TRIGGER LABELING

Language Specific Issue

To address the language specific issues, we treat event detection as a sequence labeling task rather than classification. Sentences are tagged in the BIO scheme, where each token is labeled as B-type if it is the beginning of an event trigger with event type type, or I-type if it is inside a trigger, or O otherwise. Our first labeling model is a word-based bidirectional LSTM network (BiLSTM) with a CNN layer and a CRF layer as shown in Figure 1.

**Word-based Convolution BiLSTM-CRF Model**

In this section, we introduce the components (layers) in our Convolution BiLSTM-CRF (C-BiLSTM-CRF) network one-by-one from bottom to top.

**LSTM Network**

Recurrent neural networks (RNNs) maintain a memory based on historical contextual information, which makes them a natural choice for processing sequential data. Unfortunately, it is difficult for standard RNNs to capture long range dependencies due to vanishing/exploding gradients \cite{bengio1994learning}. Long Short-Term Memory Network \cite{hochreiter1997long} is explicitly designed to solve the long-term dependency problem through purpose-built memory cells. They consist of several multiplicative gates that control the proportion of information to forget and to store in the cell states. In this paper, we apply a variation of LSTM units, Gated Recurrent Unit (GRU) \cite{cho2014learning}, which is found to be superior to LSTM on a suit of tasks by Chung et al. \shortcite{chung2014empirical}. The GRU is implemented as the following formulas:

In these formulas, the $W\_\*$ variables are the weight matrices and the $b\_\*$ variables are the biases. $\sigma(\cdot)$ is the element-wise sigmoid function and $\odot$ is the element-wise product.

Formally, we use $w=\{w\_1, w\_2, …, w\_T\}$ to represent an input sentence where $w\_t$ is the $t$-th word, and $x\_t$ is its feature vector (e.g. word embedding). At each time step $t$, a GRU takes $x\_t$ as input and computes the hidden state $h\_t$ (also called output vector) by reset gate and update gate. The reset gate $r\_t$ controls how much and what information from the previous hidden state should be reset, and the update gate $z\_t$ determines how much the unit updates its previous hidden state.

**BiLSTM Network**

For our event extraction task, if we access to both past and future contexts for a given time, we can make use of more sentence-level information and make better prediction. This can be done by bidirectional LSTM networks \cite{graves2005framewise,graves2013speech}. Figure 1(a) shows the layers of a BiLSTM trigger identification model.

A forward LSTM network computes the hidden state $\overrightarrow{h\_t}$ of the past (left) context of the sentence at word $w\_t$, while a backward LSTM network reads the same sentence in reverse and outputs $\overleftarrow{h\_t}$ given the future (right) context. In our implementation, we concatenate these two vectors to form the hidden state of a BiLSTM network, i.e. $h\_t = [\overrightarrow{h\_t}; \overleftarrow{h\_t}]$.

**Convolutional Neural Network**

Convolutional neural networks (CNNs) are originally applied to computer vision to capture salient local features \cite{lecun1998gradient}. Previous studies on event extraction \cite{nguyen2015event,chen2015event,feng2016language} have gradually shown that CNN architectures are effective to capture semantic features similar to n-grams, but represent them in a more compact way. We employ a convolutional neural network as illustrated in Figure 1(b) to extract local contextual information for each word in a sentence.

Specifically, given a sentence containing $n$ words $\{x\_1, x\_2,$ $…, x\_n\}$, for every word in the sentence, we want to extract local contextual information to help predict if the current word is an event trigger. The current word $x\_i$ along with its context constitutes the input of CNN. Let $x\_{i:i+j}$ be the window of words from $x\_i$ to $x\_{i+j}$, and $2k+1$ be the fixed context size. So the context window $c\_i$ (padded when necessary) where the current word $x\_i$ is in the middle can be written as

\begin{equation}c\_i = x\_{i-k:i+k} = [x\_{i-k}, \ldots, x\_{i}, \ldots, x\_{i+k}].\end{equation}

Before being fed into a convolution layer, each word $x\_i$ is transformed into a d-dimensional word vector $\textbf{x}\_i$ by looking up the embedding table. As a result, the original context $c\_i$ is transformed into a matrix $\textbf{c}\_i = [\textbf{x}\_{i-k}, \ldots, \textbf{x}\_i, \ldots, \textbf{x}\_{x+k}]$ of size $(2k+1) \times d$. The matrices $\textbf{c}\_1, \ldots, \textbf{c}\_n$ are then passed through a convolution layer and a max pooling layer one by one.

In the convolution layer, we utilize a set of kernels $\{\textbf{w}\_1, \textbf{w}\_2, \ldots, \textbf{w}\_m\}$ with varying widths to extract semantic features of various granularities. For every context matrix $\textbf{c}\_i$, a kernel $\textbf{w}\_j$ of width $l$ is applied to all possible windows of $l$ words inside the context (i.e., $x\_{i-k:i-k+l-1}$, …, $x\_{i+k-l+1:i+k}$). And $\textbf{w}\_j$ can be essentially seen as a weight matrix of size $l \times d$. For example, the convolution operation involves kernel $\textbf{w}\_j$ over the window $x\_{a:a+l-1}$ can be express as:

\begin{equation} s\_{j,t}=f(w\_j \cdot x\_{t:t+l-1} + b\_j), 1 \leq j \leq m, i-k \leq t \leq i+k-l+1 \end{equation}

where $b$ is a bias term and $f$ is a non-linear function such as hyperbolic tangent. The convolution result is a feature map $\textbf{s}\_j \in \mathbb{R}^{2k-l+2}$.

We then perform a max-over-time pooling operation \cite{collobert2011natural} over each feature map $\tilde{s}\_j=\max{\textbf{s}\_j}$ so that only the largest number is recorded. One property of pooling is that it produces a fixed size output vector, which enables us to apply variable kernel sizes. And by performing the max operation, we are keeping the most salient information. Finally, we take the fixed length output vector $\textbf{C}=[\tilde{s}\_1, \tilde{s}\_2, \ldots, \tilde{s}\_m]$ as a representation of local contextual information about current word.

In our implementation, the context window size is 7 (3 words to the left and to the right of a center word), and kernel sizes from 2 to 7 to encode the semantics of n-grams with various granularities. Each kernel generates 32 feature maps.

**Output Layer**

For each word $w\_i$ in the sentence, we concatenate the bidirectional sentence-level features $B\_i$ learned by BiLSTM, and the contextual semantic features $C\_i$ extracted by CNN, into a single vector $F\_i=[B\_i;C\_i]$. To compute the confidence of each label, the final output vector $F\_i \in \mathbb{R}^{2d\_{gru}+d\_{cnn}}$, where $d\_{gru}$ is the dimension of the GRU unit and $d\_{cnn}$ is the number of feature maps in CNN layer, is fed into a fully connected linear layer.

\begin{equation} O\_i = W\_sF\_i+b\_s \end{equation}

$\text{W}\_s \in \mathbb{R}^{n\_{e} \times (2d\_{gru} + d\_{cnn})}$ is the transformation matrix and $\text{O}\_i=[O\_{i,1}, \ldots, O\_{i,{n\_e}}]$ is the final output vector, where $n\_{e}$ is the number of distinct labels, and the $e$-th element $O\_{i, e}$ indicates the score for label $e$. Then $\text{O}\_i \in \mathbb{R}^{n\_e}$ is applied to a softmax function to estimate a probability distribution over all possible labels. In the end, we choose the label that obtains maximum probability as a prediction of $y\_i$.

**Errata table**

However, this word-base model still cannot solve the inconsistency problem caused by However, this word-base model still cannot solve the inconsistency problem caused by inside-word triggers. Inspired by Chen and Ji \shortcite{chen2009language}, we construct a global errata table to record some frequent appearances of tokens and triggers in the training set. In our experiments, if 80\% occurrences of a token inside a word should be labeled as triggers with the same event type, we then add this ``word$-$token$-$type'' triple into the table. During testing, if a word has an entry in the errata table, we regard its token as a trigger with the event type according to the corresponding triple directly. For example, if ``\begin{CJK}{UTF8}{gbsn}击毙$-$击$-$\emph{Attack}\end{CJK}'' and ``\begin{CJK}{UTF8}{gbsn}击毙$-$毙$-$\emph{Die}\end{CJK}'' are two triples in the errata table, word-based C-BiLSTM model can identify all inside-word triggers in S5 correctly.

**Character-based Convolution BiLSTM Model**

Despite of the effectiveness of errata table, word-based method is not a flawless solution to language specific issue, because it only recognizes triggers across words or frequent inside-word triggers appearing in training data.

Ideally, character-based methods may solve both inconsistency problems. It uses the same \texttt{\{BIO\}-type} tagging scheme as word-based model, however, to label each character rather than each word. For better understanding, result sequences of sentences S4 and S5 labeled by two models are listed in Table~\ref{six}.

As shown in Figure 2, character-based C-BiLSTM have a similar network architecture as word-based C-BiLSTM. The main difference is that character-based model tags sentence character by character, while word-based model tags word by word. They also differ in their input layers: a character-based C-BiLSTM concatenates character embedding and word embedding, while a word-based C-BiLSTM only uses word embedding.

**CRF Layer**

As mentioned in Section ~\ref{output}, regarding the final feature vector $\text{F}$ as an input to a softmax classifier is a straightforward but effective way to make independent labeling decisions. However, the independent classification decisions are limiting when there are strong dependencies between tags in a sentence. For example, in trigger labeling, \texttt{B-attack} is more likely to be followed by \texttt{B-die}, while \texttt{I-die} cannot follow \texttt{B-attack}. Output layers in C-BiLSTM models discard this kind of labeling information.

Therefore, in this section, we propose a character-based convolution BiLSTM-CRF model (C-BiLSTM-CRF) that considers the correlations between labels in neighborhoods and jointly decodes the best sequence of labels via a CRF layer. This kind of CRF architecture is similar to the ones presented in other sequence labeling tasks, such as chunking and NER \cite{\*\*\*}.

We consider a training scheme which takes into account the sentence structure: given the predictions of all tags by our network for all words in a sentence, and given a score for going from one tag to another tag, we want to encourage valid paths of tags during training, while discouraging all other paths.

We select a subset of all distinct labels and generate a heat map Figure~\ref{figure3} from the transition scores between them. We find that:

1. A CRF layer models the hard constraints nicely. For a specific event type \texttt{Type}, \texttt{I-Type} can only follow \texttt{B-Type} or \texttt{I-Type}. In other word, if current label is \texttt{I-Type}, the transition scores for previous labels, except \texttt{B-Type} and \texttt{I-Type}, are supposed to be very low. For example, as the forth column in heatmap shows, going from \texttt{B-Die} to \texttt{I-Die} is encouraged, while other transitions, like \textttt{B-Attack} to \texttt{I-Die} or \texttt{I-Injure} to \texttt{I-Die} are discouraged.
2. A CRF layer models the co-occurrence of several event types. For example, an \emph{Attack} event is usually followed by an event about \emph{Injure} or \emph{Die}. As a result, in the first row, the transition score for going from \texttt{B-Attack} to \texttt{B-Injure} and \texttt{B-Die} is much higher than other labels start with \texttt{B-}, such as \texttt{B-Be-born}.

Figure~\ref{figure43} shows whole transition matrix after training. The darkest color grids always appear right below the diagonal, which suggests that for a given label \texttt{B-type}, its most likely following label is \texttt{I-Type}.

As we can conclude from Figure~\ref{figure4}, during training, CRF layer gradually learns a score for going from one label to another label and encourages valid paths of labels, while discouraging other invalid or unusual paths.

因为对于以I开头的标签，例如I-type，它只能出现在B-type的后面，所以对于I-type这一列，得分最高的应该是B-type，在热图上就应该是在前面。

instead of modeling tagging decisions independently,

ARGUMENT LABELING

Argument labeling can be split into two subtasks: argument identification and argument classification. Argument identification aims to determine whether a candidate argument serves as a participant or an attribute with a specific role or not, and argument classification assigns a role to each identified argument. Like trigger labeling, we propose a joint argument labeling model that jointly performs argument identification and argument classification.

However, it is worth mentioning that unlike trigger labeling, argument labeling is no longer a sequence tagging task, but a simpler classification task. ACE dataset provides ground truth about entity mention, value, time expression recognition, and it guarantees that gold standard arguments are annotated exactly from those particular words. As a result, the list of candidate arguments we used for a trigger includes all and only those entity mentions, values, and time expressions that appear in the same sentence as the trigger. Training instances are created by pairing each trigger with each of its argument candidates. For instance, there are three triggers (bold words), and three entities (italic words) in S7, which together makes up nine pairs of trigger and argument candidate to be classified.

The idea of C-BiLSTM model in trigger labeling is also suitable for argument labeling: given a typed trigger recognized in previous trigger labeling stage, together with a candidate argument, we utilize a bidirectional LSTM to obtain a sentence-level feature, concatenated with a CNN-extracted local lexical feature, to predict what role the candidate plays. A special role \texttt{None} indicates the candidate does not play any role in the event. Language specific issues do not exist in argument labeling, as all candidate arguments are words. The model we proposed in this section is a word-based model. Next, we will present the detailed differences between the convolution directional LSTM models used in trigger labeling and argument labeling.

不需要用character-based方法来判断了