Toward Mention Detection Robustness with Recurrent Neural Networks

Thien Huu Nguyen[†*], **Avirup Sil**[§], **Georgiana Dinu**[§] **and Radu Florian**[§]
[†] Computer Science Department, New York University, New York, USA
[§] IBM T.J. Watson Research Center, Yorktown Heights, New York, USA
thien@cs.nyu.edu,{avi,gdinu,raduf}.us.ibm.com

Abstract

One of the key challenges in natural language processing (NLP) is to obtain good performance across application domains and languages. In this work, we investigate the robustness of the mention detection systems, one of the fundamental tasks in information extraction, via recurrent neural networks (RNNs). The advantage of RNNs over the traditional approaches is their capacity to capture long range contexts and implicitly adapt the word embeddings, trained on a large corpus, into a task-specific word representation, but still preserve the original semantic generalization to be helpful across domains. Our systematic evaluation for RNN architectures demonstrates that RNNs outperform a very strong baseline for mention detection in the cross-domain setting for English and are significantly better than the traditional methods on the similar task of named entity recognition for Dutch (up to 22% relative error reduction).

1 Introduction

One of the crucial steps toward understanding natural languages is mention detection (MD), whose goal is to identify entity mentions, whether named, nominal (*the president*) or pronominal (*he, she*), and classify them into some predefined types of interest in text such as PERSON, ORGANIZATION or LOCATION. This is an extension of the named entity recognition (NER) task which only aims to extract entity names. MD is necessary for many higher-level applications such as relation extraction, knowledge population, information retrieval, question answering and so on.

Traditionally, both MD and NER are formalized as sequential labeling problems, thereby being solved by some linear graphical models such as Hidden Markov Models (HMMs), Maximum Entropy Markov Models (MEMMs) or Conditional Random Fields (CRFs) [Lafferty *et al.*, 2001]. Although these graphical models have achieved the top performance for MD, there are still at least three problems we want to focus on this work:

(i) The first problem is the performance loss of the mention detectors when they are trained on some domain (the source domain) and applied to other domains (the target domains). The problem might originate from various mismatches between the source and the target domains (domain shifts) such as the vocabulary difference, the distribution mismatches etc [Daume, 2007; Plank and Moschitti, 2013].

(ii) Second, in mention detection, we might need to capture a long context, possibly covering the whole sentence, to correctly predict the type for a word. For instance, consider the following sentence with the pronominal "*they*":

Now, the reason that France, Russia and Germany are against war is because they have suffered much from the past war.

In this sentence, the correct type GPE[1] for "*they*" can only be inferred from its GPE references: "*France*", "*Russia*" and "*Germany*" which are far from the pronominal "*they*" of interest. The challenge is to come up with the models that can encode and utilize these long-range dependency contexts effectively.

(iii) The third challenge is to be able to quickly adapt the current techniques for MD so that they can perform well on new languages.

In this paper, we propose to address these problems for MD via recurrent neural networks (RNNs) which offer an effective recurrent mechanism to embed the sentence context into a distributed representation and employ it to decode the sentences. Besides, as RNNs replace the symbolic forms of words in the sentences with their word embeddings, the distributed representation that captures the general syntactic and semantic properties of words [Turian *et al.*, 2010], they can alleviate the lexical sparsity, induce more general feature representation, thus generalizing well across domains [Nguyen and Grishman, 2015b]. This also helps RNNs to quickly and effectively adapt to new languages which just require word embeddings as the only new knowledge we need to obtain. Finally, we can achieve the task-specific word embeddings for MD to improve the overall performance by updating the initial pre-trained word embeddings during the course of training in RNNs.

The recent emerging interest in deep learning has produced many successful applications of RNNs for NLP problems such as machine translation [Cho *et al.*, 2014a; Bahdanau *et al.*, 2015], semantic role labeling [Zhou and Xu, 2015] etc. However, to the best of our knowledge, there has been no

[*] Work carried out during an internship at IBM

[1] Geographical Political Entity

previous work employing RNNs for MD on the cross-domain and language settings so far. To summarize, the main contributions of this paper are as follows:

1. We perform a systematic investigation on various RNN architectures and word embedding techniques that are motivated from linguistic observations for MD.

2. We achieve the state-of-the-art performance for MD in the cross-domain setting with the bidirectional modeling applied to RNNs.

3. We demonstrate the portability of the RNN models for MD to new languages by their significant improvement with large margins over the best reported system for named entity recognition in Dutch.

2 Related Work

Both named entity recognition [Ando and Zhang, 2005; Ratinov and Roth, 2009; Turian *et al.*, 2010; Cherry and Guo, 2015] and mention detection [Florian *et al.*, 2006] have been extensively studied with various evaluations in the last decades: MUC6, MUC7, CoNLL'02, CoNLL'03 and ACE. The previous work on MD has examined the cascade models [Florian *et al.*, 2006], transferred knowledge from rich-resource languages to low-resource ones via machine translation [Zitouni and Florian, 2008] or improved the systems on noisy input [Florian *et al.*, 2010]. Besides, some recent work also tries to solve MD jointly with other tasks such as relation or event extraction to benefit from their inter-dependencies [Li and Ji, 2014a; Li *et al.*, 2014b]. However, none of these work investigates RNNs for MD on the cross-domain and language settings as we do in this paper.

Regarding neural networks, a large volume of work has been devoted to the application of deep learning to NLP in the last few years, centering around several network architecture such as convolutional neural networks (CNNs) [Kalchbrenner *et al.*, 2014], recurrent/recursive neural networks [Socher *et al.*, 2012; Bahdanau *et al.*, 2015], to name a few. For NER, Collobert et al. [2011] propose a CNN-based framework while Mesnil et al. [2013] and Yao et al. [2014] investigate the RNNs for the slot filling problem in spoken language understanding. Although our work also examines the RNNs, we consider the mention detection problem with an emphasis on the robustness of the models in the domain shifts and language changes which has never been explored in the literature before.

Finally, for the robustness in the domain adaptation setting, the early work has focused on the sequential labeling tasks such as part-of-speech tagging or name tagging [Daume, 2007]. Recent work has drawn attention to other information extraction tasks such as relation extraction [Plank and Moschitti, 2013; Nguyen *et al.*, 2015a] and event detection [Nguyen and Grishman, 2015b].

3 Models

We formalize the mention detection problem as a sequential labeling task. Given a sentence $X = w_1 w_2 \dots w_n$, where w_i is the i-th word and n is the length of the sentence, we want to predict the label sequence $Y = y_1 y_2 \dots y_n$ for X, where y_i is the label for w_i. The labels y_i follow the BIO2 encoding to capture the entity mentions in X. Note that this work focuses on the extraction of the entity mention heads, following Florian et al. [2006] and Li and Ji [2014a].

In order to prepare the sentence for RNNs, we first transform each word w_i into a real-valued vector using the concatenation of two vectors e_i and f_i: $w_i = [e_i, f_i]$[2], where:

- e_i is the word embedding vector of w_i, obtained by training a language model on a large corpus (discussed later).
- f_i is a binary vector encompassing different features for w_i. In this work, we are utilizing four types of features: capitalization, gazetteers, triggers (whether w_i is present in a list of trigger words[3] or not) and cache (the label that is assigned to w_i sometime before in the document).

We then enrich this vector representation by including the word vectors in a context window of v_c for each word in the sentence to capture the short-term dependencies for prediction [Mesnil *et al.*, 2013]. This effectively converts w_i into the context window version of the concatenated vectors: $x_i = [w_{i-v_c}, \dots, w_i, \dots, w_{i+v_c}]$.

Given the new input representation, we describe the RNNs to be investigated in this work below.

3.1 The Basic Models

In standard recurrent neural networks, at each time step (word position in sentence) i, we have three main vectors: the input vector $x_i \in \mathbf{R}^I$, the hidden vector $h_i \in \mathbf{R}^H$ and the output vector $o_i \in \mathbf{R}^O$ (I, H and O are the dimensions of the input vectors, the dimension of the hidden vectors and the number of possible labels for each word respectively). The output vector o_i is the probabilistic distribution over the possible labels for the word x_i and obtained from h_i via the softmax function φ:

$$o_i = \varphi(W h_i), \qquad \varphi(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}}$$

Regarding the hidden vectors or units h_i, there are two major methods to obtain them from the current input and the last hidden and output vectors, leading to two different RNN variants:

- In the Elman model, called **ELMAN**, the hidden vector from the previous step h_{i-1}, along with the input in the current step x_i, constitute the inputs to compute the current hidden state h_i:

$$h_i = \Phi(U x_i + V h_{i-1}) \tag{1}$$

- In the Jordan model, called **JORDAN**, the output vector from the previous step o_{i-1} is fed into the current hidden layer rather than the hidden vector from the previous steps h_{i-1}. The rationale for this topology is to introduce the label from the preceding step as a feature for current prediction:

$$h_i = \Phi(U x_i + V o_{i-1}) \tag{2}$$

[2]For simplicity, we are using the word w_i and its real-valued vector representation interchangeably.

[3]Trigger words are the words that are often followed by entity names in sentences such as "*president*", "*Mr.*" etc.

In the formula above, Φ is the sigmoid activation function: $\Phi(z) = \frac{1}{1+e^{-z}}$ and W, U, and V are the same weight matrices for all time steps, to be learned during training.

3.2 Gated Recurrent Units

The ELMAN and JORDAN models are basically the stacks of the standard feed-forward neural networks. Unfortunately, this stacking mechanism is prone to the "*vanishing gradient*" problem [Bengio *et al.*, 1994], making it challenging to train the networks properly in practice. This problem can be alleviated by long-short term memory units (LSTM) [Hochreiter and Schmidhuber, 1997] that propose the idea of memory cells to allow the information storage and access over a long period of time.

In this work, we use a variant of LSTM, called the *Gated Recurrent Units* (GRUs) by Cho et al. [2014a]. GRU is shown to be simpler than LSTM in terms of computation and implementation but still achieves comparable performance [Józefowicz *et al.*, 2015].

The introduction of GRUs into the models ELMAN and JORDAN amounts to two new models, named **ELMAN_GRU** and **JORDAN_GRU** respectively, with two new methods to compute the hidden vectors h_i. The formula for ELMAN_GRU is adopted directly from Cho et al. [2014b] and given below:

$$\begin{aligned} h_i &= z_i \odot \hat{h}_i + (1 - z_i) \odot h_{i-1} \\ \hat{h}_i &= \Phi(W_h x_i + U_h(r_i \odot h_{i-1})) \\ z_i &= \Phi(W_z x_i + U_z h_{i-1}) \\ r_i &= \Phi(W_r x_i + U_r h_{i-1}) \end{aligned} \tag{3}$$

where $W_h, W_z, W_r \in \mathbf{R}^{H \times I}$, $U_h, U_z, U_r \in \mathbf{R}^{H \times H}$ and $\odot$ is the element-wise multiplication operation.

We cannot directly apply the formula above to the JORDAN_GRU model since the dimensions of the output vectors o_i and the hidden vector h_i are different in general. For JORDAN_GRU, we first need to transform the output vector o_i into the hidden vector space, leading to the following formula:

$$\begin{aligned} h_i &= z_i \odot \hat{o}_i + (1 - z_i) \odot t_{i-1} \\ t_{i-1} &= T o_{i-1} \\ \hat{o}_i &= \Phi(W_o x_i + U_o(r_i \odot t_{i-1})) \\ z_i &= \Phi(W_z x_i + U_z t_{i-1}) \\ r_i &= \Phi(W_r x_i + U_r t_{i-1}) \end{aligned} \tag{4}$$

where $T \in \mathbf{R}^{H \times O}$.

3.3 The Bidirectional Networks

One of the limitations of the four basic models presented above is their incapacity to incorporate the future context information that might be crucial to the prediction in the current step. For instance, consider the first word "*Liverpool*" in the following sentence:

Liverpool *suffered an upset first home league defeat of the season, beaten 1-0 by a Guy Whittingham goal for Sheffield Wednesday.*

In this case, the correct label ORGANIZATION can only be detected if we first go over the whole sentence and then utilize the context words after "*Liverpool*" to decide its label.

The limitation of the four models originates in their mechanism to perform a single pass over the sentences from left to right and make the prediction for a word when they first encounter it. Guided by this intuition, we propose to employ the bidirectional networks to solve the MD problem.

The bidirectional networks involve three passes over the sentence, in which the first two passes are designated to encode the sentence while the third pass is responsible for decoding. The procedure for the sentence $X = x_1 x_2 \dots x_n$ is below:

(i) Run the first RNN R_{ef} from left to right over $x_1 x_2 \dots x_n$ to obtain the first hidden vector or output vector sequence (depending on whether R_{ef} is an Elman or Jordan network respectively): $R_{ef}(x_1 x_2 \dots x_n) = l_1, l_2, \dots, l_n$ (forward encoding).

(ii) Run the second RNN R_{eb} from right to left over $x_1 x_2 \dots x_n$ to obtain the second hidden vector or output vector sequence: $R_{eb}(x_n x_{n-1} \dots x_1) = r_n, r_{n-1}, \dots, r_1$ (backward encoding).

(iii) Obtain the concatenated sequence $\alpha = \alpha_1, \alpha_2, \dots, \alpha_n$ where $\alpha_i = [l_i, r_i]$.

(iv) Decode the sentence with the third RNN R_d (the decoding model) using α as the input vector, i.e, replacing x_i by α_i in the formula (1), (2), (3) and (4).

Conceptually, the encoding RNNs R_{ef} and R_{eb} can be different but in this work, for simplicity and consistency, we assume that we only have a single encoding model, i.e, $R_{ef} = R_{eb} = R_e$. Note that R_e and R_d can be any model in {ELMAN, JORDAN, ELMAN_GRU, JORDAN_GRU}.

The observation is, at the time step i, the forward hidden vector l_i represents the encoding for the past word context (from position 1 to i) while the backward hidden vector r_i is the summary for the future word context (from position n to i). Consequently, the concatenated vector $\alpha_i = [l_i, r_i]$ constitutes a distributed representation that is specific to the word at position i but still encapsulates the context information over the whole sentence at the same time. This effectively provides the networks a much richer representation to decode the sentence.

3.4 Training and Inference

We train the networks locally. In particular, each training example consists of a word x_i and its corresponding label y_i in a sentence $X = x_1 x_2 \dots x_n$ (denoted by $E = (x_i, y_i, X)$). In the encoding phase, we first compute the necessary inputs according to the specific model of interest. This can be the original input vectors $x_1, x_2, \dots, x_n$ in the four basic models or the concatenated vectors $\alpha_1, \alpha_2, \dots, \alpha_n$ in the bidirectional models. Eventually, in the decoding phase, an sequence of v_d input vectors preceding the current position i is fed into the decoding network R_d to obtain the output vector sequence. The last vector in this output sequence corresponds to the probabilistic label distribution for the current position i, to be used to compute the objective function. For example, in

the bidirectional models, the input sequence for the decoding phase is $\alpha_{i-v_d}\alpha_{i-v_d+1}\dots\alpha_i$ while the output sequence is: $R_e(\alpha_{i-v_d}\alpha_{i-v_d+1}\dots\alpha_i) = o_{i-v_d}o_{i-v_d+1}\dots o_i$.

In this work, we employ the stochastic gradient descent algorithm[4] to update the parameters via minimizing the negative log-likelihood objective function: $\mathbf{nll}(E) = -\log(o_i[y_i])$.

Finally, besides the weight matrices in the networks, the word embeddings are also optimized during training to obtain the task-specific word embeddings for MD. The gradients are computed via back-propagation and inference is performed by running the networks over the whole sentences and taking argmax over the output sequence: $y_i = \mathrm{argmax}(o_i)$.

4 Word Representation

Following Collobert et al. [2011], we pre-train word embeddings from a large corpus and employ them to initialize the word representations in the models. One of the state-of-the-art models to train word embeddings have been proposed recently in Mikolov et al. [2013b] that introduce two log-linear models, i.e the continuous bag-of-words model (CBOW) and the continuous skip-gram model (Skip-gram). The CBOW model attempts to predict the current word based on *the average of the context word vectors* while the Skip-gram model aims to predict the surrounding words in a sentence given the current word. In this work, besides the CBOW and skip-gram models, we examine a concatenation-based variant of CBOW (C-CONCAT) to train word embeddings and compare the three models to gain insights into which kind of model is effective to obtain word representations for the MD task. The objective of C-CONCAT is to predict the target word using *the concatenation of the vectors of the words surrounding it*, motivated from our strategy to decide the label for a word based on the concatenated context vectors. Intuitively, the C-CONCAT model would perform better than CBOW as the concatenation mechanism helps to assign different weights to different context words, thereby being more flexible than CBOW that applies a single weight for all the context words.

5 Experiments

5.1 Dataset

In order to investigate the robustness across domains, following the prior work [Plank and Moschitti, 2013; Nguyen *et al.*, 2015a], we utilize the ACE 2005 dataset which contains 6 domains: broadcast news (*bn*), newswire (*nw*), broadcast conversation (*bc*), telephone conversation (*cts*), weblogs (*wl*), usenet (*un*) and 7 entity types: person, organization, GPE, location, facility, weapon, vehicle. The union of *bn* and *nw* is considered as a single domain, called **news**. We take half of *bc* as the only development data and use the remaining data and domains for evaluation. Some statistics about the domains are given in Table 1. As shown in Plank and Moschitti [2013], the vocabulary of the domains is quite different.

Regarding the robustness across languages, we further evaluate the RNN models on the CoNLL 2002 dataset for Dutch Named Entity Recognition[5] [Carreras *et al.*, 2002; Tjong Kim Sang, 2002]. The CoNLL dataset comes along with the training data, validation data and test data, annotated for 4 types of entities: person, organization, location and miscellaneous.

Domain	#Docs	#Sents	#Mentions
news	332	6487	22460
bc	60	3720	9336
cts	39	5900	9924
wl	119	2447	6538
un	49	2746	6507
Total	599	21300	54765

Table 1: ACE 2005 Dataset

5.2 Resources and Parameters

In all the experiments for RNNs below, we employ the context window $v_c = 5$, the decoding window $v_d = 9$. We find that the optimal number of hidden units (or the dimension of the hidden vectors) and the learning rate vary according to the dataset. For the ACE 2005 dataset, we utilize 200 hidden units with learning rate = 0.01 while these numbers are 100 and 0.06 respectively for the Dutch CoNLL dataset. Note that the number of hidden units is kept the same in both the encoding phase and the decoding phase.

For word representation, we train the word embeddings for English from the Gigaword corpus augmented with the newsgroups data from BOLT (Broad Operational Language Technologies) (6 billion tokens) while the entire Dutch Wikipedia pages (310 million tokens) are extracted to train the Dutch word embeddings. We utilize the word2vec toolkit[6] (modified to add the C-CONCAT model) to learn the word representations. Following Baroni et al. [2014], we use the context window of 5, subsampling set to $1e$-05 and negative sampling with the number of instances set to 10. The dimension of the vectors is set to 300 to make it comparable with the word2vec toolkit. Finally, we use the standard IOB2 tagging schema for both ACE 2005 and Dutch CoNLL datasets.

5.3 Model Architecture Evaluation

In this section, we evaluate different RNN models by training the models on the **news** domain and report the performance on the development set. As presented in the previous sections, we have 4 basic models M = {ELMAN, JORDAN, ELMAN_GRU, JORDAN_GRU} and 16 bidirectional models (4 choices for the encoding and decoding models R_e, R_d in M). The performance for the basic models and the bidirectional models are shown in Table 2 and Table 3 respectively[7].

There are several important observations from the three tables:

-Elman vs Jordan: In the encoding phase, the Elman models consistently outperform the Jordan models when the same decoding model is applied in the bidirectional architecture. In the decoding phase, however, it turns out that the Jordan

[4] We try the $AdaDelta$ algorithm and the dropout regularization but do not see much difference.

[5] http://www.cnts.ua.ac.be/conll2002/ner/

[6] https://code.google.com/p/word2vec/

[7] The experiments in this section use C-CONCAT to pre-train word embeddings.

models are better most of the time over different model architectures (basic or bidirectional).

-With vs Without GRUs: It is clear from the tables that GRUs are very helpful in the encoding part of the bidirectional architecture for MD. However, for the decoding part, we can only see the clear benefit of GRUs in the basic models and the bidirectional architecture when R_e is a Jordan model.

-Regarding different model architectures, in general, the bidirectional models are more effective than the basic models, confirming the effectiveness of bidirectional modeling to achieve a richer representation for MD.

The best basic model (F1 = 81.06%) and the best bidirectional model (F1 = 82.37%) are called BASIC and BIDIRECT respectively. In the following, we only focus on these best models in the experiments.

Model(R_d)	F1
ELMAN	80.70
JORDAN	80.46
ELMAN_GRU	80.85
JORDAN_GRU	**81.06**

Table 2: The basic models' performance

R_d \ R_e	ELMAN	ELMAN_GRU
ELMAN	80.99	81.42
JORDAN	81.14	81.68
ELMAN_GRU	80.53	81.16
JORDAN_GRU	80.98	**82.37**
R_d \ R_e	JORDAN	JORDAN_GRU
ELMAN	79.12	79.64
JORDAN	79.21	80.85
ELMAN_GRU	79.80	80.41
JORDAN_GRU	79.76	81.02

Table 3: The bidirectional models' performance

5.4 Word Embedding Evaluation

Word Embeddings	Model	
	BASIC	BIDIRECT
RANDOM	79.30	79.76
FIXED	80.36	81.52
WORD2VEC	80.92	81.41
CBOW	78.61	79.74
SKIP-GRAM	81.45	81.96
C-CONCAT	81.06	**82.37**

Table 4: Word Embedding Comparison

The section investigates the effectiveness of different techniques to learn word embeddings to initialize the RNNs for MD. Table 4 presents the performance of the BASIC and BIDIRECT models on the development set (trained on **news**) when the CBOW, SKIP-GRAM and C-CONCAT techniques are utilized to obtain word embeddings from the same English corpus. We also report the performance of the models when they are initialized with the word2vec word embeddings from Mikolov et al. [2013b] (trained with the Skip-gram model on 100 billion words of Google News) (WORD2VEC). All of these word embeddings are updated during the training of the RNNs to induce the task-specific word embeddings . Finally, for comparison purpose, the performance for the following two scenarios is also included: (i) the word vectors are initialized randomly (not using any pre-trained word embeddings) (RANDOM), and (ii) the word vectors are loaded from the C-CONCAT pre-trained word embeddings but fixed during the RNN training (FIXED).

The first observation is that we need to borrow some pre-trained word embeddings and update them during the training process to improve the MD performance (comparing C-CONCAT, RANDOM and FIXED). Second, C-CONCAT is much better than CBOW, confirming our intuition in Section 4. Third, we do not see much difference in terms of MD performance when we enlarge the corpus to learn word embeddings (comparing SKIP-GRAM and WORD2VEC that is trained with the skip-gram model on a much larger corpus). Finally, we achieve the best performance when we apply the C-CONCAT technique in the BIDIRECT model. From now on, for consistency, we use the C-CONCAT word embeddings in all the experiments below.

5.5 Cross-Domain Experiments

This section evaluates the MD systems on the cross-domain settings to gain an insight into their operation when the domain changes. The state-of-the-art systems for MD have been the joint extraction system for entity mentions and relations from Li and Ji [2014a], the information networks to unify the outputs of three information extraction tasks: entity mentions, relations and events using structured perceptron from Li et al. [2014b] and the Maximum Entropy Markov Model (MEMM) system from Florian et al. [2006]. These systems extensively hand-design a large set of features (parsers, gazetteers, word clusters, coreference etc) to capture the useful structures for MD. In this work, we use the MEMM system in Florian et al. [2006] as the baseline and compare it with the RNN systems. The reason for this choice is twofold: (i) as shown in Section 5.4 of Li and Ji [2014a], the performance of the joint systems are comparable to the MEMM system in Florian et al. [2006], and (ii) similar to our work, the MEMM system in Florian et al. [2006] only focuses on the MD task while the joint systems in Li et al. [2014a; 2014b] involves the predictions for other tasks, making it less comparable to our work, especially on the cross-domain setting for MD. Evaluating the joint models in Li et al. [2014a; 2014b] on the cross-domain setting for MD is another important dimension, however, out of the scope of the current paper.

We note that the performance of the MEMM system reported in this work is obtained from the actual system in Florian et al. [2006] and the feature set of the MEMM[8] system also includes the four features we are using in the RNN models (Section 3).

Following the previous work on the cross-domain settings for the ACE 2005 dataset [Plank and Moschitti, 2013; Nguyen *et al.*, 2015a], we treat *news* as the source domain and the other domains: *bc*, *cts*, *wl* and *un* as the target domains. We then examine the systems on two scenarios: (i) the systems are trained and tested on the source domain via

[8]We also tried the CRF model with the same feature set as the MEMM system but it is worse in our case.

System	Without Features					With Features				
	In-Domain	bc	cts	wl	un	In-Domain	bc	cts	wl	un
MEMM	76.90	71.73	78.02	66.89	67.77	**82.55**	78.33	87.17	**76.70**	76.75
BASIC	79.01	**77.06**	85.42	73.00	72.93	81.99	78.75	86.51	76.60	76.94
BIDIRECT	**80.00**†	76.27†	**85.64**†	**73.79**†	**73.88**†	82.52	**79.65**†	**88.43**†	**76.70**	**77.03**

Table 5: System's Performance on the Cross-domain Setting. Cells marked with †designate the BIDIRECT models that significantly outperform ($p < 0.05$) the MEMM model on the specified domains.

	MEMM				BIDIRECT				BIDIRECT-MEMM			
	bc	cts	wl	un	bc	cts	wl	un	bc	cts	wl	un
bc	75.20	86.60	70.25	72.38	75.49	87.51	70.75	73.04	**0.29**	**0.91**†	**0.50**†	**0.66**†
cts	66.91	89.76	68.74	69.72	68.23	91.24	68.82	70.27	**1.32**†	**1.48**†	**0.08**	**0.55**†
wl	74.94	86.53	77.07	75.90	74.73	86.79	76.35	75.37	-0.21	**0.26**	-0.72	-0.53
un	72.72	86.75	72.04	73.47	73.53	88.29	73.16	74.00	**0.81**†	**1.45**†	**1.12**†	**0.53**†

Table 6: Comparison between MEMM and BIDIRECT. Cells marked with †designate the statistical significance ($p < 0.05$). The columns and rows correspond to the source and target domains respectively. BIDIRECT-MEMM implies performance substraction.

5-fold cross validation (in-domain performance), and (ii) the systems are trained on the source domain but evaluated on the target domains. Besides, in order to understand the effect of the features on the systems, we report the systems' performance both including and excluding the features described in Section 3. Table 5 presents the results.

To summarize, we find that the RNN systems significantly outperform the MEMM system across all the target domains when the features are not applied. The BIDIRECT system still yields the best performance among systems being investigated (except in domain *bc*). This is also the case when the features from Section 3 are included and demonstrates the robustness of the BIDIRECT model in the domain shifts. We further support this result in Table 6 where we report the performance of the MEMM and BIDIRECT systems (with features) on different domain assignments for the source and the target domains. Finally, we also see that the features are very useful for both the MEMM and the RNNs.

5.6 Named Entity Recognition for Dutch

The previous sections have dealt with mention detection for English. In this section, we want to explore the capacity of the systems to quickly and effectively adapt to a new language. In particular, we evaluate the systems on the named entity recognition task (the simplified version of the MD task) for Dutch using the CoNLL 2002 dataset. The state-of-the-art performance for this dataset in the CoNLL evaluation is due to Carreras et al. [2002] who utilize the AdaBoost classifier. In Nothman et al. [2013], the authors leverage data from Wikipedia and are able improve the state-of-the-art performance for Dutch. Very recently, while we are preparing this paper, Gillick el al. [2015] introduce a multilingual language processing system based on bytes and also report the performance on this dataset. Table 7 compares the systems.

We note that the system in Gillick el al. [2015] is also based on RNNs and the row labeled with * for Gillick el al. [2015] corresponds to the system trained on multiple datasets instead of the single CoNLL dataset for Dutch, so not being comparable to ours.

The most important conclusion from the table is that the RNN models in this work significantly outperform MEMM as well as the other comparable system by large margins (up to 22% reduction in relative error). This proves that the proposed RNN systems are less subject to the language changes than MEMM and the other systems. Finally, BIDIRECT is also significantly better than BASIC, testifying to its robustness across languages.

System	P	R	F1
State-of-the-art in CoNLL	77.83	76.29	77.05
Nothman et al. [2013]	-	-	78.60
Gillick el al. [2015]	-	-	78.08
*Gillick el al. [2015]**	-	-	*82.84*
MEMM	80.25	77.52	78.86
BASIC	82.98	81.53	82.25
BIDIRECT	84.08	82.82	**83.45**

Table 7: Performance on Dutch CoNLL 2002.

6 Conclusion

We systematically investigate various RNNs to solve the MD problem which suggests that bidirectional modeling is a very helpful mechanism for this task. In particular, the bidirectional model outperforms a very strong baseline of the feature-based exponential models in the cross-domain setting, thus demonstrating its robustness across domains. We also show that the bidirectional model is more portable to new languages as it is significantly better than the best reported systems for NER in Dutch (up to 22% reduction in relative error). In the future, we plan to apply the bidirectional modeling technique to other tasks as well as study the combination of different network architectures and resources to further improve the performance of the systems.

Acknowledgment

We would like to thank Ralph Grishman and Yifan He for valuable feedbacks. Thank you to the three anonymous reviewers for useful suggestions.

References

[Ando and Zhang, 2005] Rie Ando and Tong Zhang. A high-performance semi-supervised learning method for text chunking. In *ACL*, 2005.

[Bahdanau *et al.*, 2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[Baroni *et al.*, 2014] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL*, 2014.

[Bengio *et al.*, 1994] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. In *Journal of Machine Learning Research 3*, 1994.

[Carreras *et al.*, 2002] Xavier Carreras, Lluís Màrques, and Lluís Padró. Named entity extraction using adaboost. In *CoNLL*, 2002.

[Cherry and Guo, 2015] Colin Cherry and Hongyu Guo. The unreasonable effectiveness of word representations for twitter named entity recognition. In *NAACL*, 2015.

[Cho *et al.*, 2014a] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, 2014a.

[Cho, 2014b] Kyunghyun Cho. Quick introduction to natural language processing with neural networks. In *Lecture at the Ecole Polytechnique de Montreal*, 2014b.

[Collobert *et al.*, 2011] Ronan Collobert, Jason Weston, LÃl'on Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. Natural language processing (almost) from scratch. In *CoRR*, 2011.

[Daume, 2007] Hal Daume. Frustratingly easy domain adaptation. In *ACL*, 2007.

[Florian *et al.*, 2006] Radu Florian, Hongyan Jing, Nanda Kambhatla, and Imed Zitouni. Factorizing complex models: A case study in mention detection. In *ACL*, 2006.

[Florian *et al.*, 2010] Radu Florian, John Pitrelli, Salim Roukos, and Imed Zitouni. Improving mention detection robustness to noisy input. In *EMNLP*, 2010.

[Gillick *et al.*, 2015] Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. Multilingual language processing from bytes. In *arXiv preprint arXiv:1512.00103*, 2015.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. In *Neural Computation*, 1997.

[Józefowicz *et al.*, 2015] Rafal Józefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *ICML*, 2015.

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

[Lafferty *et al.*, 2001] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

[Li and Ji, 2014a] Qi Li and Heng Ji. Incremental joint extraction of entity mentions and relations. In *ACL*, 2014a.

[Li *et al.*, 2014b] Qi Li, Heng Ji, Yu Hong, and Sujian Li. Constructing information networks using one single model. In *EMNLP*, 2014b.

[Mesnil *et al.*, 2013] Gregoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. Investigation of recurrent neural network architectures and learning methods for spoken language understanding. In *Interspeech*, 2013.

[Mikolov *et al.*, 2013b] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013b.

[Nguyen and Grishman, 2015b] Thien Huu Nguyen and Ralph Grishman. Event detection and domain adaptation with convolutional neural networks. In *ACL-IJCNLP*, 2015b.

[Nguyen *et al.*, 2015a] Thien Huu Nguyen, Barbara Plank, and Ralph Grishman. Semantic representations for domain adaptation: A case study on the tree kernel-based method for relation extraction. In *ACL-IJCNLP*, 2015a.

[Nothman *et al.*, 2013] Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R Curran. Learning multilingual named entity recognition from wikipedia. In *Artificial Intelligence*, 2013.

[Plank and Moschitti, 2013] Barbara Plank and Alessandro Moschitti. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *ACL*, 2013.

[Ratinov and Roth, 2009] Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *CoNLL*, 2009.

[Socher *et al.*, 2012] Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. Semantic compositionality through recursive matrix-vector spaces. In *EMNLP*, 2012.

[Tjong Kim Sang, 2002] Erik F. Tjong Kim Sang. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *CoNLL*, 2002.

[Turian *et al.*, 2010] Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In *ACL*, 2010.

[Yao *et al.*, 2014] Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. Spoken language understanding using long short-term memory neural networks. In *IEEE SLT*, 2014.

[Zhou and Xu, 2015] Jie Zhou and Wei Xu. End-to-end learning of semantic role labeling using recurrent neural networks. In *ACL-IJCNLP*, 2015.

[Zitouni and Florian, 2008] Imed Zitouni and Radu Florian. Mention detection crossing the language barrier. In *EMNLP*, 2008.