Incorporating User Generated Content for Drug Drug Interaction Extraction Based on Full Attention Mechanism

Bo Xu[†‡], Xiufeng Shi[†‡], Zhehuan Zhao[* † ‡], Wei Zheng[§¶], Hongfei Lin[§], Zhihao Yang[§], Jian Wang[§], and Feng Xia[†‡]
[†]School of Software, Dalian University of Technology, China
[‡]Key Laboratory for Ubiquitous Network and Service Software of Liaoning, Dalian University of Technology, China
[§]College of Computer Science and Technology, Dalian University of Technology, China
[¶]College of Software, Dalian JiaoTong University, China

***Abstract*—When a patient takes multiple medications simultaneously under the treatment, it is vital for the doctor to fully comprehend all interactions between drugs in the prescription. Drug drug interaction (DDI) extraction aims to obtain interactions between drugs from biomedical literature automatically. Nowadays, researchers apply artificial intelligence and natural language processing techniques to perform DDI extraction task. Existing DDI extraction methods have utilized some kinds of external resources such as biomedical databases or ontologies to offer more knowledge and improve the performance. However, these kinds of external resources are delayed because of the hardship of updating. User generated content (UGC) is another sort of external biomedical resource which is up-to-date and can be updated rapidly. We attempt to utilize UGC resource in our deep learning DDI extraction method to provide more fresh information. We propose a DDI extraction method that merges UGC information and contextual information together by a new attention mechanism called full-attention. We conducted a series of experiments on the DDI 2013 Evaluation dataset to evaluate our method. UGC-DDI outperforms the other state-of-the-art methods and achieves a competitive F-score of 0.712.**

***Index Terms*—drug drug interaction, drug safety, attention mechanism**

I. Introduction

Undoubtedly, it will cause a serious accident if a doctor miscalculates potential interactions between miscellaneous medications which are prescribed to a single patient. But it is very difficult for doctors to keep pace with frequently updated drug interaction knowledge. Traditionally, physicians acquire new published drug drug interactions (DDIs) from two auxiliary information sources: reading massive biomedical papers to learn DDIs between the lines or querying DDIs from human-maintained biomedical databases. Obviously, it is laborious, painful, and inefficient to read tons of academical papers. As for searching DDIs from biomedical databases, it seems to be feasible. But in the consideration of the quantity of the biomedical literature, it requires a lot of resources to update, maintain and revise a professional database manually. Apparently, both two of them are not idealistic manners to detect DDIs.

DDI extraction task aims at extracting DDIs from free text of the biomedical domain. As an illustration, in the following sentence:

> Because of its primary CNS effect, caution should be used when **EQUETRO** is taken with other centrally acting drugs and **alcohol**.

DDI extraction methods attempt to make the prediction of the interaction type, (such as adverse, recommendation, or no interaction) between two recognized entities (namely **EQUETRO** and **alcohol**, highlighted in bold text).

Much related work of DDI extraction has been done such as FBK-irst [1], WBI-DDI [2], UWM-TRIADS [3], Uturku [4], and SCNN [5]. These methods utilize artificial intelligence (AI) and natural language processing (NLP) techniques to extract DDIs. Some of them introduce external resources to improve the performance of DDI extraction. WBI-DDI utilizes WordNet lexical database to extract token features. Uturku sets the presences of words in the DrugBank as a part of features in the feature engineering step. UWM-TRIADS uses a FDA Drug classification dictionary to recognize drugs in the same drug class during the post-processing phase. However, external resources used by these methods are from delayed information source such as manually-updated databases, ontologies or lexical tools. These external resources can not be updated timely to catch up with the latest information. Many experts are needed to update, review, and revise new items of these external resources.

Compared with these delayed external resources, online User Generated Content (UGC) [6] can update more frequently and timely. As its name suggests, UGC is created by users of online platforms, forums and social media websites. Con-

This work was supported in part by the Natural Science Foundation of China under Grant 61502071 and Grant 61572094 and in part by the Fundamental Research Funds for the Central Universities under Grant DUT18RC(3)004.

* Corresponding Author: z.zhao@dlut.edu.cn

sidering the count of Internet users, the quantity of UGC is very massive. UGC is also a more active form of external resources compared with external resources mentioned above [7]. Billions of users post text, photos, and videos on the Internet to share their opinions, comments, and experiences every day. There is lots of information can be mined among these content. Unfortunately, there are still no research efforts that exploit the external resources with Deep learning approach for DDI extraction. To fill this gap, we take the first attempt to investigate the feasibility and advantage of utilizing the UGC in DDI extraction task.

To our knowledge, there is no existing method that utilizes UGC to assist DDI extraction. Hence, the way to merge UGC with the model is also a challenge. Recently, self-attention mechanism is a very hot topic in the NLP community. Experimental results of dozens of papers showed that it performed very well in a lot of NLP tasks such as machine translation [8] and natural language understanding [9]. Inspired by the self-attention mechanism, we merge UGC with the word embeddings in an attentive way called ***Full-Attention*** to capture the global dependency of every token-UGC document pair comprehensively. We represent UGC in the form of low-dimensional distributed document embedding vectors. In order to capture the global dependency of every token-UGC document pair comprehensively, we build interactions between every single token and every single UGC document by scaled dot product attention.

Based on the ideas we described above, we propose a brand new DDI extraction method named *UGC-DDI*. UGC-DDI merges UGC embeddings and word embeddings together by the full-attention firstly. Then the merged vectors are concatenated with the concept embeddings and offset embeddings to generate the final token representations. At last, token representations are fed into a deep learning based classifier to make predictions of DDI types. The experimental result shows UGC-DDI outperforms existing methods on DDI 2013 Evaluation dataset.

The rest of this paper is organized as follows. Section II presents the details of the proposed method. In Section III, we describe experimental settings and results. Finally, our conclusion is presented in Section IV.

II. Method

The pipeline of our DDI extraction method is a three-step procedure:

1) In the ***preprocessing*** step, we conduct several kinds of operations to trim and condense raw sentences from DDI 2013 Evaluation dataset.
2) In the ***token representation*** step, we merge UGC embeddings and word embeddings to get attention output vectors firstly, then concatenate full attention output vectors, concept embeddings, and offset embeddings together to form token representations.
3) In the ***classifier*** step, we extract latent features of sentences using a stacked two-layer encoder which consists of a Bidirectional Long Short Term Memory (Bi-LSTM) layer and a Transformer layer, then make predictions with a Softmax densely-connected neural network layer.

Fig. 1 illustrates the basic pipeline of the UGC-DDI method.

A. Preprocessing

At the preprocessing step, three kinds of operations are conducted: *Negative Instances Filtering*, *Drug Blinding*, and *Tokenization*.

There is no interaction between drugs in negative examples. In the training dataset of DDI 2013 Evaluation, the positive to negative instances ratio is $1:5.91$. Evidently, the dataset is extremely imbalanced. Imbalanced datasets have been proved that can cause a strong bad influence on the performance of the model [10]. And previous methods [5] and [11] have verified that negative instance filtering is a practical operation to relieve the effect of the imbalanced problem. So firstly, we filter out some negative training examples to balance dataset distribution.

We follow the same negative instance filtering policies of SCNN [5] which can be summarized into two rules:

> **Rule 1**: If two drugs in a drug pair refer to the same drug, this pair will be removed under the assumption that the same drug can not interact with itself. There are two cases to be considered: the first case is two drug names are the same and the second case is one drug is the abbreviation of the other drug.
>
> **Rule 2**: If two drugs in a drug pair are in coordinate relations, this pair will be removed since it is prone to be a false positive [12].

Then, we perform an action called *drug blinding* that replaces concrete drug entity names in sentences with special tokens. The purpose of drug blinding is to enhance the generalization of the model [13]. We replace two drug entity names in every drug pair from the dataset with token **DURG1** and **DRUG2**, and replace other drug entity names in the same sentence with token **DRUGN**.

After the drug blinding step, we tokenize sentences with *GENIA Tagger* [14]. Tokenization converts a sentence into a sequence of tokens. At this step, we replace numerical tokens with the general token **NUM** and remove all symbols, special non-ASCII characters, punctuation marks, and some selected meaningless stop words.

B. Token Representation

At this step, we generate different kinds of embeddings that contains diverse information firstly, then concatenate them together to final token representations.

1) Full Attention

In full attention phase, we employ full attention mechanism to merge UGC embeddings and word embeddings.

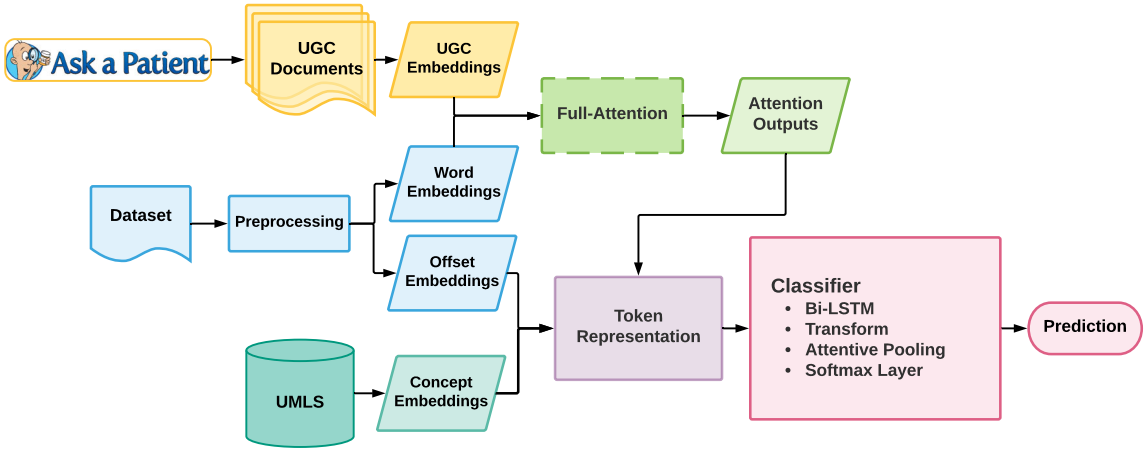

Fig. 1: The Pipeline of UGC-DDI method.

Given a sentence $S = \{t_1, t_2, \ldots, t_n\}$ from DDI 2013 Evaluation dataset which consists of n tokens, we transform this token sequence to corresponding word embeddings $emb_{\text{word}} = \{\boldsymbol{e_1}, \boldsymbol{e_2}, \ldots, \boldsymbol{e_n}\}$ by a pre-trained word embedding look-up table. Embedding vector $\boldsymbol{e_i} \in \mathbb{R}^{d_k}$ in the word embedding sequence contains contextual information of the token t_i.

Then we generate UGC embeddings with documents crawled from the website *Ask a Patient*. Ask a Patient is an online medicine ratings and reviewing forum where patients can share and compare their medication experiences on various aspects such as physical or mental feelings, dosage, durations, side effects, etc. Fig. 2 is a screenshot of Ask a Patient forum which shows patients' reviews of the **Chlorpheniramine** which is a drug used in the prevention of the symptoms of allergic conditions. We use columns *SIDE EFFECTS* and *COMMENTS* in the table to generate UGC documents. We crawled user reviews of corresponding drug names that appear in the DDI 2013 Evaluation dataset. Then UGC embeddings are trained with the crawled reviews using document embedding model [15] where each review is regard as a document. We can map each review to a UGC embedding which is a dense vector. Every drug name can have multiple comments from different patients, therefore, a drug name may correspond to multiple UGC embeddings simultaneously.

For the sentence S, we combine UGC embeddings of two drug names to obtain an unique set $emb_{\text{UGC}} = \{\boldsymbol{u_1}, \boldsymbol{u_2}, \ldots, \boldsymbol{u_g}\}$. Then, two collections of embeddings, emb_{word} and emb_{UGC}, are obtained. Finally, we merge them together by an attentive way called ***full attention***.

Full attention is inspired by the self-attention mechanism described in [8]. We propose full attention based on the *Scaled Product Attention* which is one of core concepts in the self-attention mechanism. The details of full-attention is shown in (1):

$$\boldsymbol{a_i} = \text{Attention}(\boldsymbol{e_i}, \boldsymbol{U_i}) = \text{Softmax}\left(\frac{\boldsymbol{e_i}\boldsymbol{U_i}^\top}{\sqrt{d_k}}\right)\boldsymbol{U_i} \quad (1)$$

The inputs consist of the word embedding vector $\boldsymbol{e_i}$ and a stacked UGC embedding matrix $\boldsymbol{U_i} \in \mathbb{R}^{g \times d_k}$ which is the matrix representation of the set emb_{UGC}. We compute the dot product $\boldsymbol{e_i}\boldsymbol{U_i}^\top$ to get a group of weights between current input token t_i and its corresponding UGC embeddings. We use a scaled factor d_k which is the dimension of a UGC embedding vector to reduce the large-magnitude-growth effect [8]. Then we use the Softmax function to normalize these scaled weights which stand for correlations between the input token and its UGC embeddings U_i. We can merge these two collections of embeddings depending on computed weights. Output $\boldsymbol{a_i} \in \mathbb{R}^{d_k}$ is a weighted sum of UGC embeddings $\{\boldsymbol{u_1}, \boldsymbol{u_2}, \ldots, \boldsymbol{u_g}\}$. We present a diagram of the full-attention computation procedure in Fig. 3.

Full-attention is proposed to compute alignment weights between UGC embeddings and word embeddings, we combine these two kinds of embeddings according to output weights. We calculate a scaled dot product of the $\boldsymbol{e_i}$ and $\boldsymbol{U_i}$ to get a weight distribution and then normalize it by the Softmax function. Then, we sum up UGC embeddings with weights computed above and get an UGC feature vector $\boldsymbol{a_i} \in \mathbb{R}^{d_k}$. Finally, we perform an element-wise summation over it and the original word embedding.

$$\boldsymbol{w_i} = \boldsymbol{a_i} + \boldsymbol{e_i} \quad (2)$$

The final output of full-attention $\boldsymbol{w_i} \in \mathbb{R}^{d_k}$ combines contextual information from word embeddings and external medical knowledge from UGC embeddings. The computation of full-attention is very intuitive and efficient, no extra parameter is needed.

2) Concept Embeddings

RATING	REASON	SIDE EFFECTS FOR CHLOR-TRIMETON	COMMENTS	SEX	AGE	DURATION/ DOSAGE	DATE ADDED
				F M			
5	Seasonal allergies	None as long as I take 1/2 of a 4 mg tablet. When I take a whole table (4 mg) I feel very "spacey".	I take 1/2 of a 4 mg tablet every morning because I have a lot of allergies. During the months when the Cedar trees are puffing their pollen (Dec, Jan, Feb) I sometimes have to take 1/2 tablet every 4 hours through the day and night.	F	73	8 years 2 mg 1X D	1/16/2016
3	seasonal allergies	Caused a sort of speedy feeling - a bit like I had too many cups of coffee - and caused my heart rate to go through the roof @ 120bpm while resting. Will not be taking it again.	Unfortunately I can't continue to take this because of the rise in heart rate it caused but it did relieve a lot of my symptoms. It was as good as Zyrtec, better than Claritin and totally different from Benadryl. None of the grogginess but if you are sensitive to decongestants (despite that this is NOT one), you might want to avoid it.	F	42	3 days 4mg 2X D	5/6/2014 Email
5	Allergies and Allergy Related Sympt	Unfortunately, when I first take the drug, there is a brief (45 min +/-) period where my nose becomes terribly itchy. In the evening hours I become more sleepy than without, however it's very easy to remain awake during the day.	With seasonal allergies, and my need to drive a vehicle for work, this is by far the best medicine on the market. The 4 hour is my personal preference. Directly after the itchiness, there is immediate and sustained relief of post-nasal drip, allergy related headache and watery eyes. T	F	45	30 years 4mg 2X D	4/19/2014

Fig. 2: Patients' reviews on **Chlorpheniramine** from Ask a Patient.

Obviously, DDI extraction is an interdisciplinary task that combines biomedical science, computer science, and linguistics. Due to this nature of DDI extraction, the domain-specific knowledge is also very valuable for this task. We introduce concept embeddings to add biomedical domain-specific information to the model. A concept is a biomedical entity such as drugs, proteins, genes, etc. We use the taxonomy of Unified Medical Language System (UMLS) ontology [16], and every concept is assigned a Concept Unique Identifier (CUI). Concept embeddings are low-dimensional vectors which are learned from the relations between concepts and semantics of concepts. We firstly translate a sentence from a token sequence to a CUI sequence by the *MetaMap* toolkit [17]. Then we map CUIs to corresponding concept embeddings which are trained by the concept embedding model proposed in [18]. For the sentence S, now we have a concept embedding list $emb_{\text{concept}} = \{\boldsymbol{p_1}, \boldsymbol{p_2}, \ldots, \boldsymbol{p_n}\}$. Concept embedding list and token sequence have the same length [19].

3) Offset Embeddings

Offset embeddings can supply distance information between the current token and two drug entities. We calculate distances of tokens from the current token to the two drugs, i.e. **DRUG1** and **DRUG2**, in the drug pair respectively. Then we convert two distances into two offset embeddings by an dynamic offset lookup table. The basic idea of offset embeddings is that adjacent tokens make more contributions to the final predictions. So we want the model to give more concentration to nearby words of drug entities.

For the token t_i in the sentence S, we concatenate the attention output vector $\boldsymbol{w_i}$, the concept embedding $\boldsymbol{p_i}$ and the offset embeddings $\boldsymbol{o1_i} \in \mathbb{R}^{d_o}$, $\boldsymbol{o2_i} \in \mathbb{R}^{d_o}$ together to generate the final token representation $\boldsymbol{x_i} \in \mathbb{R}^{d_k+d_k+2\times d_o}$:

$$\boldsymbol{x_i} = [\boldsymbol{w_i} + \boldsymbol{p_i} + \boldsymbol{o1_i} + \boldsymbol{o2_i}]\,. \tag{3}$$

C. Classifier

In this step, we extract latent features of sentences by a deep learning encoder and make predictions of DDI type with a Softmax densely-connected neural network.

We build a stacked two-layer encoder consists of a Bi-LSTM layer and a Transformer Layer. Bi-LSTM [20], [21] is a powerful sentence embedding architecture [22], it has been applied to extensive NLP tasks such as speech recognition [23], sequence tagging [24], dependency parsing [25], and event detection [26] as an encoder. So we choose Bi-LSTM to be one layer of our stacked encoder. Given a token representation list $X_S = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}\}$ of sentence S, we calculate two collections of outputs named hidden states: forward hidden states $\overrightarrow{\boldsymbol{h}} = \{\overrightarrow{\boldsymbol{h_1}}, \overrightarrow{\boldsymbol{h_2}}, \ldots, \overrightarrow{\boldsymbol{h_n}}\}$ and backward hidden states $\overleftarrow{\boldsymbol{h}} = \{\overleftarrow{\boldsymbol{h_1}}, \overleftarrow{\boldsymbol{h_2}}, \ldots, \overleftarrow{\boldsymbol{h_n}}\}$. We concatenate them to get the final output of Bi-LSTM layer:

$$\boldsymbol{h_i} = \left[\overrightarrow{\boldsymbol{h_i}} + \overleftarrow{\boldsymbol{h_i}}\right] \tag{4}$$

After the Bi-LSTM layer, a Transformer layer which is described in [8] follows. We append the Transformer layer at this position because Transformer model is more efficient and can capture more global dependency. The Transformer Model we use includes N Transformer blocks, and between the blocks we connect these blocks with residual connections and layer normalizations. After the process of the Transformer model, we get a list of token features $\boldsymbol{h'} = \{\boldsymbol{h'_1}, \boldsymbol{h'_2}, \ldots, \boldsymbol{h'_n}\}$. In practice, we stack the token feature list as a matrix

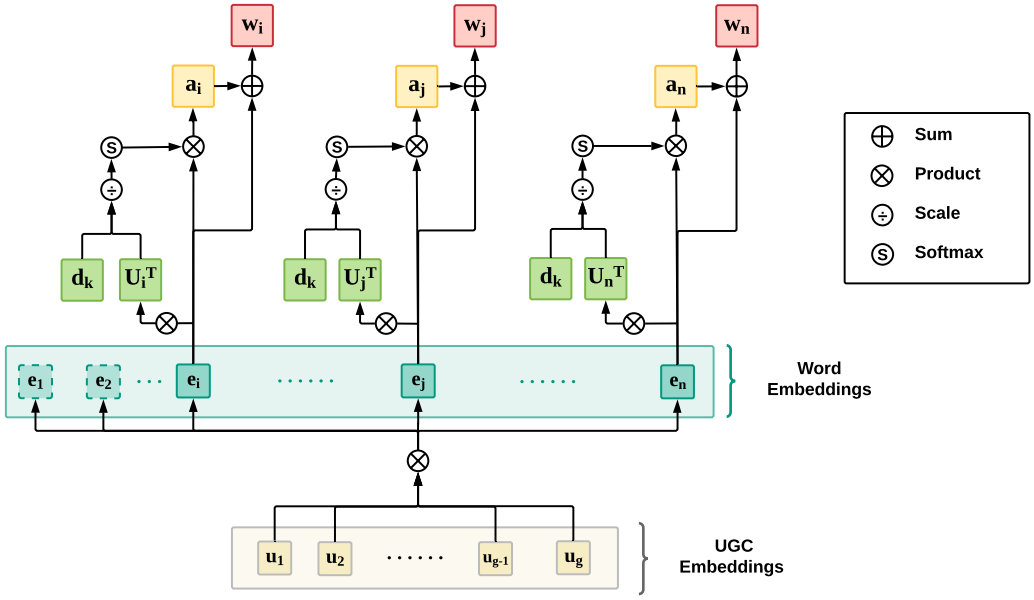

Fig. 3: The computation procedure of full-attention.

$H \in \mathbb{R}^{n \times d_h}$, d_h is the dimension of a single token feature vector.

We use an attentive pooling method to generate sentence feature vectors. Given the token feature matrix $\boldsymbol{H}_S \in \mathbb{R}^{n \times d_h}$ of sentence S, we first calculate a token weights vector $\boldsymbol{v'_S} \in \mathbb{R}^n$:

$$\boldsymbol{v}_S = \tan\left(W H_S^\top + \boldsymbol{b}\right), \tag{5}$$

$$\boldsymbol{v'}_S = Softmax\left(\boldsymbol{v}_S^\top \cdot \boldsymbol{w_v}\right), \tag{6}$$

$W \in \mathbb{R}^{d_h \times d_h}$, $\boldsymbol{b} \in \mathbb{R}^{d_h}$, and $\boldsymbol{w_v} \in \mathbb{R}^{d_h}$ are all learnable parameters.

Then we compute the final sentence embedding $\boldsymbol{emb}_\mathrm{S} \in \mathbb{R}^{d_h}$ which is a weighted sum of token features:

$$\boldsymbol{emb}_\mathrm{S} = \sum_{i=0}^{n} (\boldsymbol{v'}_S)_i \times (\boldsymbol{h'}_S)_i \tag{7}$$

After getting the sentence embedding vector, we feed it into a Softmax layer to make predictions:

$$\boldsymbol{DDI_S} = Softmax\left(W_C \times \boldsymbol{emb}_\mathrm{S}^\top + \boldsymbol{b_C}\right), \tag{8}$$

where the output $\boldsymbol{DDI_S} \in \mathbb{R}^C$ is a DDI type distribution, C is the count of the DDI types. The weight $W_C \in \mathbb{R}^{C \times d_h}$ and the bias $\boldsymbol{b_C} \in \mathbb{R}^C$ are trainable parameters.

III. Experiments

We built our method with Python and Keras deep learning library [27] running on the top of TensorFlow [28] backend. We trained our model on a NVIDIA TITAN Xp graphic card whose frame buffer size is 12GB.

TABLE I: Label distribution before and after the negative instance filtering phase.

Class	Training Set		Test Set	
	Before	After	Before	After
negative	23772	8987	4737	2049
advise	826	814	221	221
effect	1687	1592	360	357
mechanism	1319	1260	302	301
int	188	188	96	92
ratio	1:5.91	1:2.33	1:4.83	1:2.11
Total	27792	12841	5716	3020

A. Dataset Description And Evaluation Metrics

We evaluated our method on the DDI 2013 Evaluation datasets [29]. The datasets consist of $27,792$ and $5,716$ drug pairs in the training set and the test set, respectively. There is one kind of $negative$ label and four types of positive DDI labels: $advise$, $mechanism$, int, and $effect$. As we mentioned before, we filtered out some negative instances in the preprocessing step. We list the label type distribution before and after the negative instance filtering phase in the Table I.

We used official evaluation metrics, precision (P), recall (R), and F-score (F) to measure the performance of our DDI extraction method.

B. Hyper-parameters and training strategies

UGC embeddings, word embeddings and concepts embeddings used in our method are all pre-trained and their dimensions are all 200. Word embeddings [30] were induced from PubMed and PubMed Central (PMC) texts by the Word2vec model [31], [32]. We updated word embeddings during the

TABLE II: Hyper-parameter list

Name	Value
UGC embedding dimension	200
word embedding dimension	200
concept embedding dimension	200
offset embedding dimension	20
hidden state dimension	250
feedforward dimension of Transformer	512
head count of Transformer	5
block count of Transformer	4
weight decay	0.08
dropout probability	0.5
training batch size	128
test batch size	128
Adam-learning rate	0.004
Adam-$\beta - 1$	0.9
Adam-$\beta - 2$	0.999
Adam-ϵ (epsilon)	1e-08
Adam-learning rate decay	0

training process. In contrast, concept embeddings and UGC embeddings stayed static. UGC embeddings are trained by the gensim toolkit [33]. DRUG1 offset embeddings and DRUG2 offset embeddings shared the same embedding look-up table whose dimension is 20. Offset embedding vectors were initialized randomly and were also updated dynamically.

The hidden state dimension is 250 in our Bi-LSTM encoder. Our Transformer encoder consists of $N = 4$ Transformer self-attention blocks. The feedforward dimension is 512 and the count of heads of multi-head self-attention computation in the Transformer encoder is 5 which is the same as the count of our DDI types (one negative type and four positive types). To overcome the overfitting problem, we used the dropout technique in our classifier with a drop rate value 0.5. We also used several L1-L2 regularizers [34] inside our model, the weight decay value was 0.08. We trained our model with an Adam optimizer [35], the learning rate we used was 0.0004, other hyper-parameters of optimizers were all default values. Our training batch size was 128 and testing batch size was 64,. The zero-masking strategy was used to train on variance-length sequences. We list all of our hyper-parameters in Table II.

C. Experimental Result

Table III presents the evaluation results of every single DDI type. As can be seen, "int" DDI type only achieves an F-score of 0.4545 which is much lower than that of other three DDI types. It obtains a recall of 0.3125 that causes the lowest F-score. "int" DDI type is assigned when the sentence simply states that an interaction occurs and does not provide any information about the interaction [29]. Due to the ambiguous meaning of the "int" DDI type, it is error prone to misclassified instances to other specific DDI types. The "mechanism" type achieves the highest F-score, this is probably because "mechanism" describes a pharmacokinetic mechanism. Pharmacokinetic means that the effects of one drug are changed by the presence of another drug at its site of action which is relatively not easy to confuse.

1) Performance comparison

Evaluation metrics comparison with other state-of-the-art methods are showed in the Table IV. The highest values of every metric are highlighted in bold text. Our methods achieved the highest F-score of 0.712 and the highest recall of 0.668 in the comparison.

The preceding four methods in Table IV UTurku, WBI-DDI, FBK-irst, and Kim et al. are all support vector machine (SVM) based DDI extraction methods. One significant drawback of SVM based models is that they usually utilize complex human-selected lexical or syntactical features to form representations of sentences or tokens. For examples, Kim et al. which performs best among SVM based DDI extraction methods uses a linear SVM classifier with a rich set of lexical and syntactic features to extract DDI. FBK-irst utilize a hybrid kernel SVM classifier with syntax tree and dependency tree features.

SVM based methods are all highly dependent on the external lexical tools such as part-of-speech tagger, chunk tagger, and dependency parser to analyze text and construct features from the processing output. Errors from the external lexical tools can propagate to the DDI extraction method through the pipeline [38]. Selecting features manually is laborious and time-consuming compared with deep learning latent features. From another perspective, human-selected features may also perform poor on generalization. Compared with FBK-irst and Kim et al., UGC-DDI learns high-level latent features automatically from the deep learning network encoder and the attentive pooling method. It also does not depends on any external lexical or syntactical analyzer to generate features. However, UGC-DDI achieves an F-score of 0.712 which is higher than that of FBK-irst and Kim et al. by 0.061 and 0.042, respectively.

Remaining methods in Table IV are all deep learning based methods. They can be categorized into two types: convolutional neural network based methods and bidirectional long short-term memory (Bi-LSTM) network based methods. SCNN [5] is a syntax convolutional neural network based DDI extraction method. In SCNN, a novel word embedding, syntax embedding, is proposed to employ syntactical information of a sentence. SCNN also employs many hand-crafted features to improve the performance of DDI extraction that makes the extraction process more complicated. Compared with SCNN, UGC-DDI extracts no traditional feature but achieves a higher F-score, 0.712 vs. 0.686.

Joint AB-LSTM is a bidirectional long short-term memory based DDI extraction method. It joins two separate bidirectional long short-term memory network, B-LSTM and AB-LSTM to generate a sentence embedding. B-LSTM and AB-LSTM utilize two different pooling methods max poling and attentive pooling in their own model to unify token embeddings. Instead of two independent Bi-LSTM networks, UGC-DDI utilize a stacked two-layer encoder inside the classifier, two encoder layers can share the information and communicate with each other. Besides the model architecture, UGC-DDI also utilizes domain-specific knowledge such as UGC embeddings and concept embeddings in the procedure of DDI extraction. Compared with Joint AB-LSTM, UGC-DDI

TABLE III: Detailed evaluation metrics of DDI-UGC.

	TP	FP	FN	Total	P	R	F
mechanism	230	69	72	302	0.769	0.762	0.765
effect	240	101	120	360	0.704	0.667	0.685
advise	154	28	67	221	0.846	0.697	0.764
int	30	6	66	96	0.833	0.312	0.455
classification	654	204	325	979	0.762	0.668	0.712

TABLE IV: Performance comparison between methods

Methods	P	R	F
UTurku [4]	0.732	0.499	0.594
WBI-DDI [2]	0.642	0.579	0.609
FBK-irst [1]	0.646	0.656	0.651
Kim *et al.* [36]	—	—	0.670
SCNN[1] [5]	0.691	0.651	0.670
SCNN[2] [5]	0.725	0.651	0.686
Joint AB-LSTM [11]	0.745	0.645	0.694
DCNN [37]	**0.772**	0.644	0.702
Our method without UGC resource	0.756	0.656	0.702
UGC-DDI	0.762	**0.668**	**0.712**

achieves a higher F-score (0.712 vs. 0.694) with a richer set of information sources.

2) UGC feature analysis

UGC external biomedical resource is a core information source in our UGC-DDI method. To evaluate the improvement made by the external UGC resource, we compared the metrics and the loss of our method with and without UGC embeddings in Fig. 4 and Fig. 5. We conduct an extra comparison experiment which concatenates the word embeddings, concept embeddings and two offset embeddings directly as token representations. Other model components such as encoders and attentive pooling layer are all the same as the UGC-DDI method settings. We can see that the F-score of UGC-DDI method increases more rapidly and climb to a highest peak in a extremely short period (94th epoch in about half an hour). Although the method without UGC resource does not get a better performance compared with UGC-DDI, it gains a relatively good performance compared with other existing methods, either SVM based methods or deep learning based methods. As we mentioned before, UGC embeddings stayed static during the training procedure. And the full-attention mechanism does not add new parameters into the model, it just merge UGC embeddings and word embeddings based on their scale dot products and summations. No extra training cost is needed compared with the method without UGC resource. Compared with directly concatenating UGC embeddings with word embeddings or other non-attentive manner, full-attention does not increase the dimension of the token representation and enrich the information offered to the classifier. This advantage also control the magnitude of parameters of the whole network and prevent suffering from the overfitting issue.

The F-score of the method without the UGC resource is 0.702 and the F-score is improved by 1% with the utilization of the UGC resource.

IV. Conclusions

In this study, we proposed a novel methods namely UGC-DDI to extract DDI from biomedical literature automatically. We merge user generated content with local contextual information by full-attention mechanism to provide rich and fresh knowledge. User generated content are offered in the form of low-dimension embedding vectors. Attention outputs are concatenated with concept embeddings and entity offset embeddings together to be the input of the deep learning classifier of our method. The experimental result shows that our method gains the highest evaluation metrics values.

References

[1] 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," vol. 2, pp. 351–355.

[2] P. Thomas, M. Neves, T. Rocktäschel, and U. Leser, "WBI-DDI: drug-drug interaction extraction using majority voting," vol. 2, pp. 628–635.

[3] M. Rastegar-Mojarad, R. D. Boyce, and R. Prasad, "UWM-TRIADS: classifying drug-drug interactions with two-stage SVM and post-processing," vol. 2, pp. 667–674.

[4] J. Björne, S. Kaewphan, and T. Salakoski, "UTurku: drug named entity recognition and drug-drug interaction extraction using SVM classification and domain knowledge," vol. 2, pp. 651–659.

[5] Z. Zhao, Z. Yang, L. Luo, H. Lin, and J. Wang, "Drug drug interaction extraction from biomedical literature using syntax convolutional neural network," p. btw486. [Online]. Available: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btw486

[6] J. Krumm, N. Davies, and C. Narayanaswami, "User-generated content," vol. 7, no. 4, pp. 10–11.

[7] W. Duan, Q. Cao, Y. Yu, and S. Levy, "Mining online user-generated content: using sentiment analysis technique to study hotel service quality." IEEE, pp. 3119–3128.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need." [Online]. Available: http://arxiv.org/abs/1706.03762

[9] T. Shen, T. Zhou, G. Long, J. Jiang, S. Pan, and C. Zhang, "DiSAN: Directional self-attention network for RNN/CNN-free language understanding." [Online]. Available: http://arxiv.org/abs/1709.04696

[10] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," vol. 250, pp. 113–141.

[11] S. K. Sahu and A. Anand, "Drug-drug interaction extraction from biomedical text using long short term memory network." [Online]. Available: http://arxiv.org/abs/1701.08303

[12] I. Segura-Bedmar, P. Martínez, and M. Herrero-Zazo, "Lessons learnt from the DDIExtraction-2013 shared task," vol. 51, pp. 152–164. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1532046414001245

[13] S. Liu, B. Tang, Q. Chen, and X. Wang, "Drug-drug interaction extraction via convolutional neural networks," vol. 2016.

[14] Y. Tsuruoka, "GENIA tagger: Part-of-speech tagging, shallow parsing, and named entity recognition for biomedical text."

[15] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," pp. 1188–1196.

[16] O. Bodenreider, "The unified medical language system (UMLS): integrating biomedical terminology," vol. 32, pp. D267–D270.

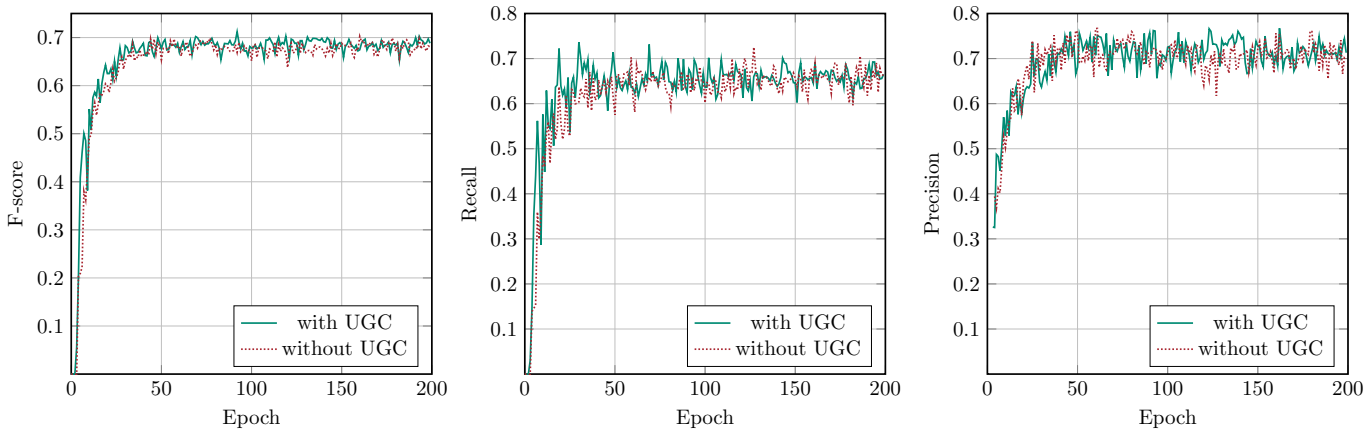

Fig. 4: Metrics comparison of methods with and without UGC resource during the training.

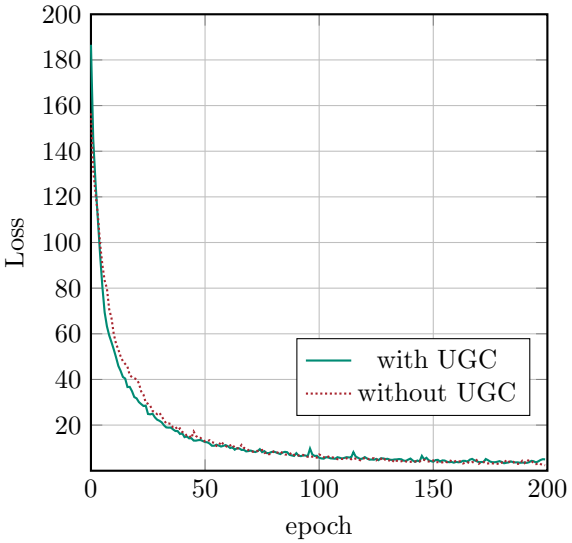

Fig. 5: Loss comparison of methods with and without UGC resource during the training.

[17] A. R. Aronson and F.-M. Lang, "An overview of MetaMap: historical perspective and recent advances," vol. 17, no. 3, pp. 229–236.

[18] L. De Vine, G. Zuccon, B. Koopman, L. Sitbon, and P. Bruza, "Medical semantic similarity with a neural language model." ACM, pp. 1819–1822.

[19] L. Wang, M. Li, J. Xie, Y. Cao, H. Liu, and Y. He, "Ontology-based systematical representation and drug class effect analysis of package insert-reported adverse events associated with cardiovascular drugs used in china," vol. 7, no. 1, p. 13819.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," vol. 9, no. 8, pp. 1735–1780.

[21] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," vol. 45, no. 11, pp. 2673–2681.

[22] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding." [Online]. Available: http://arxiv.org/abs/1703.03130

[23] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition."

[24] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF models for sequence tagging."

[25] E. Kiperwasser and Y. Goldberg, "Simple and accurate dependency parsing using bidirectional LSTM feature representations."

[26] X. Feng, B. Qin, and T. Liu, "A language-independent neural network for event detection," vol. 61, no. 9, p. 092106.

[27] F. Chollet, "Keras."

[28] S. S. Girija, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems."

[29] I. Segura-Bedmar, P. Martínez, and M. H. Zazo, "Semeval-2013 task 9: Extraction of drug-drug interactions from biomedical texts (ddiextraction 2013)," vol. 2, pp. 341–350.

[30] S. Moen and T. S. S. Ananiadou, "Distributional semantics resources for biomedical text processing," pp. 39–43.

[31] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality." [Online]. Available: http://arxiv.org/abs/1310.4546

[32] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space." [Online]. Available: http://arxiv.org/abs/1301.3781

[33] R. Řehůřek and P. Sojka, "Software framework for topic modelling with large corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, pp. 45–50.

[34] A. Y. Ng, "Feature selection, l 1 vs. l 2 regularization, and rotational invariance." ACM, p. 78.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization." [Online]. Available: http://arxiv.org/abs/1412.6980

[36] S. Kim, H. Liu, L. Yeganova, and W. J. Wilbur, "Extracting drug–drug interactions from literature using a rich feature-based linear kernel approach," vol. 55, pp. 23–30. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1532046415000441

[37] S. Liu, K. Chen, Q. Chen, and B. Tang, "Dependency-based convolutional neural network for drug-drug interaction extraction." IEEE, pp. 1074–1080.

[38] Z. Jiao, S. Sun, and K. Sun, "Chinese lexical analysis with deep bi-GRU-CRF network." [Online]. Available: http://arxiv.org/abs/1807.01882