Sequential Labeling with Bidirectional LSTM-CNNs

Jason P.C. Chiu
University of British Columbia
jsonchiu@gmail.com

Eric Nichols
Honda Research Institute Japan Co. Ltd.
e.nichols@jp.honda-ri.com

1 Introduction

Part-of-speech tagging, text chunking, and named entity recognition are fundamental tasks in NLP. High performance approaches have been dominated by applying statistical models such as CRF, SVM, or perceptron models to hand-crafted features [24, 37, 12, 27, 23]. However, Collobert et al. [6] proposed an effective neural network model that requires little feature engineering and instead learns important features from word vectors trained on large quantities of unlabeled text – an approach made possible by recent advancements in training algorithms permitting deep architectures [32] and unsupervised learning of word vectors from big data [5, 26].

Unfortunately there are many limitations to the model proposed by Collobert et al. [6]. First, it uses a simple feed-forward neural network (FFNN), which restricts the use of context to a fixed sized window around each word – an approach that discards useful long-distance relations between words. Second, being modeled around word vectors, it is unable to exploit explicit character-level information such as prefix and suffix, which could be useful especially with rare words where word vectors are poorly trained. We seek to address these issues by proposing a more powerful neural network model.

A well-studied solution for a neural network to process variable length input and have long term memory is the recurrent neural network (RNN) [14]. Recently, RNNs have shown great success in diverse NLP tasks such as speech recognition [15], machine translation [4], and language modeling [25]. The long-short term memory (LSTM) unit with the forget gate allows highly non-trivial long-distance dependencies to be easily learned [11]. For sequential labeling tasks, a bidirectional LSTM model can take into account any arbitrary amount of context on both sides of a word and eliminates the problem of limited context that applies to FFNNs [15].

Similarly, convolutional neural networks (CNN) have become popular in image processing for feature extraction. Recently, CNNs have been successfully employed to extract character-level features for POS tagging [19] and NER [7]. Other variants have been applied to tasks requiring tree structures [2]. However, character-level CNNs have not been extensively evaluated for English.

Our main contribution lies in combining these neural network models for POS tagging, chunking, and NER. We present a hybrid model of bidirectional LSTMs and CNNs that learns both character- and word-level features, presenting the first evaluation of such an architecture on well-established English evaluation datasets.

2 Model

Our neural network is inspired by the work of Collobert et al. [6], where feature vectors are computed by lookup tables and concatenated together, and then fed into a multi-layer network. Instead of a FFNN, we use the more powerful bidirectional long-short term memory (BLSTM) network. We also add CNNs to our network to induce character-level features [8, 19].

2.1 Core Features

2.1.1 Word Vectors

Our models use the 50-dimension word vectors and vector lookup table method of Collobert et al. [6][1] All words are lower-cased before passing through the lookup table to convert to their corresponding vectors. The vectors are allowed to be modified during training.

2.1.2 Character Vectors

We randomly initialized a lookup table with values drawn from an uniform distribution with range $[-0.5, 0.5]$ to output a character vector of 25 dimensions. The character set includes all unique characters in the CoNLL-2003 dataset plus the special tokens `PADDING`, which is used to pad the CNN; and `UNKNOWN`, which is used for all other characters (which appear in other datasets). The same set of random vectors were used for all experiments.

2.1.3 CNN-extracted Character Features

Figure 1 shows the CNN that extracts a fixed-length feature vector from the characters of a single word. For each word we employ a convolution and a max layer to extract a new feature vector from character vectors. Words are padded with special `PADDING` characters on both sides to fill the window of the CNN. The resulting vector is concatenated with other word-level feature vectors.

2.2 Special Features

2.2.1 Capitalization Features

As capitalization information is erased during lookup of the word vector, we evaluate Collobert et al.'s method of using a separate lookup table to add a capitalization feature with the following options: `allCaps`, `upperInitial`, `lowercase`, `mixedCaps`, `other` [6].

2.2.2 Suffix Features

Suffix features are widely used in POS tagging systems. We evaluate Collobert et al.'s approach [6], creating a lookup table of vectors for two-character suffixes. We extracted 706 unique suffixes from the CoNLL-2012 data.

2.2.3 POS Tags

As Collobert et al. [6] report that POS tags significantly improves chunking performance, we evaluate their POS tag encoding method with tags from our tagger.

2.2.4 Lexicons

Almost all state of the art NER systems make use of lexicons [31, 27, 9, 23]. We evaluate the lexicon encoding approach of Collobert et al. [6].

[1]Part of SENNA: http://ml.nec-labs.com/senna/.

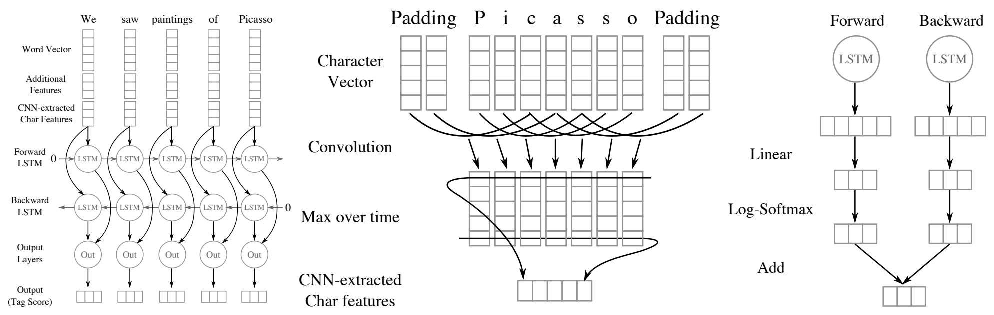

Figure 1: The BLSTM-CNN architecture. Multiple lookup tables compute the word vector and the word-level features. The CNN extracts character features from the character vectors corresponding to each word, which are concatenated and fed to the BLSTM network. The output layers decode BLSTM output into a tag category score.

2.3 Sequence Labeling with BLSTMs

Following Graves et al. [15], we employed a stacked BLSTM to transform the extracted word features into a distribution of tag scores. Figure 1 describes the network. The extracted features of each word is fed into forward and backward LSTM networks, each with multiple layers. The output of each LSTM network at each time step is fed through a linear layer and a log-softmax layer to decode into log-probabilities for each tag category, which are added together to produce the final output.

2.4 Training and Inference

We implement the neural network using the torch7 library.[2] Training and inference are done on a per-sentence level. The initial states of the LSTM are zero vectors. Except for the character and word vectors, all lookup tables are randomly initialized from the standard normal distribution. For chunking and NER, we use the BIOES tag scheme, as it was reported to perform best [31].

2.4.1 Objective Function and Inference

We train our network following Collobert et al. [6], using the Viterbi algorithm to find the maximum sentence-level log likelihood tag sequence at inference time.

2.4.2 Learning Algorithm

Training is done by mini-batch SGD with a fixed learning rate. Each mini-batch consists of multiple sentences with the same number of tokens. We apply dropout to the input and output nodes of each LSTM layer to reducing overfitting [28]. We carried out hyper-parameter selection for POS tagging and chunking with Optunity's implementation of the particle swarm algorithm,[3] however, due to time constraints NER hyper-parameter selection uses random search. Table 2 shows the final values.

3 Evaluation

Table 1 gives an overview of the datasets used. For POS tagging, we evaluated on OntoNotes 5.0 [16]. For chunking, we evaluated on the CoNLL-2000 dataset [39]. For NER, we evaluated on both the CoNLL-2003 NER shared task dataset [40] and OntoNotes 5.0 [29]. We ran each experiment multiple times and report the average and standard deviation of at least 6 successful trials.

3.1 Dataset Preprocessing

For all datasets, we do the following preprocessing:

- All digit sequences are replaced by a single "0."
- Before training, we group sentences by word length into mini-batches and shuffle them.

[2]http://torch.ch
[3]http://optunity.net

In addition, in order to handle numeric NEs in the OntoNotes NER dataset, we split tokens before and after every digit when performing NER experiments.

3.2 OntoNotes 5.0 Dataset: POS tagging

For POS tagging we applied our model to the CoNLL-2012 dataset [30] from the OntoNotes corpus. Manning [24] argued that the OntoNotes is ideal for evaluation because it corrects many tag errors in the PTB and contains text from many different domains, which provides a fairer picture of performance in the general use case. However, this means our results are not directly comparable to other published results, so in order to compare, we downloaded the publicly available systems and re-trained the models on the CoNLL-2012 training set whenever possible. Since some models could not be retrained, we also evaluate the provided models on CoNLL-2012 data.

3.3 CoNLL-2000 Dataset: Chunking

We applied our system to the chunking task on the CoNLL-2000 dataset [39]. We performed hyper parameter search with a held-out dev set that consists of around 2,000 sentences randomly selected from the training set. Then we trained the model on the entire training set using the best hyper-parameters. We experimented with tags from a POS model trained on the non-overlapping portion of the CoNLL-2012 dataset.

3.4 CoNLL-2003 Dataset: NER

We tuned the hyper-parameters (Table 2) on the dev set by random search and then trained our models on both the training and dev sets.

3.5 OntoNotes 5.0 Dataset: NER

Following Durrett and Klein [9], we used the part of the CoNLL-2012 shared task data [30] with gold-standard NE annotations. Due to time constraints we reused the CoNLL-2003 hyper-parameters and tuned the number of epochs and learning rate based on dev set performance.

4 Results and Discussion

4.1 Part-of-speech Tagging

Table 3 shows our results on POS tagging compared to other published systems. Because OntoNotes has a much wider domain than the Penn Treebank and a slightly different tag set, the publicly released models all suffer a large performance penalty when evaluated on the OntoNotes dataset. However, after retraining, most perform similar to the PTB. As our system performs significantly[4] better than the retrained models from strong systems, it is likely competitive with the state-of-the-art.

[4]Two-sample t-test, $p < 0.0001$ assuming similar variance.

Dataset	Train	Dev	Test
OntoNotes (POS)	1,299,312	163,104	169,579
CoNLL-2000	211,727	N/A	47,377
CoNLL-2003	204,567 (23,499)	51,578 (5,942)	46,666 (5,648)
OntoNotes (NER)	1,088,503 (81,828)	147,724 (11,066)	152,728 (11,257)

Table 1: Dataset size in # of tokens (# of named entities)

4.2 Chunking

Table 3 shows our results on chunking compared to other published systems on the CoNLL-2000 dataset. With just word vectors, our system readily improves over most other published systems. With POS tags, however, our system significantly outperforms all other systems[5] except for Shen and Sarkar [34], which uses a voting ensemble of models trained on different tag schemes.

4.3 Named Entity Resolution

Table 3 shows the results for all datasets. Our best model is competitive with other state of the art systems on the CoNLL-2003 dataset, and for the OntoNotes dataset, to the best of our knowledge we have surpassed the previous highest reported results on all of precision, recall, and F1-score. In particular, word vectors and character-level features (produced by the CNN) contributed most of the performance, suggesting that when given enough data, the neural network is able to learn the relevant features for NER without feature engineering. Detailed NER evaluation can be found in [3].

4.4 Special Features

4.4.1 CNNs vs. Capitalization Features

Table 3 show that BLSTM-CNN models significantly[6] outperform the BLSTM models when only word vectors are used. Moreover, for POS-tagging and NER on OntoNotes, the BLSTM models using all hand-crafted features cannot surpass the BLSTM-CNN models using only word vectors. Furthermore, while capitalization improves BLSTM-CNN a small amount for every task except POS tagging, the effect disappears when special features are added. These results suggest character-level CNNs can replace explicit character-level features.

4.4.2 POS tagging: Suffix Feature

Table 3 shows that the suffix feature slightly degrades performance and capitalization does not influence performance, suggesting that the CNN has learned more generalized character-level features.

4.4.3 Chunking: POS Tag Feature

While our BLSTM-CNN model, with only word vectors, improves over all previous neural network models, adding a POS tag feature with our tagger significantly[7] improves results and allows our system to rival the best published result by Shen and Sarkar [34]. We suspect that this is because our tagger is more accurate.

4.4.4 NER: Lexicon Features

Table 3 shows that on both the CoNLL-2003 and OntoNotes 5.0 datasets, adding Collobert et al. [6] lexicon features to our model provides a large gain,[8] showing that lexical knowledge is still crucial to performance.

Hyper-parameters	OntoNotes (POS)	CoNLL-2000 (Chunk)	CoNLL-2003 (NER)	OntoNotes (NER)
CNN width	5	5	3	3
CNN output	49	66	20	20
LSTM states	275	350	200	200
LSTM layers	2	2	2	2
Learning rate	0.0121	0.0145	0.0166	0.008
Epochs	20	20	100	18
Dropout	0.56	0.50	0.63	0.63
Mini-batch	9	9	9	9

Table 2: Final hyper-parameter values for all experiments

5 Related Research

5.1 Part-of-speech Tagging

POS taggers are typically classifiers trained on various windowed, hand-crafted features, with bidirectional decoding, and evaluated on the PTB. These include maxent with bidirectional dependency networks [41, 24], SVMs [13], and other learning algorithms [35, 37, 42, 38], who all use similar features. Søgaard [36] used semi-supervised condensed nearest neighbor with SVMTools [13] and UnSuPOS [1] output to achieve the non-NN state-of-the-art.

Competitive neural network approaches started with the FFNN of Collobert et al. [6]. Dos Santos and Zadrozny [8] added character-level CNNs to Collobert's network. POS tagging has also been attempted with bidirectional RNN-CNNs in German [19], and for English, BLSTMs [43] and bidirectional LSTM-CRFs [17]. Our model combines these ideas into a BLSTM-CNN network which achieves state of the art performance on OntoNotes with minimal feature engineering. The best reported result on PTB is Ling et al. [22], which is similar but uses BLSTMs with compositional character vectors.

5.2 Chunking

Common approaches to chunking include SVM [18, 44, 20], Winnow [45], CRFs [33], and guided learning [12]. Shen and Sarkar [34] achieve the current state of the art with ensemble voting over multiple chunk annotations.

Neural network approaches are similar to POS tagging, and include FFNNs [6], BLSTMs [43], and BLSTM-CRFs [17]. We differ by using character-level CNNs.

5.3 Named Entity Recognition

Most recent approaches to NER has been characterized by feature engineering-dependent models, such as CRFs. Ratinov and Roth [31] used non-local features, gazetteer extracted from Wikipedia, and Brown-cluster-like word representations. In leu of a lexicon, Lin and Wu [21] used phrase features obtained from k-means clustering of search engine query logs, and Passos et al. [27] trained phrase vectors. Finkel and Manning [10] did joint parsing and NER. Training an NER system together with entity linking has recently been shown to improve the state of the art. Durrett and Klein [9] achieved state of the art results on OntoNotes by combining coreference resolution, entity linking, and NER into a single CRF model with cross-task interaction factors. Luo et al. [23] achieved the current CoNLL-2003 state of the art by training a joint NER/entity linking model.

Collobert et al. [6] employed a deep FFNN and word vectors to achieve near state of the art results on POS tagging, chunking, NER, and SRL. We build on their approach, sharing their word vectors, feature encoding method, and objective functions. Recently, CharWNN [7] augmented the FFNN of Collobert et al. [6] with character-level CNNs to improve performance on Spanish

[5]Two-sample t-test, $p < 0.0001$ assuming similar variance.

[6]Wilcoxon rank sum test, $p < 0.0005$.

[7]Wilcoxon rank sum test, $p < 0.0001$.

[8]CoNLL-2003: Wilcoxon rank sum test, $p < 0.001$.

[9]Evaluation on OntoNotes 5.0 done by Pradhan et al. [29]

[10]CoNLL-2003 results from [23]. OntoNotes results from [9].

[11]We attempted to retrain using the released Python script, but we are unsure of the cause of the disappointing result.

[12]It was unclear whether or not they evaluated their system on the CoNLL-2012 split of the OntoNotes dataset.

[13]Numbers taken from [23]. It is unclear which figure is correct.

Model	POS Tagging			Chunking	NER					
	PTB	OntoNotes / CoNLL-2012		CoNLL-2000	CoNLL-2003			OntoNotes 5.0		
	-	**Existing**	**Retrained**	**F1**	**Prec.**	**Recall**	**F1**	**Prec.**	**Recall**	**F1**
Sha and Pereira [33]	-	-	-	94.38	-	-	-	-	-	-
Gimenez and Marquez [13]	97.16	93.03	97.02	-	-	-	-	-	-	-
Shen and Sarkar [34]	-	-	-	**95.23**	-	-	-	-	-	-
Shen et al. [35]	97.33	**93.21**	-	-	-	-	-	-	-	-
Finkel and Manning [10][9]	-	-	-	-	-	-	-	84.04	80.86	82.42
Lin and Wu [21]	-	-	-	-	-	-	90.90	-	-	-
Ratinov and Roth [31][10]	-	-	-	-	91.20	90.50	90.80	82.00	84.95	83.45
Spoustová et al. [37]	97.44	92.90	-	-	-	-	-	-	-	-
Gesmundo [12]	-	-	-	94.56	-	-	-	-	-	-
Manning [24]	97.32	92.22	97.31	-	-	-	-	-	-	-
Søgaard [36]	97.50	-	91.41[11]	-	-	-	-	-	-	-
Tsuruoka et al. [42]	97.28	93.16	97.39	-	-	-	-	-	-	-
Passos et al. [27][12]	-	-	-	-	-	-	90.90	-	-	82.30
Durrett and Klein [9]	-	-	-	-	-	-	-	85.22	82.89	84.04
Sun [38]	97.36	-	-	-	-	-	-	-	-	-
Luo et al. [23][13]	-	-	-	-	**91.50**	91.40	**91.20**	-	-	-
Collobert et al. [6]	97.20	-	-	93.63	-	-	88.67	-	-	-
Collobert et al. [6] + special	97.29	92.96	-	94.32	-	-	89.59	-	-	-
Ling et al. [22]	**97.78**	-	-	-	-	-	-	-	-	-
Huang et al. [17]	97.55	-	-	94.46	-	-	90.10	-	-	-
Wang et al. [43]	97.26	-	-	94.59	-	-	89.64	-	-	-
BLSTM	-	-	95.43 (± 0.05)	90.33 (± 0.19)	80.38	73.95	77.03 (± 0.53)	79.68	75.97	77.77 (± 0.37)
BLSTM + vec	-	-	96.55 (± 0.04)	94.28 (± 0.09)	88.31	87.11	87.71 (± 0.30)	82.85	82.59	82.72 (± 0.23)
BLSTM + vec + caps + special	-	-	97.58 (± 0.02)	95.11 (± 0.11)	90.33	91.22	90.77 (± 0.61)	85.89	86.35	86.12 (± 0.26)
BLSTM-CNN	-	-	97.35 (± 0.03)	93.05 (± 0.20)	82.95	83.75	83.35 (± 0.35)	82.58	82.49	82.53 (± 0.40)
BLSTM-CNN + vec	-	-	**97.64** (± 0.03)	94.58 (± 0.10)	90.22	90.93	90.57 (± 0.45)	86.05	86.37	86.21 (± 0.20)
BLSTM-CNN + vec + caps	-	-	**97.64** (± 0.03)	94.65 (± 0.12)	90.28	91.10	90.69 (± 0.29)	86.16	86.54	86.35 (± 0.28)
BLSTM-CNN + vec + special	-	-	97.59 (± 0.02)	95.15 (± 0.08)	90.74	**91.53**	91.13 (± 0.15)	**86.16**	**86.65**	**86.40** (± 0.21)

Table 3: Results for POS tagging, chunking, and NER with various feature sets. The three sections are, in order, non-neural network models, neural network models, and our models. "lex" = Collobert's lexicon, "caps" = capitalization features, "special" = task-specific special features (Section 2.2). Standard deviations are in parentheses.

and Portuguese NER. We have successfully incorporated similar character-level CNNs into our model.

Our approach is most similar to the recent Bi-LSTM-CRF model for POS tagging, chunking, and NER presented by Huang et al. [17], the BLSTM RNN model proposed by Wang et al. [43], and the Bi-RNN model for POS tagging presented by Labeau et al. [19]. Our approach differs from Huang et al. [17] in that we use character-level CNNs instead of hand-crafting word features. Unlike Labeau et al. [19], we employed LSTMs.

6 Conclusion

We have shown that our neural network model, which incorporates a bidirectional LSTM and a character-level CNN, achieves results competitive with the state-of-the-art in POS tagging, chunking, and NER, significantly improving over previous neural networks. Our model reports a new state-of-the-art on OntoNotes for POS tagging and NER without any hand-crafted features or external knowledge. Evaluation showed that our character-level CNNs learned more effective character-level features than hand-crafted features like capitalization and suffix, further reducing the burden of feature engineering. We also found a sweet spot in knowledge engineering: by adding POS tags from our tagger to chunking and publicly-available lexicons to NER, our system is competitive with the CoNLL-2000 and 2003 states-of-the-art. We are also currently working on a Japanese system.

References

[1] Chris Biemann. Unsupervised part-of-speech tagging employing efficient graph clustering. In Proc. of ACL 2006, pages 7–12.

[2] Phil Blunsom, Edward Grefenstette, Nal Kalchbrenner, et al. A convolutional neural network for modelling sentences. In Proc. of ACL 2014.

[3] Jason P. C. Chiu and Eric Nichols. Named entity recognition with bidirectional lstm-cnns. CoRR, abs/1511.08308, 2015.

[4] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In Proc. of SSST-8, pages 103–111, 2014.

[5] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proc. of ICML 2008, pages 160–167.

[6] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. JMLR, 12:2493–2537, 2011.

[7] Cıcero dos Santos, Victor Guimaraes, RJ Niterói, and Rio de Janeiro. Boosting named entity recognition with neural character embeddings. In NEWS 2015.

[8] Cıcero dos Santos and Bianca Zadrozny. Learning character-level representations for part-of-speech tagging. In Proc. of ICML-14, 2014.

[9] Greg Durrett and Dan Klein. A joint model for entity analysis: Coreference, typing, and linking. TACL, 2:477–490, 2014.

[10] Jenny Rose Finkel and Christopher D Manning. Joint parsing and named entity recognition. In Proc. of NAACL-HLT 2009, pages 326–334, 2009.

[11] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with LSTM. Neural Computation, 12(10), 2000.

[12] Andrea Gesmundo. Bidirectional sequence classification for tagging tasks with guided learning. Proc. of TALN 2011.

[13] Jesús Giménez and Lluis Marquez. Svmtool: A general pos tagger generator based on support vector machines. In Proc. of ICLRE 2004.

[14] Christoph Goller and Andreas Kuchler. Learning task-dependent distributed representations by backpropagation through structure. In Proc. of Neural Networks 1996, volume 1, pages 347–352. IEEE.

[15] Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In Proc. of ICASSP 2013, pages 6645–6649. IEEE, 2013.

[16] Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. OntoNotes: the 90% solution. In Proc. of NAACL-HLT 2006.

[17] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF models for sequence tagging. CoRR, abs/1508.01991, 2015.

[18] Taku Kudo and Yuji Matsumoto. Chunking with support vector machines. In Proc. of NAACL-HLT 2001.

[19] Matthieu Labeau, Kevin Löser, and Alexandre Allauzen. Non-lexical neural architecture for fine-grained pos tagging. In Proc. of EMNLP 2015.

[20] Yue-Shi Lee and Yu-Chieh Wu. A robust multilingual portable phrase chunking system. Expert Systems with Applications, 33(3):590–599, 2007.

[21] Dekang Lin and Xiaoyun Wu. Phrase clustering for discriminative learning. In Proc. of ACL-IJCNLP 2009, pages 1030–1038.

[22] Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. Finding function in form: Compositional character models for open vocabulary word representation. In Proc. of EMNLP 2015, pages 1520–1530.

[23] Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. Joint entity recognition and disambiguation. In Proc. of EMNLP 2015, pages 879–888.

[24] Christopher D Manning. Part-of-speech tagging from 97% to 100%: is it time for some linguistics? In Computational Linguistics and Intelligent Text Processing, pages 171–189. Springer, 2011.

[25] Tomas Mikolov, Stefan Kombrink, Anoop Deoras, Lukar Burget, and Jan Cernocky. Rnnlm-recurrent neural network language modeling toolkit. In Proc. of ASRU 2011, pages 196–201.

[26] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In Proc. of NIPS 2013, pages 3111–3119.

[27] Alexandre Passos, Vineet Kumar, and Andrew McCallum. Lexicon infused phrase embeddings for named entity resolution. Proc. of CoNLL-2014, page 78.

[28] Vu Pham, Théodore Bluche, Christopher Kermorvant, and Jérôme Louradour. Dropout improves recurrent neural networks for handwriting recognition. In Proc. of ICFHR 2014, pages 285–290. IEEE, 2014.

[29] Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. Towards robust linguistic analysis using ontonotes. In Proc. of CoNLL 2013, pages 143–52.

[30] Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In Proc. of CoNLL-Shared Task, pages 1–40, 2012.

[31] Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In Proc. of CoNLL 2009, pages 147–155, 2009.

[32] David Rumelhart, Geoffrey Hinton, and Ronald Williams. Learning representations by back-propagating errors. Nature, pages 323–533, 1986.

[33] Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In Proc. of NAACL-HLT 2003, pages 134–141.

[34] Hong Shen and Anoop Sarkar. Voting between multiple data representations for text chunking. Springer, 2005.

[35] Libin Shen, Giorgio Satta, and Aravind Joshi. Guided learning for bidirectional sequence classification. In Proc. of ACL 2007, pages 760–767.

[36] Anders Søgaard. Semisupervised condensed nearest neighbor for part-of-speech tagging. In Proc. of ACL-HLT 2011, pages 48–52.

[37] Drahomíra "johanka" Spoustová, Jan Hajič, Jan Raab, and Miroslav Spousta. Semi-supervised training for the averaged perceptron pos tagger. In Proc. of EACL 2009, pages 763–771.

[38] Xu Sun. Structure regularization for structured prediction. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, NIPS 2014, pages 2402–2410.

[39] Erik F Tjong Kim Sang and Sabine Buchholz. Introduction to the conll-2000 shared task: Chunking. In Proc. of the 2nd workshop on learning language in logic, pages 127–132, 2000.

[40] Erik F Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Proc. of CoNLL 2003, pages 142–147.

[41] Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In Proc. of NAACL-HLT 2003, pages 173–180.

[42] Yoshimasa Tsuruoka, Yusuke Miyao, and Jun'ichi Kazama. Learning with lookahead: can history-based models rival globally optimized models? In Proc. of CoNLL 2011, pages 238–246.

[43] Peilu Wang, Yao Qian, Frank K Soong, Lei He, and Hai Zhao. Learning distributed word representations for bidirectional lstm recurrent neural network. In Proc. of ICASSP 2016.

[44] Yu-Chieh Wu and Chia-Hui Chang. Efficient text chunking using linear kernel with masked method. Knowledge-Based Systems, 20(3):209–219, 2007.

[45] Tong Zhang, Fred Damerau, and David Johnson. Text chunking based on a generalization of winnow. JMLR, 2:615–637, 2002.