Open Information Extraction from Question-Answer Pairs

Nikita Bhutani [*]
University of Michigan
nbhutani@umich.edu

Yoshihiko Suhara
Megagon Labs
yoshi@megagon.ai

Wang-Chiew Tan
Megagon Labs
wangchiew@megagon.ai

Alon Halevy
Megagon Labs
alon@megagon.ai

H. V. Jagadish
University of Michigan
jag@eecs.umich.edu

Abstract

Open Information Extraction (OPENIE) extracts meaningful structured tuples from free-form text. Most previous work on OPENIE considers extracting data from one sentence at a time. We describe NEURON, a system for extracting tuples from question-answer pairs. Since real questions and answers often contain precisely the information that users care about, such information is particularly desirable to extend a knowledge base with.

NEURON addresses several challenges. First, an answer text is often hard to understand without knowing the question, and second, relevant information can span multiple sentences. To address these, NEURON formulates extraction as a multi-source sequence-to-sequence learning task, wherein it combines distributed representations of a question and an answer to generate knowledge facts. We describe experiments on two real-world datasets that demonstrate that NEURON can find a significant number of new and interesting facts to extend a knowledge base compared to state-of-the-art OPENIE methods.

1 Introduction

Open Information Extraction (OPENIE) (Banko et al., 2007) is the problem of extracting structured data from a text corpus, without knowing a priori which relations will be extracted. It is one of the primary technologies used in building knowledge bases (KBs) that, in turn, power question answering (Berant et al., 2013). The vast majority of previous work on OPENIE extracts structured information (e.g., triples) from individual sentences.

This paper addresses the problem of extracting structured data from conversational question-answer (CQA) data. Often, CQA data contains precisely the knowledge that users care about. As such, this data offers a goal-directed method for extending existing knowledge bases. Consider, for example, a KB about a hotel that is used to power its website and/or a conversational interface for hotel guests. The KB provides information about the hotel's services: complimentary breakfast, free wifi, spa. However, it may not include information about the menu/times for the breakfast, credentials for the wifi, or the cancellation policy for a spa appointment at the hotel. Given the wide range of information that may be of interest to guests, it is not clear how to extend the KB in the most effective way. However, the conversational logs, which many hotels keep, contain the actual questions from guests, and can therefore be used as a resource for extending the KB. Following examples illustrate the kind of data we aim to extract:

Example 1. *Q: Does the hotel have a gym?*
A: It is located on the third floor and is 24/7.
Tuple: ⟨*gym, is located on, third floor*⟩

Example 2. *Q: What time does the pool open?*
A: 6:00am daily.
Tuple: ⟨*pool, open, 6:00am daily*⟩

As can be seen from these examples, harvesting facts from CQA data presents significant challenges. In particular, the system must interpret information collectively between the questions and answers. In this case, it must realize that *'third floor'* refers to the location of the *'gym'* and that *6:00am* refers to the opening time of the pool. OPENIE systems that operate over individual sentences ignore the discourse and context in a QA pair. Without knowing the question, they either fail to or incorrectly interpret the answer.

This paper describes NEURON, an end-to-end system for extracting information from CQA data. We cast OPENIE from CQA as a multi-source sequence-to-sequence generation problem to explicitly model both the question and answer

[*] Part of the work was done while the author was at Megagon Labs.

in a QA pair. We propose a multi-encoder, constrained-decoder framework that uses two encoders to encode each of the question and answer to an internal representation. The two representations are then used by a decoder to generate an output sequence corresponding to an extracted tuple. For example, the output sequence of Example 2 is:

$\langle arg_1\rangle$ pool $\langle /arg_1\rangle\langle rel\rangle$ open $\langle /rel\rangle\langle arg_2\rangle$ 6:00am daily $\langle /arg_2\rangle$

While encoder-decoder frameworks have been used extensively for machine translation and summarization, there are two key technical challenges in extending them for information extraction from cQA data. First, it is vital for the translation model to learn constraints such as, arguments and relations are sub-spans from the input sequence, output sequence must have a valid syntax (e.g., $\langle arg_1\rangle$ must precede $\langle rel\rangle$). These and other constraints can be integrated as *hard* constraints in the decoder. Second, the model must recognize auxiliary information that is irrelevant to the KB. For example, in the hotel application, NeurON must learn to discard greetings in the data. Since existing facts in the KB are representative of the domain of the KB, this prior knowledge can be incorporated as *soft* constraints in the decoder to rank various output sequences based on their relevance. Our contributions are summarized below:

- We develop NeurON, a system for extracting information from cQA data. NeurON is a novel multi-encoder constrained-decoder method that explicitly models both the question and the answer of a QA pair. It incorporates vocabulary and syntax as *hard* constraints and prior knowledge as *soft* constraints in the decoder.
- We conduct comprehensive experiments on two real-world cQA datasets. Our experimental results show that the use of hard and soft constraints improves the extraction accuracy and NeurON achieves the highest accuracy in extracting tuples from QA pairs compared with state-of-the-art sentence-based models, with a relative improvement as high as 13.3%. NeurON's higher accuracy and ability to discover 15-25% tuples that are not extracted by state-of-the-art models make it suitable as a tuple extraction tool for KB extension.
- We present a case study to demonstrate how a KB can be extended iteratively using tuples extracted using NeurON. In each iteration, only relevant tuples are included in the KB. In turn, the extended KB is used to improve relevance scoring for subsequent iterations.

2 Task Formulation

In this work, we choose to model an OpenIE extraction from a question-answer (QA) pair as a tuple consisting of a single relation with two arguments, where the relation and arguments are contiguous spans from the QA pair. Formally, let (q, a) be a QA pair, where question $q = (q_1, q_2, ..., q_m)$ and answer $a = (a_1, a_2, ..., a_n)$ are word sequences. The output is a triple (arg_1,rel,arg_2) extracted from (q, a). The output triple can be naturally interpreted as a sequence $y = (y_1, y_2, ..., y_o)$ where y_i is either a word or a placeholder tag ($\langle arg_1\rangle$, $\langle rel\rangle$, $\langle arg_2\rangle$) that marks relevant portions of the triple. In OpenIE, the extracted tuple should be asserted by the input QA pair. Formulating this, therefore, requires the vocabulary of y to be restricted to the vocabulary of (q, a) and placeholder tags.

Following this definition, our aim is to directly model the conditional probability $p(y|q, a)$ of mapping input sequences q and a into an output sequence:

$$P(y|q, a) = \prod_{i=1}^{o} p(y_i|y_1, \ldots, y_{i-1}, q, a). \quad (1)$$

In our formulation, a triple is generated as a sequence: a *head* argument phrase arg_1, followed by a relation phrase *rel* and a *tail* argument phrase arg_2. It is possible to consider different orderings in the output sequence (such as (rel,arg_1,arg_2)). However, the goal of OpenIE is to identify the relation phrase that holds between a pair of arguments. Our representation is, thus, consistent with this definition as it models the relation phrase to depend on the head argument.

3 NeurON: A Multi-Encoder Constrained-Decoder Model

Overview of NeurON We propose to extract tuples using a variation of an encoder-decoder RNN architecture (Cho et al., 2014) operating on variable-length sequences of tokens. Fig. 1 shows the architecture of NeurON. It uses two encoders to encode question and answer sequences in a QA pair separately into fixed-length vector representations. A decoder then decodes the vector representations into a variable-length sequence corresponding to the tuple. The decoder is integrated

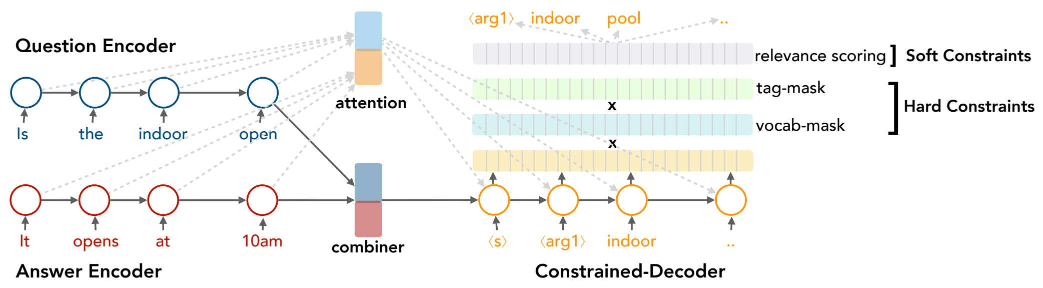


Figure 1: Multi-Encoder, Constrained-Decoder model for tuple extraction from (q, a).

with a set of *hard constraints* (e.g., output vocabulary) and *soft constraints* (e.g., relevance scoring) suited for the extraction task.

3.1 Multiple Encoders

Given an input QA pair, two RNN encoders separately encode the question and answer. The question encoder converts q into hidden representation $h^q = (h^q_1, ..., h^q_m)$ and the answer encoder converts a into $h^a = (h^a_1, ..., h^q_n)$, where $h^q_t = \text{lstm}(q_t, h^q_{t-1})$ is a non-linear function represented by the long short-term memory (LSTM) cell. The combiner combines the encoders' states and initializes the hidden states $\mathbf{h}$ for the decoder:

$$h = \tanh(W_c[h^q \circ h^a]),$$

where $\circ$ denotes concatenation. The decoder stage uses the hidden states to generate the output y with another LSTM-based RNN. The probability of each token is defined as:

$$p(y_t) = \text{softmax}((s_t \circ c^q_t \circ c^a_t)W_y), \qquad (2)$$

where s_t denotes the decoder state, $s_0 = h$ and $s_t = \text{lstm}((y_{t-1} \circ c^q_t \circ c^a_t)W_s, s_{t-1})$. The decoder is initialized by the last hidden state from the combiner. It uses the previous output token at each step. Both W_y and W_s are learned matrices. Each decoder state is concatenated with *context vectors* derived from the hidden states of the encoders. Context vector c_t is the weighted sum of the encoder hidden states, i.e. $c^q_t = \sum_{i=1}^{m} \alpha_{t_i} h^q_i$, where α_{t_i} corresponds to an attention weight. The attention model (Bahdanau et al., 2015) helps the model learn to focus on specific parts of the input sequences, instead of solely relying on hidden vectors of the decoders' LSTM. This is crucial for extraction from (q, a) pairs where input sequences tend to be long.

3.2 Constrained Decoder

The decoder finds the best hypothesis (i.e., the best output sequence) for the given input representations. Typically, the output sequence is generated, one unit at a time, using *beam search*. At each time step, the decoder stores the top-k scoring partial sequences, considers all possible single token extensions of them, and keeps k most-likely sequences based on model's probabilities (Eq. 1). As soon as the $\langle/S\rangle$ symbol is appended, the sequence is removed from the beam and added to the set of complete sequences. The most-likely complete sequence is finally generated.

Hard Constraints While such encoder-decoder models typically outperform conventional approaches (Cho et al., 2014; Zoph and Knight, 2016; Xiong et al., 2017) on a wide variety of tasks including machine translation and question answering, the accuracy and training efficiency has been shown to improve when the model is integrated with the constraints of the output domain (Xiao et al., 2016; Yin and Neubig, 2017). Motivated by these, NEURON allows constraints relevant to information extraction to be incorporated in the model. Specifically, we describe how the decoder can enforce vocabulary and structural constraints on the output.

- **Vocabulary constraints.** Since the arguments and relations in the extracted tuples typically correspond to the input QA pair, the decoder must constraint the space of next valid tokens when generating the output sequence. NEURON uses a masking technique in the decoder to mask the probability of tokens (as in Eq. 2) that do not appear in the input (q, a) pair. Specifically, it computes a binary mask vector v, where $|v|$ is vocabulary size and $v_i = 1$ if and only if i-th token appears in q or a. The probability of each token is modified as:

$$p(y_t) = \text{softmax}((s_t \circ c^q_t \circ c^a_t)W_y \otimes v), \qquad (3)$$

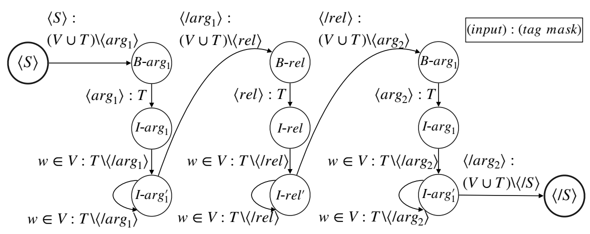


Figure 2: State diagram for tag masking rules. V is the vocabulary including placeholder tags, T is the set of placeholder tags.

where $\otimes$ indicates element-wise multiplication.

• **Structural constraints.** For the output sequence to correspond to a valid tuple with non-empty arguments, the decoding process must conform to the underlying grammar of a tuple. For instance, decoding should always begin in the $\langle S \rangle$ state, where only $\langle arg_1 \rangle$ can be generated. In subsequent time steps, all other placeholders except $\langle /arg_1 \rangle$ should be restricted to ensure a non-empty argument. Once $\langle /arg_1 \rangle$ is generated, $\langle rel \rangle$ must be generated in the next time step and so on. The various states and grammar rules can be described as a finite state transducer as shown in Figure 2.

Depending upon the state, NeurON generates a mask r based on this grammar and uses r to further modify the probabilities of the tokens as follows:

$$p(y_t) = \text{softmax}((s_t \circ c_t^q \circ c_t^a) W_y \otimes v \otimes r). \quad (4)$$

Soft Constraints OpenIE systems are typically used to extract broad-coverage facts to extend existing KBs. Facts already existing in the KB are representative of the domain of the KB. It is, therefore, useful to incorporate this prior knowledge in the extraction itself. NeurON is able to use prior knowledge (incorporated as soft constraints) in the decoder to understand the relevance of candidate extractions and adjust the ranking of various output sequences accordingly. To see why such soft constraints can be useful, consider the example:

Example 3. *Q: "Is the pool open?"*
A: "I am sorry but our pool reopens at 7:00am."
Tuple: ⟨I, am, sorry⟩; ⟨pool, reopens at, 7:00am⟩

Both the tuple facts are correct given the input QA pair but only the second tuple contains useful information. Filtering such irrelevant facts is difficult without additional evidence.

The multi-encoder and constrained-decoder in NeurON are jointly optimized to maximize the log probability of output sequence conditioned on the input sequences. At inference, the decoder estimates the likelihood of various candidate sequences and generates the sequence with the highest likelihood. As shown in Eq. 1, this likelihood is conditioned solely on the input (q, a) pair, thus increasing the possibility of obtaining facts that may be correct but irrelevant. Instead, if a relevance scoring function were integrated at extraction time, the candidate output sequences could be re-ranked so that the predicted output sequence is likely to be both correct and relevant.

Learning a relevance scoring function can be modeled as a KB completion task, where missing facts have to be inferred from existing ones. A promising approach is to learn vector representations of entities and relations in a KB by maximizing the total plausibility of existing facts in the KB (Wang et al., 2017). For a new candidate output sequence, its plausibility can be predicted using the learned embeddings for the entities and relation in the sequence.

In NeurON, we learn the entity and relation embeddings using Knowledge Embedding (KE) methods such as TransE (Bordes et al., 2013) and HolE (Nickel et al., 2016). Note that NeurON is flexible with how the relevance scoring function is learned or which KE method is chosen. In this paper, we use TransE for evaluation. TransE computes the plausibility score S of a tuple $y = \langle arg_1, rel, arg_2 \rangle$ as:

$$S(y) = ||\mathbf{v}_{arg_1} + \mathbf{v}_{rel} - \mathbf{v}_{arg_2}||,$$

where $\mathbf{v}_{arg_1}$, $\mathbf{v}_{rel}$, and $\mathbf{v}_{arg_2}$ are embedding vectors for arg_1, rel, and arg_2 respectively. Following (Jain et al., 2018), we compute the embedding vectors of out-of-vocabulary arguments (and relations) as the average of embedding vectors of known arguments (and relations). We generate the most-likely output based on its conditional probability and plausibility score:

$$\hat{y} = \underset{y}{\text{argmax}}(\log P(y|q, a) + \gamma \log S(y)). \quad (5)$$

To implement the tuple relevance scoring function, we employ the re-ranking approach, which is a common technique for sequence generation methods (Luong et al., 2015b). Our re-ranking method first obtains candidates from a beam search decoder and then re-ranks the candidates based on the objective function (Eq. 5).

4 Experiments

We evaluated the performance of NeurON on two cQA datasets. In our analysis, we find that integrating hard and soft constraints in the decoder improved the extraction performance irrespective of the number of encoders used. Also, 15-25% of the tuples extracted by NeurON were not extracted by state-of-the-art sentence-based methods.

4.1 Datasets and Data Preparation

ConciergeQA is a real-world internal corpus of 33,158 QA pairs collected via a multi-channel communication platform for guests and hotel staff. Questions (answers) are always made by guests (staff). An utterance has 36 tokens on average, and there are 25k unique tokens in the dataset. A QA utterance has 2.8 sentences on average, with the question utterance having 1.02 sentences on average and answer utterance having 1.78 sentences on average.

AmazonQA (Wan and McAuley, 2016; McAuley and Yang, 2016) is a public dataset with 314,264 QA pairs about electronic products on *amazon.com*. The dataset contains longer and more diverse utterances than the *ConciergeQA* dataset: utterances have an average of 45 tokens and the vocabulary has more than 50k unique tokens. A QA utterance has 3.5 sentences on average. The question utterances had 1.5 sentences on average and the answer having 2 sentences.

For training NeurON, we bootstrapped a large number of high-quality training examples using a state-of-the-art OpenIE system. Such bootstrapping has been shown to be effective in information extraction tasks (Mausam et al., 2012; Saha et al., 2017). The StanfordIE (Angeli et al., 2015) system is used to extract tuples from QA pairs for training examples. To further obtain high-quality tuples, we filtered out tuples that occur too infrequently (< 5) or too frequently (> 100). For each tuple in the set, we retrieved all QA pairs that contain all the content words of the tuple and included them in the training set. This helps create a training set encapsulating the multiplicity of ways in which tuples are expressed across QA pairs. We randomly sampled 100 QA pairs from our bootstrapping set and found 74 of them supported the corresponding tuples. We find this quality of bootstrapped dataset satisfactory, since the seed tuples for bootstrapping could be noisy as they were generated by another OpenIE system.

Instance type	*ConciergeQA*	*AmazonQA*
Exclusively from question	13.9%	13.8%
Exclusively from answer	25.8%	17.6%
Ambiguous	36.9%	29.8%
Jointly from Q-A	23.4%	38.8%

Table 1: Various types of training instances.

Dataset	Q-A	|V|	Train	Dev	Test (Q-A)
ConciergeQA	33k	25k	1.25M	128k	2,905
AmazonQA	314k	50k	1.43M	159k	39,663

Table 2: Training, Dev and Test splits.

Our bootstrapped dataset included training instances where a tuple matched (a) tokens in the questions exclusively, (b) tokens in the answers exclusively, (c) tokens from both questions and answers. Table 1 shows the distribution of the various types of training instances. Less than 40% (30%) of ground truth tuples for *ConciergeQA* (*AmazonQA*) exclusively appear in the questions or answers. Also, 22.1% (37.2%) of ground truth tuples for *ConciergeQA* (*AmazonQA*) are extracted from the combination of questions and answers. These numbers support our motivation of extracting tuples from QA pairs. We used standard techniques to construct training/dev/test splits so that QA pairs in the three sets are disjoint. Table 2 shows the details of the various subsets.

4.2 Baseline Approaches

We compared NeurON with two methods that can be trained for tuple extraction from QA pairs: BiLSTM-CRF (Huang et al., 2015) and NeuralOpenIE (Cui et al., 2018). BiLSTM-CRF is a sequence tagging model that has achieved state-of-the-art accuracy on POS, chunking, NER and OpenIE (Stanovsky et al., 2018) tasks. For OpenIE, the model predicts boundary labels (e.g., B-ARG1, I-ARG1, B-ARG2, O) for the various tokens in a QA pair. NeuralOpenIE is an encoder-decoder model that generates a tuple sequence given an input sequence. Since it uses a single encoder, we generate the input sequence by concatenating the question and answer in a QA pair. We trained all the models using the same training data.

4.3 Performance Metrics

We examine the performance of different methods using three metrics: *precision*, *recall*, and *relative coverage* (RC). Given a QA pair, each system returns a sequence. We label the sequence correct if it matches one of the ground-truth tuples for the QA pair, incorrect otherwise. We then measure precision of a method (i.e., # of correct predictions of the method / # of question-answer pairs) and recall (i.e., # of correct predictions of the method / # of correct predictions of any method) following (Stanovsky and Dagan, 2016). To compare the coverage of sequences extracted by NeurON against the baseline method, we compute relative coverage of NeurON as the fraction of all correct predictions that were generated exclusively by NeurON. Specifically,

$$RC = \frac{|\mathrm{TP}_{\mathrm{NeurON}} \backslash \mathrm{TP}_{baseline}|}{|\mathrm{TP}_{\mathrm{NeurON}} \bigcup \mathrm{TP}_{baseline}|},$$

where TP denotes the correct predictions.

4.4 Model Training and Optimization

We implemented NeurON using OpenNMT-tf (Klein et al., 2017) , an open-source neural machine translation system that supports multi-source encoder-decoder models. We implemented NeuralOpenIE using the same system. We used the open-source implementation of BiLSTM-CRF (Reimers and Gurevych, 2017). For fair comparison, we used identical configurations for NeurON and NeuralOpenIE. Each encoder used a 3-layer bidirectional LSTM and the decoder used a 3-layer bidirectional LSTM. The models used 256-dimensional hidden states, 300-dimensional word embeddings, and a vocabulary size of 50k. The word embeddings were initialized with pre-trained GloVe embeddings (glove.6B) (Pennington et al., 2014). We used an initial learning rate of 1 and optimized the model with stochastic gradient descent. We used a decay rate of 0.7, a dropout rate of 0.3 and a batch size of 64. The models were trained for 1M steps for the *ConciergeQA* dataset and 100k steps for the *AmazonQA* dataset. We used TESLA K80 16GB GPU for training the models. We trained the KE models for relevance scoring using our bootstrapped training dataset. For integrating the relevance scoring function, we experimented with different values for γ and found it not have a major impact within a range of 0.02 to 0.2. We used a value of 0.05 in all the experiments.

4.5 Experimental Results

The BiLSTM-CRF model showed extremely low (2-15%) precision values. Very few of the tagged sequences (32-39%) could be converted to a tuple. Most tagged sequences had multiple relations and arguments, indicating that it is difficult to learn how to tag a sequence corresponding to a tuple. The model only learns how to best predict tags for each token in the sequence, and does not take into account the long-range dependencies to previously predicted tags. This is still an open problem and is outside the scope of this paper.

Method	P	R	RC
NeuralOpenIE (baseline)	0.769	0.580	-
+ hard constraints	0.776	0.585	-
+ hard and soft constraints	0.796	0.600	-
NeurON (our method)	0.791	0.597	0.224
+ hard constraints	0.792	0.597	0.204
+ hard and soft constraints	**0.807**	**0.608**	**0.245**

Table 3: Precision (P), Recall (R), and Relative Coverage (RC) results on *ConciergeQA*.

Tables 3 and 4 show the performance of NeuralOpenIE and NeurON on the two cQA datasets. NeurON achieves higher precision on both the datasets. This is because NeuralOpenIE uses a single encoder to interpret the question and answer in the same vector space, which leads to lower performance. Furthermore, concatenating the question and answer makes the input sequence too long for the decoder to capture long-distance dependencies in history (Zhang et al., 2016; Toral and Sánchez-Cartagena, 2017). Despite the attention mechanism, the model ignores past alignment information. This makes it less effective than the dual-encoder model used in NeurON.

The tables also show that incorporating task-specific hard constraints helps further improve the overall precision and recall, regardless of the methods and the datasets. Re-ranking the tuples based on the soft constraints derived from the existing KB further improves the performance of both methods in *ConciergeQA* and NeuralOpenIE in *AmazonQA*. The existing KB also helps boost the likelihood of a correct candidate tuple sequence that was otherwise scored to be less likely. Lastly, we found that NeurON has significant relative coverage; it discovered significant additional, unique tuples missed by NeuralOpenIE.

Table 4 shows a slight decrease in performance for NeurON after soft constraints are added. This is likely caused by the lower quality KE model due to the larger vocabulary in *AmazonQA*. In contrast, even with the lower quality KE model, Neu-

Method	P	R	RC
NeuralOpenIE (baseline)	0.557	0.594	-
+ hard constraints	0.563	0.601	-
+ hard and soft constraints	0.571	0.610	-
NeurON (our method)	0.610	0.652	0.139
+ hard constraints	**0.631**	**0.674**	**0.164**
+ hard and soft constraints	0.624	0.666	0.149

Table 4: Precision (P), Recall (R), and Relative Coverage (RC) results on *AmazonQA* dataset.

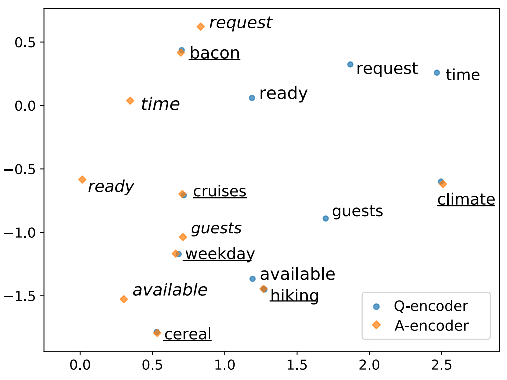


Figure 3: Example embedding vectors from question and answer encoders. Underlines denote similar embedding vectors in both the encoders.

ralOpenIE improved slightly. This is likely because the NeuralOpenIE model, at this stage, still had a larger margin for improvement. We note however that learning the best KE model is not the focus of this work.

AmazonQA is a more challenging dataset than *ConciergeQA*: longer utterances (avg. 45 tokens vs. 36 tokens) and richer vocabulary ($>$ 50k unique tokens vs. $<$ 25k unique tokens). This is reflected in lower precision and recall values of both the systems on the *AmazonQA* dataset. While the performance of end-to-end extraction systems depends on the complexity and diversity of the dataset, incorporating hard and soft constraints alleviates the problem to some extent.

End-to-end extraction systems tend to outperform rule-based systems on extraction from cQA datasets. We observed that training data for *ConciergeQA* had a large number ($>$ 750k) dependency-based pattern rules, of which $<$ 5% matched more than 5 QA pairs. The set of rules is too large, diverse and sparse to train an accurate rule-based extractor. Even though our training data was generated by bootstrapping from a rule-based extractor StanfordIE, we found only 51.5% (30.7%) of correct tuples from NeurON exactly matched the tuples from StanfordIE in *ConciergeQA* (*AmazonQA*). This indicates that NeurON combined information from question and answer, otherwise not accessible to sentence-wise extractors. As an evidence, we found 11.4% (6.1%) of tuples were extracted from answers, 16.8% (5.0%) from questions, while 79.6% (82.5%) combined information from questions and answers in *ConciergeQA* (*AmazonQA*).

Multiple Encoders: Our motivation to use different encoders for questions and answers is based on the assumption that they use different vocabulary and semantics. We found that there were 8k (72k) unique words in questions, 18k (114k) unique words in answers, and the Jaccard coefficient between two vocabulary sets was 0.25 (0.25) in *ConciergeQA* (*AmazonQA*), indicating that two sources use significantly different vocabulary. Also, the same word can have different meanings depending on a speaker, and thus such words in the two sources should be embedded differently. To visualize the embedding vectors of common words in *ConciergeQA*, we mapped them into 2D space using t-SNE (Maaten and Hinton, 2008). Fig. 3 shows that *subjective* words that represents speakers attitude (e.g., “ready”, “guests”, “time”) had significantly different embeddings in the question and answer encoders. In contrast, *objective* words such as menu, or activity (e.g., “bacon”, “cruise”, “weekday”) had similar embeddings although the two encoders do not directly share the embedding parameters. This indicates that multiple encoders not only capture the different meanings in questions and answers but also retain consistent meanings for words that keep the same meanings in the two sources.

Relevance Scoring: We compared with another NeurON model that uses HolE (Nickel et al., 2016) for relevance scoring. Both the HolE and TransE models achieved the same precision of 80.7%, with HolE achieving slightly higher recall (+1.4%). This suggests that incorporating relevance scoring in NeurON can robustly improve the extraction accuracy, regardless of the choice of the knowledge embedding method. We also estimated the upper-bound precision by evaluating if the correct tuple was included in the top-500 candidates. The upper-bound precision was 85.0% on *ConciergeQA*, indicating that there is still room for improvement on incorporating relevance scoring.

1	**Q**: Tell me what the username and password is for WiFi **A**: Absolutely! Both the username and passcode is C800. **StanfordIE**: ⟨ passcode, is, C800 ⟩ **NEURON**: ⟨ password, is, C800 ⟩
2	**Q**: Do hotel guys have ice? **A**: There is an ice machine on first floor lobby. **StanfordIE**: ⟨ hotel, do, ice ⟩ **NEURON**: ⟨ hotel, have, ice machine ⟩
3	**Q**: Is there a charge for parking a rental car on the property? **A**: Self-parking will be $15 per night. **StanfordIE**: None **NEURON**: ⟨ parking, will, charge ⟩
4	**Q**: arrange late check out for tomorrow? **A**: I have notated a 12 pm check out. Normal check out time is at 11 am. **StanfordIE**: ⟨ normal check, is at, 11 am ⟩ **NEURON**: ⟨ check, is at, 11 ⟩

Table 5: Examples of successful cases (**1** and **2**) and failed cases (**3** and **4**) from test data.

4.6 Error Analysis

We examined a random sample of 100 errors shared by all the systems across the tested datasets. Arguably, encoder-decoder models suffer when extracting tuples from long utterances (avg. of 54 tokens), contributing to 43% of the errors. 34% of the incorrectly extracted tuples used words that were shared across the two sources. This indicates that the extractor makes errors when resolving ambiguity in tokens. 28% of the error cases used informal language that is generally difficult for any extractor to understand. We show some examples (1 and 2 in Table 5) where NEURON successfully combined information across two sources and examples (3 and 4 in Table 5) where it failed.

We further examined three different scenarios: a) errors are shared by both NEURON and NEURALOPENIE, b) errors are made exclusively by NEURON, c) errors are made exclusively by NEURALOPENIE. For each scenario, we examined a random sample of 100 errors. We categorize the different sources of errors and report the results in Table 6. As shown, NEURON is superior on longer utterances compared to NEURALOPENIE (54 tokens vs. 49 tokens). However, ambiguity in tokens in the two sources is a concern for NEURON because it has the flexibility to interpret the question and answer differently. Not surprisingly, informal utterances are hard to translate for both the systems.

5 Case Study - KB Extension

The extracted tuples from NEURON can be used to extend a KB for a specific domain. However, automatically fusing the tuples with existing facts in the KB can have limited accuracy. This can be due to noise in the source conversation, no prior knowledge of join rules and more. One possible solution is to design a human-in-the-loop system that iteratively extracts tuples and filters them based on human feedback (Fig. 4). In each iteration, a set of tuples is annotated by human annotators based on their relevance to the domain of the KB. The tuples marked relevant are added to the KB and the relevance scoring function is updated for extracting more relevant tuples from the corpus in the next iteration.

Error Category	$\overline{N}, \overline{B}$	$N, \overline{B}$	$\overline{N}, B$
long utterances	43%	45%	40%
avg. length of utterance	54 tokens	49 tokens	54 tokens
ambiguity	34%	36%	48%
informal language	28%	36%	34%

Table 6: Different errors $\overline{N}$ and $\overline{B}$ made by NEURON (N) and NEURALOPENIE (B) respectively.

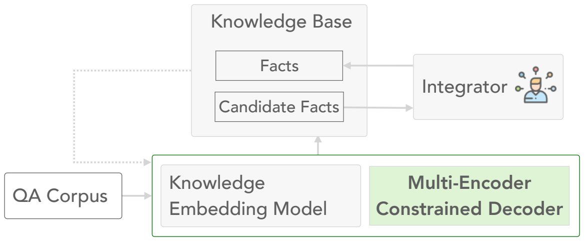


Figure 4: Human-in-the-loop system for extending a domain-specific KB.

We conducted a crowdsourced experiment[1], simulating the first iteration of the procedure i.e., when no KE model is available. We collected annotations on top-5 tuples extracted by NEURON for 200 QA pairs in the *ConciergeQA* dataset. For reliability, we hired five workers for each extraction. The workers were asked to judge if a tuple is relevant to the hotel domain and represents concrete information to be added to a KB. We found precision@5 was 41.4%, and NEURON extracted at least one useful tuple for 83.0% of the 200 QA pairs. Overall, the system added 243 unique tuples (out of 414 tuples extracted by NEURON) to the KB. We also collected annotations for the tuples extracted by NEURALOPENIE. The precision@5 and recall@5 values were 41.3% and 79.0% respectively. Although the precision values are quite similar, NEURON can extract correct tuples from more QA pairs than NEURALOPENIE. While the precision can further be improved, the preliminary results support that NEURON is a good candidate for extraction in

[1] https://www.figure-eight.com/

a human-in-the-loop system for KB extension. We did not use any sophisticated methods for ranking tuples in our experiment. Thus, a better ranking algorithm might lead to improved precision.

6 Related Work

There is a long history of OpenIE systems for extracting tuples from plain text. They are built on hand-crafted patterns over an intermediate representation of a sentence (e.g., POS tags (Yates et al., 2007; Fader et al., 2011), dependency trees (Bhutani et al., 2016; Mausam et al., 2012)). Such rule-based systems require extensive engineering when the patterns become diverse and sparse. Recently, OpenIE systems based on end-to-end frameworks, such as sequence tagging (Stanovsky et al., 2018) or sequence-to-sequence generation (Cui et al., 2018), have been shown to alleviate such engineering efforts. However, all these systems focus on sentence-level extraction. We are the first to address the problem of extracting tuples from question-answer pairs.

Our proposed system is based on an encoder-decoder architecture, which was first introduced by Cho et al. for machine translation. Attention mechanisms (Bahdanau et al., 2015; Luong et al., 2015b) have been shown to be effective for mitigating the problem of poor translation performance on long sequences. Their model can learn how much information to retrieve from specific parts of the input sequence at decoding time. There is abundant research on generalizing such frameworks for multiple tasks, specially by employing multiple encoders. Using multiple encoders has been shown to be useful in mutli-task learning (Luong et al., 2015a), multi-source translation (Zoph and Knight, 2016) and reading comprehension (Xiong et al., 2017). We are the first to explore a multi-source encoder-decoder architecture for extracting tuples from cQA datasets.

Traditional encoder-decoder architectures are not tailored for information extraction and knowledge harvesting. To make them suitable for information extraction, the sequence generation must be subjected to several constraints on the vocabulary, grammar etc. Recently, grammar structures have been integrated into encoder-decoder models (Iyer et al., 2017; Zhang et al., 2017). There are variations such as Pointer Networks (Vinyals et al., 2015) that yield a succession of pointers to tokens in the input sequence. All these studies share a common idea with our paper, which is to enforce constraints at sequence generation time. Since we focus on extraction from cQA datasets, our work is broadly related to the literature on relation extraction (Savenkov et al., 2015; Hixon et al., 2015; Wu et al., 2018) and ontology extraction (S and Kumar, 2018) from community generated question-answer datasets. However, we differ in our underlying assumption that the relations and entities of interest are not known in advance. Alternatively, a cQA dataset could be transformed into declarative sentences (Demszky et al., 2018) for a conventional OpenIE system. However, such a two-stage approach is susceptible to error propagation. We adopt an end-to-end solution that is applicable to generic cQA datasets.

7 Conclusions and Future Work

We have presented NeurON, a system for extracting structured data from QA pairs for the purpose of enriching knowledge bases. NeurON uses a multi-encoder, constrained-decoder framework to generate quality tuples from QA pairs.

NeurON achieves the highest precision and recall in extracting tuples from QA pairs compared with state-of-the-art sentence-based models, with a relative improvement as high as 13.3%. It can discover 15-25% more tuples which makes it suitable as a tuple extraction tool for KB extension.

There are several directions for future research. One interesting direction is to investigate whether NeurON can be extended to work on *open-domain QA corpus*, which may not be restricted to any specific domain.

Acknowledgement

We thank Tom Mitchell and the anonymous reviewers for their constructive feedback. This work was supported in part by the UM Office of Research.

References

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proc. ACL '15/IJCNLP '15*, pages 344–354.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. ICLR '15*.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007.

Open information extraction from the web. In *Proc. IJCAI '07*, pages 2670–2676.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proc. EMNLP '13*, pages 1533–1544.

Nikita Bhutani, HV Jagadish, and Dragomir Radev. 2016. Nested propositions in open information extraction. In *Proc. EMNLP '16*, pages 55–64.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proc. NIPS '13*, pages 2787–2795.

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 *Proc. EMNLP '14*, pages 1724–1734.

Lei Cui, Furu Wei, and Ming Zhou. 2018. Neural open information extraction. In *Proc. ACL '18*, pages 407–413.

Dorottya Demszky, Kelvin Guu, and Percy Liang. 2018. Transforming question answering datasets into natural language inference datasets. *arXiv preprint arXiv:1809.02922*.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proc. EMNLP '11*, pages 1535–1545.

Ben Hixon, Peter Clark, and Hannaneh Hajishirzi. 2015. Learning knowledge graphs for question answering through conversational dialog. In *Proc. NAACL-HLT '15*, pages 851–861.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proc. ACL '17*, pages 963–973.

Prachi Jain, Shikhar Murty, Mausam, and Soumen Chakrabarti. 2018. Mitigating the effect of out-of-vocabulary entity pairs in matrix factorization for KB inference. In *Proc. IJCAI '18*, pages 4122–4129.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL '17 (System Demonstrations)*, pages 67–72.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015a. Multi-task sequence to sequence learning. In *Proc. ICLR '16*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015b. Effective approaches to attention-based neural machine translation. In *Proc. EMNLP '15*, pages 1412–1421.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, Oren Etzioni, et al. 2012. Open language learning for information extraction. In *Proc. EMNLP '12*, pages 523–534.

Julian McAuley and Alex Yang. 2016. Addressing complex and subjective product-related queries with customer reviews. In *Proc. WWW '16*, pages 625–635.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In *Proc. AAAI '16*, pages 1955–1961.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proc. EMNLP '14*, pages 1532–1543.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *Proc. EMNLP '17*, pages 338–348.

Subhashree S and P Sreenivasa Kumar. 2018. Enriching domain ontologies using question-answer datasets. In *Proc. CoDS-COMAD '18*, pages 329–332.

Swarnadeep Saha, Harinder Pal, et al. 2017. Bootstrapping for numerical open ie. In *Proc. ACL '17*, pages 317–323.

Denis Savenkov, Wei-Lwun Lu, Jeff Dalton, and Eugene Agichtein. 2015. Relation extraction from community generated question-answer pairs. In *Proc. NAACL-HLT '15*, pages 96–102.

Gabriel Stanovsky and Ido Dagan. 2016. Creating a large benchmark for open information extraction. In *Proc. EMNLP '16*.

Gabriel Stanovsky, Julian Michael, Luke Zettlemoyer, and Ido Dagan. 2018. Supervised open information extraction. In *Proc. ACL '18*, pages 885–895.

Antonio Toral and Víctor M. Sánchez-Cartagena. 2017. A multifaceted evaluation of neural versus phrase-based machine translation for 9 language directions. In *Proc. EACL '17*, pages 1063–1073.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Proc. NIPS '15*, pages 2692–2700.

Mengting Wan and Julian McAuley. 2016. Modeling ambiguity, subjectivity, and diverging viewpoints in opinion question answering systems. In *Proc. ICDM '16*, pages 489–498.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743.

Zeqiu Wu, Xiang Ren, Frank F. Xu, Ji Li, and Jiawei Han. 2018. Indirect supervision for relation extraction using question-answer pairs. In *Proc. WSDM '18*, pages 646–654.

Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *Proc. ACL '16*, pages 1341–1350.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. In *Proc. ICLR '17*.

Alexander Yates, Michael Cafarella, Michele Banko, Oren Etzioni, Matthew Broadhead, and Stephen Soderland. 2007. TextRunner: Open information extraction on the web. In *Proc. NAACL-HLT '07 (Demonstrations)*, pages 25–26.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proc. ACL '17*, pages 440–450.

Biao Zhang, Deyi Xiong, and Jinsong Su. 2016. Cseq2seq: Cyclic sequence-to-sequence learning. *arXiv preprint arXiv:1607.08725*.

Yaoyuan Zhang, Zhenxu Ye, Yansong Feng, Dongyan Zhao, and Rui Yan. 2017. A constrained sequence-to-sequence neural model for sentence simplification. *arXiv preprint arXiv:1704.02312*.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. In *Proc. NAACL-HLT '16*, pages 30–34.

A Supplementary Material

This supplementary material contains details of the analysis settings described in Section 4 and additional results not reported in the main paper.

A.1 Word Embedding Analysis

We investigated the embedding layers of the question encoder and the answer encoder of the NEURON model trained on the *ConciergeQA* dataset.

For robust analysis, we ignored non-English words and any words that contained numerical digits (e.g., #18D, $10). We used PYENCHANT[2] for filtering English words. For the remaining words, we find their embedding vectors from the two encoders, concatenate them to create a single matrix. This ensures that same embedding vectors are mapped to the same point in the visualization space. We used t-SNE[3] to map embedding vectors into 2D space for visualization.

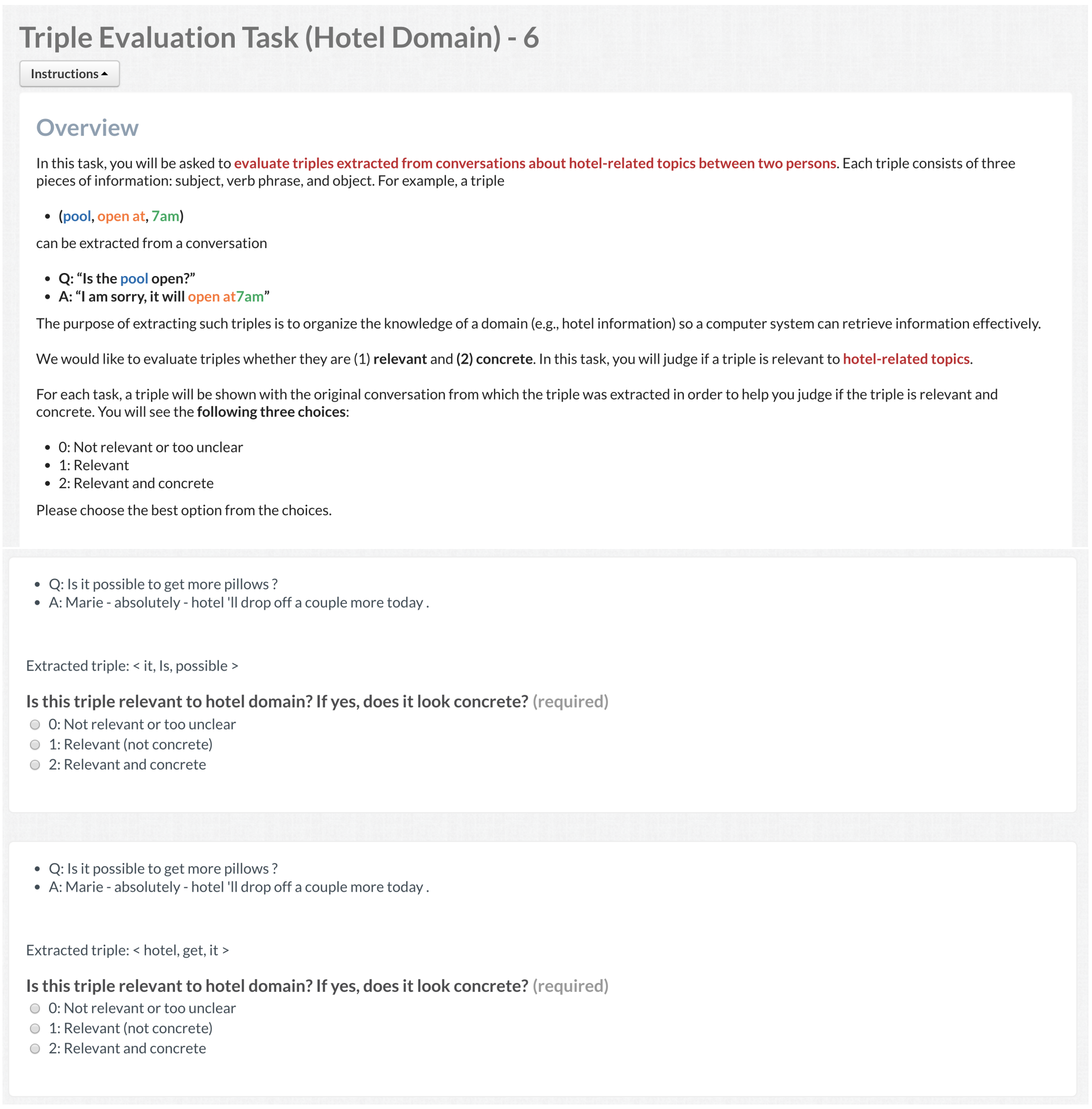
A.2 Crowdsourced Evaluation

Figure 5 shows the instructions and examples of the crowdsouced task. In the crowdsourcing task, crowdsourced workers were asked to judge after reading an extracted tuple with the original QA pair. Since it is difficult to define the usefulness of the tuples without assuming a KB, we used *relevance* and *concreteness* as criteria to grade extracted tuples. Specifically, each worker was asked to choose one option from the three options: `Not relevant or unclear` (0), `Relevant` (1), `Relevant and concrete` (2).

We set $0.05 as payment for each annotation. We carefully created 51 test questions which were used to filter out untrusted judgments and workers. The platform increases the number of annotators so each tuple should always have 5 trusted annotators. The 5 annotations for each tuple were aggregated into a single label with a confidence value that takes into account the accuracy rates of the annotators based on the test questions.

[2] v2.0.0 `https://github.com/rfk/pyenchant`

[3] TSNE v0.20.0 `https://scikit-learn.org/stable/` with default configuration



Triple Evaluation Task (Hotel Domain) - 6

Instructions ▴

Overview

In this task, you will be asked to **evaluate triples extracted from conversations about hotel-related topics between two persons**. Each triple consists of three pieces of information: subject, verb phrase, and object. For example, a triple

- (pool, open at, 7am)

can be extracted from a conversation

- **Q: "Is the pool open?"**
- **A: "I am sorry, it will open at7am"**

The purpose of extracting such triples is to organize the knowledge of a domain (e.g., hotel information) so a computer system can retrieve information effectively.

We would like to evaluate triples whether they are (1) **relevant** and **(2) concrete**. In this task, you will judge if a triple is relevant to **hotel-related topics**.

For each task, a triple will be shown with the original conversation from which the triple was extracted in order to help you judge if the triple is relevant and concrete. You will see the **following three choices**:

- 0: Not relevant or too unclear
- 1: Relevant
- 2: Relevant and concrete

Please choose the best option from the choices.

- Q: Is it possible to get more pillows ?
- A: Marie - absolutely - hotel 'll drop off a couple more today .

Extracted triple: < it, Is, possible >

Is this triple relevant to hotel domain? If yes, does it look concrete? (required)

- ○ 0: Not relevant or too unclear
- ○ 1: Relevant (not concrete)
- ○ 2: Relevant and concrete

- Q: Is it possible to get more pillows ?
- A: Marie - absolutely - hotel 'll drop off a couple more today .

Extracted triple: < hotel, get, it >

Is this triple relevant to hotel domain? If yes, does it look concrete? (required)

- ○ 0: Not relevant or too unclear
- ○ 1: Relevant (not concrete)
- ○ 2: Relevant and concrete

Figure 5: Screenshot of the instructions and examples of the crowdsourced task.