arXiv:1701.08303v2 [cs.CL] 13 Aug 2017

Drug-Drug Interaction Extraction from Biomedical Text Using Long Short Term Memory Network

Sunil Kumar Sahu
IIT Guwahati, India
sunil.sahu@iitg.ernet.in

Ashish Anand
IIT Guwahati, India
anand.ashish@iitg.ernet.in

ABSTRACT

Simultaneous administration of multiple drugs can have synergistic or antagonistic effects as one drug can affect activities of other drugs. Synergistic effects lead to improved therapeutic outcomes, whereas, antagonistic effects can be life-threatening, may lead to increased healthcare cost, or may even cause death. Thus identification of unknown drug-drug interaction (DDI) is an important concern for efficient and effective healthcare. Although multiple resources for DDI exist, they are often unable to keep pace with rich amount of information available in fast growing biomedical texts. Most existing methods model DDI extraction from text as a classification problem and mainly rely on handcrafted features. Some of these features further depend on domain specific tools. Recently neural network models using latent features have been shown to give similar or better performance than the other existing models dependent on handcrafted features. In this paper, we present three models namely, *B-LSTM*, *AB-LSTM* and *Joint AB-LSTM* based on long short-term memory (LSTM) network. All three models utilize word and position embedding as latent features and thus do not rely on explicit feature engineering. Further use of bidirectional long short-term memory (Bi-LSTM) networks allow implicit feature extraction from the whole sentence. The two models, *AB-LSTM* and *Joint AB-LSTM* also use attentive pooling in the output of Bi-LSTM layer to assign weights to features. Our experimental results on the SemEval-2013 DDI extraction dataset show that the *Joint AB-LSTM* model outperforms all the existing methods, including those relying on handcrafted features. The other two proposed LSTM models also perform competitively with state-of-the-art methods.

Keywords

Information Extraction, Recurrent Neural Network, Long Short Term Memory, Attention Model

1. INTRODUCTION

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2017 ACM. ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

There has been a significant rise in a number of persons taking multiple drugs at the same time. According to the numbers released in 2010 by the US Centers for Disease Control and Prevention, one in ten Americans is on five or more medications [17]. Similar statistics can be expected from other countries as well. When multiple drugs are administered together, there is an inevitable risk of a drug affecting activities of other drugs. Effect of DDI can be either synergistic or antagonistic. Adverse drug reaction (ADR) is an example of antagonistic effect [6]. With the rise in people taking multiple drugs, it is very important to have DDI information available in structured form. DrugBank[1] and Stockley[2] are examples of knowledge bases (KBs) keeping DDI information in a structured form. However, keeping KBs update with growing rate of biomedical literature is a challenging task [30, 29]. PubMed, a database of biomedical articles, contains about 27 million citations[3] and approximately 0.8 million articles are being added annually[4]. The growing rate of this number in recent years has necessitated the development of efficient automatic tools for information extraction across biomedical domain [24, 31].

Identifying DDIs in text is the process of recognizing how two drugs in a given sentence are related [29]. We illustrate different interaction types between two drugs through examples in Table1. The two pairs (*Fluoxetine*, *Phenelzine*) and (*Crocin*, *Phenelzine*) in the sentence [S_1] fall into *Advice* interaction category as they are suggested to be not taken together. Similarly, interacting drug pairs (*PGF2alpha*, *Oxytocin*) in the sentence [S_2] and (*Ketamine*, *Halothane*) in [S_3] belong to *Effect* and *Mechanism* categories respectively, since impact and mechanism of impact are present in the respective sentences. Sentence [S_4] does not say anything more than that the two drugs (*Warfarin* and *Rifampin*) are interacting, so it falls into *Interaction* (*Int*) class. Identifying this kind of information can also be useful for other applications such as drug repurposing, semantic search and other information retrieval tasks.

Realizing the importance of automatic extraction of drug interaction information, two challenges were organized. The first DDI extraction challenge [30], organized as a workshop in the SEPLN 2011[5] conference, focused on the DDI extrac-

[1]https://www.drugbank.ca/
[2]https://www.medicinescomplete.com/mc/alerts/current/drug-interactions.htm
[3]https://www.ncbi.nlm.nih.gov/pubmed
[4]https://www.nlm.nih.gov/bsd/stats/cit_added.html
[5]http://www.uhu.es/sepln2011/index.php/en.html

$[S_1]$	**Fluoxetine** and crocin should not be administered to patients receiving **phenelzine**	*Advice(Fluoxetine, Phenelzine)*
$[S_1]$	Fluoxetine and **crocin** should not be administered to patients receiving **phenelzine**	*Advice(Crocin, Phenelzine)*
$[S_2]$	Both **PGF2alpha** and **Oxytocin** induced dopamine release in the nucleus accumbens	*Effect(PGF2alpha, Oxytocin)*
$[S_3]$	The half life of **ketamine** in plasma and brain was longer in the presence of **halothane**	*Mechanism(Ketamine, Halothane)*
$[S_4]$	The drug interaction between **warfarin** and **rifampin** is not well known	*Int (Warfarin, Rifampin)*

Table 1: Examples illustrating different types of interactions between two drugs. All examples are taken from the dataset described in Sec. 3.

tion task. The aim of the task was to identify whether there was an interaction between a given pair of drugs mentioned within a sentence. It was assumed that the drug names present in sentences were given. Extension of the first challenge was organized as SemEval 2013 Task-9 [29]. In this challenge, two tasks were designed: first task concentrated on drug name recognition and their classification, and the second focused on DDI classification from biomedical text. In this work, we focus on the second task of the SemEval 2013 Task-9 only.

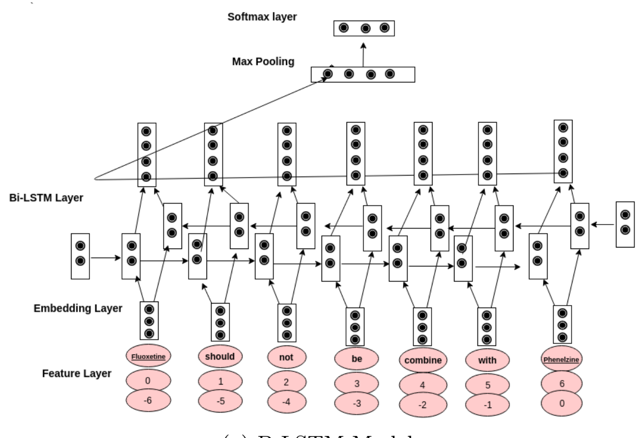
Existing methods can be classified into two categories: one-stage and two-stage methods. In one-stage methods [3, 13, 36], a multi-class classifier is used to map a sentence with two target drugs either into one of the interacting classes or into the negative class. On the other hand two-stage methods [26, 2], as the name suggests, break the problem into two steps. The first step builds a binary classifier to determine whether an interaction exists between two target drugs or not. Only those sentences with target drug pairs, which fall into positive category of the binary classifier in the previous step, are considered as input to multi-class classifier of the second step. These methods can further be divided into two categories, methods relying on handcrafted features and methods using latent features. In the first category mainly support vector machines (SVMs) with linear or non-linear kernels have been used in several studies [8, 4, 18]. All of these methods are dependent on manually engineered features such as PoS tag, chunk tag, trigger words, shortest dependency tree and syntax tree. Methods using non-linear kernels map structure features (dependency tree and syntax tree) into real values. Such methods have been successfully used for other similar relation extraction tasks including ADRs extraction from biomedical texts [12, 11, 14, 37] and from social media texts [38], protein-protein interaction extraction from biomedical text [25], relation between genes and diseases [5], and relations between medical concepts [27]. Although such methods have been shown to perform well, they require manually crafted features. Extraction of these features is however dependent on other NLP tools. Inherent noise and cost of such tools may adversely affect the performance of models dependent on these features. Methods using latent features and belonging to the second category, are result of re-emergence of deep learning models as a powerful alternative to conventional feature based models. Some notable studies [20, 40] for the DDI extraction tasks are based on convolution neural networks (CNNs) and have been shown to achieving superior performance than the existing state-of-the-art methods. We discuss more about these methods in section 4.2.

In this work, we also rely on latent features learned by neural network models. As opposed to works in [20, 40], which used CNN models, we use LSTM based neural network models [16]. CNN models require pooling on continuous n grams built on entire sentence to obtain constant length features. Here n is the length of convolution or filter. It may cause problems for the sentences of large length and/or for sentences having important clues lying far away from each other. To overcome this issue we use Bi-LSTM with two different pooling techniques for encoding variable length features. Theoretically, a Bi-LSTM can preserve information about the past and future words while reading [16]. Therefore when we apply pooling on the output of Bi-LSTM, we can get features containing information about complete context from whole sentence. This is in contrast to the CNN models which extract features based on n gram of the sentence. With this intuition, we propose three models, namely: *B-LSTM*, *AB-LSTM* and *Joint AB-LSTM* for the DDI extraction task. Here *B-LSTM* and *AB-LSTM* uses a Bi-LSTM for encoding word and position features. *B-LSTM* uses max pooling and *AB-LSTM* uses attentive pooling on the outputs of Bi-LSTM to get fixed length features over complete sentence. On the other hand, *Joint AB-LSTM* being an ensemble of *B-LSTM* and *AB-LSTM* uses two Bi-LSTMs, one with max pooling and another with attentive pooling. In each of these models we use fully connected neural network in output layer.

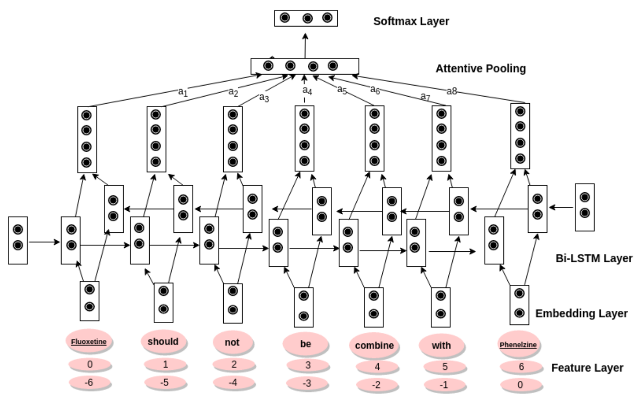
All the three proposed models give either competitive performance compared to the existing methods or have achieved new state-of-the-art performance. The two important features of the models are: all of them belong to one-stage category and use simple features. None of the chosen features explicitly extract syntactic information hidden in a sentence. Among the three proposed models, *Joint AB-LSTM* outperforms all existing models for the DDI extraction tasks. Analysis of our results indicate that all models finds it difficult to make correct prediction for drug pairs present in a long sentence having too many other drug entries. If we compare between CNN and LSTM models, then LSTM models are generally found to have better prediction for longer sentences than the CNN model. We believe that new models should work in the direction of mitigating above issues to bring significant improvement.

2. MODEL ARCHITECTURE

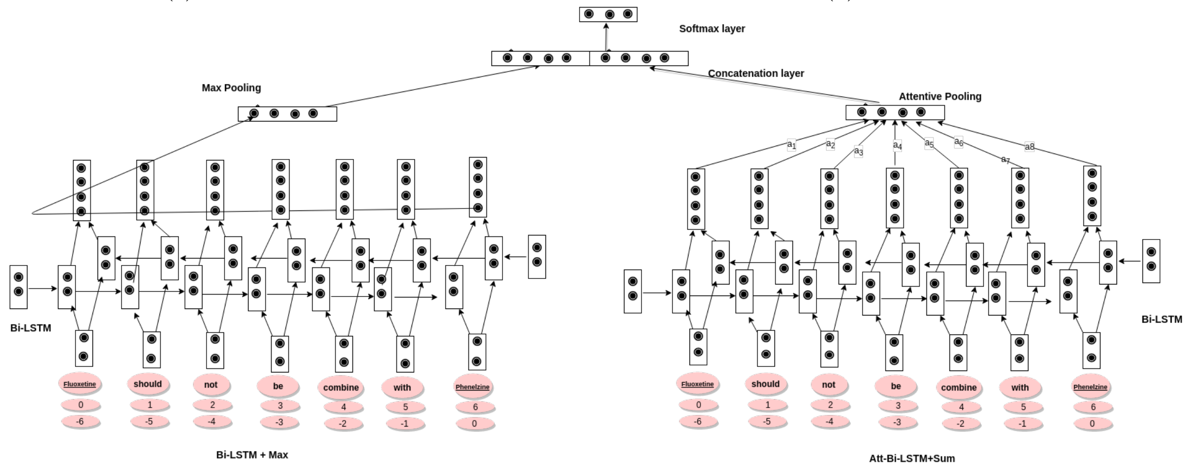
We present three LSTM based models namely, *B-LSTM*, *AB-LSTM* and *Joint AB-LSTM* for the DDI extraction task. We assume that the two targeted drug names are given in a sentence and model has to classify it into one of the five categories: *Advice*, *Effect*, *Mechanism*, *Int*, *Negative*. Section 3 describes about the task in detail. Architecture of the three proposed models are shown in Figure 1. Each model uses embedding features as input in first layer and learn fixed length vector representation through subsequent layers. Score for each possible class is computed in the final layer and decision is made using these scores. We now briefly explain each components of the three models.



(a) B-LSTM Model



(b) AB-LSTM Model



(c) Joint AB-LSTM Model

Figure 1: Block diagram of all three models

Feature Layer

We represent each word in the sentence with three discrete local features, namely: *word* (W), $distance_1$ (P_1), $distance_2$ (P_2). Here W denotes the exact word appeared in the sentence. P_1 indicates distance (in terms of words) from the first drug name [9, 28]. This value will be zero for the first targeted drug name. P_2 is similar to P_1 but considers distance from the second targeted drug name. This way a word $w \in D^1 \times D^2 \times D^3$, where D^i is the dictionary for i^{th} local features. This feature layer constitutes the first layer for all models.

Embedding Layer

In embedding layer, each discrete feature is mapped to a real-valued vector representation using a lookup or embedding matrix. Let us say M^i is an embedding matrix for i^{th} feature. Here each column of M^i is a vector for the value in i^{th} feature. Mapping can be done by taking product of one hot vector of feature value with its embedding matrix [9]. Suppose $a_j^{(i)}$ is the one hot vector for j^{th} feature value of i^{th} feature then embedding layer will output:

$$f_j^{(i)} = M^i.a_j^{(i)} \tag{1}$$

$$x^i = f_1^{(i)} \oplus f_2^{(i)} \oplus f_3^{(i)} \tag{2}$$

Here $\oplus$ is concatenation operation so $x^i \in \mathbb{R}^{(n_1 + \ldots. n_3)}$ is feature vector for i^{th} word in sentence and n_k is dimension of k^{th} feature. Pre-trained word vectors are used for word embedding matrix and other feature matrices are initialized with random values.

Bi-LSTM Layer

Recurrent neural network is a powerful model for modeling sequential data [21]. It is a network with loop, allowing information to persist throughout the sequence. However it may suffer with vanishing or exploding gradient problems [10, 22] for instances of longer sentence. LSTM aims to overcome this problem by using gate and memory mechanism. 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 final output of LSTM will be calculated on the basis of new states of memory cell.

Consider $x^1x^2.....x^m$ is the sequence of feature vectors of a sentence, where m is the length of sentence and $x^t \in \mathbb{R}^d$

is a vector obtained by concatenating all feature vector of t^{th} word. Let $h_l^{(t-1)}$ and $c_l^{(t-1)}$ be previous hidden and cell states of LSTM layer (LSTM$_l$) respectively, then current hidden state ($h_l^{(t)}$), cell state $c_l^{(t)}$ and output of Bi-LSTM ($z^{(t)}$) can be computed as

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

where σ is sigmoid activation function, $*$ is an element wise product, $U_l^{(i)}$, $U_l^{(f)}$, $U_l^{(o)}$, $U_l^{(g)} \in \mathbb{R}^{N \times d}$, $W_l^{(i)}$, $W_l^{(o)}$, $W_l^{(f)}$, $W_l^{(g)} \in \mathbb{R}^{N \times N}$, b_l^i, b_l^f, b_l^o, $b_l^g \in \mathbb{R}^N$, $h_l^{(0)}$, $c_l^{(0)} \in \mathbb{R}^N$ are learning parameters for LSTM$_l$. Here d is dimension of input feature vector and N is hidden layer size. $h_l^{(t)}$ is output of LSTM$_l$ at time step t. We compute $h_r^{(t)}$ in similar manner as $h_l^{(t)}$ by reversing the words of sentence. A separate LSTM$_r$ is used for this calculation. The final output for t^{th} word by Bi-LSTM would be:

$$z^{(t)} = (h_l^{(t)} \oplus h_r^{(t)}) \tag{3}$$

Pooling Layer

The objective of pooling layer is to get a fixed length features from a variable length word features. We experiment with two different kinds of pooling schemes:

(A) Max Pooling

Max pooling takes one optimal over complete sequence. As Bi-LSTM accumulates information in both forward and backward directions, each node is assumed to have information of complete sentence. *Max* pooling takes the maximum over sentence assuming all important and relevant information are accumulated in that position. Let $z^1 z^2 ... z^m$ ($z^i \in \mathbb{R}^N$) be the sequence of vectors obtained after concatenating forward and backward LSTM output of each word then:

$$z = \max_{1 \le i \le (m)} [z^i], \tag{4}$$

where $z \in \mathbb{R}^N$ is dimension wise max of entire z^i's.

(B) Attentive Pooling

Max pooling may fail to perform well when important clues for the DDIs are present in different clauses or lie faraway in the sentence. Consider an example belonging to *mechanism* class: *Preliminary evidence suggests that cimetidine$_{Drug}$ inhibits mebendazole$_{Drug}$ metabolism and may result in an increase in plasma concentrations of mebendazole$_{Drug}$*. The first clue *inhibits mebendazole metabolism* indicates interaction class likely to be *effect* but the second clue *increase in plasma concentration* makes it belong to the *mechanism* class. Taking one optimal over the complete sentence may incorrectly classify this instance. To overcome this issue we use *attentive* pooling which takes optimal based on weighted linear combination of feature vectors. Weights of the feature vectors are computed using attention mechanism which assign weights based on importance of that features [1, 39, 41]. The attention mechanism produces a vector α of size equal to length of sentence. The values in this vector are the weights we would assign to each word feature vectors. Weighted linear combination of Bi-LSTM outputs and attention weights are the output of attentive pooling layer. Let $Z \in \mathbb{R}^{N \times m}$ be the matrix of outputs obtained by Bi-LSTM then, output of attentive pooling would be:

$$\begin{aligned}
H &= tanh(Z) \\
\alpha &= Softmax(w^{aT} H) \\
z &= \alpha Z^T,
\end{aligned} \tag{5}$$

where $w^a \in \mathbb{R}^N$ is the learning parameter, $\alpha \in \mathbb{R}^m$ is attention weights and $z \in \mathbb{R}^N$ would be the output of attentive pooling layer. Important thing to notice here is that α would be different for every sentence, i.e., indicating relevant context words may appear in distinct positions in different sentences.

Fully Connected and Softmax

Output of pooling layer would be a fixed length vector. This vector undergoes non-linear transformation by applying *tanh* activation and which is then fed to fully connected neural layer. In fully connected layer we maintain the number of node equals to number of class.

$$\begin{aligned}
h^3 &= tanh(h^2) \\
p(y|x) &= Softmax(h^{3T} W^o + b^o)
\end{aligned} \tag{6}$$

Here h^2 would be the output of pooling layer, $W^o \in \mathbb{R}^{N \times C}$, $b^o \in \mathbb{R}^C$ are parameters of fully connected neural network and C is number of class in our model. To make classification, we use *softmax* function in the output of fully connected layer. *Softmax* will give normalized probability score for each class.

Training and Implementation

All three models use cross entropy loss function for training the entire network. Adam's technique [19] is used for optimization. We use batch size of 200 for training each model. Implementation[6] is done in python language using *Tensorflow*[7] package.

2.1 B-LSTM Model

B-LSTM is similar to the model proposed in [20, 28] for DDI extraction and clinical relation extraction tasks. Here we use Bi-LSTM in place of convolution neural network used in [20, 28]. As shown in figure 1a, this model apply *Max* pooling on the output of Bi-LSTM to get optimal fixed length features. Max pooling is obtained from Equation 4 for every instance. These features are then fed to fully connected neural layer followed by *softmax* layer to obtain final classification.

2.2 AB-LSTM Model

Figure 1b is the graphical representation of *AB-LSTM* model. In this case we apply *attentive* pooling on Bi-LSTM output. Attention weights are obtained from Equation 5 for

[6]Code for reprodocing the results is available at https://github.com/sunilitggu/DDI-extraction-through-LSTM
[7]https://www.tensorflow.org

each sentence. Output of attentive pooling layer was used as features and passed to fully connected and *softmax* layers to obtain final classification.

2.3 Joint AB-LSTM Model

The objective of using *Joint AB-LSTM* is to take the advantage of both *max* and attentive pooling techniques. As shown in figure 1c *Joint AB-LSTM* model uses two separate modules each with a Bi-LSTM network. Both Bi-LSTM take same feature vectors as input and produce output for every word in the sentence. We applied *Max* pooling on the first and *attentive* pooling on the second Bi-LSTM layer to get features from both the modules. Concatenation of both features are used for classification using fully connected and *softmax* layers.

3. DATASET DESCRIPTION

We obtained the dataset from the shared challenge SemEval-2013 Task-9 [29, 15]. This dataset contains annotated sentences from two sources, Medline abstracts and DrugBank database. MedLine contains biomedical research articles and DrugBank contains manually entered texts collected from various sources and verified by accredited experts. The dataset is annotated with following four kinds of interactions:

Advice: The text states an opinion or recommendation related to the simultaneous use of the two drugs, e.g."*alpha-blockers* should not be combined with *uroxatral*".

Effect : The sentence notes the effect of the drug-drug interaction or pharmacodynamic mechanism of interaction. For example "*Warfarin* users who initiated *fluoxetine* had an increased risk of hospitalization for gastrointestinal bleeding".

Mechanism : The sentence describes a pharmacokinetic mechanism, as in "*Paroxetine* reduce the plasma concentration of *endoxifen* by about 20%".

Int : The text mentions a drug interaction without providing any other information. For example, "This is typical of the interaction of *meperidine* and *MAOIs.*".

Dataset provides the training and test instances as sentences. If a sentence has more than two drug names, all possible pairs of drugs in the sentence have been separately annotated. This way single sentence having multiple drug names leads to separate instances of drug pairs and corresponding interaction. Statistics of the dataset is shown in Table 2. Finally the objective of model is to identify exact class of interaction (one of the four types) or no interaction. In our subsequent discussions, we refer the task as *DDI classification.*

3.1 Pre-processing

The following pre-processing is done in the dataset before using it in our model:

- *Genia tagger*[8] is used for tokenization. All digits are normalized by replacing them with a special token *DG* and all letters are changed to lowercase.

- The two targeted drug names are replaced with *DRUG-A* and *DRUG-B* respectively, and other drug names in the same sentence are replaced with *DRUG-N*. Earlier studies [26, 20] have reported that this step helps in improving the model performance.

[8]http://www.nactem.ac.uk/GENIA/tagger/

Corpus	Training Set		Test Set	
	Before	After	Before	After
Documents	714	714	191	191
Pairs	27774	16495	5716	4025
Positive DDIs	4018	3844	979	979
Negative DDIs	23756	12651	4737	3046
Mechanism	1318	1264	302	302
Effect	1685	1620	360	360
Advice	826	820	221	221
Int	189	140	96	96

Table 2: Statistics of DDI classification dataset before and after negative instance filtering

3.2 Negative Instance Filtering

Consideration of all possible pairs of drug names in a sentence as separate instances for our model made the resultant dataset very imbalanced. We have 1:5.9 ratio of positive to negative instances. However, one can follow some strategies to remove negative instances. Earlier studies [40, 18, 20] have shown positive impact of negative instance filtering. We filter negative samples based on the following rules:

1. If both targeted drug mentions have the same name, remove the corresponding instance. Assumption behind this rule is drug doesn't interact with itself. We use string matching on both drug names to identify such cases.

2. Remove the instance, if one drug is a kind of or a special case of the other one in the corresponding sentence. To identify such cases, we use regular expression by observing patterns in the dataset. "*DRUG-A (DRUG-B)*", "*DRUG-A such as DRUG-B*" are examples of such patterns.

3. If both target drugs appear in same coordinate structure, then remove the corresponding instance. We use several regular expressions based on observing the patterns in training set to filter out such instances. Examples of one such pattern is "*DRUG-A , (DRUG-N ,)*$^+$*DRUG-B*".

Similar to [20], our rules have not eliminated any positive instances from the test set. However, 144 positive instances (54 *Mechanism*, 65 *Effects*, 49 *Int* and 6 *Advice*) are removed from the training set. Table 2 summarizes statistics of dataset before and after filtering.

Models	***Dropout***	l_2 ***regu.***
CNN*[1]	0.7	0.1
B-LSTM	0.7	0.001
AB-LSTM	0.7	0.0001
Joint AB-LSTM	1.0	0.0001

Table 3: Values of different regularization parameters used in the three models.

4. EXPERIMENT DESIGN

We train and evaluate three LSTM models on the DDI classification task. As mentioned earlier, if a sentence contains more than two drug names then all possible pairs with the sentence constitute separate instances/samples. We use the same evaluation scheme as used in the challenge [29].

Models	**Before**			**After**			$\triangle$
	Precision	**Recall**	**F1 Score**	**Precision**	**Recall**	**F1 Score**	
CNN*	62.11	63.63	62.86	66.14	60.87	63.40	0.86
B-LSTM	69.07	64.35	66.63	70.62	66.80	68.66	3.05
AB-LSTM	70.75	60.06	64.97	73.34	62.41	67.43	3.79
Joint AB-LSTM	67.77	66.80	67.28	74.47	64.96	69.39	3.14

Table 4: Performance improvement after filtering negative instance from the dataset. $\triangle$ indicates percentage of relative improvement in F1 score.

Class	**B-LSTM**			**AB-LSTM**			**Joint AB-LSTM**		
	P	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
Advice	$\downarrow$	$\uparrow$	$\uparrow$	$\downarrow$	$\uparrow$	$\downarrow$	$\downarrow$	$\uparrow$	$\uparrow$
Mechanism	$\uparrow$	$\downarrow$	$\uparrow$	$\uparrow$	$\sim$	$\uparrow$	$\uparrow$	$\downarrow$	$\uparrow$
Effect	$\sim$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\downarrow$	$\uparrow$
Int	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\downarrow$	$\sim$	$\downarrow$

Table 5: Relative change in class-wise performance measures after negative filtering. $\downarrow$ indicates value got reduced after filtering step compared to the value obtained using the complete data. Similarly $\uparrow$ indicates increase in value and $\sim$ indicates value remain almost same after filtering step.

4.1 Hyper-parameters

As there is no separate development or validation set available, we divided the original training dataset into two parts, 80% as training and rest 20% as validation sets. Hyper-parameters are tuned using this validation set. Hidden layer size in *B-LSTM* and *AB-LSTM* are kept as 200, and 150 for *Joint AB-LSTM*. Pre-trained word embedding of 100 dimensions and distance embedding of 10 dimensions are used in all three LSTM models. Word embedding are obtained using *GloVe* tool [23] on a corpus of PubMed open source articles [35]. We used both l_2 regularization and dropout [33] techniques for regularization. We applied dropout only on the output of the pooling layers. Different values of the regularization parameters are shown in the Table 3. Next to decide about the early-stopping criteria (i.e. number of epochs neural network models will be trained), we again split the original training dataset into two parts containing 95% and 5% of all samples. Now we trained the model on 95% set using above selected hyper-parameters and selected the epoch giving the best result on the left out 5% of the original training data. Trained model on the 95% set is used on the independent test set and corresponding results are discussed in the respective sections.

4.2 Baseline Methods for comparison

We compare performance of the three proposed models with several baseline methods. Approaches based on conventional features, kernel methods and on neural networks are included as baseline methods. Below we briefly describe about the baseline methods, where superscript one ($*^1$) indicates one stage and superscript two ($*^2$) indicates two stages methods:

Linear Methods: In this class of methods, a linear classifier is used to identify the correct class of interaction for each instance. All instances are represented by a vector of manually designed features. **UTurku**^{1} used Turku event extraction system (TEES) [2] for drug interaction extraction. The major features used by TEES comes from dependency parsing and domain dependent resources such as MetaMap. **UWM-TRIADS**^{2} [26] and **Kim**^{2} [18] are two stage methods. In both the stages SVM with contextual, lexical, semantic and tree structured features was used.

Kernel Methods: Kernel methods are powerful techniques for utilizing graph based features in any natural language processing task. **WBI-DDI**^{2} and **FBK irst**^{2} are two stage methods [7, 36] for DDI classification. First stage of both models used different kernels methods for utilizing syntax tree and dependency tree features. In the second stage WBI-DDI^2 used TEES and FBK irst^2 used SVM with non-linear kernel for classification. **NIL UCM**^{1} used multi-class SVM as kernel methods in one stage framework.

Neural Network Methods: Neural network or deep learning methods use latent features in place of manually designed features. This class of algorithms use neural network to encode word level features for generation of sentence level features. Final classification happens with sentence level features. **SCNN**^{1,2} [40] used convolution neural network with max pooling layer to learn higher level discriminative features over entire sentence. SCNN also utilized PoS tags and dependency tree based features apart from latent word embedding and distance embedding features. **MV-RNN**^{1} [34] is also a neural network based model. In particular, MV-RNN used recursive neural network [32] for learning embedding of sentence or part of sentence recursively and thus obtained final vector is used for classification. To analyze the performance of RNN and CNN based models, we implemented CNN with max pooling technique for this task. We implemented the similar model, as discussed in [20], due to unavailability of the source code and referred to it as CNN*^{1}. Initially, the results were obtained with similar parameter settings as discussed in [20] but we could not obtain the similar results, as mentioned in that research article. So we tuned the model hyper-parameters and used them in subsequent analysis. The difference in performance could be due to change in training dataset after negative instance filtering as well as due to use of different pre-trained embedding vectors.

5. RESULTS AND DISCUSSIONS

5.1 Effect of Negative Instance Filtering

Following our filtering rules, a large number of negative instances along with few positive instances are removed from the dataset. However none of the positive instances from *test set* are removed. There was a significant change in imbalance ratios of interaction classes due to the filtering step. For example, imbalance ratio changed from 1:32.6 to 1:19.1 for the interaction class *Advice* and from 1:146 to 1:116.8 for the interaction class *Int*. Table 4 shows performance of different models on *test set* while training is done using either complete or filtered dataset. All models when using filtered dataset gave improved performance in terms of F1 score. All three LSTM models obtained more than 3% of relative improvement with the use of filtered dataset. Performance improvement can be attributed to two factors: cleaner data and reduction of imbalance. Both these factors are result of the data filtering step.

We next look at class-wise performance of all LSTM models to see how change in imbalance ratio affected the model performance at individual class level. For this we compared the relative changes in *precision*, *recall* and *F1 score*, when models trained on the filtered dataset vs the complete dataset. The results are summarized in the Table 5. The overall F1 score for individual class increases in almost all the cases. For *Advice* class, all three models have similar effect on precision and recall. All three LSTM models have obtained better precision but recall got reduced or remained almost same for the *Mechanism* class. For the *Effect* class, *Joint AB-LSTM* model obtained reduced recall, but all the models have obtained better or similar precision values after filtering. The performance pattern of the *B-LSTM* and *AB-LSTM* models were same for the *Int* class. It is very difficult to reason out the effect of filtering step due to inconsistent effect of it on the three models.

Models	***Precision***	***Recall***	***F1 Score***
SCNN^2 [40]	68.5	61.0	64.5
CNN*^{1}	62.11	63.63	62.86
B-LSTM^1	69.07	64.35	66.63
AB-LSTM^1	70.75	60.06	64.97
Joint AB-LSTM^1	67.77	66.80	67.28

Table 6: Performance comparison of proposed models with existing models on complete dataset, i.e. without using negative instance filtering, for DDI classification task.

5.2 Comparison with Baseline Methods

Comparison based on the complete dataset: First we compare the three proposed models with other existing models on the complete dataset, i.e., without using the filtering step. Best results out of five runs are shown in Table 6. Here, we have included only those methods which explicitly mention about the similar steps and report corresponding results. *B-LSTM* and *Joint AB-LSTM* models have outperformed all other models. We performed t-test (one-sided null hypothesis: mean performance of model 1 is not less than that of model 2) on mean performance of model pairs. Mean performance of a model was calculated as the average of F1 score obtained in five different runs of the model. *B-LSTM* and *Joint AB-LSTM* models outperformed CNN*^{1} model at the significance level of 0.05. In particular p-values of CNN*^{1} vs *B-LSTM*, *AB-LSTM*, and *Joint AB-LSTM* comparisons were $0.0007, 0.04, 0.01$ respectively. Among the LSTM models, no single model outperforms other models at the above chosen significance level.

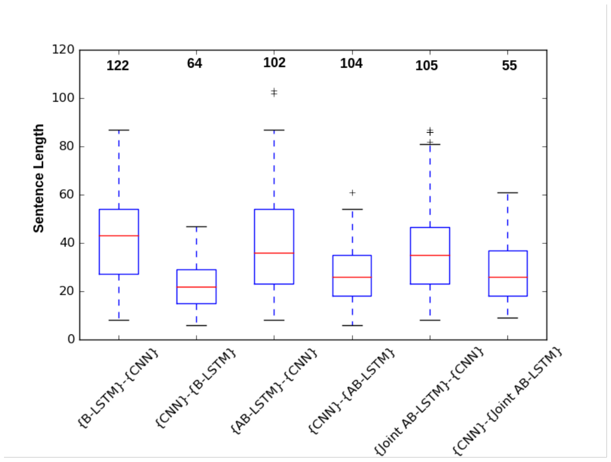
Method	**Precision**	**Recall**	**F Score**
UTurku^1 [2]	73.2	49.9	59.4
UWM-TRIADS^2 [26]	43.9	50.5	47.0
Kim^2 [18]	-	-	67.0
NIL UCM^1 [4]	53.5	50.1	51.7
WBI-DDI^2 [36]	64.2	57.9	60.9
FBK irst^2 [8]	64.6	65.6	65.1
SCNN^1 [40]	69.1	65.1	67.0
SCNN^2 [40]	72.5	65.1	68.6
MV-RNN^1 [34]	52.0	48.0	50.0
CNN$^{\star 1}$	66.14	60.87	63.40
B-LSTM^1	70.62	**66.80**	68.66
AB-LSTM^1	73.34	62.41	67.43
Joint AB-LSTM^1	**74.47**	64.96	**69.39**

Table 7: Performance comparison between the proposed methods and top-ranking approaches on the test data. Performance is measured based on precision, recall and F1 score. The highest scores are highlighted in bold.

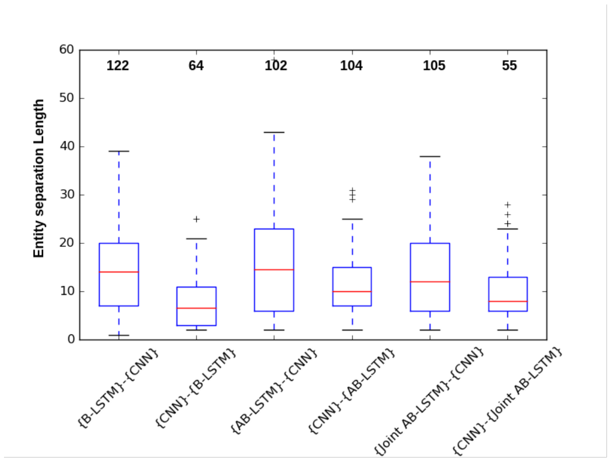
Comparison based on the filtered dataset: Table 7 provides a comparison of our models with previous approaches. *Joint AB-LSTM* model obtained the best F1 score of 69.39%. There is 3.6% of relative improvement in F1 score on comparison with the best performing method (Kim^2) among feature based linear and kernel methods. In comparison to SCNN^2 model, all three proposed models gave similar results for this task. SCNN^2, a convolution neural network based model, uses higher order grammatical features based on parts of speech and shortest dependency path apart from pre-trained word embedding and position embedding features. If we remove shortest dependency path features from SCNN^2 model, its performance decreases to 63.8% from 68.6%. On the other hand, apart from *Joint AB-LSTM*, the other two models also gave relatively much better performance than SCNN^2 with only using simple features.

Among the three proposed models, *Joint AB-LSTM* always performs better than the other two methods. Based on the McNemar test *B-LSTM*, *AB-LSTM*, and *Joint AB-LSTM* models outperformed CNN*^{1} model with p-values of 0.0005, 0.001 and 8.4×10^{-9} respectively. As results indicate all LSTM based models outperform CNN*^{1} model significantly. Among the LSTM models, *Joint AB-LSTM* and *B-LSTM* models outperform *AB-LSTM* with p-values of 0.005 and 0.03 respectively on the DDI classification task. Further, the McNemar test suggests that there is no significant difference in performance of the two models on the DDI-classification task.

Class wise Performance Analysis: We compare class wise performance of the proposed models with existing models in Table 8. We observe that no single model outperforms others for all classes. *FBK irst* obtained the best performance for *Int* class. *B-LSTM*, *AB-LSTM* and *Joint AB-LSTM* obtained the best performance for *Mechanism*, *Effect* and *Advice* classes respectively. However, *Joint AB-LSTM* outperformed all other models on aggregate performance measure using macro-average F1 score. All models finds it easier to detect *Advice* interaction types compared to the instances of the other three interaction types. Simi-



(a) Box plot for sentence length (number of words)



(b) Box plot for entity separation (number of words)

Figure 2: Boxplot for sentence length and entity separation length in instances. Here {X}-{Y} represent instances correctly predicted by X but not by Y.

Models	*Advice*	*Mechanism*	*Effect*	*Int*	*MAVG*
UTurku [2]	63.0	58.2	60.0	50.7	58.7
UWM-TRIADS[26]	53.2	44.6	44.9	42.1	47.2
Kim [18]	72.5	69.3	66.2	48.3	64.1
FBK irst [8]	69.2	67.9	62.8	**54.7**	64.8
NIL_UCM [4]	61.3	51.5	48.9	42.7	53.5
WBI-DDI[36]	63.2	61.8	61.1	51.1	59.7
MV-RNN[1] [34]	57.0	46.0	49.0	49.0	50.25
CNN*[1]	69.04	63.98	63.00	45.07	60.27
B-LSTM	75.92	**72.66**	65.15	47.40	65.28
AB-LSTM	69.68	68.06	**68.28**	54.16	65.04
Joint AB-LSTM	**80.26**	72.26	65.46	44.11	**65.52**

Table 8: Performance comparison between the proposed methods and top-ranking approaches on the test data for DDI classification. Performance are measured through F1-Score for each class and Macro Average (MAVG). The highest scores are highlighted in bold.

Models	*Precision*	*Recall*	*F Score*
Joint AB-LSTM	74.47	64.96	69.39
Joint AB-LSTM - { P }	70.62	66.80	68.66
Joint AB-LSTM - { (P + X) }	71.21	61.89	66.22

Table 9: Contribution of each feature in **Joint AB-LSTM** model. Here P refers random position embedding for both P_1 and P_2, and X refers pre-trained word vector embedding

larly all models find it most difficult to detect *Int* interaction types. The worse performance on the *Int* class can be attributed to insufficient training data. *Effect* interaction class was found to be the second most difficult class to detect by most of the models compared in this analysis.

5.3 Feature Analysis

In order to validate the importance of each feature, we further analyzed the performance of *Joint AB-LSTM* model by removing feature types one by one. It can be observed from Table 9 that the use of pre-trained word embedding is important feature. About 1.1% of relative decrement is observed if the model does not use position embedding. On the other hand, removal of position embedding as well as use of random vectors in place of pre-trained word vectors lead to 4.6% of relative decrements in the model's performance. This analysis clearly indicates the importance of word and position embedding features.

5.4 LSTM vs CNN models

Earlier we discussed that intuitively it seems LSTM models are likely to perform better for longer sentences compared to CNN model. We performed an analysis on our results to see whether that is the case or not. For this we analyze length as well as entity separation between two targeted drugs of all those sentences which were predicted correctly by one model but incorrectly by the other. Figures 2a and 2b show the box plots for sentence length and separation length between targeted drugs. Here {X}-{Y} represent instances correctly predicted by X but not by Y. In all of these cases length represent number of words and numbers present at top of the boxes represent number of instances present in that category. From the figures we can observe that the proposed LSTM models performed better than CNN in both scenarios of instances having longer sentences and having larger entity separation length.

5.5 Error Analysis

Apart from imbalance issue, we try to find out whether there is any other factor which is adversely affecting performance of the models. For this we look at average sentence lengths of correctly and incorrectly classified instances by each of the four models for the DDI classification task (Table 10). We observe that, average sentence length and entity separation length for incorrectly classified instances are always high compared to the correctly classified sentences. Another aspect of incorrectly predicted instances was, presence of multiple drug entities in many such instances. Repetitive drug entities is more likely to behave like a noise, which may cause neural models to lose relevant information from other words likely to be contextually important. Hence a better strategy is required to deal with such cases. Considering limited context along with removal of repetitive entities can be one way to deal with extra large sentences.

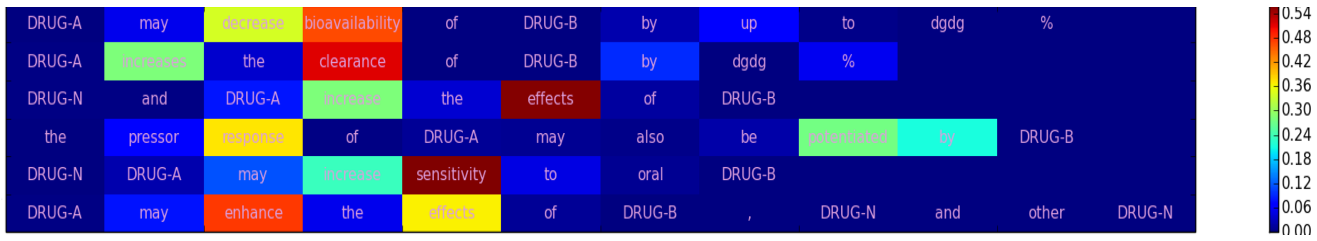


Figure 3: Heat map of attention weights indicating importance of relevant words.

Model	Sentence Length		Entity Separation	
	True	False	True	False
CNN*	26.19$_{(13.38)}$	42.51$_{(21.00)}$	11.24$_{(9.39)}$	15.17$_{(12.21)}$
B-LSTM	29.12$_{(16.19)}$	37.34$_{(22.49)}$	12.15$_{(9.55)}$	13.92$_{(13.03)}$
AB-LSTM	28.52$_{(17.57)}$	37.54$_{(20.19)}$	11.12$_{(10.11)}$	14.26$_{(12.65)}$
Joint AB-LSTM	28.52$_{(14.50)}$	39.97$_{(21.70)}$	12.81$_{(9.49)}$	14.19$_{(11.86)}$

Table 10: Mean and standard deviations (in subscript) of length of sentence for True Positive and False Negative instance in DDI classification task

5.6 Visual Analysis

In order to confirm that the model is able to learn attention weights based on importance of words, we visualize attention weights of some of the sentences after training *Joint AB-LSTM*. Figure 3 is the heat map of attention weights for 6 instances of test set. Here every line is a sentence with two targeted drug names replaced with special tokens *DRUG-A* and *DRUG-B* and darkness in red color indicate heedfulness. Figure shows that our model can select important words based on the task. For example in the sentence *"DRUG-A may enhance the effects of DRUG-B , DRUG-N and other DRUG-N"*, model is able to assign high weights to *"may enhance the effects"* very well. Similarly, in the sentence *"DRUG-N and DRUG-A increase the effects of DRUG-B"*, the model is assigning high weights to the words *increase* and *effect*.

6. CONCLUSION

In this work we proposed three LSTM based models *B-LSTM*, *AB-LSTM* and *Joint AB-LSTM* for DDI classification task. All the three models use simple word and distance embedding as features and learn higher level feature representation using Bi-LSTM network. Two of the proposed models also utilized neural attention mechanism to get higher level feature representation. To the best of our knowledge, it is the first study to use LSTM and attention mechanism for DDI extraction task. Performance of all three models are compared with the existing methods on SemEval-2013 DDI extraction dataset. *Joint AB-LSTM* model achieves state-of-the-art for the DDI classification task. Performance of the other two models, *B-LSTM* and *AB-LSTM*, are also found to be competitive. Analysis of the results indicates the following important points: imbalance and noise adversely affect all models, *Advice* interaction class is easiest to predict, repetitive entries of other drug names negatively effect all models, and models are likely to make incorrect classification for longer sentences.

7. ADDITIONAL AUTHORS

8. REFERENCES

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[2] J. Björne, S. Kaewphan, and T. Salakoski. Uturku: drug named entity recognition and drug-drug interaction extraction using svm classification and domain knowledge. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval 2013)*, pages 651–659, 2013.

[3] T. Bobic, J. Fluck, and M. Hofmann-Apitius. Scai: Extracting drug-drug interactions using a rich feature vector. *Atlanta, Georgia, USA*, page 675, 2013.

[4] B. Bokharaeian and A. DIAZ. NIL_UCM: Extracting Drug-Drug interactions from text through combination of sequence and tree kernels. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 644–650, Atlanta, Georgia, USA, June 2013. Association for Computational Linguistics.

[5] À. Bravo, J. Piñero, N. Queralt-Rosinach, M. Rautschka, and L. I. Furlong. Extraction of relations between genes and diseases from text and large-scale data analysis: implications for translational research. *BMC bioinformatics*, 16(1):1, 2015.

[6] R. Businaro. Why we need an efficient and careful pharmacovigilance. *J Pharmacovigilance*, 1(4):1000e110, 2013.

[7] M. F. M. Chowdhury and A. Lavelli. Exploiting the scope of negations and heterogeneous features for relation extraction: A case study for drug-drug interaction extraction. In *HLT-NAACL*, pages 765–771, 2013.

[8] M. F. M. Chowdhury and A. Lavelli. Fbk-irst: a multi-phase kernel based approach for drug-drug interaction detection and classification that exploits linguistic information. *Atlanta, Georgia, USA*, 351:53, 2013.

[9] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, Nov. 2011.

[10] A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[11] H. Gurulingappa, A. Mateen-Rajpu, and L. Toldo. Extraction of potential adverse drug events from medical case reports. *Journal of biomedical semantics*, 3(1):1, 2012.

[12] H. Gurulingappa, L. Toldo, A. M. Rajput, J. A. Kors, A. Taweel, and Y. Tayrouz. Automatic detection of adverse events to predict drug label changes using text and data mining techniques. *Pharmacoepidemiology and drug safety*, 22(11):1189–1194, 2013.

[13] N. D. Hailu, L. E. Hunter, and K. B. Cohen. Ucolorado som: extraction of drug-drug interactions from biomedical text using knowledge-rich and knowledge-poor features. *Atlanta, Georgia, USA*, page 684, 2013.

[14] R. Harpaz, A. Callahan, S. Tamang, Y. Low, D. Odgers, S. Finlayson, K. Jung, P. LePendu, and N. H. Shah. Text mining for adverse drug events: the promise, challenges, and state of the art. *Drug safety*, 37(10):777–790, 2014.

[15] M. Herrero-Zazo, I. Segura-Bedmar, P. MartÃ■nez, and T. Declerck. The {DDI} corpus: An annotated corpus with pharmacological substances and drugâĂŞdrug interactions. *Journal of Biomedical Informatics*, 46(5):914 – 920, 2013.

[16] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997.

[17] S. Hutson. Researchers sound alarm on'silent'drug interactions. *Nature medicine*, 17(1):6–6, 2011.

[18] S. Kim, H. Liu, L. Yeganova, and W. J. Wilbur. Extracting drug–drug interactions from literature using a rich feature-based linear kernel approach. *Journal of biomedical informatics*, 55:23–30, 2015.

[19] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[20] S. Liu, B. Tang, Q. Chen, and X. Wang. Drug-drug interaction extraction via convolutional neural networks. *Computational and mathematical methods in medicine*, 2016, 2016.

[21] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048, 2010.

[22] R. Pascanu, T. Mikolov, and Y. Bengio. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063, 2012.

[23] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, 2014.

[24] P. Przybyła, M. Shardlow, S. Aubin, R. Bossy, R. E. de Castilho, S. Piperidis, J. McNaught, and S. Ananiadou. Text mining resources for the life sciences. *Database*, 2016:baw145, 2016.

[25] L. Qian and G. Zhou. Tree kernel-based proteinâĂŞprotein interaction extraction from biomedical literature. *Journal of Biomedical Informatics*, 45(3):535 – 543, 2012.

[26] M. Rastegar-Mojarad, R. D. Boyce, and R. Prasad. Uwm-triads: classifying drug-drug interactions with two-stage svm and post-processing. In *Proceedings of the 7th International Workshop on Semantic Evaluation*, pages 667–674, 2013.

[27] B. Rink, S. Harabagiu, and K. Roberts. Automatic extraction of relations between medical concepts in clinical texts. *Journal of the American Medical Informatics Association*, 18(5):594–600, 2011.

[28] S. K. Sahu, A. Anand, K. Oruganty, and N. Gattu. Relation extraction from clinical texts using domain invariant convolutional neural network. In *Proceedings of the 15th Workshop on Biomedical Natural Language Processing*, pages 206–215, 2016.

[29] I. Segura-Bedmar, P. Martínez, and M. Herrero Zazo. Semeval-2013 task 9 : Extraction of drug-drug interactions from biomedical texts (ddiextraction 2013). In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 341–350, Atlanta, Georgia, USA, June 2013. Association for Computational Linguistics.

[30] I. Segura Bedmar, P. Martinez, and D. Sánchez Cisneros. The 1st ddiextraction-2011 challenge task: Extraction of drug-drug interactions from biomedical texts. 2011.

[31] A. Singhal, R. Leaman, N. Catlett, T. Lemberger, J. McEntyre, S. Polson, I. Xenarios, C. Arighi, and Z. Lu. Pressing needs of biomedical text mining in biocuration and beyond: opportunities and challenges. *Database*, 2016:baw161, 2016.

[32] R. Socher, C. C.-Y. Lin., C. D. Manning, and A. Y. Ng. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *ICML*, 2011.

[33] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[34] V. Suárez-Paniagua and I. Segura-Bedmar. Extraction of drug-drug interactions by recursive matrix-vector spaces. In *6thInternational Workshop on Combinations of Intelligent Methods and Applications (CIMA 2016)*, page 65, 2016.

[35] M. TH, S. Sahu, and A. Anand. Evaluating distributed word representations for capturing semantics of biomedical concepts. In *Proceedings of BioNLP 15*, pages 158–163, Beijing, China, July 2015. Association for Computational Linguistics.

[36] P. Thomas, M. Neves, T. Rocktäschel, and U. Leser. Wbi-ddi: drug-drug interaction extraction using majority voting. In *Second Joint Conference on Lexical and Computational Semantics (* SEM)*, volume 2, pages 628–635, 2013.

[37] R. Xu and Q. Wang. Large-scale automatic extraction of side effects associated with targeted anticancer drugs from full-text oncological articles. *Journal of biomedical informatics*, 55:64–72, 2015.

[38] M. Yang, M. Kiang, and W. Shang. Filtering big data from social media–building an early warning system for adverse drug reactions. *Journal of biomedical informatics*, 54:230–240, 2015.

[39] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. Hierarchical attention networks for document classification.

[40] Z. Zhao, Z. Yang, L. Luo, H. Lin, and J. Wang. Drug drug interaction extraction from biomedical literature using syntax convolutional neural network. *Bioinformatics*, page btw486, 2016.

[41] P. Zhou, W. Shi, J. Tian, Z. Qi, B. Li, H. Hao, and B. Xu. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 207–212, Berlin, Germany, August 2016. Association for Computational Linguistics.