Recurrent neural network models for disease name recognition using domain invariant features

Sunil Kumar Sahu and **Ashish Anand**
Department of Computer Science and Engineering
Indian Institute of Technology Guwahati
Assam, India - 781039
{sunil.sahu, anand.ashish}@iitg.ernet.in

Abstract

Hand-crafted features based on linguistic and domain-knowledge play crucial role in determining the performance of disease name recognition systems. Such methods are further limited by the scope of these features or in other words, their ability to cover the contexts or word dependencies within a sentence. In this work, we focus on reducing such dependencies and propose a domain-invariant framework for the disease name recognition task. In particular, we propose various end-to-end recurrent neural network (RNN) models for the tasks of disease name recognition and their classification into four pre-defined categories. We also utilize convolution neural network (CNN) in cascade of RNN to get character-based embedded features and employ it with word-embedded features in our model. We compare our models with the state-of-the-art results for the two tasks on NCBI disease dataset. Our results for the disease mention recognition task indicate that state-of-the-art performance can be obtained without relying on feature engineering. Further the proposed models obtained improved performance on the classification task of disease names.

1 Introduction

Automatic recognition of disease names in biomedical and clinical texts is of utmost importance for development of more sophisticated NLP systems such as information extraction, question answering, text summarization and so on (Rosario and Hearst, 2004). Complicate and inconsistent terminologies, ambiguities caused by use of abbreviations and acronyms, new disease names, multiple names (possibly of varying number of words) for the same disease, complicated syntactic structure referring to multiple related names or entities are some of the major reasons for making automatic identification of the task difficult and challenging (Leaman et al., 2009). State-of-the-art disease name recognition systems (Mahbub Chowdhury and Lavelli, 2010; Doğan and Lu, 2012; Dogan et al., 2014) depends on user defined features which in turn try to capture context keeping in mind above mentioned challenges. Feature engineering not only requires linguistic as well as domain insight but also is time consuming and is corpus dependent.

Recently window based neural network approach of (Collobert and Weston, 2008; Collobert et al., 2011) got lot of attention in different sequence tagging tasks in NLP. It gave state-of-art results in many sequence labeling problems without using many hand designed or manually engineered features. One major drawback of this approach is its inability to capture features from outside window. Consider a sentence *"Given that the skin of these adult mice also exhibits signs of de novo hair-follicle morphogenesis, we wondered whether human pilomatricomas might originate from hair matrix cells and whether they might possess beta-catenin-stabilizing mutations"* (taken *verbatim* from PMID: 10192393), words such as *signs* and *originate* appearing both sides of the word *"pilomatricomas"*, play important role in deciding it is a disease. Any model relying on features defined based on words occurring within a fixed window of neighboring words will fail to capture information of influential words occurring outside this window.

Our motivation can be summarized in the following question: *can we identify disease name and categorize them without relying on feature en-*

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2216–2225,
Berlin, Germany, August 7-12, 2016.

gineering, domain-knowledge or task specific resources? In other words, we can say this work is motivated towards mitigating the two issues: first, feature engineering relying on linguistic and domain-specific knowledge; and second, bring flexibility in capturing influential words affecting model decisions irrespective of their occurrence anywhere within the sentence. For the first, we used character-based embedding (likely to capture orthographic and morphological features) as well as word embedding (likely to capture lexico-semantic features) as features of the neural network models.

For the second issue, we explore various recurrent neural network (RNN) architectures for their ability to capture long distance contexts. We experiment with bidirectional RNN (Bi-RNN), bidirectional long short term memory network (Bi-LSTM) and bidirectional gated recurrent unit (Bi-GRU). In each of these models we used sentence level log likelihood approach at the top layer of the neural architecture. The main contributions of the work can be summarized as follows

- Domain invariant features with various RNN architectures for the disease name recognition and classification tasks,
- Comparative study on the use of character based embedded features, word embedding features and combined features in the RNN models.
- Failure analysis to check where exactly our models are failed in the considered tasks.

Although there are some related works (discussed in sec 6), this is the first study, to the best of our knowledge, which comprehensively uses various RNN architectures without resorting to feature engineering for disease name recognition and classification tasks.

Our results show near state-of-the-art performance can be achieved on the disease name recognition task. More significantly, the proposed models obtain significantly improved performance on the disease name classification task.

2 Methods

We first give overview of the complete model used for the two tasks. Next we explained embedded features used in different neural network models. We provide short description of different RNN models in the section 2.3. Training and inference strategies are explained in the section 2.4.

2.1 Model Architectures

Similar to any named entity recognition task, we formulate the disease mention recognition task as a token level sequence tagging problem. Each word has to be labeled with one of the defined tags. We choose *BIO model* of tagging, where *B* stands for beginning, *I* for intermediate and *O* for outsider or other. This way we have two possible tags for all entities of interest, i.e., for all disease mentions, and one tag for other entities.

Generic neural architecture is shown in the figure 1. In the very first step, each word is mapped to its embedded features.

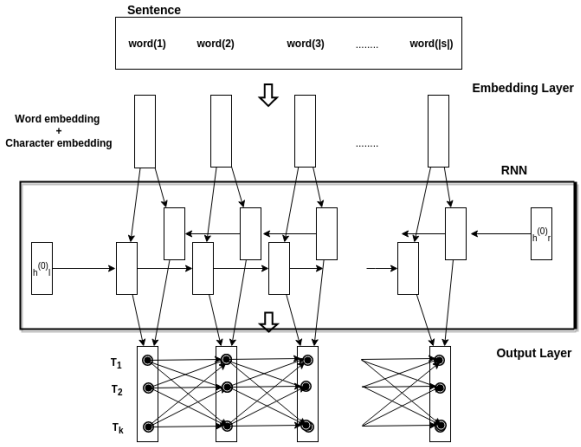

Figure 1: Generic bidirectional recurrent neural network with sentence level log likelihood at the top-layer for sequence tagging task

We call this layer as *embedding layer*. This layer acts as input to hidden layers of RNN model. We study the three different RNN models, and have described them briefly in the section 2.3. Output of the hidden layers is then fed to the output layer to compute the scores for all tags of interest (Collobert et al., 2011; Huang et al., 2015). In output layer we are using sentence level log likelihood, to make inference. Table 1 briefly describes all notations used in the paper.

2.2 Features

Distributed Word Representation (WE)

Distributed word representation or word embedding or simply word vector (Bengio et al., 2003; Collobert and Weston, 2008) is the technique of learning vector representation of a word in a given

Symbols	Explanation
V	Vocabulary of words $(v_1, v_2 ... v_{\|V\|})$
C	Vocabulary of characters $(c_1, c_2 .. c_{\|C\|})$
T	Tag set $(t_1, t_2 ... t_{\|T\|})$
d^{we}	Dimension of word embedding
d^{chr}	Dimension of character embedding
d^{ce}	Dimension of character level word embedding
$M^{we} \in \mathbb{R}^{d^{we} X \|V\|}$	word embedding matrix, where every column M_i^{we} is a vector representation of corresponding word v_i in *V*
$M^{cw} \in \mathbb{R}^{d^{chr} X \|C\|}$	character embedding matrix, where every column M_i^{cw} is a vector representation of corresponding character c_i in *C*.
$w^{(i)} \in \mathbb{R}^{d^{we}}$	word embedding of v_i
$y^{(i)} \in \mathbb{R}^{d^{ce}}$	character level word embedding of v_i
$x^{(i)} \in \mathbb{R}^{d^{we}+d^{ce}}$	feature vector of word $w^{(i)}$. We get this after concatenating $w^{(i)}$ and $y^{(i)}$
$z^{(i)} \in \mathbb{R}^{\|T\|}$	score for i^{th} word in sentence at output layer of neural network. Here j^{th} element will indicate the score for t_j^{th} tag.
W_*^*, U_*^*, V_*^*	Parameters of different neural networks

Table 1: Notation

corpus. Word vectors are present in columns of matrix M^{we}. We can get this vector by taking product of matrix M^{we} and *one hot vector* of v_i.

$$w^{(i)} = M^{we} h^{(i)} \quad (1)$$

Here $h^{(i)}$ is the one hot vector representation of i^{th} word in *V*. We use pre-trained 50 dimensional word vectors learned using skipgram method on a biomedical corpus (Mikolov et al., 2013b; Mikolov et al., 2013a; TH et al., 2015).

Character Level Word Embedding (CE)

Word embedding preserve syntactic and semantic information well but fails to seize morphological and shape information. However, for the disease entity recognition task, such information can play an important role. For instance, letter *-o-* in the word *gastroenteritis* is used to combine various body parts *gastro* for *stomach*, *enter* for *intestines*, and *itis* indicates *inflammation*. Hence taken together it implies *inflammation of stomach and intestines*, where *-itis* play significant role in determining it is actually a disease name.

Character level word embedding was first introduced by (dos Santos and Zadrozny, 2014) with the motivation to capture word shape and morphological features in word embedding. Character level word embedding also automatically mitigate the problem of out of vocabulary words as we can embed any word by its characters through character level embedding. In this case, a vector is initialized for every character in the corpus. Then we learn vector representation for any word by applying CNN on each vector of character sequence of that word as shown in figure 2. These character vectors will get update while training RNN in supervised manner only. Since number of characters in the dataset is not high we assume that every character vectors will get sufficient updation while training RNN itself.

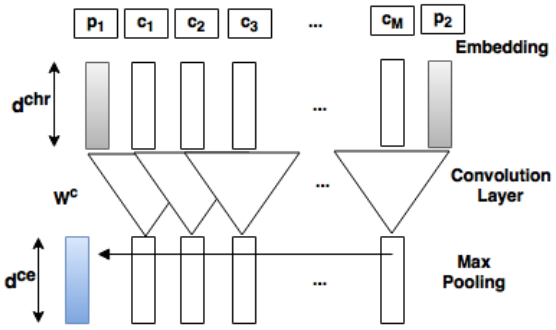

Figure 2: CNN with Max Pooling for Character Level Embedding (p_1 and p_2 are padding). Here, filter length is 3.

Let $\{p_1, c_1, c_2 ... c_M, p_2\}$ is sequence of characters for a word with padding at beginning and ending of word and let $\{a_l, a_1, a_2 ... a_M, a_r\}$ is its sequence of character vector, which we obtain by multiplying M^{cw} with one hot vector of corresponding character. To obtain character level word embedding we need to feed this in convolution neural network (CNN) with max pooling layer (dos Santos and Zadrozny, 2014). Let $W^c \in$

$\mathbb{R}^{d^{ce}X(d^{chr}Xk)}$ is a filter and b^c bias of CNN, then

$$[y^{(i)}]_j = \max_{1<m<M}[W^c q^{(m)} + b^c]_j \quad (2)$$

Here k is window size, $q^{(m)}$ is obtained by concatenating the vector of $(k-1)/2$ character left to $(k-1)/2$ character right of c_m. Same filter will be used for all window of characters and max pooling operation is performed on results of all. We learn 100 dimensional character embedding for all characters in a given dataset (avoiding case sensitivity) and 25 dimensional character level word embedding from character sequence of words.

2.3 Recurrent Neural Network Models

Recurrent Neural Network (RNN) is a class of artificial neural networks which utilizes sequential information and maintains history through its intermediate layers (Graves et al., 2009; Graves, 2013). We experiment with three different variants of RNN, which are briefly described in subsequent subsections.

Bi-directional Recurrent Neural Network

In Bi-RNN, context of the word is captured through past and future words. This is achieved by having two hidden components in the intermediate layer, as schematically shown in the fig 1. One component process the information in forward direction (left to right) and other in reverse direction. Subsequently outputs of these components then concatenated and fed to the output layer to get score for all tags of the considered word. Let $x^{(t)}$ is a feature vector of t^{th} word in sentence (concatenation of corresponding embedding features w^{t_i} and y^{t_i}) and $h_l^{(t-1)}$ is the computation of last hidden state at $(t-1)^{th}$ word, then computation of hidden and output layer values would be:

$$h_l^{(t)} = tanh(U^l x^{(t)} + W^l h_l^{(t-1)})$$

$$z^{(t)} = V(h_l^{(t)} : h_r^{(t)}) \quad (3)$$

Here $U^l \in \mathbb{R}^{n_H \times n_I}$ and $W^l \in \mathbb{R}^{n_H \times n_H}$, where n_I is input vector of length $d^{we} + d^{ce}$, n_H is hidden layer size and $V \in \mathbb{R}^{n_O \times (n_H + n_H)}$ is the output layer parameter. $h_l^{(t)}$ and $h_r^{(t)}$ correspond to left and right hidden layer components respectively and $h_r^{(t)}$ is calculated similarly to $h_l^{(t)}$ by reversing the words in the sentence. At the beginning $h_l^{(0)}$ and $h_r^{(0)}$ are initialized randomly.

Bi-directional Long Short Term Memory Network

Traditional RNN models suffer from both vanishing and exploding gradient (Pascanu et al., 2012; Bengio et al., 2013). Such models are likely to fail where we need longer contexts to do the job. These issues were the main motivation behind the LSTM model (Hochreiter and Schmidhuber, 1997). LSTM layer is just another way to compute a hidden state which introduces a new structure called a memory cell (c_t) and three gates called as input (i_t), output (o_t) and forget (f_t) gates.

These gates are composed of sigmoid activation function and responsible for regulating information in memory cell. The input gate by allowing incoming signal to alter the state of the memory cell, regulates proportion of history information memory cell will keep. On the other hand, the output gate regulates what proportion of stored information in the memory cell will influence other neurons. Finally, the forget gate can modulate the memory cells and allowing the cell to remember or forget its previous state. Computation of memory cell ($c^{(t)}$) is done through previous memory cell and candidate hidden state ($g^{(t)}$) which we compute through current input and the previous hidden state. The final output of hidden state would be calculated based on memory cell and forget gate.

In our experiment we used model discussed in (Graves, 2013; Huang et al., 2015). Let $x^{(t)}$ is feature vector for t^{th} word in a sentence and $h_l^{(t-1)}$ is previous hidden state then computation of hidden ($h_l^{(t)}$) and output layer ($z^{(t)}$) of LSTM would be.

$$\begin{aligned}
i_l^{(t)} &= \sigma(U_l^{(i)} x^{(t)} + W_l^{(i)} h_l^{(t-1)} + b_l^i) \\
f_l^{(t)} &= \sigma(U_l^{(f)} x^{(t)} + W_l^{(f)} h_l^{(t-1)} + b_l^f) \\
o_l^{(t)} &= \sigma(U_l^{(o)} x^{(t)} + W_l^{(o)} h_l^{(t-1)} + b_l^o) \\
g_l^{(t)} &= tanh(U_l^{(g)} x^{(t)} + W_l^{(g)} h_l^{(t-1)} + b_l^g) \\
c_l^{(t)} &= c_l^{(t-1)} * f_l + g_l * i_l \\
h_l^{(t)} &= tanh(c_l^{(t)}) * o_l
\end{aligned}$$

Where σ is sigmoid activation function, $*$ is a element wise product, $U_l^{(i)}, U_l^{(f)}, U_l^{(o)}, U_l^{(g)} \in \mathbb{R}^{n_H \times n_I}$ and $W_l^{(i)}, W_l^{(o)}, W_l^{(f)}, W_l^{(g)} \in \mathbb{R}^{n_H \times n_H}$, where n_I is input size ($d^{we} + d^{ce}$) and n_H is hidden layer size. We compute $h_r^{(t)}$ in similar manner as $h_l^{(t)}$ by reversing the all words of sentence. Let $V \in \mathbb{R}^{n_O \times (n_H + n_H)}$ (n_O size of output layer) is

the parameter of output layer of LSTM then computation of output layer will be:

$$z^{(t)} = V(h_l^{(t)} : h_r^{(t)}) \quad (4)$$

Bi-directional Gated Recurrent Unit Network

A gated recurrent unit (GRU) was proposed by (Cho et al., 2014) to make each recurrent unit to adaptively capture dependencies of different time scales. Similar to the LSTM unit, the GRU has gating units reset *r* and update z gates that modulate the flow of information inside the unit, however, without having a separate memory cells. The resulting model is simpler than standard LSTM models.

We follow (Chung et al., 2014) model of GRU to transform the extracted word embedding and character embedding features to score for all tags. Let $x^{(t)}$ embedding feature for *t*th word in sentence and $h_l^{(t-1)}$ is computation of hidden state for $(t-1)$th word then computation of GRU would be:

$$z_l^{(t)} = \sigma(U_l^{(z)} x^{(t)} + W_l^{(z)} h_l^{(t-1)} + b_l^{(z)})$$
$$r_l^{(t)} = \sigma(U_l^{(r)} x^{(t)} + W_l^{(r)} h_l^{(t-1)} + b_l^{(r)})$$
$$\tilde{h}_l^{(t)} = \tanh(U_l^{(h)} x^{(t)} + W_l^{(h)} h_l^{(t-1)} * r_l + b_l^{(h)})$$
$$h_l^{(t)} = z_l^{(t)} * \tilde{h}_l + (1 - z_l^{(t)}) * h_l^{(t-1)}$$

$$z^{(t)} = V(h_l^{(t)} : h_r^{(t)}) \quad (5)$$

Where $*$ is pair wise multiplication, $U_l^{(z)}, U_l^{(r)}$, $U_l^{(h)}, U_l^{(h)} \in \mathbb{R}^{n_H \times n_I}$ and $W_l^{(z)}, W_l^{(r)} W_l^{(h)} \in \mathbb{R}^{n_H \times n_H}$ are parameters of GRU. $V \in \mathbb{R}^{n_O \times (n_H + n_H)}$ is output layer parameter. Computation of $h_r^{(t)}$ is done in similar manner as $h_l^{(t)}$ by reversing the words of sentence.

2.4 Training and Inference

Equations 3, 4 and 5 are the scores of all possible tags for t^{th} word sentence. We follow *sentence-level log-likelihood (SLL)* (Collobert et al., 2011) approach equivalent to linear-chain CRF to infer the scores of a particular tag sequence for the given word sequence. Let $[w]_1^{|s|}$ is sentence and $[t]_1^{|s|}$ is the tag sequence for which we want to find the joint score, then score for the whole sentence with the particular tag sequence would be:

$$s([w]_1^{|s|}, [t]_1^{|s|}) = \sum_{1 \le i \le |s|} (W_{t_{i-1}, t_i}^{trans} + z_{t_i}^{(i)}), \quad (6)$$

where W^{trans} is transition score matrix and $W_{i,j}^{trans}$ is indicating the transition score moving from tag t_i to t_j; t_j is tag for the j^{th} word; $z_{t_i}^{(i)}$ is the output score from the neural network model for the tag t_i of i^{th} word. To train our model we used cross entropy loss function and adagrad (Duchi et al., 2010) approach to optimize the loss function. Entire neural network parameters, word embedding, character embedding and W^{trans} (transition score matrix used in the SLL) was updated during training. Entire code has been implemented using theano (Bastien et al., 2012) library in python language.

3 Experiments

3.1 Dataset

We used NCBI dataset (Doğan and Lu, 2012), the most comprehensive publicly available dataset annotated with disease mentions, in this work. NCBI dataset has been manually annotated by a group of medical practitioners for identifying diseases and their types in biomedical articles. All disease mentions were categorized into four different categories, namely, *specific disease*, *disease class*, *composite disease* and *modifier*. A word is annotated as *specific disease*, if it indicates a particular disease. *Disease class* category indicates a word describing a family of many specific diseases, such as autoimmune disorder. A string signifying two or more different disease mentions is annotated with *composite mention*. *Modifier* category indicates disease mention has been used as modifiers for other concepts. This dataset is a extension of the AZDC dataset (Leaman et al., 2009) which was annotated with disease mentions only and not with their categories. Statistics of the dataset is mentioned in the Table 2.

Corpus	***Train set***	***Dev set***	***Test set***
sentences	5661	939	961
disease	5148	791	961
spe. dis.	2959	409	556
disease class	781	127	121
modifier	1292	218	264
comp. men.	116	37	20

Table 2: Dataset statistics. spe. dis. : specific disease and comp. men.: composite mention

In our evaluation we used this dataset in two settings, **A**: *disease mention recognition*, where all

Task	Model	Validation Set			Test Set		
		Precision	Recall	F1 Score	Precision	Recall	F1 Score
A	NN+CE	76.98	75.80	76.39	78.51	72.75	75.52
	Bi-RNN+CE	71.96	74.90	73.40	74.14	72.12	73.11
	Bi-GRU+CE	76.28	74.14	75.19	76.03	69.81	72.79
	Bi-LSTM+CE	81.52	72.86	**76.94**	76.98	75.80	**76.39**
B	NN+CE	67.27	53.45	59.57	67.90	49.95	57.56
	Bi-RNN+CE	61.34	56.32	58.72	60.32	57.28	58.76
	Bi-GRU+CE	61.94	59.11	**60.49**	62.56	56.50	59.38
	Bi-LSTM+CE	61.82	57.03	59.33	64.74	55.53	**59.78**

Table 3: Performance of various models using 25 dimensional CE features, A:Disease name recognition, B: Disease classification task

disease types are flattened into a single category and, the **B**: *disease class recognition*, where we need to decide exact categories of disease mentions. It is noteworthy to mention that the Task B is more challenging as it requires model to capture semantic contexts to put disease mentions into appropriate categories.

4 Results and Discussion

Evaluation of different models using CE

We first evaluate the performance of different RNNs using only character embedding features. We compare the results of RNN models with window based neural network (Collobert et al., 2011) using sentence level log likelihood approach (NN + CE). For the window based neural network, we considered window size 5 (two words from both left and right, and one central word) and same settings of character embedding were used as features. The same set of parameters are used in all experiments unless we mention specifically otherwise. We used exact matching scheme to evaluate performance of all models.

Table 3 shows the results obtained by different RNN models with only character level word embedding features. For the task A (Disease name recognition) Bi-LSTM and NN models gave competitive performance on the test set, while Bi-RNN and Bi-GRU did not perform so well. On the other hand for the task B, there is $2.08\% - 3.8\%$ improved performance (F1-score) shown by RNN models over the NN model again on the test set. Bi-LSTM model obtained F1-score of 59.78% while NN model gave 57.56%. As discussed earlier, task B is difficult than task A as disease category is more likely to be influenced by the words falling outside the context window considered in window based methods. This could be reason for RNN models to perform well over the NN model. This hypothesis will be stronger if we observe similar pattern in our other experiments.

Evaluation of different models with WE and WE+CE

Next we investigated the results obtained by the various models using only 50 dim word embedding features. The first part of table 4 shows the results obtained by different RNNs and the window based neural network (NN). In this case RNN models are giving better results than the NN model for both the tasks. In particular performance of Bi-LSTM models are best than others in both the tasks. We observe that for the task A, RNN models obtained 1.2% to 3% improvement in F1-score than the baseline NN performance. Similarly 2.55% to 4% improvement in F1-score are observed for the task B, with Bi-LSTM model obtaining more than 4% improvement.

In second part of this table we compare the results obtained by various models using the features set obtained by combining the two feature sets. If we look at performance of individual model using three different set of features, model using only word embedding features seems to give consistently best performance. Among all models, Bi-LSTM using word embedding features obtained best F1-scores of 79.13% and 63.16% for the tasks A and B respectively.

Importance of tuning pre-trained word vectors

We further empirically evaluate the importance of updating of word vectors while training. For this, we performed another set of experiments, where pre-trained word vectors are not updated while

Task	Model	Validation Set			Test Set		
		Precision	Recall	F1 Score	Precision	Recall	F1 Score
A	NN+WE	81.86	76.82	79.26	80.32	73.58	76.81
	Bi-RNN+WE	84.14	77.46	80.67	82.49	73.58	77.78
	Bi-GRU+WE	84.51	78.23	81.25	82.32	75.16	78.58
	Bi-LSTM+WE	85.13	77.72	**81.26**	84.87	74.11	**79.13**
B	NN+WE	65.33	56.43	60.55	64.23	57.14	60.48
	Bi-RNN+WE	63.62	56.84	60.04	67.47	57.50	62.09
	Bi-GRU+WE	66.42	57.41	61.59	68.25	58.58	63.05
	Bi-LSTM+WE	67.48	58.01	**62.39**	68.97	58.25	**63.16**
A	NN+WE+CE	76.37	78.62	**77.48**	74.92	75.16	75.04
	Bi-RNN+WE+CE	76.10	75.03	75.56	77.01	72.33	74.59
	Bi-GRU+WE+CE	77.73	76.44	77.08	78.04	73.38	**75.63**
	Bi-LSTM+WE+CE	76.94	77.34	77.14	76.10	74.11	75.09
B	NN+WE+CE	67.60	56.70	61.67	67.60	56.70	61.67
	Bi-RNN+WE+CE	60.94	61.34	61.14	64.36	60.90	62.58
	Bi-GRU+WE+CE	61.58	61.99	**61.78**	61.92	63.85	**62.87**
	Bi-LSTM+WE+CE	64.92	58.61	61.60	61.14	60.54	60.84

Table 4: Performance of various models using 50 dimensional WE features. A:Disease name recognition, B: Disease classification task

training. Results obtained on the validation dataset of the Task A are shown in the Table 5. One can observe that performance of all models have deteriorated. Next, instead of using pre-trained word vectors, we initialized each word with zero vector but kept updating them while training. Although performance (Table 6) deteriorated (compare to Table 4) but not as much as in table 5. This observation highlights the importance of tuning word vectors for a specific task during training.

Model	*P*	*R*	*F*
NN+WE	74.02	67.86	70.81
Bi-RNN+WE	72.17	64.40	68.06
Bi-GRU+WE	77.06	70.55	73.66
Bi-LSTM+WE	77.32	73.75	75.49

Table 5: Performance of different models with *50 dim embedded vectors* in **Task A** *validation set* **when word vectors are not getting updated while training**

Comparison with State-of-art

At the end we are comparing our results with state-of-the art results reported in (Doğan and Lu, 2012) on this dataset using BANNER (Leaman and Gonzalez, 2008) in table 7. BANNER is a CRF based bio entity recognition model, which uses general linguistic, orthographic, syntactic dependency features. Although the result reported in (Doğan and Lu, 2012) (F1-score = 81.8) is better than that of our RNN models but it should be noted that competitive result (F1-score = 79.13%) is obtained by the proposed Bi-LSTM model which does not depend on any feature engineering or domain-specific resources and is using only word embedding features trained in unsupervised manner on a huge corpus.

Model	*P*	*R*	*F*
NN+RV	81.64	74.01	77.64
Bi-RNN+RV	82.32	72.73	77.2
Bi-GRU+RV	82.48	74.14	78.08
Bi-LSTM+RV	83.41	72.73	77.70

Table 6: Results of different models with *50 dim random vectors* in **Task A** *validation set*

For the task B, we did not find any paper except (Li, 2012). Li (2012) used linear soft margin support vector (SVM) machine with a number of hand designed features including dictionary based features. The best performing proposed model shows more than 37% improvement in F1-score (benchmark: 46% vs Bi-LSTM+WE: 63.16%).

5 Failure Analysis

To see where exactly our models failed to recognize diseases, we analyzed the results carefully.

Task	Model	Validation Set			Test Set		
		P	R	F	P	R	F
A	Bi-LSTM+WE	85.13	77.72	81.26	84.87	74.11	79.13
	BANNER (Doğan and Lu, 2012)	-	-	**81.9**	-	-	**81.8**
B	Bi-LSTM+WE	67.48	58.01	**62.39**	68.97	58.25	**63.16**
	SM-SVM(Li, 2012)	-	-	-	66.1	35.2	46.0

Table 7: Comparisons of our best model results and state-of-art results. SM-SVM :Soft Margin Support Vector Machine

We found that significant proportion of errors are coming due to use of acronyms of diseases and use of disease form which is rarely appearing in our corpus. Examples of few such cases are *"CD", "HNPCC","SCA1"*. We observe that this error is occurring because we do not have exact word embedding for these words. Most of the acronyms in the disease corpus were mapped to rare-word embedding[1]. Another major proportion of errors in our results were due to difficulty in recognizing nested forms of disease names. For example, in all of the following cases: *"hereditary forms of 'ovarian cancer'" , "inherited 'breast cancer'", "male and female 'breast cancer'"*, part of phrase such as *ovarian cancer* in *hereditary forms of ovarian cancer*, *breast cancer* in *inherited breast cancer* and *male and female breast cancer* are disease names and our models are detecting this very well. However, according to annotation scheme if any disease is part of nested disease name, annotators considered whole phrase as a single disease. So even our model is able to detect part of the disease accurately but due to the exact matching scheme, this will be false positive for us.

6 Related Research

In biomedical domain, named entity recognition has attracted much attention for identification of entities such as genes and proteins (Settles, 2005; Leaman and Gonzalez, 2008; Leaman et al., 2009) but not as much for disease name recognition. Notable works, such as of Chowdhury and Lavelli (2010), are mainly conditional random field (CRF) based models using lots of manually designed template features. These include linguistic, orthographic, contextual and dictionary based features. However, they have evaluated their model on the AZDC dataset which is small compared to the NCBI dataset, which we have considered in this study. Nikfarjam et al. (2015) have proposed a CRF based sequence tagging model, where cluster id of embedded word as an extra feature with manually engineered features is used for adverse drug reaction recognition in tweets.

Recently deep neural network models with minimal dependency on feature engineering have been used in few studies in NLP including NER tasks (Collobert et al., 2011; Collobert and Weston, 2008). dos Santos et al. (2015) used deep neural network based model such as window based network to recognize named entity in Portuguese and Spanish texts. In this work, they exploit the power of CNN to get morphological and shape features of words in character level word embedding, and used it as feature with concatenation of word embedding. Their results indicate that CNN are able to preserve morphological and shape features through character level word embedding. Our models are quite similar to this model but we used different variety of RNN in place of window based neural network.

Labeau et al. (2015) used Bi-RNN with character level word embedding only as a feature for PoS tagging in German text. Their results also show that with only character level word embedding we can get state-of-art results in PoS tagging in German text. Our model used word embedding as well as character level word embedding together as features and also we have tried more sophisticated RNN models such as LSTM and GRU in bi-directional structure. More recent work of Huang et al. (2015) used LSTM and CRF in variety of combination such as only LSTM, LSTM with CRF and Bi-LSTM with CRF for PoS tagging, chunking and NER tasks in general texts. Their results shows that Bi-LSTM with CRF gave best results in all these tasks. These two works have used either Bi-RNN with character embedding features or Bi-LSTM with word embedding

[1]we obtained pre-trained word-embedding features from (TH et al., 2015) and in their pre-processing strategy, all words of frequency less than 50 were mapped to rare-word.

features in general or news wire texts, whereas in this work we compare the performance of three different types of RNNs: Bi-RNN, Bi-GRU and Bi-LSTM with both word embedding and character embedding features in biomedical text for disease name recognition.

7 Conclusions

In this work, we used three different variants of bidirectional RNN models with word embedding features for the first time for disease name and class recognition tasks. Bidirectional RNN models are used to capture both forward and backward long term dependencies among words within a sentence. We have shown that these models are able to obtain quite competitive results compared to the benchmark result on the disease name recognition task. Further our results have shown a significantly improved results on the relatively harder task of disease classification which has not been studied much. All our results were obtained without putting any effort on feature engineering or requiring domain-specific knowledge. Our results also indicate that RNN based models perform better than window based neural network model for the two tasks. This could be due to the implicit ability of RNN models to capture variable range dependencies of words compared to explicit dependency on context window size of window based neural network models.

Acknowledgments

We acknowledge the use of computing resources made available from the Board of Research in Nuclear Science (BRNS), Dept of Atomic Energy (DAE) Govt. of India sponsered project (No.2013/13/8-BRNS/10026) by Dr Aryabartta Sahu at Department of Computer Science and Engineering, IIT Guwahati.

References

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March.

Yoshua Bengio, Nicolas Boulanger-Lewandowski, and Razvan Pascanu. 2013. Advances in optimizing recurrent networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8624–8628. IEEE.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November.

Rezarta Islamaj Dogan, Robert Leaman, and Zhiyong Lu. 2014. NCBI disease corpus: A resource for disease name recognition and concept normalization. *Journal of Biomedical Informatics*, 47:1–10.

Cicero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *International Conference on Machine Learning (ICML)*, volume 32, pages 1818–1826. JMLR W&CP.

Cicero dos Santos, Victor Guimaraes, RJ Niterói, and Rio de Janeiro. 2015. Boosting named entity recognition with neural character embeddings. *Proceedings of NEWS 2015 The Fifth Named Entities Workshop*, page 9.

Rezarta Islamaj Doğan and Zhiyong Lu. 2012. An improved corpus of disease mentions in pubmed citations. In *Proceedings of the 2012 Workshop on Biomedical Natural Language Processing*, BioNLP '12, pages 91–99, Stroudsburg, PA, USA. Association for Computational Linguistics.

John Duchi, Elad Hazan, and Yoram Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. Technical Report UCB/EECS-2010-24, EECS Department, University of California, Berkeley, Mar.

Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. 2009. A novel connectionist system for unconstrained handwriting recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(5):855–868, May.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November.

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

Matthieu Labeau, Kevin Lser, and Alexandre Allauzen. 2015. Non-lexical neural architecture for fine-grained pos tagging. In Llus Mrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton, editors, *EMNLP*, pages 232–237. The Association for Computational Linguistics.

Robert Leaman and Graciela Gonzalez. 2008. Banner: An executable survey of advances in biomedical named entity recognition. In Russ B. Altman, A. Keith Dunker, Lawrence Hunter, Tiffany Murray, and Teri E. Klein, editors, *Pacific Symposium on Biocomputing*, pages 652–663. World Scientific.

Robert Leaman, Christopher Miller, and G Gonzalez. 2009. Enabling recognition of diseases in biomedical text with machine learning: corpus and benchmark. *Proceedings of the 2009 Symposium on Languages in Biology and Medicine*, 82(9).

Gang Li. 2012. Disease mention recognition using soft-margin svm. *Training*, 593:5–148.

Md. Faisal Mahbub Chowdhury and Alberto Lavelli. 2010. Disease mention recognition with specific features. In *Proceedings of the 2010 Workshop on Biomedical Natural Language Processing*, BioNLP '10, pages 83–90, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Azadeh Nikfarjam, Abeed Sarker, Karen OConnor, Rachel Ginn, and Graciela Gonzalez. 2015. Pharmacovigilance from social media: mining adverse drug reaction mentions using sequence labeling with word embedding cluster features. *Journal of the American Medical Informatics Association*, page ocu041.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063.

Barbara Rosario and Marti A. Hearst. 2004. Classifying semantic relations in bioscience texts. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

B. Settles. 2005. ABNER: An open source tool for automatically tagging genes, proteins, and other entity names in text. *Bioinformatics*, 21(14):3191–3192.

MUNEEB TH, Sunil Sahu, and Ashish Anand. 2015. Evaluating distributed word representations for capturing semantics of biomedical concepts. In *Proceedings of BioNLP 15*, pages 158–163, Beijing, China, July. Association for Computational Linguistics.