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

Bo Xu, *Member, IEEE*, Xiufeng Shi, Yueqin Yin, Zhehuan Zhao, Wei Zheng, Hongfei Lin, Zhihao Yang, Jian Wang, and Feng Xia, *Senior Member, IEEE*

***Abstract*— It is crucial for doctors to fully understand the interaction between drugs in prescriptions, especially when a patient takes multiple medications at the same time during treatment. The purpose of drug drug interaction (DDI) extraction is to automatically obtain the interaction between drugs from biomedical literature. Current state-of-the-art approaches for DDI extraction task are based on artificial intelligence and natural language processing. While such existing DDI extraction methods can provide more knowledge and enhance the performance through external resources such as biomedical databases or ontologies, due to the difficulty of updating, these external resources are delayed. In fact, user generated content (UGC) is another kind of external medical resources that can be quickly updated. We are trying to use UGC resources to provide more available information for our deep learning DDI extraction method. In this paper, we present a DDI extraction approach through a new attention mechanism called full-attention which can combine the UGC information with contextual information. We conducted a series of experiments on the DDI 2013 Evaluation dataset to evaluate our method. Experiments show improved performance compared with the state of the art and UGC-DDI model achieves a competitive F-score of 0.712.**

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

I. Introduction

UNDOUBTEDLY, if the doctor misjudged the possible interaction between the various drugs prescribed to the patient, it would cause a serious accident. But it is a huge challenge for doctors to keep up with the frequently updated knowledge of drug interactions. Traditionally, doctors obtain newly published drug-drug interactions (DDIs) mainly from two auxiliary information sources: querying DDIs from biomedical databases or reading a large number of biomedical papers. Apparently, 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. Obviously, neither of these methods is an idealistic method for detecting DDIs.

The DDI extraction task is designed to extract DDIs from the free text of the biomedical field. 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**.

The DDI extraction method attempts to predict the type of 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 use artificial intelligence (AI) and natural language processing (NLP) techniques to extract DDIs. Some of them enhance the performance of DDI extraction through introducing external resources. WBI-DDI uses WordNet lexical database to extract token features. UWM-TRIADS uses an FDA Drug classification dictionary to recognize drugs in the same drug class during the post-processing phase. Uturku sets the presences of words in the DrugBank as a part of features in the feature engineering step. However, the external resources used by these methods come from deferred sources of information, such as manually updated databases, ontologies, or vocabulary tools that cannot be updated in time to catch up with the latest information. Many experts are needed to update, review, and revise new items for these external resources.

Manuscript received March 30, 2019; accepted March 31, 2019. Date of publication May 27, 2019; date of current version June 28, 2019. 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: Zhehuan Zhao.)*

B. Xu, X. Shi, Z. Zhao, and F. Xia are with the Key Laboratory for Ubiquitous Network and Service Software of Liaoning, School of Software, Dalian University of Technology, Dalian 116024, China (e-mail: z.zhao@dlut.edu.cn).

Y. Yin is with the School of Software, Dalian University of Technology, Dalian 116024, China.

W. Zheng is with the College of Software, Dalian JiaoTong University, Dalian 116028, China.

H. Lin, Z. Yang, and J. Wang are with the Computer Science and Technology Department, Dalian University of Technology, Dalian 116024, China.

Digital Object Identifier 10.1109/TNB.2019.2919188

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. Considering 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, still, no research effectively exploits 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.

As far as we know, there is no existing method which apply UGC to 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***. 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. The details of the proposed method are given in Section II. In Section III, we describe experimental settings and results. Finally, Section IV shows our conclusion.

II. Method

The process of our DDI extraction method is divided into a total of three steps:

1) In the ***preprocessing*** step, we implement several types of operations from DDI 2013 Evaluation dataset to trim and condense the raw sentences.
2) In the ***token representation*** step, we first merge UGC embeddings and word embeddings to get attention output vectors, 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 use a Softmax densely-connected neural network layer to make predictions.

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

A. Preprocessing

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

In negative examples, there is no interaction between the drugs. In the training dataset of DDI 2013 Evaluation, the positive to negative instances ratio is 1 : 5.91. Obviously, the dataset is very 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 mitigate the impact 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 specific drug entity names in sentences with special tokens. The purpose of drug blinding is to enhance the generalization of the model [13]. We use two token **DURG1** and **DRUG2** to replace the two drug entity names in every drug pair in the dataset, and use the token **DRUGN** to replace the other drug entity names in the same sentence.

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 first generate different types of embeddings that contains diverse information, then concatenate them together to final token representations.

1) Full Attention

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

For a sentence $S = \{t_1, t_2, \ldots, t_n\}$ consists of n tokens which is selected from DDI 2013 Evaluation dataset, we transform this token sequence to corresponding word embeddings

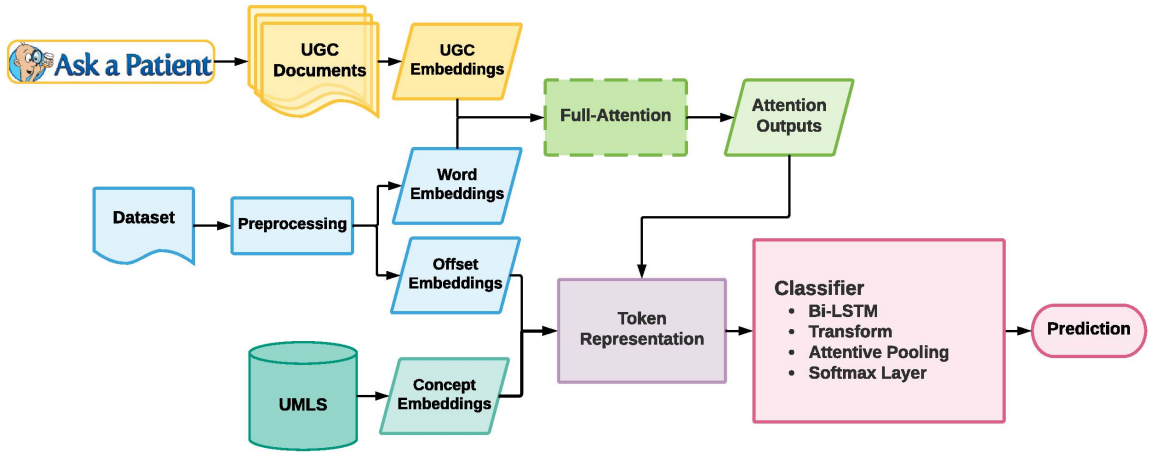

Fig. 1. The pipeline of UGC-DDI method.

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.

$emb_{\text{word}} = \{\boldsymbol{e_1}, \boldsymbol{e_2}, \ldots, \boldsymbol{e_n}\}$ through 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 based on documents crawled from the website www.askapatient.com *Ask a Patient*. Ask a Patient is an online medicine rating and reviewing forum where patients can share their side effects of drugs or success stories and learn from the experience of real people who have taken drug treatments. 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 generate UGC documents using columns *SIDE EFFECTS* and *COMMENTS* in the table. We crawled user reviews that included the drug names which appeared in the DDI 2013 Evaluation dataset. The crawled reviews are then used for UGC embeddings training by the document embedding model [15], where every review is treated as a document. We apply an UGC embedding to every review, to get 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 each sentence S, we combine the UGC embeddings of the two drug names contained therein to get a unique set $emb_{\text{UGC}} = \{\boldsymbol{u_1}, \boldsymbol{u_2}, \ldots, \boldsymbol{u_g}\}$. On this basis, we get two collections of embeddings, emb_{word} and emb_{UGC}. Finally, we use an attentive way called ***full attention*** to merge these two kinds of embeddings together.

Inspired by the Transformer architecture [8] which performed self-attention mechanism on machine translation, we propose full attention mechanism. We define the full-attention mechanism as follows which is shown in (1):

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

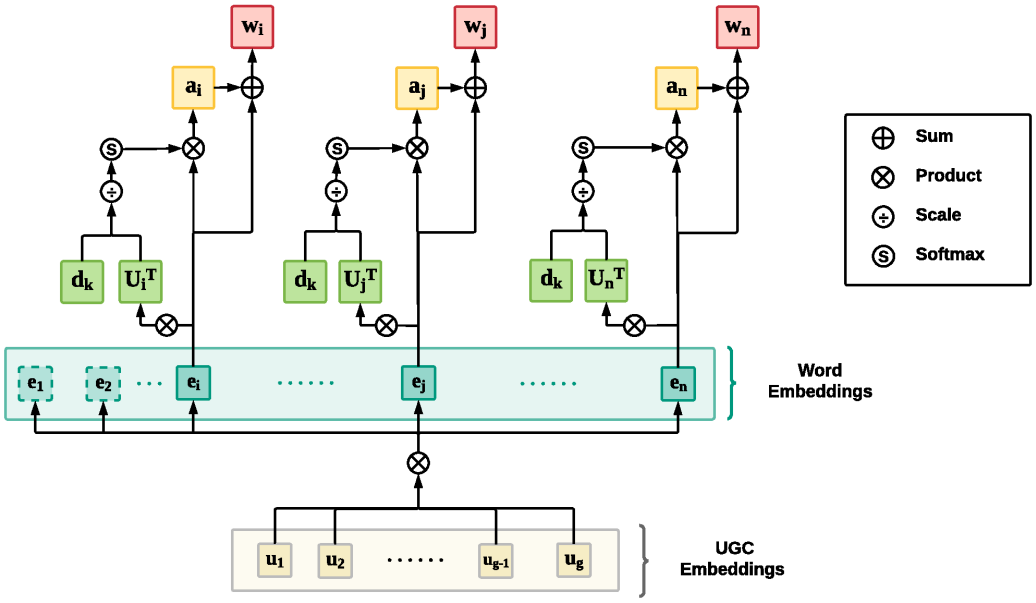

Fig. 3. The computation procedure of full-attention.

Out network takes a sequence of words as well as UGC documents as input. Let $\boldsymbol{e_i}$ be the vector representation of an element in the input sequence and a stacked UGC embedding matrix $\boldsymbol{U_i} \in \mathbb{R}^{g \times d_k}$ be the matrix representation of the set emb_{UGC}. We compute the dot product $\boldsymbol{e_i U_i^{\mathsf{T}}}$ of current input token t_i with its corresponding UGC embeddings to obtain the weights on UGC documents. We use a scaled factor d_k which is the dimension of an UGC embedding vector to reduce the large-magnitude-growth effect [8]. Then we apply a Softmax function to normalize these scaled weights which determine the correlations between the input token and the corresponding UGC embeddings U_i. We can merge these two collections of embeddings depending on computed attention score. Then we sum up the weighted UGC embeddings $\{\boldsymbol{u_1}, \boldsymbol{u_2}, \ldots, \boldsymbol{u_g}\}$ to obtain the output $\boldsymbol{a_i} \in \mathbb{R}^{d_k}$. We present a diagram of the full-attention computation procedure in Fig. 3.

We propose full attention mechanism to calculate the alignment weights of UGC embeddings and word embeddings, and combine these two kinds of embeddings according to the output weights. We calculate a scaled dot product of the $\boldsymbol{e_i}$ and $\boldsymbol{U_i}$ to obtain a weight distribution and then pass the result through a Softmax operation. Softmax normalizes the calculated scores so they are all positive and add up to 1. Then we multiply each UGC embedding by the softmax score and then sum up the weighted UGC embeddings to obtain an UGC feature vector $\boldsymbol{a_i} \in \mathbb{R}^{d_k}$. Finally, we perform an element-wise summation over $\boldsymbol{a_i}$ and the original word embedding $\boldsymbol{e_i}$.

$$\boldsymbol{w_i} = \boldsymbol{a_i} + \boldsymbol{e_i} \tag{2}$$

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

2) Concept Embeddings

Obviously, DDI extraction is an interdisciplinary task which combines biomedical science, computer science, and linguistics. Because of this nature of DDI extraction, knowledge in specific areas is also valuable for this task. We introduce concept embeddings to our model in order to add biomedical domain-specific information. A concept can be regarded as a biomedical entity such as drugs, genes, proteins, etc. We assign a Concept Unique Identifier (CUI) to every concept using the taxonomy of Unified Medical Language System (UMLS) ontology [16]. Concept embeddings are learned low-dimensional vectors which can be trained from the relations between concepts and semantics of concepts. We firstly use the *MetaMap* toolkit [17] to translate a sentence from a token sequence to a CUI sequence.

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 obtain a concept embedding list $emb_{\text{concept}} = \{\boldsymbol{p_1}, \boldsymbol{p_2}, \ldots, \boldsymbol{p_n}\}$. Concept embedding list and the sentence S have the same length [19].

3) Offset Embeddings

Offset embeddings can provide distance information between the current token and two drug entities. We calculate the token distance from the current token to the drug in drug pair, i.e. **DRUG1** and **DRUG2** respectively. These two distances are then mapped to two offset embeddings by looking up a dynamic offset lookup table. The basic idea of offset embeddings is that adjacent tokens make a greater contribution to the final predictions. So we want the model to give more attention to words nearby the 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 $\boldsymbol{x_i} \in \mathbb{R}^{d_k+d_k+2\times d_o}$:

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

C. Classifier

In this step, we use a deep learning encoder to extract the latent features of the sentence and use a Softmax densely-connected neural network to predict the DDI type.

Our model takes a hierarchical approach which is a stacked two-layer encoder. The first layer is a Bi-LSTM layer and then a Transformer Layer is employed.

Bi-LSTM [20], [21] has been a powerful sentence embedding architecture [22], giving state-of-the-art results in sequence tagging [23], speech recognition [24], dependency parsing [25], and event detection [26]. Consequently, Bi-LSTM is a suitable choice for the first layer of our stacked encoder.

Given a sentence S, we first calculate the token representation list $X_S = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}\}$ through the first two steps. Then we calculate two collections of 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 obtain the final output of Bi-LSTM layer by concatenating these two hidden states:

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

After the Bi-LSTM layer, we add a Transformer layer [8]. We apply the Transformer layer at this position to our model due to the Transformer's higher efficiency and stronger ability to capture global dependency. The Transformer Model we use includes N Transformer blocks, and between the blocks we has a residual connection around them and then followed by a layer-normalization step. After processing the Transformer model, a list of token features $\boldsymbol{h'} = \{\boldsymbol{h'_1}, \boldsymbol{h'_2}, \ldots, \boldsymbol{h'_n}\}$ is obtained. In practice, we packing our token feature list into a matrix $\mathrm{H} \in \mathbb{R}^{n\times d_h}$, where d_h is the dimension of a single token feature vector.

We generate sentence feature vectors using an attentive pooling method. Given the token feature matrix $\boldsymbol{H}_S \in \mathbb{R}^{n\times d_h}$ of the 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 learned parameters.

Then we sum up the weighted token feature vectors to generate the finals sentence embedding $\boldsymbol{emb}_\mathrm{S} \in \mathbb{R}^{d_h}$.

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

After getting the sentence embedding vector, we feed it into a Softmax layer for prediction:

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

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

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 learned parameters.

III. EXPERIMENTS

We built our method using Python and Keras deep learning library [27] which is 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.

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 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

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

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

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 the 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 as 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, in comparison with other state-of-the-art methods are shown 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 by comparison.

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**

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 example, 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 external lexical tools can propagate to the DDI extraction method through the pipeline [38]. Selecting features manually is laborious and time-consuming compared to deep learning latent features. From another perspective, human-selected features generally perform poorly. Compared to 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. Also, generating features does not depend on any external lexical or syntactical analyzer. 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 and bidirectional long short-term memory (Bi-LSTM) network based methods. SCNN [5] is a syntactical 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 to 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 networks; 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,

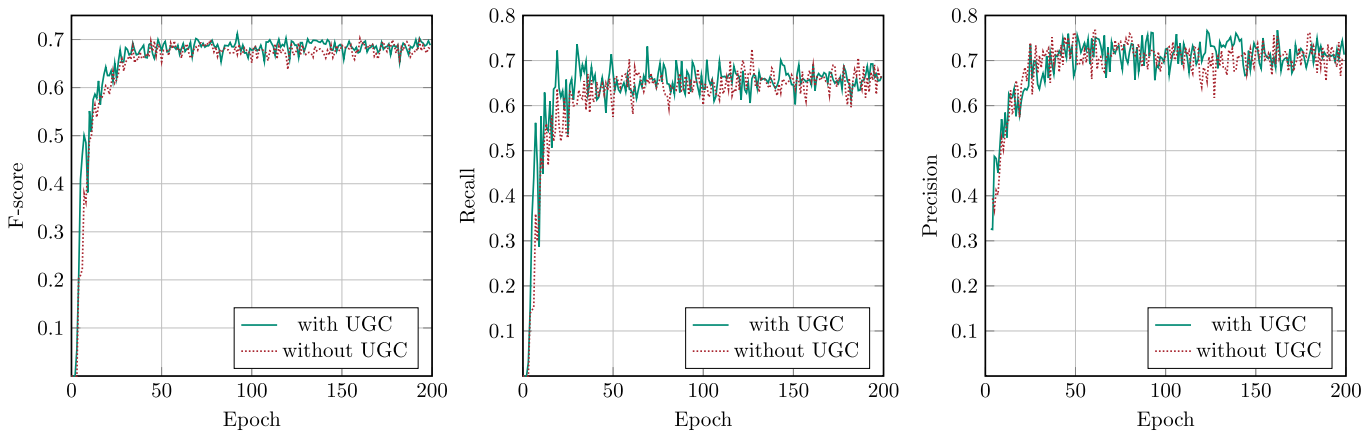

Fig. 4. Metrics comparison of methods with and without UGC resource during the training.

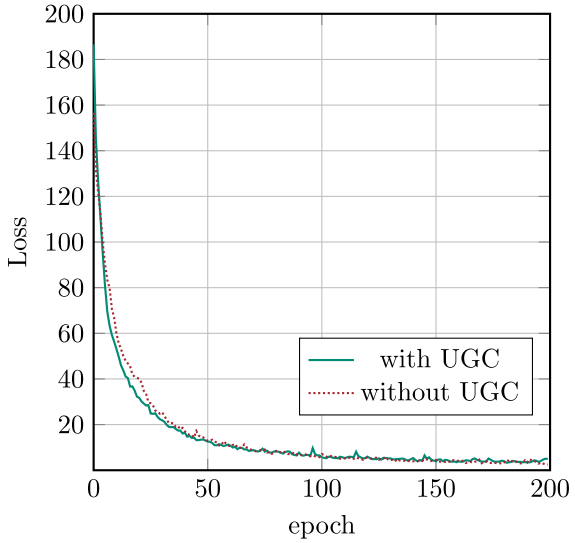

Fig. 5. Loss comparison of methods with and without UGC resource during the training.

UGC-DDI utilizes a stacked two-layer encoder inside the classifier, two encoder layers can share information by communicating with each other. Besides the model architecture, UGC-DDI also utilizes domain-specific knowledge such as UGC and concept embeddings in the procedure of DDI extraction. Compared to Joint AB-LSTM, UGC-DDI 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 the core data source in our UGC-DDI method. To evaluate the improvement made by the external UGC resource, we compared the metrics and the loss values 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 climbs to a highest peak in an extremely short period (94th epoch in about half an hour). Although the method without the UGC resource does not indicate an improved performance when compared to UGC-DDI, it gains relatively good performance in comparison to other existing methods, either SVM-based methods or deep-learning based methods. As we mentioned before, UGC embeddings remained static during the training procedure. The full-attention mechanism does not add new parameters into the model, rather merely merges UGC and word embeddings based on their scale dot products and summations. No extra training cost is required as compared to the method without the UGC resource. In comparison to directly concatenating UGC embeddings with word embeddings or other non-attentive manner, full-attention does not increase the dimension of the token representation and enriches the information offered to the classifier. This advantage also controls the magnitude of parameters of the whole network and prevents suffering from overfitting.

The F-score of the method without 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 method namely UGC-DDI to extract DDI from biomedical literature automatically. We merge user generated content with local contextual information through a full-attention mechanism to provide rich and fresh knowledge. User generated content is offered in the form of low-dimension embedding vectors. Attention outputs are concatenated with concept embeddings and entity offset embeddings to be the input of the deep learning classifier of our method. The experimental result shows that our method gains the highest evaluation metrics.

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," in *Proc. 2nd Joint Conf. Lexical Comput. Semantics*, vol. 2, 2013, pp. 351–355.

[2] P. Thomas, M. Neves, T. Rocktäschel, and U. Leser, "WBI-DDI: Drug-drug interaction extraction using majority voting," in *Proc. 2nd Joint Conf. Lexical Comput. Semantics (SEM)*, vol. 2, 2013, 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," in *Proc. 2nd Joint Conf. Lexical Comput. Semantics*, vol. 2, 2013, 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," in *Proc. 2nd Joint Conf. Lexical Comput. Semantics*, vol. 2, 2013, 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," *Bioinformatics*, vol. 32, no. 22, pp. 3444–3453, 2016. [Online]. Available: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioi%nformatics/btw486

[6] J. Krumm, N. Davies, and C. Narayanaswami, "User-generated content," *IEEE Pervasive Comput.*, vol. 7, no. 4, pp. 10–11, Oct. 2008.

[7] W. Duan, Q. Cao, Y. Yu, and S. Levy, "Mining online user-generated content: Using sentiment analysis technique to study hotel service quality." in *Proc. 46th Hawaii Int. Conf. Syst. Sci.*, Mar. 2013, pp. 3119–3128.

[8] A. Vaswani *et al.*, "Attention is all you need," Jun. 2017, *arXiv:1706.03762*. [Online]. Available: https://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," Sep. 2017, *arXiv:1709.04696*. [Online]. Available: https://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," *Inf. Sci.*, vol. 250, pp. 113–141, Nov. 2013.

[11] S. K. Sahu and A. Anand, "Drug-drug interaction extraction from biomedical text using long short term memory network," Jan. 2017, *arXiv:1701.08303*. [Online]. Available: https://arxiv.org/abs/1701.08303

[12] I. Segura-Bedmar, P. Martínez, and M. Herrero-Zazo, "Lessons learnt from the DDIExtraction-2013 shared task," *J. Biomed. Inform.*, vol. 51, pp. 152–164, Oct. 2014. [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," *Comput. Math. Methods Med.*, vol. 2016, Dec. 2015, Art. no. 6918381.

[14] Y. Tsuruoka and J. Tsujii, "Bidirectional inference with the easiest-first strategy for tagging sequence data," in *Proc. Conf. Human Lang. Technol. Empirical Methods Natural Lang. Process.*, 2005, pp. 467–474.

[15] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1188–1196.

[16] O. Bodenreider, "The unified medical language system (UMLS): Integrating biomedical terminology," *Nucleic Acids Res.*, vol. 32, pp. D267–D270, Jan. 2004.

[17] A. R. Aronson and F.-M. Lang, "An overview of MetaMap: Historical perspective and recent advances," *J. Amer. Med. Inform. Assoc.*, vol. 17, no. 3, pp. 229–236, 2010.

[18] L. De Vine, G. Zuccon, B. Koopman, L. Sitbon, and P. Bruza, "Medical semantic similarity with a neural language model," in *Proc. 23rd ACM Int. Conf. Conf. Inf. Knowl. Manage.*, 2014, 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," *Sci. Rep.*, vol. 7, no. 1, p. 13819, 2017.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.

[22] Z. Lin *et al.*, "A structured self-attentive sentence embedding," Mar. 2017, *arXiv:1703.03130*. [Online]. Available: https://arxiv.org/abs/1703.03130

[23] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF models for sequence tagging," Aug. 2015, *arXiv:1508.01991*. [Online]. Available: https://arxiv.org/abs/1508.01991

[24] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," Feb. 2014, *arXiv:1402.1128*. [Online]. Available: https://arxiv.org/abs/1402.1128

[25] E. Kiperwasser and Y. Goldberg, "Simple and accurate dependency parsing using bidirectional LSTM feature representations," *Trans. Assoc. Comput. Linguistics*, vol. 4, pp. 313–327, Dec. 2016.

[26] X. Feng, B. Qin, and T. Liu, "A language-independent neural network for event detection," *Sci. China Inf. Sci.*, vol. 61, no. 9, 2018, Art. no. 092106.

[27] F. Chollet *et al.*, "Keras," GitHub, 2015. [Online]. Available: https://github.com/fchollet/keras

[28] M. Abadi *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," in *Proc. 12th USENIX Symp. Oper. Syst. Design Implement. (OSDI)*, 2016, pp. 265–283.

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

[30] S. Moen and T. S. S. Ananiadou, "Distributional semantics resources for biomedical text processing," in *Proc. LBM*, 2013, pp. 39–43.

[31] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," Oct. 2013, *arXiv:1310.4546*. [Online]. Available: https://arxiv.org/abs/1310.4546

[32] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," Jan. 2013, *arXiv:1301.3781*. [Online]. Available: https://arxiv.org/abs/1301.3781

[33] R. Rehurek and P. Sojka, "Software framework for topic modelling with large corpora," in *Proc. LREC Workshop Challenges NLP Frameworks*, 2010, pp. 45–50.

[34] A. Y. Ng, "Feature selection, L 1 vs. L 2 regularization, and rotational invariance." in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, p. 78.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Dec. 2014, *arXiv:1412.6980*. [Online]. Available: https://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," *J. Biomed. Inform.*, vol. 55, pp. 23–30, Jun. 2015. [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," in *Proc. IEEE Int. Conf. Bioinf. Biomed. (BIBM)*, Dec. 2016, pp. 1074–1080.

[38] Z. Jiao, S. Sun, and K. Sun, "Chinese lexical analysis with deep bi-GRU-CRF network," Jul. 2018, *arXiv:1807.01882*. [Online]. Available: https://arxiv.org/abs/1807.01882