Selective Decoding for Cross-lingual Open Information Extraction

Sheng Zhang
Johns Hopkins University
zsheng2@jhu.edu

Kevin Duh
Johns Hopkins University
kevinduh@cs.jhu.edu

Benjamin Van Durme
Johns Hopkins University
vandurme@cs.jhu.edu

Abstract

Cross-lingual open information extraction is the task of distilling facts from the source language into representations in the target language. We propose a novel encoder-decoder model for this problem. It employs a novel selective decoding mechanism, which explicitly models the sequence labeling process as well as the sequence generation process on the decoder side. Compared to a standard encoder-decoder model, selective decoding significantly increases the performance on a Chinese-English cross-lingual open IE dataset by 3.87-4.49 BLEU and 1.91-5.92 F_1. We also extend our approach to low-resource scenarios, and gain promising improvement.

1 Introduction

Cross-lingual open information extraction is defined as the task of extracting facts from the source language (e.g., Chinese text in Fig. 1(a)) and representing them in the target language (e.g. English predicate-argument information in Fig. 1(c))[1]. It is a challenging task and of great importance to solve the cross-lingual portability issues of various NLP systems which are in the support of open information extraction (Sudo et al., 2004). Additionally, there is often a great demand for rapid access to information across languages, especially when a large-scale incident occurs (Lu et al., 2016).

Conventional solutions decompose the task as a pipeline of machine translation followed by open information extraction (or vice versa), which causes a deviation since machine translation attaches equal importance to adequacy and fluency of the intermediate translation results (Snover et al., 2009), whereas the final goal focuses more on extracting correct predicates and arguments.

Recently Zhang et al. (2017a) proposes an end-to-end solution that outperforms the conventional pipeline solutions. They recast cross-lingual open IE as a sequence-to-sequence learning problem by converting the target facts in the tree structure (Fig. 1(c)) into a linear form called linearized PredPatt[2] (Fig. 1(d)), and employ a stan-

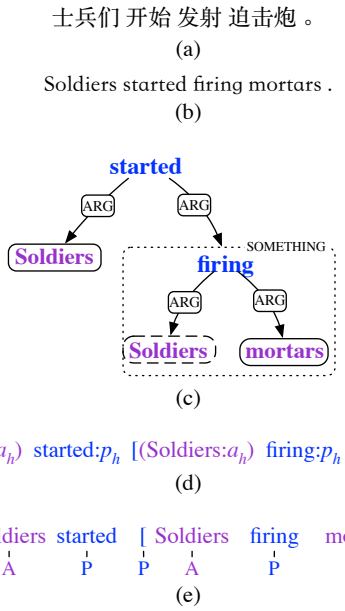

Figure 1: Example of cross-lingual open IE: Chinese input text (a), English translation (b), English predicate-argument information (c), linearized PredPatt output (d) and output with separated predicate and argument labels (e).

[1]The predicate-argument information is normally represented by relation tuples. Here, we use a richer representation (i.e., a tree structure) adopted by PredPatt (White et al., 2016), a lightweight tool for identifying predicate-argument information, available at https://github.com/hltcoe/PredPatt.

[2]Linearized PredPatt is converted from the PredPatt tree structure by taking an in-order traversal of every node in the tree. See Zhang et al. (2017a) for details.

Proceedings of the The 8th International Joint Conference on Natural Language Processing, pages 832–842,
Taipei, Taiwan, November 27 – December 1, 2017 ©2017 AFNLP

dard encoder-decoder model to address the problem (from (a) to (d) in Fig. 1). In the linearized PredPatt (Fig. 1(d)), special labels are appended to tokens as type indications. Brackets and parentheses have to be inserted to delimit predicate and argument spans. Such a workaround inevitably expands the vocabulary space and increases the burdens on the decoder, which is not ideal for sparse data scenarios and limits the overall performance.

In this paper, we reformulate cross-lingual open IE as a sequence generation and labeling problem (from (a) to (e) in Fig. 1) by separating the predicate and argument labels from the target linearized PredPatt, and removing unnecessary parentheses. We propose a novel encoder-decoder model which employs a selective decoding mechanism to explicitly model the sequence labeling as well as the sequence generation process. The new model substantially reduces the vocabulary space, eases the burden on the decoder, and leads to a significant gain of performance. And the natural of the selective decoding mechanism enables a joint training strategy that optimizes sequence generation and labeling simultaneously. In addition, we introduce an adapted beam search algorithm to further improve the prediction quality.

Experimental results demonstrate that our model employing the selective decoding mechanism significantly outperforms the previous end-to-end solution not only on a Chinese-English dataset, but also in low-resource cross-lingual open IE scenarios.

2 Problem Formulation

Our goal is to learn a model which directly maps a sentence input X in the source language into a sentence Y in the target language, and simultaneously labels each token in Y with type information T. For cross-lingual open IE, the types are *predicate* and *argument*. It is important to label types because they are used to annotate predicates and arguments in the generated tokens. Formally, we regard the input as a sequence $X = x_1, \cdots, x_{|X|}$, and the output as two sequences (Y, T): (1) the sentence in target language $Y = y_1, \cdots, y_{|Y|}$, and (2) the type information $T = t_1, \cdots, t_{|Y|}$, where $t_i \in \mathcal{T}$ is the label for the token y_i, and $|X|$ and $|Y|$ are the length of the sequence X and Y respectively. Our model maps X into (Y, T) using a conditional probability which is decomposed as:

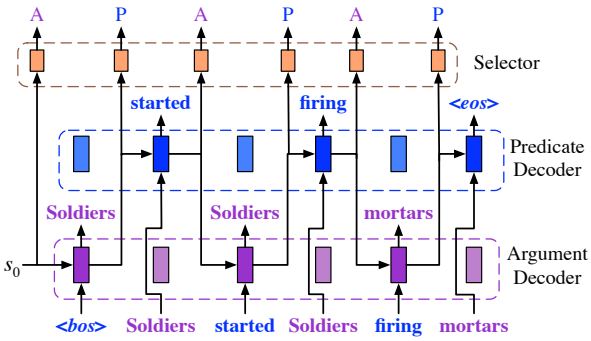

Figure 2: Selective decoding process. (Brackets and attention layers are omitted.)

$$P(Y, T \mid X) = \prod_{i=1}^{|Y|} P(y_i, t_i \mid y_{<i}, t_{<i}, X)$$
$$= \prod_{i=1}^{|Y|} P(y_i \mid y_{<i}, t_{\leq i}, X) P(t_i \mid y_{<i}, t_{<i}, X) \quad (1)$$

where $y_{<i} = y_1 \cdots y_{i-1}$ and $t_{\leq i} = t_1 \cdots t_i$.

Equation (1) can be interpreted as at each decoding time step, the model first decides which type (label) of tokens to generate, and then generates a token for that type.

3 Proposed Model

To learn the factored conditional probabilities as shown in Equation (1), we propose a novel encoder-decoder model with the selective decoding mechanism: on the encoder side, an input sentence X is encoded into vector representations; on the decoder side, the selective decoding mechanism employs multiple decoders each of which learns the conditional probability of decoding a specific type of token (i.e., $P(y_i \mid y_{<i}, t_{\leq i}, X)$), and a selector learning to decide which type of decoder to use (i.e., $P(t_i \mid y_{<i}, t_{<i}, X)$) at each decoding time step.

Fig. 2 illustrates the selective decoding process for the example shown in Fig. 1(e). s_0 is the initial decoder hidden state initialized by the last hidden state of the encoder. Special tokens $\langle bos \rangle$ and $\langle eos \rangle$ are added to the beginning and the end of the sequence to indicate the start and the finish of decoding. The connections at each decoding time step are dynamically changing according to the decision of the selector. Specifically, at the decoding time step i, firstly the selector on the top decides to use which type of decoder $D_{t_i} \in \{D_{\mathrm{P}}, D_{\mathrm{A}}\}$[3], and

[3] D_{P} stands for the predicate decoder, and D_{A} for the argument decoder.

then the decoder D_{t_i} decodes the token y_i which is naturally given the label t_i.

In addition to distinguish between labels, the multiple decoders used by the selective decoding mechanism has two prominent advantages over the single standard RNN decoder: (1) Multiple decoders learn different conditional probability distributions for predicate and argument generation respectively. For instance, given the same input token "*wanted*", the predicate decoder would like to next generate tokens such as "*to*" and "*by*" which starts a prepositional phrase, whereas the argument decoder would be in favor of tokens such as "*a*" and "*him*" which starts a direct object. (2) Multiple decoders reduce the decoder vocabulary size, which eases the burden of sequence generation. Moreover, we propose an efficient architecture that supports batch training of the model. The details of the architecture are described in the **Decoder with Selective Decoding** section.

3.1 Encoder

The encoder employs a bi-directional recurrent neural network (Schuster and Paliwal, 1997) to encode the input sequence $X = x_1, \cdots, x_{|X|}$ into a sequence of hidden states $\boldsymbol{h} = h_1, \cdots, h_{|X|}$. Each hidden state h_i in $\boldsymbol{h}$ is a concatenation of a left-to-right hidden state $\overrightarrow{h_i} \in \mathbb{R}^n$ and a right-to-left hidden state $\overleftarrow{h_i} \in \mathbb{R}^n$,

$$h_i = \begin{bmatrix} \overleftarrow{h}_i \\ \overrightarrow{h}_i \end{bmatrix} = \begin{bmatrix} \overleftarrow{f}(x_i, \overleftarrow{h}_{i+1}) \\ \overrightarrow{f}(x_i, \overrightarrow{h}_{i-1}) \end{bmatrix},$$

where $\overleftarrow{f}$ and $\overrightarrow{f}$ are two L-layer stacked LSTMs units (Hochreiter and Schmidhuber, 1997).

3.2 Decoder with Selective Decoding

Unlike the single standard RNN decoder (Bahdanau et al., 2014), which recurrently uses the same decoder to generate tokens, our model dynamically selects different decoders at each decoding time step to generate tokens (Fig. 2). However, since the decoding path may be different for each input sequence X, directly running the selective decoding process suffers from a key technical issue: it does not support batched computation, which makes them slow and unwieldy for large-scale NLP tasks (Bowman et al., 2016).

To address this issue, we introduce a general decoding architecture that is applicable to all selective decoding processes. The detailed connection

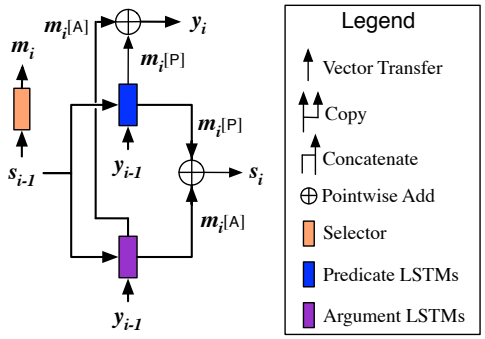

Figure 3: Detailed connection at a decoding step. (Attention layers are ommited.)

in the architecture is shown in Fig. 3. At each decoding time step, the model feeds the input token and the previous hidden state to all types of decoders, and use a mask vector created by the selector to select the decoder output to generate tokens and update the hidden state.

Formally, let $s_i \in \mathbb{R}^n$ denote the hidden state at decoding time step i. The last left-to-right hidden state $\overrightarrow{h}_{|X|}$ from the encoder is used to initialize the first hidden state s_0 in the decoder.

Selector: At the decoding time step i, given the sequence of encoder hidden states $\boldsymbol{h}$ and the previous decoder hidden state s_{i-1}, the selector computes the conditional probability of t_i (i.e., the type of decoder to use) as:

$$\begin{aligned} P(t_i \mid y_{<i}, t_{<i}, X) &= g(t_i, s_{i-1}, \boldsymbol{h}) \\ &= \text{softmax}(U_o s_{i-1} + C_o c_{i-1} + b_o)[t_i], \end{aligned} \quad (2)$$

where $U_o \in \mathbb{R}^{|\mathcal{T}| \times n}$, $C_o \in \mathbb{R}^{|\mathcal{T}| \times n}$ and $b_o \in \mathbb{R}^{|\mathcal{T}|}$ are weight matrices and bias.[4] $[t_i]$ indexes the element of a vector that corresponds to the type t_i.

Attention: The context vector c_{i-1} captures the attention to the encoder side (Bahdanau et al., 2014; Luong et al., 2015), computed as a weighted sum of encoder hidden states: $c_{i-1} = \sum_j^{|X|} a_{(i-1)j} h_j$. The weight $a_{(i-1)j}$ is computed by:

$$a_{(i-1)j} = \frac{\exp\left(\text{score}(s_{i-1}, h_j)\right)}{\sum_{j'=1}^{|X|} \exp\left(\text{score}(s_{i-1}, h_{j'})\right)}, \quad (3)$$

where $\text{score}(s_{i-1}, h_j) = s_{i-1}^{\intercal} W_a h_j$, and $W_a \in \mathbb{R}^{n \times 2n}$ is a transform matrix.

Hidden State Update: According to the conditional probability $P(t_i \mid y_{<i}, t_{<i}, X)$, a mask vector $\boldsymbol{m}_i$ is created, which is used to mask out the

[4] $|\mathcal{T}|$ is the number of token types. In the example shown in Fig. 3, $\mathcal{T} = \{\texttt{P}, \texttt{A}\}$, and $|\mathcal{T}| = 2$.

decoders' hidden states,

$$\boldsymbol{m}_i[t_i] = \begin{cases} 1, \text{ if } t_i = \underset{t'_i \in \mathcal{T}}{\text{argmax}} P(t'_i \mid y_{<i}, t_{<i}, X) \\ 0, \text{ otherwise} \end{cases}$$

Then the hidden state s_i for the decoding time step i is computed by:

$$s_i = \sum_{t_i \in \mathcal{T}} \boldsymbol{m}_i[t_i] f_{t_i}(y_{i-1}, s_{i-1}, c_i)$$

where c_i is the context vector capturing the *attention*, computed in the same way as Equation (3). f_{t_i} is L-layer stacked LSTMs for the type t_i. In Fig. 3, there is an L-layer stacked LSTMs for generating predicate tokens f_{P}, and another L-layer stacked LSTMs for generating argument tokens f_{A}. They have untied parameters.

Token Generation: The conditional probability of the token y_i with the type t_i is defined as:

$$\begin{aligned} &P(y_i \mid y_{<i}, t_{\leq i}, X) = g'(y_{i-1}, s_{i-1}, \boldsymbol{h}, \boldsymbol{m}_i) \\ &= \text{softmax}(U'_o s_i + C'_o c_i + b'_o)[y_i], \end{aligned} \tag{4}$$

where $U'_o \in \mathbb{R}^{|\mathcal{V}| \times n}$, $C'_o \in \mathbb{R}^{|\mathcal{V}| \times n}$ and $b_o \in \mathbb{R}^{|\mathcal{V}|}$ are weight matrices and bias.[5]

3.3 Training

In the training procedure, our optimization objective is to minimize the negative log-likelihood of the sequence Y and its type information T given the input sequence X over the training data, defined as:

$$\text{minimize} - \sum_{(X,Y,T) \in \mathcal{D}} \log P(Y, T \mid X)$$

According to Equation (1), the log-likelihood $\log P(Y, T \mid X)$ can be decomposed as:

$$\sum_{i=1}^{|Y|} [\log P(y_i \mid y_{<i}, t_{\leq i}, X) + \log P(t_i \mid y_{<i}, t_{<i}, X)],$$

where $P(y_i \mid y_{<i}, t_{\leq i}, X)$ models the sequence generation process, and $P(t_i \mid y_{<i}, t_{<i}, X)$ models the sequence labeling process. They are computed by Equations (4) and (2) respectively. The decomposition of the log-likelihood into these two parts enables a joint optimization for the sequence generation and labeling process simultaneously.

We use the Adam optimizer (Kingma and Ba, 2014) and mini-batch gradient to solve this optimization problem. To prevent overfitting, we apply dropout operators (Srivastava et al., 2014) to non-recurrent connections between LSTM layers.

[5] $|\mathcal{V}|$ is the vocabulary size of the target language.

3.4 Inference

In the inference procedure, we predict the sequence Y and its type information T for an input sequence X according to:

$$(\hat{Y}, \hat{T}) = \underset{(Y',T') \in \mathcal{V}^{|Y'|} \times \mathcal{T}^{|Y'|}}{\text{argmax}} P(Y', T' \mid X)$$

$\mathcal{V}^{|Y'|} \times \mathcal{T}^{|Y'|}$ is the set of all possible (Y', T') pairs. And $(\hat{Y}, \hat{T})$ can be directly converted to the form of linearized PredPatt which is used for evaluation.

However, it is impractical to iterate over all these (Y', T') pairs during inference: here, we use beam search to generate tokens and labels as shown in Algorithm 1.

Algorithm 1 Beam search for selective decoding

Input: X - sequence in the source language
Output: (Y, T) - sequence in the target language and its type information

$step \leftarrow 0$
$b \leftarrow$ beam size
$l \leftarrow$ max decoder length
$fw\_hid \leftarrow \textsf{Encoder}(X)$
TERMINATED_STATES.$init(\varnothing)$
$start\_state \leftarrow \text{State}(\langle bos \rangle, fw\_hid)$
BEAM.$init(\{start\_state\})$
while BEAM $\neq \varnothing$ and $step < l$ **do**
  $step \leftarrow step + 1$
  ACTIVE_STATES.$init(\varnothing)$
  for $state$ in BEAM **do**
    ▷ *Select the decoder.*
    $t \leftarrow \textsf{Selector}(state)$
    ▷ *Generate the next token.*
    $states \leftarrow \textsf{Decoder}_t(state)$
    ▷ *Update the active states.*
    ACTIVE_STATES.$update(states)$
  BEAM.$clean()$
  ▷ *Get the top-k candidates.*
  for $state$ in $top(b,$ ACTIVE_STATES$)$ **do**
    if $state.decoded\_token = \langle eos \rangle$ **then**
      $b \leftarrow b - 1$
      ▷ *Move out the terminated states.*
      TERMINATED_STATES.$add(state)$
    else
      ▷ *Update the beam.*
      BEAM.$add(state)$
TERMINATED_STATES.$update($BEAM$)$
$state \leftarrow top(1,$ TERMINATED_STATES$)$
return $(state.Y, state.T)$

The beam is used to increase the search space for the sequence Y in the target language. At each decoding time step, we first greedily select the type of decoder, and then generate candidate tokens from the selected decoder to update the beam. When the special token $\langle eos \rangle$ is generated, we remove the candidate sequence from the beam.

4 Related Work

The model we propose in this paper is adapted from the RNN encoder-decoder architectures which have been successfully applied to a wide range of NLP tasks such as machine translation (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Bahdanau et al., 2014), image description generation (Karpathy and Fei-Fei, 2015; Vinyals et al., 2015b), syntactic parsing (Vinyals et al., 2015a), question answering (Hermann et al., 2015), summarization (Rush et al., 2015), and semantic parsing (Dong and Lapata, 2016).

As a novel variation of the encoder-decoder architecture, our model provides a general solution to tasks involving translation and labeling. cross-lingual open IE is an example of this kind of task. The end-to-end solution proposed by Zhang et al. (2017a) used a vanilla attention-based encoder-decoder model to achieve results which outperform the traditional pipeline solutions. Compared to the vanilla encoder-decoder model, our model splits the joint task into two concurrent tasks (i.e., labeling and translating), which are jointly learnt by a selector and multiple decoders. This eases the burden of the decoder by shifting the labeling task to the selector. As a result, our model requires a smaller vocabulary for the target language.

The selective decoding mechanism can be viewed as having different types of decoders stacking together and adding a hard gate to the RNN unit, through which the bit of information will be either totally kept or dropped. It may seem redundant since the RNN gated unit already has the sophisticated gating mechanism such as the GRU unit (Cho et al., 2014) and the LSTM unit (Hochreiter and Schmidhuber, 1997). However, we think that the selective decoding mechanism is a complement to the gated unit: rather than having a soft pointwise control, the selective decoding mechanism adopts a hard vectorwise control to explicitly select a certain type of information which corresponds to the predicate or the argument by keeping one and dropping the others, whereas the GRU/LSTM gated unit itself learns to memorize long short-term dependencies. Similar mechanisms have been used in neural machine translating (Tu et al., 2016) and image caption generation (Xu et al., 2015) to explicitly control the influence from source or target contexts. The experiments in § 5 also confirms our point: our model using the selective decoding mechanism significantly improves the performance, compared to the standard encoder-decoder model.

Regarding to open IE systems for generating training data, PredPatt has shown promising performance on large-scale open IE benchmarks (Zhang et al., 2017c). Compared to other existing open IE systems (Banko et al., 2007; Fader et al., 2011; Angeli et al., 2015), PredPatt uses manual language-agnostic patterns on UD, which makes it a well-founded component across languages. Additionally, the underlying structure constructed by PredPatt has been shown to be a well-formed syntax-semantics interface (Zhang et al., 2017b).

5 Experiments

We describe the hyper-parameters setting for experiments, evaluate our approach in two kinds of scenarios, and compare the results of our approach and the other comparing approaches.

5.1 Hyper-parameters

On the encoder side, both the forward RNN and the backward RNN have 2-layer stacked LSTMs with 500 hidden units. On the decoder side, all types of decoders are 2-layer stacked LSTMs with 500 hidden units. All LSTM parameters are sampled from $\mathcal{U}(-0.1, 0.1)$. The dropout rate is set to 0.3. The word embedding size is 300 for input tokens on both the encoder side and the decoder side. We use open-source GloVe vectors (Pennington et al., 2014) trained on Common Crawl 840B with 300 dimensions[6] to initialize the word embeddings on the decoder side. The mini-batch size is set to 64 and the step size set to 50. Gradients are clipped when their norms are greater than 5 (Pascanu et al., 2013). For simplicity, we use vanilla softmax over the decoder vocabulary as opposed to more efficient alternatives such as sampled softmax (Jean et al., 2015). The vocabulary size is set to 40,000. The number of epochs

[6] https://nlp.stanford.edu/projects/glove/

is 20. Early stopping is used to avoid overfitting. The beam size is 5. Before feeding into the encoder, we reverse the input sentences (Sutskever et al., 2014).

5.2 Chinese-English

5.2.1 Dataset

We first evaluate our approach on the Chinese-English dataset (Zhang et al., 2017a), which contains pairs of Chinese sentences and English linearized PredPatt. Table 1 shows the number of data for training, validation and test.

#Train	#Valid	#Test
941,040	10,000	39,626

Table 1: Number of data used for Chinese-English cross-lingual open IE.

5.2.2 Comparisons

Our approach (**Selective Decoding**) is compared against four other approaches: (1) **Joint Seq2Seq**, which trains a standard encoder-decoder model on the Chinese-English dataset described in Table 1; (2) **Joint Moses**, which trains a phrase-based machine translation system, Moses (Koehn et al., 2007), directly on the same data; (3) **Pipeline-S** which consists of a Moses system that translates Chinese sentence to English *sentence*, followed by SyntaxNet Parser (Andor et al., 2016) for Universal Dependency parsing on English, and PredPatt (White et al., 2016) for predicate-argument identification; and (4) **Pipeline-N** is the same as Pipeline-S except that the Moses system is replaced by OpenNMT (Klein et al., 2017), a neural machine translation system.

5.2.3 Evaluation Metrics

For evaluation, we directly convert the output by our approach (e.g. Fig. 1(e)) to the form of linearized PredPatt (e.g., Fig. 1(d)), and follow the same manner in Zhang et al. (2017a), using the cased BLEU score and the token-level F_1 score to evaluate the results, since the generation of linearized PredPatt involves translation and information extraction. We also compute the recoverability: the number of outputs can not be recovered to the tree structure (e.g., Fig. 1(c)).

5.2.4 Evaluation using BLEU

Table 2 shows the cased BLEU scores of linearized PredPatt and linearized predicates[7] on the test set. Selective Decoding significantly improves the performance on both of them. Compared to the previous best approach (Joint Seq2Seq), Selective Decoding improves the BLEU score of linearized PredPatt to 23.88, and the score of linearized predicates to 25.42.

Approach	Linearized PredPatt	Linearized Predicate
Pipeline-S	17.19	17.24
Pipeline-N	18.03	18.59
Joint Moses	18.34	16.43
Joint Seq2Seq	18.94	21.55
Selective Decoding	**23.88**	24.81
- pretrained embeddings	23.67	**25.42**
- beam search	22.07	23.94

Table 2: Evaluation results (BLEU) of linearized PredPatt and linearized predicates on the test set.

We also report two ablation variants of Selective Decoding, i.e., without the pretrained word embeddings for parameter initialization (-pretrained embeddings), and without beam search, only using greedy search during inference (-beam search). As shown in Table 2, while the pretrained word embeddings moderately improve the BLEU score of linearized PredPatt, they have slightly negative impact on linearized predicates. Beam search helps improve the BLEU score of both.

Selective Decoding explicitly models sequence generation and sequence labeling, which enables a standalone evaluation of the sequence generation process (i.e., the final output without predicate and argument labels). To make a baseline comparison, we train an OpenNMT system (Klein et al., 2017) directly on the same data ignoring the labels.

OpenNMT	Selective Decoding
24.92	**25.16**

Table 3: Evaluation results (BLEU) of sequence generation on the test set.

Table 3 shows the BLEU score of sequence generation on the test set. Selective Decoding achieves higher BLEU than OpenNMT. It demonstrates that the selective decoding mechanism

[7] In linearized predicates, arguments are replaced by placeholders. For example, the linearized PredPatt in Fig. 1(d) becomes "[?arg started:p_h Sth:= [?arg firing:p_h ?arg]]" after replacement.

learning with extra labels helps improve the quality of sequence generation. We also notice that the BLEU score (25.16 in Table 2) of the final linearized PredPatt from Selective Decoding is even higher than OpenNMT (24.92 in Table 3). Hence, we can draw a conclusion that simply placing a labeler atop the OpenNMT system to tackle the cross-lingual open IE problem will not narrow the gap in BLEU between itself and our Selective Decoding approach.

5.2.5 Evaluation using F_1

Approach	Predicate	Argument
Pipeline-S	24.24	33.54
Pipeline-N	24.41	33.51
Joint Moses	25.11	38.90
Joint Seq2Seq	25.79	34.44
Selective Decoding	**31.71**	40.81
- pretrained embeddings	31.56	**40.81**
- beam search	30.06	39.06

Table 4: Evaluation results (F_1) of predicates and arguments on the test set.

We compute the token-level F_1 score (Liu et al., 2015) of predicates and arguments. As shown in Table 4, Selective Decoding substantially improves the F_1 score of both predicates and arguments. In the ablation test, pretrained word embeddings slightly improve F_1 of predicates, but have no improvement on F_1 of arguments. Beam search helps improve the score of both.

5.2.6 Recoverability

We compute the number of the linearized PredPatt outputs from which the tree structure representation can not be recovered, including the empty outputs and the outputs which have unmatched brackets, or have zero or multiple heads for an argument or a predicate. As shown in Table 5, compared to the previous best Joint Seq2Seq approach, Selective Decoding further reduces the number of unrecoverable outputs by one order of magnitude.

Pipeline-S	Pipeline-N	Joint Moses	Joint Seq2Seq	Selective Decoding
5,965	6,014	33,178	557	**53**

Table 5: Number of unrecoverable outputs.

5.2.7 Analysis

To analyze the difference between Selective Decoding and the previous best approach Joint Seq2Seq, we plot the BLEU scores of the linearized PredPatt on the test set with respect to the lengths of the reference. As shown in Fig. 4, when the reference length is greater than 20, the linearized PredPatt generated by Selective Decoding gets notably better BLEU scores, especially for the reference length around 30. However, when the reference length is shorter than 11, the performance of Selective Decoding drops below the Joint Seq2Seq approach.

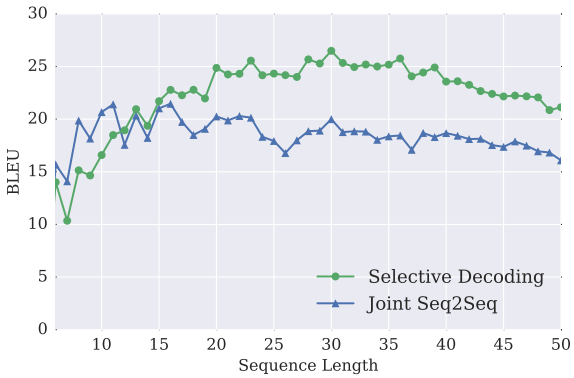

Figure 4: BLEU scores of the linearized PredPatt on the test set w.r.t. the lengths of the references.

To explain this performance drop, we randomly sample an example from the test set, where the reference length is shorter than 11, and the BLEU score of the linearized PredPatt generated by Joint Seq2Seq is higher than Selective Decoding. The example is shown in Table 6.

Input sentence and its English translation:

我 哪怕 有 千分之一 的 希望 呢 , 我 死 活 都 要 给 他 做 最后 的
(*Even if there was only a one thousandth of a hope , er , live or die I would give him my all.*)

Reference[8]:

(1) (**I**) would **give** (**him**) (my **all**)

Selective Decoding:

(1) Even if (**we**) **have** (a _UNK per cent **hope**)

(2) uh , (**I**) would **have** SOMETHING[9]

(3) SOMETHING := (**I**) **give** (**him**) (the final **thing**)

Joint Seq2Seq:

(1) (**I**) **wish** (**everyone**) (last **hope**)

Table 6: Example outputs with the reference length shorter than 11.

In this example (Table 6), the Chinese input sentence has a grammatical error: the object modified by "最后 的" is missing. Additionally, the reference linearized PredPatt output in this example is incomplete: the fact related to the concessive clause is missing. Although here Joint Seq2Seq gets the better BLEU score against the incomplete reference, Selective Decoding is able to better generlize over the train data: the facts it generates are much closer to the original input sentence, and even better than the reference.

Input sentence and its English translation:
结果 , 民主党 失去 了 列举 布什 " 罪状 " 的 良机 ,
(*As a result, the Democratic party lost a good opportunity to list the ' charges ' against Bush.*

Reference:

(1) As (a **result**) , (the Democratic **party**) **lost** (a good **opportunity**)

(2) (a good **opportunity**) **list** (the ' **charges** ' against Bush)

Selective Decoding:

(1) As (a **result**) , (the Democratic **Party**) **lost** (the good **opportunity**)

(2) (the good **opportunity**) **cite** (**Bush**)

Joint Seq2Seq:

(1) (The **result**) **is** SOMETHING

(2) SOMETHING := (the Democratic **Party**) **lost** (his **opportunity**)

(3) (his **opportunity**) **give** (**him**) (good **opportunity**)

Table 7: Example outputs with the reference length longer than 20.

Another example where the reference length is greater than 20 is shown in Table 7. In this example, Selective Decoding generates the same number of facts as the reference, and the meaning of the facts is closer to the reference than Joint Seq2Seq: though not perfect, Selective Decoding captures "列举 布什 ' 罪状 '" ("list the 'charges' against Bush") by generating "**cite** (**Bush**)", whereas Joint Seq2Seq fails to generate any thing related.

[8]The predicate tokens are colored blue, and the argument tokens are colored purple. Head tokens are underlined in bold. Token labels and brackets are omitted.

[9]"SOMETHING" is a special argument used to indicate that the argument is a proposition.

5.3 Low-resource Scenarios

One of the goals of cross-lingual open IE is to extract facts from languages for which few NLP resources and tools are available, and represent the facts in the language for which plenties of resources and tools can be used. Therefore, we extend the experiments to cross-lingual open IE from 5 languages to English in a low-resource setting.

5.3.1 Datasets

Task	#Train	#Valid	#Test
uzb-eng	31,581	1,373	1,373
tur-eng	20,774	903	903
amh-eng	12,140	527	527
som-eng	10,702	465	465
yor-eng	5,787	251	251

Table 8: Number of data used for cross-lingual open IE in low-resource scenarios.

To prepare the experiment datasets, we first collect bitexts from DARPA LORELEI language packs (Strassel and Tracey, 2016). The source languages of the bitexts are Uzbek, Turkish, Amharic, Somali, and Yoruba.[10]

We then run a process similar to Zhang et al. (2017a) to generate pairs of source-language sentences and English linearized PredPatt: first, we employ SyntaxNet Parser (Andor et al., 2016) to generate Universal Dependency parses for the English sentences, and then run PredPatt (White et al., 2016) to generate English linearized PredPatt from the Universal Dependency parses. We remove empty sequences and very long sequences (length>50) in the pairs, and randomly split them into training, validation and test sets in the ratio of 23:1:1. The detailed number of pairs for each experiment is shown in Table 8.

5.3.2 Baseline

To compare with Selective Decoding, we implement the Joint Seq2Seq approach which uses a standard encoder-decoder model as the baseline.

5.3.3 Evaluation Results

We train the models of Selective Decoding and Joint Seq2Seq respectively using the same hyper-

[10]These bitexts are from LDC2016E29 (uzb-eng); LDC2014E115 (tur-eng); LDC2016E86 and LDC2016E87 (amh-eng); LDC2016E90 and LDC2016E91 (som-eng); LDC2016E104 and LDC2016E105 (yor-eng).

parameters setting described in § 5.1, and evaluate the results on the test set by computing the cased BLEU score of the linearized PredPatt, and the token-level F_1 scores.

Task	Joint S2S	Selective Decoding
uzb-eng	8.66	**10.76**
tur-eng	7.18	**7.47**
amh-eng	7.18	**8.37**
som-eng	10.61	**13.06**
yor-eng	11.31	**12.19**

Table 9: Evaluation results in low-resource cross-lingual open IE scenarios: BLEU of linearized PredPatt.

Table 9 shows the evaluation results using BBLEU. Selective Decoding outperforms the Joint Seq2Seq approach by 0.29-2.45, which is expected since Selective Decoding employs the decoder solely for sequence generation and the selector solely for sequence labeling.

Task	Predicate		Argument	
	Joint S2S	Selective Decoding	Joint S2S	Selective Decoding
uzb-eng	**12.50**	12.46	19.57	**24.08**
tur-eng	**9.89**	6.49	17.39	**17.76**
amh-eng	8.44	**8.82**	17.31	**18.58**
som-eng	13.64	**13.91**	22.81	**25.38**
yor-eng	**11.97**	10.74	22.61	**25.57**

Table 10: Evaluation results in low-resource cross-lingual open IE scenarios: the token-level F_1 of predicates and arguments.

The F_1 score measure is shown in Table 10 where both the F_1 scores of predicates and arguments are computed seperately. For predicates, Selective Decoding shows no advantage to the Joint Seq2Seq approach. In the tur-eng task, its F_1 score is obviously worse than the baseline. However, Selective Decoding in general shows promising results in the argument F_1 scores.

6 Conclusions

In this paper, we recast cross-lingual open IE as a more general problem, which involves sequence generation and sequence labeling. We propose a novel encoder-decoder model which employs the selective decoding mechanism to explicitly model the sequence generation and sequence labeling process. Experimental results show our approach achieves consistent and significant improvements in a variety of cross-lingual open IE scenarios.

Since the selective decoding mechanism is not limited to cross-lingual open IE, we believe that it will also benefit other NLP tasks which can be generalized as jointly doing sequence generation and sequence labeling. In the future, we plan to investigate its effectiveness to tasks such as cross-lingual information retrieval.

Acknowledgments

Thank you to the anonymous reviewers for their feedback. This work was supported in part by the JHU Human Language Technology Center of Excellence (HLTCOE), and DARPA LORELEI. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes. The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements of DARPA or the U.S. Government.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany. Association for Computational Linguistics.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, Beijing, China. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2670–2676, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association*

for Computational Linguistics (Volume 1: Long Papers), pages 1466–1477, Berlin, Germany. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany. Association for Computational Linguistics.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China. Association for Computational Linguistics.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA. Association for Computational Linguistics.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Zhengzhong Liu, Teruko Mitamura, and Eduard Hovy. 2015. Evaluation algorithms for event nugget detection: A pilot study. In *Proceedings of the 3rd Workshop on EVENTS at the NAACL-HLT*, pages 53–57.

Di Lu, Xiaoman Pan, Nima Pourdamghani, Shih-Fu Chang, Heng Ji, and Kevin Knight. 2016. A multimedia approach to cross-lingual entity knowledge transfer. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 54–65, Berlin, Germany. Association for Computational Linguistics.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of The 30th International Conference on Machine Learning*, pages 1310–1318.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Matthew Snover, Nitin Madnani, Bonnie J Dorr, and Richard Schwartz. 2009. Fluency, adequacy, or hter?: exploring different human judgments with a tunable mt metric. In *Proceedings of the Fourth*

Workshop on Statistical Machine Translation, pages 259–268. Association for Computational Linguistics.

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

Stephanie Strassel and Jennifer Tracey. 2016. Lorelei language packs: Data, tools, and resources for technology development in low resource languages. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. 2004. Cross-lingual information extraction system evaluation. In *Proceedings of the 20th international Conference on Computational Linguistics*, page 882. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Zhaopeng Tu, Yang Liu, Zhengdong Lu, Xiaohua Liu, and Hang Li. 2016. Context gates for neural machine translation. *arXiv preprint arXiv:1608.06043*.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015a. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2773–2781.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015b. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164.

Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2016. Universal decompositional semantics on universal dependencies. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1713–1723, Austin, Texas. Association for Computational Linguistics.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, volume 14, pages 77–81.

Sheng Zhang, Kevin Duh, and Benjamin Van Durme. 2017a. Mt/ie: Cross-lingual open information extraction with neural sequence-to-sequence models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 64–70, Valencia, Spain. Association for Computational Linguistics.

Sheng Zhang, Rachel Rudinger, Kevin Duh, and Ben Van Durme. 2017b. Ordinal common-sense inference. *Transactions of the Association for Computational Linguistics*.

Sheng Zhang, Rachel Rudinger, and Ben Van Durme. 2017c. An evaluation of predpatt and open ie via stage 1 semantic role labeling. In *Proceedings of the 12th International Conference on Computational Semantics (IWCS)*, Montpellier, France.