Confidence order for joint extraction of entities and relations

Shuo Qiu and Yves Lepage

Waseda University
2-7 Hibikino, Wakamatsu-ku, Kitakyushu-shi, 808-0135 Fukuoka-ken, Japan
qiu.shuo@akane.waseda.jp, yves.lepage@waseda.jp

Abstract

Recent methods for joint entities and relation extraction using triplets do not model the confidence with which these triplets are extracted. We model the confidence of a triplet conditioned on previous predictions. We design a loss function in an improved sequence-to-sequence model, that estimates the true parameters for the computation of the confidence score. Evaluation shows that our method significantly outperforms previous methods. It is also more stable and leaves the possibility of tuning against a customised precision-recall trade-off.

1. Introduction

Recognition of entities and extraction of relations are two basic tasks in information extraction. Joint extraction of entities and relations consists in performing both at the same time. Given the sentence, "Born$_1$ in$_2$ Seattle$_3$,$_4$ Washington$_5$,$_6$ William$_7$ Henry$_8$ Gates$_9$ is$_{10}$ the$_{11}$ principle$_{12}$ founder$_{13}$ of$_{14}$ Microsoft$_{15}$ Corporation$_{16}$.$_{17}$" as input, and { birthplace, founder, company } as a set of predefined relations, we want to directly extract (William Henry Gates, birthplace, Seattle), (William Henry Gates, birthplace, Washington), (Microsoft Corporation, founder, William Henry Gates), (William Henry Gates, company, Microsoft Corporation).

Earlier methods followed a pipeline approach. They identified relations after all entities are recognised. This does not model the correlation between the two tasks. In particular, relations between entities may not be unique. They may overlap (Zeng, 2018) or exhibit some correlation. For instance, the triplet (Microsoft Corporation, founder, William Henry Gates) and the triplet (William Henry Gates, company, Microsoft Corporation) are built on the same pair of entities.

Recently, methods which directly extract entities and relations using triplets (see below, Section 2.1.) gave a new perspective on the task. (Zheng, 2017) proposed a tagging scheme which integrates the relation extraction step into this tagging scheme. For each token in a sentence, it assigns a tag that already indicates which relation this token can be involved in. However, the method cannot handle overlapped triplets. (Zeng, 2018) proposed a sequence-to-sequence based method, which models the extraction result as a sequence of triplets. The relation extraction step takes place just after two correlated entities are extracted.

Triplet-based methods aggregate two processes which utilise similar information, while the pipeline methods aggregate similar processes (entity recognition and relation extraction) together. Intuitively, the information needed to recognise the two entities William Henry Gates and Washington should be located close to the information related to their relation birthplace. The information for recognising the two entities William Henry Gates and Microsoft Corporation should also be located in the neighbourhood of the information needed to extract the relation founder. Hence, it is more natural to extract a triplet before starting to extract another one, as the information for this other triplet is often supported by the same part of the sentence.

However, triplet-based methods do not model the plausibility of each triplet. Knowing this plausibility can bring two important benefits: flexibility in tuning and better selection of relevant triplets. As for flexibility, it will be easier to tune either against precision or recall to better fit for different downstream tasks like augmenting a knowledge base. As for better selection, rather than letting the model decide which triplets should be discarded, leaving this to a final selection process brings more freedom to select actually relevant triplets.

The contribution of this paper is a method to achieve the goal of outputting triplets of entities and relation in confidence order. We design a probabilistic model using a neural network, which approximates this goal by minimising a loss function. We study several termination methods using tuneable thresholds so as to determine the confidence value for each triplet at each step. We implement the probabilistic model using a sequence-to-sequence model. Our model significantly improves over the results reported in (Zeng, 2018). A case study shows that our loss function has the ability of outputting more valid triplets than baseline loss functions.

2. Preliminaries

2.1. Triplets

Inside a given sentence x, a triplet consists of two entities a single relation between them, usually subject, object and relation. The relation must belong to a predefined relation type set $\mathcal{R}$. An entity is defined by its support, i.e., its start and end positions. A triplet is thus defined by four indices, beginning and end of subject and object, i.e., ($subB$, $subE$, $objB$, and $objE$) and a relation r between the subject and the object. It is noted by $(subB, objB, subE, objE, r)$.

For example, with sentence given in Section 1.), the entity Washington is represented as (5, 5), William Henry Gates as (7, 9), and the triplet (William Henry Gates, birthplace, Washington) as (5, 7, 5, 9, birthplace).

2.2. Probabilistic model

In the following section, we explain our notations. Given a sentence, the goal is to exactly predict a set of reference triplets $\mathcal{Y}$. Here, in addition, we want to model the probability $p(\mathcal{Y}|x)$ that the set of triplet $\mathcal{Y}$ is the actual

final result for the given sentence x. Triplets are output one at a time, at each time step. At each time step t, we thus estimate how likely of a triplet y_t is plausible out of all possible triplets, given the previous triplets output $y_{<t}$ This probability is noted $p(y_t|y_{<t}, x)$.

A special null triplet y_t^τ will be used for termination. Its probability reflects how unreasonable further prediction is.

The probability of a triplet at time step t, $p(y_t|y_{<t}, x)$, can be estimated using six factors. For a time step t, we can estimate the probability for each token in the sentence to be an index in the triplet, i.e., the beginning of the subject or the object, $subB$ or $objB$, or their end, $subE$ or $objE$, in that order. This makes four factors. We can estimate the probability of each relation to be the relation in the triplet. This is a fifth factor. We denote each of these factors as y_t^1, $\ldots$, y_t^5. So the model estimates $p(y_t^s|\hat{y}_t^{<s}, \hat{y}_{<t}, x)$ for each step s in $\{1, \ldots, 5\}$ at each time step t.

Finally, we can estimate the termination probability at time step t for y_t^τ as $p(y_t^\tau, \hat{y}_{<t+1}, x)$.

Combining all above factors, the probability of a triplet at time step t is $p(y_t^\tau, \hat{y}_{<t+1}, x) \times \prod_{s=1}^{5} p(y_t^s|\hat{y}_{<t}^{<s}, \hat{y}_{<t}, x)$. In the rest of this paper, we note $p(y_t^s)$ for $p(y_t^s|\hat{y}_t^{<s}, \hat{y}_{<t}, x)$, and $p(y_t^\tau)$ for $p(y_t^\tau, \hat{y}_{<t+1}, x)$. Of these two probabilities, $p(y|y_{<t}, x)$ stands for the fact that the triplet is valid at this time step, out of all the possible triplets in the sentence, and $p(y^\tau|y, y_{<t}, x)$ expresses how much it is an invalid triplet.

3. Method

3.1. Loss function

An embarrassing problem is that no order for triplets is provided in actual benchmarks. Ranking the triplets in arbitrary order to treat them as a sequence is a view adopted in the MultiDecoder system (Zeng, 2018). A negative-log-likelihood loss function solves the problem..

We propose a different loss function to model the plausibility of order. Instead of maximising on one triplet, we maximise the probabilities of all possible predictions at each time step by taking the mean log-likelihood over all possible predictions.

The possible predictions for t, s are the corresponding elements in the set of possible triplets $\mathcal{Y}_t^s$ at time step t, s. We define $\mathcal{Y}_t^s$ based on our probabilistic model: for each time step t, we estimate $p(y_t^s|y_t^{<s}, y_{<t}, x)$ at this time step. Note that $y_{<t}$ are predicted triplets before t. Under this assumption, we expect the model to predict what in $\mathcal{Y}$ has not been predicted yet. It is the set difference between $\mathcal{Y}$ and the set of all $y_{<t}$, denoted by $\mathcal{Y}_t$.

As an additional condition on $y_t^{<s}$, the factors for a triplet at a given time step should be consistent in one triplet. Thus, triplets in $\mathcal{Y}_t$ which do not match elements predicted previously $y_t^{<s}$ should be excluded. Hence, $\mathcal{Y}_t^s$ consists of all triplets in $\mathcal{Y}_t$ which match previous predictions $y_t^{<s}$.

Our loss function can be separated into two cases:

Not all reference triplets have been predicted: $\mathcal{Y} \neq \emptyset$. In this case, we sum up all negative log-likelihoods over all possible predictions:

$$l(x, \mathcal{Y}, s, t) = \frac{1}{|\mathcal{Y}_t^s|} \sum_{y \in \mathcal{Y}_t^s} -\log\left(p(y^s)(1 - p(y_t^\tau))\right) \quad (1)$$

$$l(x, \mathcal{Y}, s, t) = \frac{1}{|\mathcal{Y}_t^s|} \sum_{y \in \mathcal{Y}_t^s} -\log\left(p(y^s)(1 - p(y_t^\tau))\right) \quad (2)$$

Let us call n_t the number of generator time steps such that $|\mathcal{Y}_t^s| \neq 0$ at decoder time step t. We let $l(x, \mathcal{Y}, s, t) = 0$ for all s, t such that $|\mathcal{Y}_t^s| = 0$. The final loss at decoder time step t is normalised so that the case $\mathcal{Y}^s = \emptyset$ does not affect the final result.

$$l(x, \mathcal{Y}, t) = \frac{1}{n_t} \sum_{s=1}^{5} l(x, \mathcal{Y}, s, t) \quad (3)$$

To train termination probabilities, we use the simple loss:

$$l_\tau(x, \mathcal{Y}, t) = -\log p(y_t^\tau) \quad (4)$$

All reference triplets have been predicted: $\mathcal{Y} = \emptyset$. In this case, we maximise the termination probability directly using Formula (4).

$$l(x, \mathcal{Y}, t) = l_\tau(x, \mathcal{Y}, t) = -\log p(y_t^\tau) \quad (5)$$

3.2. Termination methods (TM)

The above loss functions defined on sets of triplets allow us to compute a confidence score for each prediction.

Baseline termination method A simple termination method consists in stopping when the termination class is predicted, i.e., when it has the largest probability. This can be a probability larger than the probability of each other triplet:

$$p(y_t) < p(y_t^\tau) \quad (6)$$

Termination method with threshold Since the set loss ranks prediction in plausibility order, we can sample first n reference as top n prediction. So, we can introduce a threshold to baseline termination method to control the number of triplets predicted.

$$p(y_t) < \alpha p(y_t^\tau) \quad \text{i.e.,} \quad \frac{p(y_t)}{p(y_t^\tau)} < \alpha \quad (7)$$

α is a tuneable threshold valued from 0 to infinity. The left fraction can be considered as a confidence score which measures how much the prediction set is close to the reference set until decoder time step t.

Other termination methods By replacing the numerator and the denominator in the left fraction, we obtain different ways to calculate a confidence score. Options for the numerator are: 1, $p(y_t)$, $\prod_t p(y_t)$. $p(y_t)$ is a predicted triplet probability which should be small after all valid prediction is predicted. $\prod_t p(y_t)$ is the accumulated probability over all previously predicted triplets; this measures the probability over all outputted triplets. The denominator can be replaced by 1 or $p(y_t^\tau)$. The left of Figure 3 graphs recall–precision for such variations of the numerator and the denominator.

3.3. Model

We use an improved version of MultiDecoder (Zeng, 2018) to approximate our probabilistic model. They use a seq2seq model which consists of an encoder and a decoder. Given a sentence consisting of words represented by word embeddings $\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_{|x|}$, the encoder outputs a representation for each word in sentence,

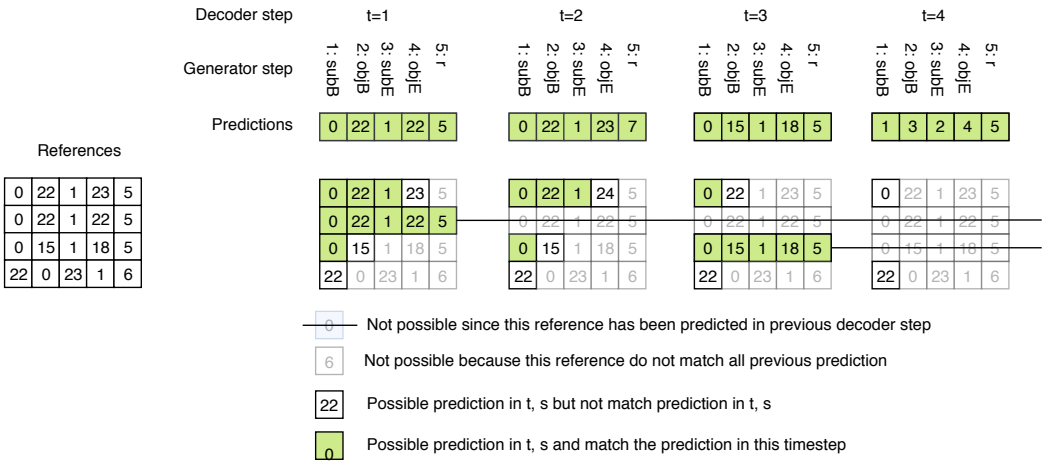


Figure 1: Prediction example that illustrates our loss function. Notice the decoder time steps t and the generator steps s.

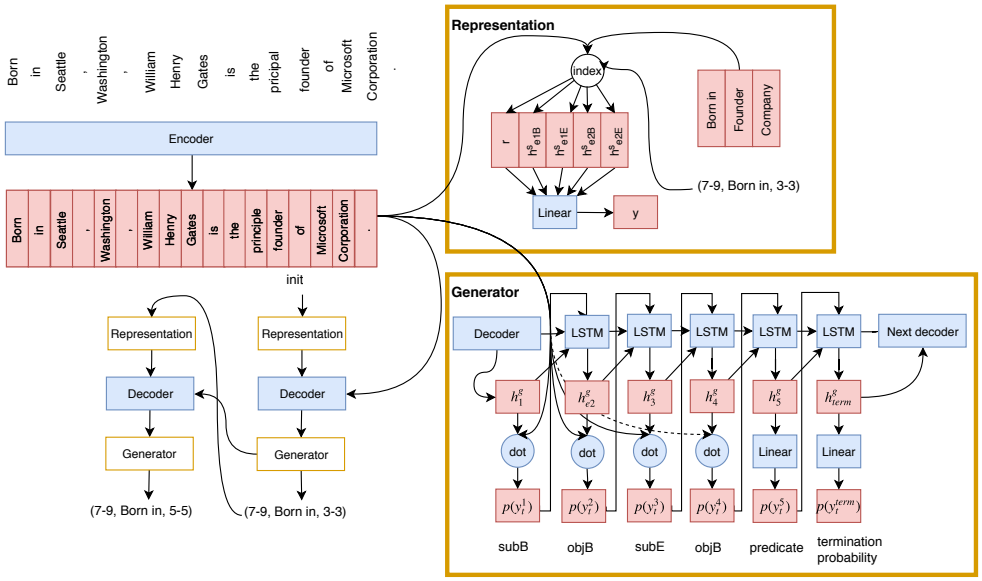


Figure 2: Overview of our model. The red boxes are vectors, the blue boxes are functions.

denoted by $\mathbf{h}^w_1, \mathbf{h}^w_2, \cdots, \mathbf{h}^w_{|x|}$, with $\mathbf{h}^w_i \in \mathbb{R}^{d_h}$, or $\mathbf{H}^w = [\mathbf{h}^w_1, \mathbf{h}^w_2, \cdots, \mathbf{h}^w_{|x|}]^T$, $\mathbf{H}^w \in \mathbb{R}^{|x| \times d_h}$. d_h is the hidden size of the seq2seq model. At each decoder time step, the decoder receives a vector representation of a triplet predicted during the previous time step as input, and the hidden state from the generator as a hidden state. It transforms the hidden state to initialise the current generator It also outputs a vector to generate the probability of the next triplet.

We provide improvements to the MultiDecoder model in the two following ways:

One triplet at each decoder step The MultiDecoder model has several decoders, one per triplet. Differently, we predict one triplet at each decoder time step. We build a triplet representation and a generator to convert triplet into a fixed-size hidden vector, and a generator to generate a triplet probability from a fixed-size hidden vector.

Dedicated termination probability The MultiDecoder model adds an extra class "NA" to in entities and relations to infer when to terminate. Instead, we use a dedicated termination step in the generator.

3.3.1. Triplet representation

We represent a triplet by a fixed-size hidden vector of size d_h. For that, we use a one-hot representation for relation types in relation set $\mathcal{R}$, denoted as $\mathbf{r}_1, \cdots, \mathbf{r}_{|\mathcal{R}|}$, with $\mathbf{r}_i \in \mathbb{R}^{\mathcal{R}}$. We also use the encoder output to represent indices. We concatenate all vectors representing the indices in a triplet and use a linear transformation followed by a non-linear function to get a d_h sized vector, as below.

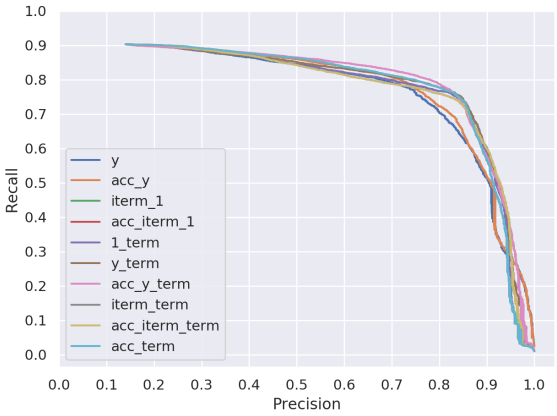
$$\mathbf{y} = \tanh(\mathbf{W}_{\mathbf{repr}} \begin{bmatrix} \mathbf{r}_{\mathbf{r}} \\ \mathbf{h}^e_{subB} \\ \mathbf{h}^e_{subE} \\ \mathbf{h}^e_{objB} \\ \mathbf{h}^e_{objE} \end{bmatrix}), \tag{8}$$

with trainable parameters $\mathbf{W}_{repr} \in \mathbb{R}^{d_h \times (d_r + 4d_h)}$.

3.3.2. Generator

To generate a triplet y_t from a hidden vector $\mathbf{h}^t$ at each decoder time step, we use a cascade of steps to fit for our probabilistic model. They are explained below.

To transform the hidden states among steps, we use



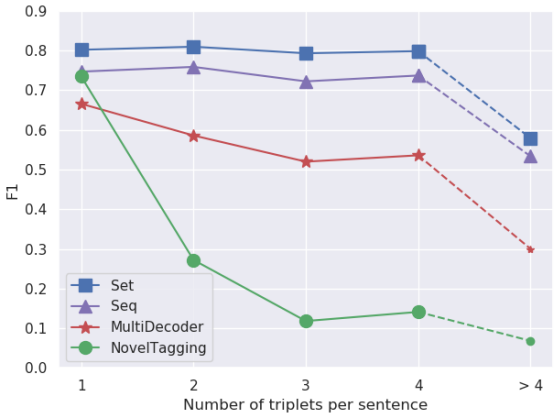


Figure 3: Precision-Recall curves for different termination methods on the *left*. F1 score w.r.t. the number of reference triplets per sentence on the *right*.

	Precision	Recall	F1
NovelTagging	0.624 ± N/P	0.317 ± N/P	0.420 ± N/P
MultiDecoder	0.610 ± N/P	0.566 ± N/P	0.587 ± N/P
Base (repr. + gen.)	0.724 ± 0.004	0.717 ± 0.005	0.720 ± 0.004
+ set loss + baseline TM	0.774 ± 0.008	**0.778 ± 0.005**	0.777 ± 0.003
+ set loss + best TM	**0.825 ± 0.007**	0.759 ± 0.006	**0.791 ± 0.004**

Table 1: Experimental results. Results are reported in $\bar{x} \pm s$ form, where $\bar{x}$ indicates the mean and s indicates the standard deviation. Performance for compared methods are copied from (Zeng, 2018) (no standard deviation provided).

an LSTM (Hochreiter and Schmidhuber, 1997). It is initialised with the decoder's hidden state and a cell state. Each step receives the representation of the prediction in the previous time step as input, and outputs a hidden state $\mathbf{h}^g_{s+1}$ for the prediction in the next time step. We also feed the hidden state to the next decoder step. Figure 2 explains this.

Steps 1–4: Generate indices We use a copy-attention mechanism as in the MultiDecoder model to generate each index. For each s, we estimate the probability of y^s_t by calculating the attention score between the hidden state and the memory:

$$p(y^s_t) = \text{softmax}(\mathbf{H}^e \mathbf{h}^g_s) \tag{9}$$

y^s_t is just the index with the highest probability. The representation $\mathbf{h}^e_{y^s_t}$ is used as input to the generator LSTM to get the hidden state $\mathbf{h}^g_{s+1}$ for the next index.

Step 5: Generate relation We use a linear transformation combined with a softmax to estimate the probability $p(y^4_t)$ of a relation:

$$p(y^4_t) = \text{softmax}(\mathbf{W}_5 \mathbf{h}^g_5) \tag{10}$$

$\mathbf{W}_5 \in \mathbb{R}^{d_h \times d_h}$ is a trainable parameter matrix. We then transform the one-hot vector of the relation into vector of fixed size d_h using another linear transformation. The result is used as input to the generator LSTM to generate $\mathbf{h}^g_\tau$ similarly as in the previous steps.

Step 6: Generate termination score This step is dedicated for predicting a termination probability $p(y^\tau_t)$. We use a sigmoid for that:

$$p(y^\tau_t) = \text{sigmoid}(\mathbf{W}_\tau \mathbf{h}^g_\tau) \tag{11}$$

$\mathbf{W}_\tau \in \mathbb{R}^{1 \times d_h}$ is a trainable parameter matrix.

4. Experiments

4.1. Datasets, metrics, hyperparameters, and compared methods

We conduct experiments on part of the public data-set produced by (Ren, 2017). It consists of about 294,000 examples of training data sampled from New York Times News from 1989 to 2007. 228,000 examples do not contain any triplet. We filter out such examples or sentences with more than 100 words and obtain more than 59,000 training examples. We select 10,000 examples out of them as our validation set, and 10,000 examples as our evaluation set, as (Zeng, 2018) do.

We use a bi-LSTM version of the seq2seq model (Luong et al., 2015) with input feed, global attention, dropout. Word embeddings are trained on the training set using Word2Vec (Mikolov, 2013). As in (Zeng, 2018), the embedding size is 100, the hidden size is 1,000.

We compare our method to two other triplet-based methods described in Section 1.: NovelTagging (Zheng, 2017) and MultiDecoder (Zeng, 2018). We use standard micro Precision, Recall and F1 score to evaluate the results. An entity is correct if the first token in the entity is correct. A triplet is correct if the two entities and the relation in it are correct.

4.2. Results

Table 1 gives results of evaluation. Our base method with model improvement alone drastically increases the F1 score compared to the two other triplet-based methods. The models with set loss still increase their final

	Baseline loss					Set loss + acc-y-τ				
	subject	relation	object	p	r	subject	relation	object	p	r
Prime Minister Paul Martin made a campaign promise to ban handguns in Canada to put an end to a rash of gang shootings in Toronto.										
1	Paul Martin	contains	Toronto	P	N	Canada	contains	Toronto	P	P
2	Paul Martin	nationality	Toronto	N	N	Paul Martin	nationality	Canada	P	P
3	Canada	admin div	Toronto	N	P	Canada	admin div	Toronto	N	P
4	Canada	admin div	Toronto	N	N	Toronto	admin d/c	Canada	N	P
5	Canada	admin div	Toronto	N	N	Paul Martin	nationality	Canada	N	N
6	Canada	admin div	Toronto	N	N	Canada	contains	Toronto	N	N
Like so many other people here, Pedro, a landscaper from Chiapas, Mexico, is desperately trying to get out of Biloxi.										
1	Mexico	contains	Chiapas	P	P	Mexico	contains	Chiapas	P	P
2	Chiapas	admin d/c	Mexico	P	P	Mexico	admin div	Chiapas	N	P
3	Mexico	admin div	Chiapas	P	P	Chiapas	admin d/c	Mexico	N	P
4	Biloxi	admin div	Chiapas	N	N	Mexico	capital	Chiapas	N	N
5	Mexico	admin div	Chiapas	N	N	Pedro	nationality	Mexico	N	P
6	Mexico	admin div	Chiapas	N	N	Chiapas	contains	Biloxi	N	N

Table 2: Results for two sentences by two systems: baseline loss and set loss + acc-y-τ. The p column indicates if the triplet was kept by the termination method (P) or not (N). The r column indicates if triplet is reference triplet (P) or not (N). admin div stands for administration division. Its first entity should be a country. admin d/c is the reverse relation.

F1 scores. This empirically proves that avoiding order selection helps. The baseline termination method (TM) performs well. However, it does not allow to tune the precision-recall as other termination methods do for which the precision-recall curve is given in Figure 3 (*left*): a balance between precision and recall is possible if the threshold is tuned (see remark on flexibility in Section 1.).

The right graph in of Figure 3 shows the performance w.r.t. the number of triplets per sentence. Since the number of examples with more than four triplets is small, we report five cases where the number of triplets per example is 1, 2, 3, 4 or greater than 4. The set loss + $\frac{\prod_t p(y_t)}{p(y_t^\tau)}$ termination method consistently outperforms other models in all cases expect for precision in the case of one triplet per example. The F1 score of set loss + $\frac{\prod_t p(y_t)}{p(y_t^\tau)}$ is relatively stable with regard to number of triplets per example. As the number of triplets per sentence increase, the precision of our methods increases, and the recall decreases. It means that the ratio of predicted triplets to the number of reference triplets become smaller as the number of triplets increases.

4.3. Case study

Table 2 gives results for two sentences in two systems.

In the first example, the set loss system achieves a higher score than the baseline loss. It achieved this by correctly predicting the reference triplets. However, the termination method terminates at time step 2, and discards the third and fourth predictions, which are valid. Observe that the baseline loss has mistaken Toronto as a country.

The second example has three reference triplets. The baseline loss system performs better than the set loss system: it terminates correctly in this case. The set loss system failed at determining when to terminate. This corresponds to the observation in the left graph of Figure 3, where the recall of the set loss system is lower than precision because it did not make adequate predictions. However, this example shows the potential of the set loss system: the relation capital is actually possible here. Also, the relation nationality is actually supported by the sentence; it was predicted in spite of the fact that it was not present in the knowledge base.

5. Conclusion

In this paper, we focused on modelling the plausibility of triplets, which is missing in previous triplet-based methods for joint extraction of entities and relations. We proposed a loss function which takes into account all possible triplets at one time step. Experiments showed that by modelling the plausibility of triplets, we can not only improve the performance but also give the model the flexibility to tune for precision or recall. Experiments also showed that our method is more stable with respect to the number of triplets per sentence.

6. References

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

Luong, Thang, Hieu Pham, and Christopher D. Manning, 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Mikolov, Tomas et al., 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*.

Ren, Xiang et al., 2017. Cotype: Joint extraction of typed entities and relations with knowledge bases. In *26th Int. Conf. on WWW*.

Zeng, Xiangrong et al., 2018. Extracting relational facts by an end-to-end neural model with copy mechanism. In *ACL*.

Zheng, Suncong et al., 2017. Joint extraction of entities and relations based on a novel tagging scheme. In *ACL*.