A Multi-Task Learning Framework for Extracting Drugs and Their Interactions from Drug Labels

Tung Tran[1], Ramakanth Kavuluru[1,2], and Halil Kilicoglu[3]

[1]Department of Computer Science, University of Kentucky, Lexington, KY, USA
[2]Division of Biomedical Informatics, Department of Internal Medicine, University of Kentucky, Lexington, KY, USA
[3]Lister Hill National Center for Biomedical Communications, National Library of Medicine, Bethesda, MD, USA

Abstract

Preventable adverse drug reactions as a result of medical errors present a growing concern in modern medicine. As drug-drug interactions (DDIs) may cause adverse reactions, being able to extracting DDIs from drug labels into machine-readable form is an important effort in effectively deploying drug safety information. The DDI track of TAC 2018 introduces two large hand-annotated test sets for the task of extracting DDIs from structured product labels with linkage to standard terminologies. Herein, we describe our approach to tackling tasks one and two of the DDI track, which corresponds to named entity recognition (NER) and sentence-level relation extraction respectively. Namely, our approach resembles a multi-task learning framework designed to jointly model various sub-tasks including NER and interaction type and outcome prediction. On NER, our system ranked second (among eight teams) at 33.00% and 38.25% F1 on Test Sets 1 and 2 respectively. On relation extraction, our system ranked second (among four teams) at 21.59% and 23.55% on Test Sets 1 and 2 respectively.

1 Introduction

Preventable adverse drug reactions (ADRs) introduce a growing concern in the modern healthcare system as they represent a large fraction of hospital admissions and play a significant role in increased health care costs [1]. Based on a study examining hospital admission data, it is estimated that approximately three to four percent of hospital admissions are caused by adverse events [2]; moreover, it is estimated that between 53% and 58% of these events were due to medical errors [3] (and are therefore considered preventable). Such preventable adverse events have been cited as the eighth leading cause of death in the U.S., with an estimated fatality rate of between 44,000 and 98,000 each year [4]. As drug-drug interactions (DDIs) may lead to preventable ADRs, being able to extract DDIs from structured product labeling (SPL) documents for prescription drugs is an important effort toward effective dissemination of drug safety information. The Text Analysis Conference (TAC) is a series of workshops aimed at encouraging research in natural language processing (NLP) and related applications by providing large test collections along with a standard evaluation procedure. The Drug-Drug Interaction Extraction from Drug Labels track of TAC 2018 [5], organized by the U.S. Food and Drug Administration (FDA) and U.S. National Library of Medicine (NLM), is established with the goal of transforming the contents of SPLs into a machine-readable format with linkage to standard terminologies.

We focus on the first two tasks of the DDI track involving named entity recognition (NER) and relation extraction (RE). Task 1 is focused on identifying **mentions** in the text corresponding to precipitants, interaction triggers, and interaction effects. Precipitants are defined as substances, drugs, or a drug class involved in an interaction. Task 2 is focused on identifying sentence-level interactions; concretely, the goal is
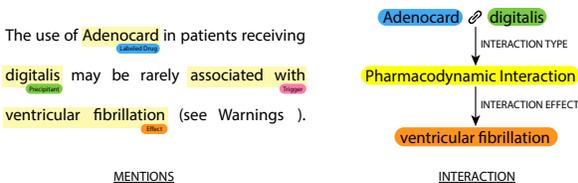
1

Figure 1: An example illustrating the DDI task

to identify the interacting precipitant, the type of the interaction, and outcome of the interaction. The interaction outcome depends on the interaction type as follows. Pharmacodynamic (PD) interactions are associated with a specified *effect* corresponding to a span within the text that describes the outcome of the interaction. Naturally, it is possible for a precipitant to be involved in multiple PD interactions. Pharmacokinetic (PK) interactions are associated with a label from a fixed vocabulary of National Cancer Institute (NCI) Thesaurus codes indicating various levels of increase/decrease in functional measurements. For example, consider the sentence: "There is evidence that treatment with phenytoin leads to to decrease intestinal absorption of furosemide, and consequently to lower peak serum furosemide concentrations." Here, *phenytoin* is involved in a PK interaction with the label drug, *furosemide*, and the type of PK interaction is indicated by the NCI Thesaurus code C54615 which describes a decrease in the maximum serum concentration (C_{max}) of the label drug. Lastly, *unspecified* (UN) interactions are interactions with an outcome that is not explicitly stated in the text and usually indicated through cautionary statements. Figure 1 features a simple example of a PD interaction that is extracted from the drug label for **Adenocard**, where the precipitant is *digitalis* and the effect is "ventricular fibrillation."

2 Materials and Methods

Herein, we describe the training and testing data involved in this task and the metrics used for evaluation. In Section 2.3, we describe our modeling approach, our deep learning architecture, and our training procedure.

2.1 Datasets

Each drug label is a collection of sections (e.g., DOSAGE & ADMINISTRATION, CONTRAINDICATIONS, and WARNINGS) where each section contains one or more sentences. Each sentence is annotated with a list of zero or more *mentions* and *interactions*. The training data released for this task contains 22 drug labels, referred to as Training-22, with gold standard annotations. Two test sets of 57 and 66 drug labels, referred to as Test Set 1 and 2 respectively, with gold standard annotations are used to evaluate participating systems. As Training-22 is a relatively small dataset, we additionally utilize an external dataset with 180 annotated drug labels dubbed NLM-180 [6] (more later). We provide summary statistics about these datasets in Table 1. Test Set 1 closely resembles Training-22 with respect to the sections that are annotated. However, Test Set 1 is more sparse in the sense that there are more sentences per drug label (144 vs. 27), with a smaller proportion of those sentences having gold annotations (23% vs. 51%). Test Set 2 is unique in that it contains annotations from only two sections, namely DRUG INTERACTIONS and CLINICAL PHARMACOLOGY, the latter of which is not represented in Training-22 (nor Test Set 1). Lastly, Training-22, Test Set 1, and Test Set 2 all vary with respect to the distribution of interaction types, with Training-22, Test Set 1, and Test Set 2 containing a higher proportion of PD, UN, and PK interactions respectively.

2.2 Evaluation Metrics

We used the official evaluation metrics for NER and relation extraction based on the standard precision, recall, and F1 micro-averaged over exactly matched entity/relation annotations. For either task, there are two matching criteria: *primary* and *relaxed*. For entity recognition, *relaxed* matching considers only entity bounds while *primary* matching considers entity bounds as well as the type of the entity. For relation extraction, *relaxed* matching only considers precipitant drug (and their bounds) while *primary* match-

	NLM-180 *	Training-22	Test Set 1	Test Set 2
Number of Drug Labels	180	22	57	66
Mean number of sentences per Drug Label	32	27	144	64
Mean number of words per sentence	23	24	22	23
Proportion of *annotated* sentences	27%	51%	23%	23%
Mean number of mentions per *annotated* sentence	4.0	3.8	3.7	3.6
Proportion of mentions that are Precipitant	57%	53%	56%	55%
Proportion of mentions that are Trigger	20%	28%	30%	33%
Proportion of mentions that are SpecificInteraction	23%	19%	14%	12%
Proportion of interactions that are Pharmacodynamic	47%	49%	33%	28%
Proportion of interactions that are Pharmacokinetic	25%	21%	28%	47%
Proportion of interactions that are Unspecified	28%	30%	39%	25%

Table 1: Characteristics of datasets
* Statistics for NLM-180 were computed on mapped examples (based on our own annotation mapping scheme) and not based on the original dataset.

ing comprehensively considers precipitant drugs and, for each, the corresponding interaction type and interaction outcome. As relation extraction evaluation takes into account the bounds of constituent entity predictions, relation extraction performance is heavily reliant on entity recognition performance. On the other hand, we note that while NER evaluation considers *trigger* mentions, *triggers* are ignored when evaluating relation extraction performance.

2.3 Methodology

We propose a multi-task learning framework for extracting drug-drug interactions from drug labels. The framework involves branching paths for each training objective (corresponding to sub-tasks) such that parameters of earlier layers (i.e., the context encoder) are shared.

Modeling Approach. Since only drugs involved in an interaction (precipitants) are annotated in the ground truth, we model the task of precipitant recognition and interaction type prediction jointly. We accomplish this by reducing the problem to a sequence tagging problem via a novel NER tagging scheme. That is, for each precipitant drug, we additionally encode the associated interaction type. Hence, there are five possible tags: **T** for trigger, **E** for effects, and **D**, **K**, and **U** for precipitants

with pharmaco**d**ynamic, pharmaco**k**inetic, and **u**nspecified interactions respectively. As a pre-processsing step, we identify the label drug in the sentence, if it is mentioned, and bind it to a generic entity token (e.g. "LABELDRUG"). We additionally account for label drug aliases, such as the generic version of a brand-name drug, and bind them to the same entity token. Table 2 shows how the tagging scheme is applied to the simple example in Figure 1. A drawback is that simplifying assumptions must be made that will hamper recall; e.g., we only consider non-overlapping mentions (more later).

O	O	O	O	O	O	O	**B-D**
The	use	of	LABELDRUG	in	patients	receiving	digitalis
O	O	O	**B-T**	**I-T**	**B-E**	**I-E**	O
may	be	rarely	associated	with	ventricular	fibrillation	.

Table 2: Example of the tagging scheme

Once we have identified the precipitant offsets (as well as of triggers/effects) and the interaction type for each precipitant, we subsequently predict the outcome or consequence of the interaction (if any). To that end, we consider all entity spans annotated with **K** tags and assign them a label from a static vocabulary of 20 NCI concept codes corresponding to PK consequence (i.e., multiclass classification) based on sentence-context. Likewise, we consider all entity spans annotated with **D** tags and link them to mention spans annotated with **E** tags; we accomplish this

3

via binary classification of all pairwise combinations. For entity spans annotated with **U** tags, no outcome prediction is made.

Neural Network Architecture. Our proposed deep neural network is illustrated in Figure 2. We utilize Bi-directional Long Short-Term Memory networks (Bi-LSTMs) and convolutional neural networks (CNNs) designed for natural language processing as building blocks for our architecture [7, 8]. Entity recognition and outcome prediction share common parameters via a Bi-LSTM context encoder that composes a context representation at each timestep based on input words mapped to dense embeddings and character-CNN composed representations. We use the same character-CNN representation as described in a prior work [9]; however, in this work, we omit the character type embedding. A Bi-LSTM component is used to annotate IOB tags for joint entity recognition and interaction type prediction (or, NER prediction) while a CNN with two separate dense output layers (one for PK and one for PD interactions) is used for outcome prediction. We consider NER prediction to be the main objective with outcome prediction playing a secondary role. When predicting outcome, the contextual input is arranged such that candidate entity (and effect) mentions are bound to generic tokens; the resulting representation is referred to as "entity-bound word embeddings" in Figure 2.

Notation. We denote $\text{BiLSTM}(\cdot) : \mathbb{R}^{n \times d_{\text{in}}} \mapsto \mathbb{R}^{n \times d_{\text{out}}}$ as an abstract function, representing a standard bi-directional recurrent neural network with LSTM units, where n is the number of input vector representations (e.g., word embeddings) in the sequence and d_{in} and d_{out} are the dimensionality of the input and output representations respectively. We similarity denote $\text{CNN}^{[h_1,\ldots,h_k]}(\cdot) : \mathbb{R}^{n \times d_{\text{in}}} \mapsto \mathbb{R}^{d_{\text{out}}}$ to represent a standard CNN that maps an $n \times d_{\text{in}}$ matrix to a vector representation of length d_{out}, where $[h_1,\ldots,h_k]$ is a list of window (or kernel) sizes that are used in the convolution.

Context Encoder. Let the input be a sentence of length n represented as a matrix $S \in \mathbb{R}^{n \times d}$, where each row corresponds to a word embedding of length d. Moreover, let $W^i \in \mathbb{R}^{m \times d_{char}}$ represent the word at position i of the sentence such that each of the m rows correspond to a character embedding of length d_{char}. The purpose of the context encoder is to encode each word of the input with surrounding linguistic features and long-distance dependency information. To that end, we employ the use of a Bi-LSTM network to encode S as a context matrix $C \in \mathbb{R}^{n \times d_{\text{context}}}$ where d_{context} is a hyperparameter of the network. Concretely,

$$C = \text{BiLSTM} \begin{pmatrix} S_1 \parallel \text{CNN}^{[3]}(W^1) \\ \vdots \\ S_n \parallel \text{CNN}^{[3]}(W^n) \end{pmatrix} \quad (1)$$

where S_i denotes the ith row of S and $\parallel$ is the vector concatenation operator. Essentially, for each word, we compose character representations using a CNN with a window size of three and concatenate them to pre-trained word embeddings; we stack the concatenated vectors as rows of a new matrix that is ultimately fed as input to the Bi-LSTM context encoder. The ith row of C, denoted as C_i, represents the entire context centered at the ith word. As an implementation detail, we chose n and m to be the maximum sentence and word length (according to the training data) respectively and pad shorter examples with *zero* vectors.

NER Objective. The network for the NER objective manifests as a stacked Bi-LSTM architecture when we consider both the context encoder and the entity recognition component. Borrowing from residual networks [10], we reinforce the input by concatenating word embeddings to the intermediate context vectors before feeding it to the second Bi-LSTM layer. Concretely, the final entity recognition matrix $R \in \mathbb{R}^{n \times d_{\text{ner}}}$ is composed such that

$$R = \text{BiLSTM} \begin{pmatrix} C_1 \parallel S_1 \\ \vdots \\ C_n \parallel S_n \end{pmatrix}. \quad (2)$$
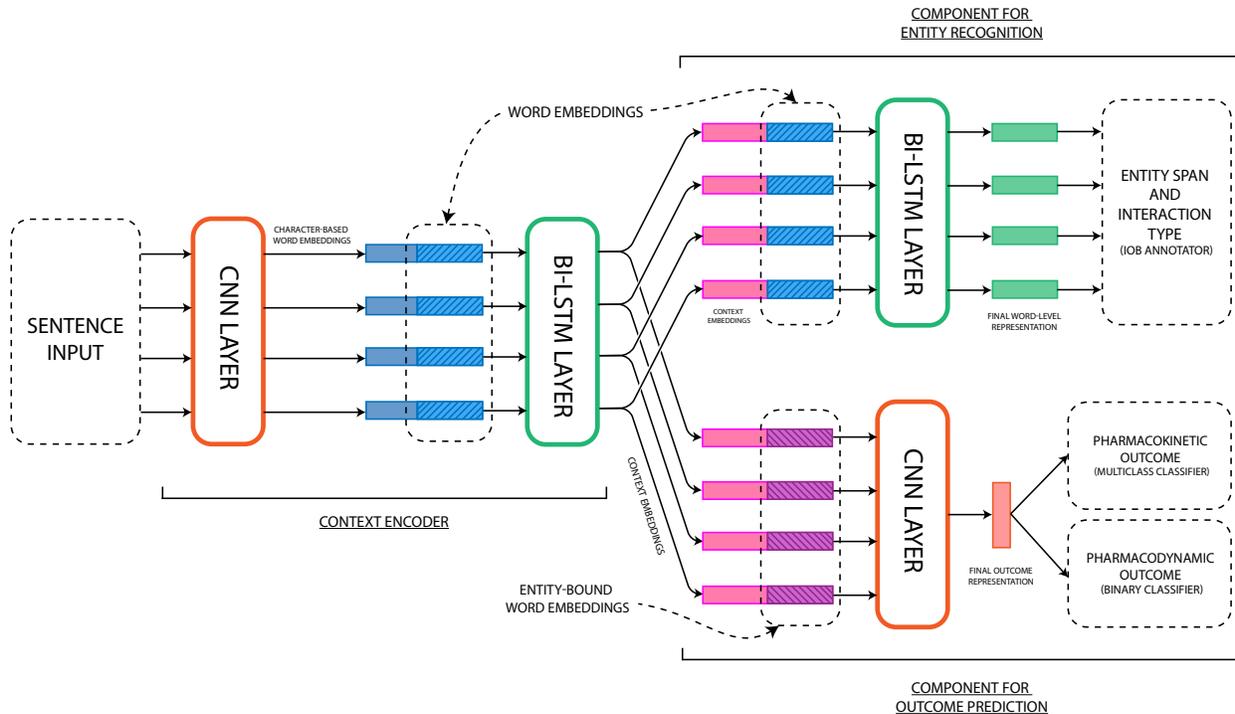
4

Figure 2: The multi-task neural network for DDI extraction

The output at each position $i = 1, \ldots, n$ is

$$\mathbf{q}^i = W^{\mathrm{ner}} R_i + \mathbf{b}^{\mathrm{ner}}$$

where R_i is the ith row of R and $W^{\mathrm{ner}} \in \mathbb{R}^{\ell \times d_{\mathrm{ner}}}$ and $\mathbf{b}^{\mathrm{ner}} \in \mathbb{R}^{\ell}$ are network parameters such that $\ell = 11$ denotes the number of possible IOB tags such as **O**, **B-K**, **I-K** and so on. In order to obtain a categorical distribution, we apply the SoftMax function to $\mathbf{q}^i$ such that

$$\mathbf{p}^i = \mathrm{SoftMax}(\mathbf{q}^i)$$

where $\mathbf{p}^i$ is the vector of probability estimates serving as a categorical distribution over ℓ tags for the word at position i. We optimize by computing the standard categorical cross-entropy loss for each of the n individual tag predictions. The final loss to be optimized is the mean over all n individually-computed losses.

A *stacked* Bi-LSTM architecture improves over a single Bi-LSTM architecture given its capacity to learn *deep* contextualized embeddings. While we showed that the stacked approach is better for this particular task in Section 3.1, it is not necessarily the case that a stacked approach

is better in general. We offer an alternative explanation and motivation for using a stacked architecture for this particular problem based on our initial intuition as follows. First, we note that a standalone Bi-LSTM is not able to handle the inference aspect of NER, which entails learning IOB constraints. As an example, in the IOB encoding scheme, it is not possible for a **I-D** tag to immediately follow a **B-E** tag; in this way, the prediction of a tag is directly dependent on the prediction of neighboring tags. This inference aspect is typically handled by a linear-chain CRF. We believe that a stacked Bi-LSTM at least partially handles this aspect in the sense that the first Bi-LSTM (the context encoder) is given the opportunity to form independent preliminary decisions while the second Bi-LSTM is tasked with to making final decisions (based on preliminary ones) that are *more* globally consistent with respect to IOB constraints.

Outcome Objective. To predict outcome, we construct a secondary branch in the network path that involves convolving over the word and

context embeddings made available in earlier layers. We first define a relation representation $\mathbf{v} \in \mathbb{R}^{d_{\text{rel}}}$ that is produced by convolving with window sizes 3, 4, and 5 over the context vectors concatenated to entity-bound[1] versions of the original input; concretely,

$$\mathbf{v} = \text{CNN}^{[3,4,5]} \begin{pmatrix} C_1 \parallel S'_1 \\ \vdots \\ C_n \parallel S'_n \end{pmatrix}.$$

where S' is the entity-bound version of S. Based on this outcome representation, we compose two separate softmax outputs: one for PK interactions and one for PD interactions. Concretely, the output layers are

$$\mathbf{p}^{\text{PK}} = \text{SoftMax}(W^{\text{PK}}\mathbf{v} + \mathbf{b}^{\text{PK}})$$

and

$$\mathbf{p}^{\text{PD}} = \text{SoftMax}(W^{\text{PD}}\mathbf{v} + \mathbf{b}^{\text{PD}})$$

where $\mathbf{p}^{\text{PK}} \in \mathbb{R}^{\ell^{\text{PK}}}$ and $\mathbf{p}^{\text{PD}} \in \mathbb{R}^{\ell^{\text{PD}}}$ are probability estimates serving as a categorical distribution over the outcome label space for PD and PK respectively and W^{PD}, W^{PK}, $\mathbf{b}^{\text{PK}}$, and $\mathbf{b}^{\text{PD}}$ are parameters of the network. For PK, $\ell^{\text{PK}} = 20$ given there are 20 possible NCI Thesaurus codes corresponding to PK outcomes. For PD, $\ell^{\text{PD}} = 2$ as it is a binary classification problem to assess whether the precipitant and effect pair encoded by S' are linked. We optimize using the standard categorical cross-entropy loss on both objectives.

Training Data. In NLM-180, there is no distinction between triggers and effects; moreover, PK effects are limited to coarse-grained (binary) labels corresponding to *increase* or *decrease* in function measurements. Hence, a direct mapping from NLM-180 to Training-22 is impossible. As a compromise, NLM-180 "triggers" were mapped to Training-22 *triggers* in the case of unspecified and PK interactions. For PD interactions, we instead mapped NLM-180 "triggers" to

Training-22 *effects*, which we believe to be appropriate based on our manual analysis of the data. Since we do not have both *trigger* and *effect* for every PD interaction, we opted to ignore trigger mentions altogether in the case of PD interactions to avoid introducing mixed signals. While trigger recognition has no bearing on relation extraction performance, this policy has the effect of reducing the recall upperbound on NER by about 25% (more later on upperbound). To overcome the lack of fine-grained annotations for PK outcome in NLM-180, we deploy the well-known bootstrapping approach [11] to incrementally annotate NLM-180 PK outcomes using Training-22 annotations as seed examples. To mitigate the problem of semantic drift, in each bootstrap cycle, we re-annotated by hand predictions that were not consistent with the original NLM-180 coarse annotations (i.e., active learning [12]).

Training Procedure. We train the three objective losses (NER, PK outcome, and PD outcome) in an interleaved fashion at the minibatch [13] level. We use word embeddings of size 200 pre-trained on the PubMed corpus [14] as input to the network; these are further modified during back-propagation. For the character-level CNN, we set the character embedding size to 24 with 50 filters over a window size of 3; the final character-CNN composition is therefore of length 50. For each Bi-LSTM, the hidden size is set to 100 such that context vectors are 200 in length. For outcome prediction, we used window sizes of 3, 4, and 5 with 50 filters per window size; the final vector representation for outcome prediction is therefore 150 in length.

A held-out development set of 4 drug labels is used for tuning and validation. The models are trained for 30 epochs with check-pointing; only the check-point with the best performance on the development set is kept for testing. We dynamically set the mini-batch size N_b as a function of the number of examples N such that the number of training iterations is roughly 300 per epoch (and also constant regardless of training data size); concretely, $N_b = \lfloor N/300 \rfloor + 1$. As a form of regularization, we apply dropout [15]

---

[1]We refer to the process of generating examples for relation classification, wherein mentions of candidate entities in the context are replaced with generic tokens that are also learned during back-propagation, as "entity-binding."

at a rate of 50% on the hidden representations immediately after a Bi-LSTM or CNN composition. The outcome objectives are trained such that the gradients of the context encoder weights are downscaled by an order of magnitude (i.e., one tenth) to encourage learning at the later layers. When learning on the NER objective – the main branch of the network – the gradients are not downscaled in the same manner. Moreover, when training on the NER objective, we upweight the loss penalty on "relation" tags (non-**O** tags) by a factor of 10, which forces the model to prioritize differentiation between different types of interactions over span segmentation. We additionally upweight the loss penalty by a factor of 3 on Training-22 examples compared to NLM-180 examples. We optimize using the Adam [16] optimization method. These hyper-parameters were tuned during initial experiments.

3 Results and Discussion

In this section, we present and discuss the results of our cross-validation experiments. We then describe the "runs" that were submitted as challenge entries and present our official challenge results. We discuss these results in Section 3.3.

3.1 Validation Results

We present the results of our initial experiments in Table 3. Evaluations were produced as as result of 11-fold cross-validation over Training-22 with two drug labels per fold. Instead of macro-averaging over folds, and thereby weighting each fold equally, we evaluate on the union of all 11 test-fold predictions.

The upperbound in Table 3 is produced by reducing Training-22 (with gold labels) to our sequence-tagging format and then reverting it back to the original official XML format. Lowered recall is mostly due to simplifying assumptions; e.g., we only consider non-overlapping mentions. For coordinated disjoint cases such as "X and Y inducers", we only considered "Y inducers" in our simplifying assumption. Imperfect precision is due to discrepancies between the tokenization scheme used by our method and that used to produce gold annotations; this leads to the occasional mismatch in entity offsets during evaluation.

Using a stacked Bi-LSTM trained on the original 22 training examples (Table 3; row 1) as our baseline, we make the following observations. Incorporating NLM-180 resulted in a significant boost of more than 20 F1-points in relation extraction performance and more than 10 F1-points in NER performance (Table 3; row 2), despite the lowered upperbound on NER recall as mentioned in Section 2.3. Adding character-CNN based word representations improved performance marginally, more so for NER than relation extraction (Table 3; row 3). We also implemented several tweaks to the pre-processing and post-processing aspects of the model based on preliminary error analysis including (1) using drug class mentions (e.g., "diuretics") as proxies if the drug label is not mentioned directly; (2) removing modifiers such as *moderate*, *strong*, and *potent* so that output conforms to official annotation guidelines; and (3) purging predicted mentions with only stopwords or generic terms such as "drugs" or "agents." These tweaks improved performance by more than two F1-points across both metrics (Table 3; row 4).

Stacked architecture. Based on early experiments with simpler models tuned on *relaxed* matching (not shown in Table 3 and not directly comparable to results displayed in Table 3), we found that a stacked Bi-LSTM architecture improves over a single Bi-LSTM by approximately four F1-points on relation extraction (55.59% vs. 51.55% F1 tuned on the *relaxed* matching criteria). We moreover found that omitting word embeddings as input at the second Bi-LSTM results in worse performance at 52.91% F1.

Temporal Convolution Networks. We also experimented with using Temporal Convolution Networks (TCNs) [17] as a "drop-in" replacement for Bi-LSTMs. Our attempts involved replacing only the second Bi-LSTM with a TCN (Table 3; row 4) as well as replacing both Bi-LSTMs with TCNs (Table 3; row 5). The re-

Model / Data	Entity (Primary)			Relation (Primary)		
	P (%)	**R** (%)	**F** (%)	**P** (%)	**R** (%)	**F** (%)
Stacked Bi-LSTM / Training-22	37.22	40.74	38.90	18.76	23.50	20.86
Stacked Bi-LSTM / Training-22 + NLM-180	49.45	49.79	49.62	42.54	43.56	43.05
Char-CNN + Stacked Bi-LSTM / Training-22 + NLM-180	51.63	50.97	51.30	43.09	44.31	43.69
Char-CNN + Stacked Bi-LSTM + Tweaks / Training-22 + NLM-180	53.72	53.76	**53.74**	46.35	45.66	**46.00**
Char-CNN + Bi-LSTM + TCN / Training-22 + NLM-180	48.82	50.72	49.75	39.68	41.17	39.68
Char-CNN + Stacked TCN / Training-22 + NLM-180	41.46	48.44	44.68	32.15	36.68	34.27
Upperbound due to simplifying assumptions	99.21	74.56	85.14	97.49	81.44	88.74

Table 3: Preliminary results based on 11-fold cross validation over Training-22 with two held-out drug labels per fold. When NLM-180 is incorporated, the training data used for each fold consists of 20 non-held out drug labels from Training-22 and all 180 drug labels from NLM-180.

sults of these early experiments were not promising and further fine-tuning may be necessary for better performance.

3.2 Official Test Results

Our final system submission is based on a stacked Bi-LSTM network with character-CNNs trained on both Training-22 and NLM-180 (corresponding to row 4 of Table 3). We submitted the following three runs based on this architecture:

1. A single model.

2. An ensemble over *ten* models each trained with randomly initialized weights and a random development split. Intuitively, models collectively "vote" on predicted annotations that are kept and annotations that are discarded. A unique annotation (entity or relation) has one vote for each time it appears in one of the *ten* model prediction sets. In terms of implementation, unique annotations are incrementally added (to the final prediction set) in order of descending vote count; subsequent annotations that conflict (i.e., overlap based on character offsets) with existing annotations are discarded. Hence, we loosely refer to this approach as "voting-based" ensembling.

3. A single model with pre/post-processing rules to handle modifier coordinations; for example, "X and Y inducers" would be correctly identified as two distinct entities cor-

responding to "X inducers" and "Y inducers." Here, we essentially encoded "X and Y inducers" as a single entity when training the NER objective; during test time, we use simple rules based on pattern matching to split the joint "entity" into its constituents.

Eight teams participated in task 1 while four teams participated in task 2. We record the relative performance of our system (among others in the top 5) on the two official test sets in Table 4. For each team, we only display the performance of the best run for a particular test set. Methods are grouped by the data used for training and ranked in ascending order of *primary* relation extraction performance followed by entity recognition performance. We also included a single model trained solely on Training-22, that was not submitted, for comparison. Our voting-based ensemble performed best among the three systems submitted by our team on both NER and relation extraction. In the official challenge, this model placed *second* overall on both NER and relation extraction.

Tang et al. [21] boasts the top performing system on both tasks. In addition to Training-22 and NLM-180, the team trained and validated their models on a set of 1148 sentences sampled from DailyMed labels that were manually annotated according to official annotation guidelines. Hence, strictly speaking, their method is not directly comparable to ours given the significant difference in available training data.

8

Training Data	Method	Entity (Primary)			Relation (Relaxed)			Relation (Primary)		
		P (%)	**R** (%)	**F** (%)	**P** (%)	**R** (%)	**F** (%)	**P** (%)	**R** (%)	**F** (%)
Training-22	Ours (Single model)[1]	21.74	33.84	26.47	32.95	34.57	33.74	15.49	15.42	15.45
Training-22 + NLM-180	Zhang and Kordjamshidi [18]	17.00	15.86	16.41	-	-	-	-	-	-
	Akhtyamova and Cardiff [19]	37.96	20.39	26.53	-	-	-	-	-	-
	Ours (Modifier Coordination)	28.63	27.48	28.04	38.97	30.62	34.30	21.94	16.57	18.88
	Ours (Single model)	29.64	31.58	30.58	38.16	33.80	35.85	21.28	18.09	19.55
	Dandala et al. [20]	41.94	23.19	29.87	46.60	29.78	36.34	25.24	16.10	19.66
	Ours (Ensemble)	29.50	37.45	**33.00**	40.55	38.36	**39.43**	22.08	21.13	**21.59**
Training-22 + NLM-180 + HS[2]	Tang et al. [21]	55.23	38.32	**45.25**	71.70	45.46	**55.64**	54.43	32.76	**40.90**

(a) System performance on Official Test Set 1

Training Data	Method	Entity (Primary)			Relation (Relaxed)			Relation (Primary)		
		P (%)	**R** (%)	**F** (%)	**P** (%)	**R** (%)	**F** (%)	**P** (%)	**R** (%)	**F** (%)
Training-22	Ours (Single model)[1]	27.77	33.31	30.29	38.38	40.85	39.58	16.17	15.39	15.77
Training-22 + NLM-180	Zhang and Kordjamshidi [18]	17.13	21.89	19.22	-	-	-	-	-	-
	Akhtyamova and Cardiff [19]	37.76	24.07	29.40	-	-	-	-	-	-
	Ours (Modifier Coordination)	34.92	30.33	32.46	46.72	36.13	40.75	21.39	15.67	18.09
	Dandala et al. [20]	44.61	29.31	35.38	50.07	36.86	42.46	22.99	16.83	19.43
	Ours (Single model)	35.29	33.47	34.36	44.93	37.64	40.96	22.51	17.71	19.82
	Ours (Ensemble)	36.68	40.02	**38.28**	49.51	44.27	**46.74**	22.53	21.13	**23.55**
Training-22 + NLM-180 + HS[2]	Tang et al. [21]	51.23	42.39	**46.39**	66.99	49.58	**56.98**	48.92	34.49	**40.46**

(b) System performance on Official Test Set 2

Table 4: Comparison of our method with that of other teams in the top 5. Only the best performing method of each team is shown; methods are grouped by available training data and ranked in ascending order by relation extraction (primary) performance followed by entity recognition performance.

[1] This model was not submitted and is shown for reference only

[2] HS refers to a private dataset of 1148 sentences manually-annotated by Tang et al. [21] according to official guidelines

3.3 Discussion

While precision was similar between the three systems (with exceptions), we observed that our ensemble-based system benefited mostly from improved recall. This aligns with our initial expectation (based on prior experience with deep learning models) that an ensemble-based approach would improve stability and accuracy with deep neural models. Although including NLM-180 as training data resulted in significant performance gains during 11-fold cross validation, we find that the same improvements were not as dramatic on either test sets despite the 800% gain in training data. As such, we offer the following analysis. First, we suspect that there may be a semantic or annotation drift between these datasets as annotation guidelines evolve over time and as annotators become more experienced. To our knowledge, the datasets were annotated in the following order: NLM-180, Training-22, and finally Test Sets 1 and 2; moreover, Test Sets 1 and 2 were annotated by separate groups of annotators. Second, having few but higher quality examples may be more advantageous than having many but lower quality examples, at least for this particular task where evaluation is based on matching exact character offsets. Finally, we note that the top performing system exhibits superior performance on Test Set 1 compared to Test Set 2; interestingly, we observe an inverse of the scenario in our own system. This may be an indicator that our system struggles with data that is more "sparse" (as previously defined in Section 2.1).

4 Conclusion

We presented a method for jointly extracting precipitants and their interaction types as part of a multi-task framework that additionally detects interaction outcome. Among three "runs", a ten model voting-ensemble was our best performer. In future efforts, we will experiment with Graph Convolution Networks [22] over dependency trees as a "drop-in" replace for Bi-LSTMs to assess its suitability for this task.

Acknowledgements

References

[1] Patrick J McDonnell and Michael R Jacobs. Hospital admissions resulting from preventable adverse drug reactions. *Annals of Pharmacotherapy*, 36(9):1331–1336, 2002.

[2] Troyen A Brennan, Lucian L Leape, Nan M Laird, Liesi Hebert, A Russell Localio, Ann G Lawthers, Joseph P Newhouse, Paul C Weiler, and Howard H Hiatt. Incidence of adverse events and negligence in hospitalized patients: results of the harvard medical practice study i. *New England journal of medicine*, 324(6):370–376, 1991.

[3] Eric J Thomas, David M Studdert, Joseph P Newhouse, Brett IW Zbar, K Mason Howard, Elliott J Williams, and Troyen A Brennan. Costs of medical injuries in utah and colorado. *Inquiry*, pages 255–264, 1999.

[4] Molla S Donaldson, Janet M Corrigan, Linda T Kohn, et al. *To err is human: building a safer health system*, volume 6. National Academies Press, 2000.

[5] U.S. National Institute of Standards and Technology. Text Analysis Conference (TAC) 2018. `https://tac.nist.gov/2018/index.html`.

[6] National Library of Medicine. NLM-DDI CD corpus: DailyMed cardiovascular product labels annotated with drug-drug interactions. `https://lhce-brat.nlm.nih.gov/NLMDDICorpus.htm`.

[7] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics.

[8] Tung Tran and Ramakanth Kavuluru. Predicting mental conditions based on "history of present illness" in psychiatric notes with deep neural networks. *Journal of Biomedical Informatics*, pages S138–S148, 2017.

[9] Tung Tran and Ramakanth Kavuluru. An end-to-end deep learning architecture for extracting protein-protein interactions affected by genetic mutations. *Journal of Biological Databases and Curation (Database)*, 2018:1–13, 2018.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] Rosie Jones, Andrew McCallum, Kamal Nigam, and Ellen Riloff. Bootstrapping for text learning tasks. In *IJCAI-99 Workshop on Text Mining: Foundations, Techniques and Applications*, volume 1, 1999.

[12] Burr Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012.

[13] Quoc V Le, Jiquan Ngiam, Adam Coates, Abhik Lahiri, Bobby Prochnow, and Andrew Y Ng. On optimization methods for deep learning. In *Proceedings of the 28th International Conference on Machine Learning*, pages 265–272. Omnipress, 2011.

[14] Sampo Pyysalo, Filip Ginter, Hans Moen, Tapio Salakoski, and Sophia Ananiadou. Distributional semantics resources for biomedical text processing. In *Proceedings of 5th International Symposium on Languages in Biology and Medicine*, pages 39–44, 2013.

[15] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference for Learning Representations (ICLR 2015)*, 2015.

[17] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[18] Yue Zhang and Parisa Kordjamshidi. PE-TU participation at TAC 2018 drug-drug interaction extraction from drug labels. In *Proceedings of the 2018 Text Analysis Conference (TAC 2018)*, 2018.

[19] Liliya Akhtyamova and John Cardiff. Extracting drug-drug interactions with character-level and dependency-based embeddings. In *Proceedings of the 2018 Text Analysis Conference (TAC 2018)*, 2018.

[20] Bharath Dandala, Diwakar Mahajan, and Ananya Poddar. IBM Research system at TAC 2018: Deep learning architectures for drug-drug interaction extraction from structured product labels. In *Proceedings of the 2018 Text Analysis Conference (TAC 2018)*, 2018.

[21] Siliang Tang, Qi Zhang, Tianpeng Zheng, Mengdi Zhou, Zhan Chen, Lixing Shen, Xiang Ren, Yueting Zhuang, Shiliang Pu, and Fei Wu Wu. Two step joint model for drug drug interaction extraction. In *Proceedings of the 2018 Text Analysis Conference (TAC 2018)*, 2018.

[22] Yuhao Zhang, Peng Qi, and Christopher D Manning. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.