feature learned deep learning system gives comparable results with the existing state-of-the-art systems for English NER. When systems which do not perform feature engineering are considered, we achieve the best F-Score on CoNLL English NER task. The performance of our system on out-of-domain task is also encouraging. Best F-Score is observed for Spanish NER using our feature learned deep learning approach. This is an encouraging result for the applicability of a feature learned deep learning based NER system for resource scarce languages. We believe that learning both character-level features and gazetteer features in an unsupervised way is the way to go for improving the performance of this NER system on both in-domain and out-of-domain tasks. We would like to study the performance of our NER system on resource scarce and morphologically rich languages which presents a different challenge.

References

1. Rie Kubota Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *J. Mach. Learn. Res.*, 6:1817–1853, December 2005.
2. Cristian Cardellino. Spanish Billion Words Corpus and Embeddings, March 2016.
3. Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*, pages 1724–1734, 2014.
4. Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November 2011.
5. Cicero dos Santos, Victor Guimaraes, RJ Niterói, and Rio de Janeiro. Boosting named entity recognition with neural character embeddings. *Proceedings of NEWS 2015 The Fifth Named Entities Workshop*, page 9, 2015.
6. John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July 2011.
7. Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah Smith. Sparse overcomplete word vector representations. In *ACL 2015*, 2015.
8. Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. Named entity recognition through classifier combination. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 168–171. Association for Computational Linguistics, 2003.
9. Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991, 2015.
10. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

11. Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751, 2013.
12. Alexandre Passos, Vineet Kumar, and Andrew McCallum. Lexicon infused phrase embeddings for named entity resolution. *CoRR*, abs/1404.5367, 2014.
13. Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1532–1543, 2014.
14. Cicero D. Santos and Bianca Zadrozny. Learning character-level representations for part-of-speech tagging. In Tony Jebara and Eric P. Xing, editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1818–1826. JMLR Workshop and Conference Proceedings, 2014.
15. M. Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45(11):2673–2681, Nov 1997.
16. Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809, 2011.
17. Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics, 2012.
18. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014.
19. Erik F. Tjong Kim Sang. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*, COLING-02, pages 1–4. Association for Computational Linguistics, 2002.
20. Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 142–147. Association for Computational Linguistics, 2003.
21. Joseph Turian, Lev Ratinov, and Yoshua Bengio. 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. Association for Computational Linguistics, 2010.
22. Veronika Vincze, István Nagy T, and Gábor Berend. Multiword expressions and named entities in the wiki50 corpus, 2011.