Commonsense Knowledge Aware Conversation Generation with Graph Attention

Hao Zhou\textsuperscript{1}, Tom Young\textsuperscript{2}, Minlie Huang\textsuperscript{1,*}, Haizhou Zhao\textsuperscript{3}, Jingfang Xu\textsuperscript{3} and Xiaoyan Zhu\textsuperscript{1}

\textsuperscript{1} Conversational AI Group, AI Lab., Dept. of Computer Science, Tsinghua University
Beijing National Research Center for Information Science and Technology, China
\textsuperscript{2} School of Information and Electronics, Beijing Institute of Technology, China
\textsuperscript{3} Sogou Inc., Beijing, China

tuxchow@gmail.com, tom@sentic.net, aihuang@tsinghua.edu.cn

Abstract

Commonsense knowledge is vital to many natural language processing tasks. In this paper, we present a novel open-domain conversation generation model to demonstrate how large-scale commonsense knowledge can facilitate language understanding and generation. Given a user post, the model retrieves relevant knowledge graphs from a knowledge base and then encodes the graphs with a \textit{static graph attention} mechanism, which augments the semantic information of the post and thus supports better understanding of the post. Then, during word generation, the model attentively reads the retrieved knowledge graphs and the knowledge triples within each graph to facilitate better generation through a \textit{dynamic graph attention} mechanism. This is the first attempt that uses large-scale commonsense knowledge in conversation generation. Furthermore, unlike existing models that use knowledge triples (entities) separately and independently, our model treats each knowledge graph as a whole, which encodes more structured, connected semantic information in the graphs. Experiments show that the proposed model can generate more appropriate and informative responses than state-of-the-art baselines.

1 Introduction

Semantic understanding, particularly when facilitated by commonsense knowledge or world facts, is essential to many natural language processing tasks [Wang et al., 2017; Lin et al., 2017], and undoubtedly, it is a key factor to the success of dialogue or conversational systems, as conversational interaction is a semantic activity [Egglins and Slade, 2005]. In open-domain conversational systems, commonsense knowledge is important for establishing effective interactions, since socially shared commonsense knowledge is the set of background information people intended to know and use during conversation [Minsky, 1991; Marková et al., 2007; Speer and Havasi, 2012; Souto, 2015].

*Corresponding author: Minlie Huang.

Recently, a variety of neural models has been proposed for conversation generation [Ritter et al., 2011; Shang et al., 2015]. However, these models tend to generate generic responses, which are unable to respond appropriately and informatively in most cases, because it is challenging to learn semantic interactions \textit{merely} from conversational data [Ghazvininejad et al., 2017] without deep understanding of user input, and the background knowledge and the context of conversation. A model can understand conversations better and thus respond more properly if it can access and make full use of large-scale commonsense knowledge. For instance, to understand a post-response pair “Don’t order drinks at the restaurant, ask for free water” and “Not in Germany. Water cost more than beer. Bring you own water bottle”, we need commonsense knowledge such as (water, AtLocation, restaurant), (free, RelatedTo, cost), etc.

Some prior studies have been conducted to introduce external knowledge in conversation generation [Han et al., 2015; Ghazvininejad et al., 2017; Zhu et al., 2017]. The knowledge used in these models is either unstructured texts [Ghazvininejad et al., 2017] or domain-specific knowledge triples [Zhu et al., 2017]. Therefore, such models face with two issues when they are applied to open-domain, open-topic conversation generation. \textbf{First}, they are highly dependent on the quality of unstructured texts or limited by the small-scale, domain-specific knowledge. \textbf{Second}, they usually make use of knowledge triples (entities) separately and independently, instead of treating knowledge triples as a whole in a graph. Thus, they are unable to represent the semantics of a graph via linked entities and relations.

To address the two issues, we propose a commonsense knowledge aware conversational model (CCM) to facilitate knowledge understanding and generation in open-domain conversational systems. We use a large-scale commonsense knowledge [Speer and Havasi, 2012] to help understand the background information of a given post, and to facilitate response generation with such knowledge. The model retrieves a few knowledge graphs for each post and then use the graphs to respond more informatively and appropriately, as shown in Figure 1. To fully leverage the retrieved graphs in conversation generation, two novel graph attention mechanisms are designed. A static graph attention mechanism encodes...
Moonlight lacks the ultraviolet rays of sunlight. I don’t think that’s a lack of uv.
Moonlight lacks the ultraviolet rays of sunlight. I’m not sure what you’re saying.

Figure 1: (Better viewed in color) Two response examples by our model (the first line) and Seq2Seq (second) with/without considering commonsense knowledge, respectively.

the retrieved graphs for a post to augment the semantic representation of the post, which can help understand the post. A dynamic graph attention mechanism attentively reads the knowledge graphs and the triples in each graph, and then uses the semantic information from the graphs and triples for better response generation.

In summary, this paper makes the following contributions:

- This work is the first attempt that uses large-scale commonsense knowledge in neural conversation generation. Supported by such knowledge, our model can understand the dialogue better and thus respond more appropriately and informatively.
- Instead of treating knowledge triples (or entities) separately and independently, we devise static and dynamic graph attention mechanisms to treat the knowledge triples as a graph, from which we can better interpret the semantics of an entity from its neighboring entities and relations.

2 Related Work

Open-domain Conversational Models

Recently, sequence-to-sequence models [Sutskever et al., 2014; Bahdanau et al., 2014] have been successfully applied to large-scale conversation generation, including neural responding machine [Shang et al., 2015], hierarchical recurrent models [Serban et al., 2015], and many others [Sordoni et al., 2015]. These models developed various techniques to improve the content quality of generated responses, including diversity promotion [Li et al., 2016; Shao et al., 2017], considering additional information [Xing et al., 2017; Mou et al., 2016], and handling unknown words [Gu et al., 2016]. However, generic or meaningless responses are still commonly seen in these models due to the inability of good understanding of the user input or other context.

Unstructured Texts Enhanced Conversational Models

Several studies incorporated unstructured texts as external knowledge into conversation generation [Ghazvininejad et al., 2017; Long et al., 2017]. [Ghazvininejad et al., 2017] used memory network which stores unstructured texts to improve conversation generation. [Long et al., 2017] applied a convolutional neural network to extract knowledge from unstructured texts to generate multi-turn conversations. However, these models largely depend on the quality of unstructured texts, which may introduce noise in conversation generation if the texts are irrelevant.

Structured Knowledge Enhanced Conversational Models

There exist some models that introduced high-quality structured knowledge for conversation generation [Han et al., 2015; Zhu et al., 2017; Xu et al., 2017]. [Xu et al., 2017] incorporated a structured domain-specific knowledge base into conversation generation with a recall-gate mechanism. [Zhu et al., 2017] presented an end-to-end knowledge grounded conversational model using a copy network [Gu et al., 2016]. However, these studies are somehow limited by the small domain-specific knowledge base, making them not applicable for open-domain, open-topic conversation generation. By contrast, our model applies a large-scale commonsense knowledge base to facilitate both the understanding of a post and the generation of a response, with novel graph attention mechanisms.

3 Commonsense Conversational Model

3.1 Background: Encoder-decoder Framework

First of all, we introduce a general encoder-decoder framework which is based on sequence-to-sequence (seq2seq) learning [Sutskever et al., 2014]. The encoder represents a post sequence \( X = x_1x_2 \cdots x_n \) with hidden representations \( H = h_1h_2 \cdots h_n \), which is briefly defined as below:

\[
 h_t = \text{GRU}(h_{t-1}, e(x_t)),
\]

where \( e(x_t) \) is the embedding of the word \( x_t \), and GRU is a gated recurrent unit [Cho et al., 2014].

The decoder takes as input a context vector \( e_t \) and the embedding of a previously decoded word \( e(y_{t-1}) \), and updates its state \( s_t \) using another GRU:

\[
 s_t = \text{GRU}(s_{t-1}, [c_{t-1}; e(y_{t-1})]),
\]

where \( [c_{t-1}; e(y_{t-1})] \) is the concatenation of the two vectors, serving as input to the GRU network. The context vector \( c_{t-1} \) is an attentive read of \( H \), which is a weighted sum of the encoder’s hidden states as \( c_{t-1} = \sum_{k=1}^{n} \alpha_k h_k \), and \( \alpha_k \) measures the relevance between state \( s_{t-1} \) and hidden state \( h_k \). Refer to [Bahdanau et al., 2014] for more details.

The decoder generates a token by sampling from the output probability distribution which can be computed as follows:

\[
 y_t \sim \alpha_t = P(y_t \mid y_{\leq t}, c_t) = \text{softmax}(W_\alpha s_t),
\]

where \( y_{\leq t} = y_1y_2 \cdots y_{t-1} \), the words already generated.

\footnote{Throughout the paper, a normal letter denotes a discrete symbol while a bolded letter denotes a vector.}
3.2 Task Definition and Overview

Our problem is formulated as follows: Given a post \( X = x_1x_2 \cdots x_n \) and some commonsense knowledge graphs \( G = \{g_1, g_2, \cdots, g_{N_G}\} \), the goal is to generate a proper response \( Y = y_1y_2 \cdots y_m \). Essentially, the model estimates the probability: 

\[
P(Y|X,G) = \prod_{i=1}^{m} P(y_i|y_{<i}, X, G).
\]

The graphs are retrieved from a knowledge base using the words in a post as queries, and each word corresponds to a graph in \( G \). Each graph consists of a set of triples \( g_i = \{\tau_1, \tau_2, \cdots, \tau_{N_{gi}}\} \) and each triple (head entity, relation, tail entity) is denoted as \( \tau = (h, r, t) \).

We adopt TransE [Bordes et al., 2013] to represent the entities and relations in the knowledge base. In order to bridge the representation gap between knowledge base and unstructured conversational texts, we adopt a MLP for the purpose: a knowledge triple \( \tau \) is represented by \( k = (h, r, t) = \text{MLP}(\text{TransE}(h, r, t)) \), where \( h/r/t \) are the transformed TransE embeddings for \( h/r/t \) respectively.

We adopt TransE [Bordes et al., 2013] to represent the entities and relations in the knowledge base. In order to bridge the representation gap between knowledge base and unstructured conversational texts, we adopt a MLP for the purpose: a knowledge triple \( \tau \) is represented by \( k = (h, r, t) = \text{MLP}(\text{TransE}(h, r, t)) \), where \( h/r/t \) are the transformed TransE embeddings for \( h/r/t \) respectively.

The overview of our commonsense conversational model (CCM) is presented in Figure 2. The knowledge interpreter takes as input a post \( X = x_1x_2 \cdots x_n \) and retrieved knowledge graphs \( G = \{g_1, g_2, \cdots, g_{N_G}\} \) to obtain knowledge-aware representations at each word position, by concatenating a word vector and its corresponding knowledge graph vector. A knowledge graph vector represents a knowledge graph for the corresponding word in \( X \) through a static graph attention mechanism. The knowledge-aware generator generates a response \( Y = y_1y_2 \cdots y_m \) with our dynamic graph attention mechanism. At each decoding position, it attentively reads the retrieved graphs and the entities in each graph, and then generates a generic word in the vocabulary or an entity in the

knowledge graphs. The entity is selected by attending on the graphs and the triples within each graph.

3.3 Knowledge Interpreter

The knowledge interpreter is designed to facilitate the understanding of a post. It augments the semantics of a word by including the corresponding graph vector for the word, as shown in Figure 3. The knowledge interpreter uses each word \( x_t \) in a post as the key entity to retrieve a graph \( g_i = \{\tau_1, \tau_2, \cdots, \tau_{N_{gi}}\} \) (the yellow parts) from the entire commonsense knowledge base. Each retrieved graph consists of a key entity (the red dots), its neighboring entities (the blue dots) and relations between entities. For common words (e.g., of) which match no entity in the commonsense knowledge graph, a knowledge graph that contains a special symbol Not_A_Fact (the grey dots) is used. Then, the knowledge interpreter computes the graph vector \( g_i \) of the retrieved graph using the static graph attention mechanism. After concatenating the word vector \( w(x_t) \) and the knowledge graph vector \( g_i \), the concatenated vector \( e(x_t) = [w(x_t); g_i] \) is obtained and fed to the GRU cell of the encoder (see Eq. 1).

### Static Graph Attention

The static graph attention mechanism is designed to generate a representation for a retrieved knowledge graph, inspired by [Velickovic et al., 2017]. The major difference to [Velickovic et al., 2017] lies in that our graph attention encodes more structured semantic information by considering not only all nodes in a graph but also relations between nodes.

The static graph attention generates a static representation for a graph, which will be used to augment the semantics of a word in a post.

Formally, the static graph attention mechanism takes as input the knowledge triple vectors \( K(g_i) = \{k_1, k_2, \cdots, k_{N_{gi}}\} \) in the retrieved knowledge graph \( g_i \), to produce a graph vector \( \hat{g}_i \) as follows:
Dynamic Graph Attention Mechanism

A dynamic graph attention mechanism is a hierarchical, top-down process. It first attentively reads all the knowledge graphs and then attentively reads all the triples in each graph for final word generation. Given the decoder state $s_t$, it first attends on the knowledge graph vectors $\{g_1, g_2, \ldots, g_{NC}\}$ to compute the probability of using of each graph $g_i$, which is defined as below:

$$c^i_t = \sum_{i=1}^{NC} \alpha^g_i g_i,$$  \hspace{1cm} (9)

$$\alpha^g_i = \frac{\exp(\beta^g_i)}{\sum_{j=1}^{NC} \exp(\beta^g_j)},$$  \hspace{1cm} (10)

$$\beta^g_i = (W_r r_n + W_t t_n),$$  \hspace{1cm} (11)

where $(h_n, r_n, t_n) = k_n$, $W_h, W_r, W_t$ are weight matrices for head entities, relations, and tail entities, respectively.

The attention weight measures the association of a relation $r_n$ to a head entity $h_n$ and a tail entity $t_n$.

Essentially, a graph vector $g_i$ is a weighted sum of the head and tail vectors $[h_n; t_n]$ of the triples contained in the graph.

### 3.4 Knowledge Aware Generator

The knowledge aware generator is designed to generate a response through making full use of the retrieved knowledge graphs, as shown in Figure 4. The knowledge aware generator plays two roles: 1) attentively reading the retrieved graphs to obtain a graph-aware context vector, and using the vector to update the decoder’s state; 2) adaptively choosing a generic word or an entity from the retrieved graphs for word generation. Formally, the decoder updates its state as follows:

$$s_{t+1} = \text{GRU}(s_t, [c_t; c^g_t; c^K_t; e(y_t)]),$$  \hspace{1cm} (7)

$$e(y_t) = [w(y_t); k_j],$$  \hspace{1cm} (8)

where $e(y_t)$ is the concatenation of the word vector $w(y_t)$ and the previous knowledge triple vector $k_j$ from which the previous word $y_t$ is selected.

$c_t$ is the context vector as used in Eq. 2, $c^g_t$ and $c^K_t$ are context vectors attended on knowledge graph vectors $\{g_1, g_2, \ldots, g_{NC}\}$ and knowledge triple vectors $\{K(g_1), K(g_2), \ldots, K(g_{NC})\}$ respectively.

**Dynamic Graph Attention**

The dynamic graph attention mechanism is a hierarchical, top-down process. It first attentively reads all the knowledge graphs and then attentively reads all the triples in each graph for final word generation. Given the decoder state $s_t$, it first attends on the knowledge graph vectors $\{g_1, g_2, \ldots, g_{NC}\}$ to compute the probability of using of each graph $g_i$, which is defined as below:

$$c^i_t = \sum_{i=1}^{NC} \alpha^g_i g_i,$$  \hspace{1cm} (9)

$$\alpha^g_i = \frac{\exp(\beta^g_i)}{\sum_{j=1}^{NC} \exp(\beta^g_j)},$$  \hspace{1cm} (10)

$$\beta^g_i = (W_r r_n + W_t t_n),$$  \hspace{1cm} (11)

where $(h_n, r_n, t_n) = k_n$, $W_h, W_r, W_t$ are weight matrices for head entities, relations, and tail entities, respectively.

The attention weight measures the association of a relation $r_n$ to a head entity $h_n$ and a tail entity $t_n$.

Essentially, a graph vector $g_i$ is a weighted sum of the head and tail vectors $[h_n; t_n]$ of the triples contained in the graph.

### 3.4 Knowledge Aware Generator

The knowledge aware generator is designed to generate a response through making full use of the retrieved knowledge graphs, as shown in Figure 4. The knowledge aware generator plays two roles: 1) attentively reading the retrieved graphs to obtain a graph-aware context vector, and using the vector to update the decoder’s state; 2) adaptively choosing a generic word or an entity from the retrieved graphs for word generation. Formally, the decoder updates its state as follows:

$$s_{t+1} = \text{GRU}(s_t, [c_t; c^g_t; c^K_t; e(y_t)]),$$  \hspace{1cm} (7)

$$e(y_t) = [w(y_t); k_j],$$  \hspace{1cm} (8)

where $e(y_t)$ is the concatenation of the word vector $w(y_t)$ and the previous knowledge triple vector $k_j$ from which the previous word $y_t$ is selected.

$c_t$ is the context vector as used in Eq. 2, $c^g_t$ and $c^K_t$ are context vectors attended on knowledge graph vectors $\{g_1, g_2, \ldots, g_{NC}\}$ and knowledge triple vectors $\{K(g_1), K(g_2), \ldots, K(g_{NC})\}$ respectively.

**Dynamic Graph Attention**

The dynamic graph attention mechanism is a hierarchical, top-down process. It first attentively reads all the knowledge graphs and then attentively reads all the triples in each graph for final word generation. Given the decoder state $s_t$, it first attends on the knowledge graph vectors $\{g_1, g_2, \ldots, g_{NC}\}$ to compute the probability of using of each graph $g_i$, which is defined as below:

$$c^i_t = \sum_{i=1}^{NC} \alpha^g_i g_i,$$  \hspace{1cm} (9)

$$\alpha^g_i = \frac{\exp(\beta^g_i)}{\sum_{j=1}^{NC} \exp(\beta^g_j)},$$  \hspace{1cm} (10)

$$\beta^g_i = (W_r r_n + W_t t_n),$$  \hspace{1cm} (11)

where $(h_n, r_n, t_n) = k_n$, $W_h, W_r, W_t$ are weight matrices for head entities, relations, and tail entities, respectively.

The attention weight measures the association of a relation $r_n$ to a head entity $h_n$ and a tail entity $t_n$.

Essentially, a graph vector $g_i$ is a weighted sum of the head and tail vectors $[h_n; t_n]$ of the triples contained in the graph.

### 3.4 Knowledge Aware Generator

The knowledge aware generator is designed to generate a response through making full use of the retrieved knowledge graphs, as shown in Figure 4. The knowledge aware generator plays two roles: 1) attentively reading the retrieved graphs to obtain a graph-aware context vector, and using the vector to update the decoder’s state; 2) adaptively choosing a generic word or an entity from the retrieved graphs for word generation. Formally, the decoder updates its state as follows:

$$s_{t+1} = \text{GRU}(s_t, [c_t; c^g_t; c^K_t; e(y_t)]),$$  \hspace{1cm} (7)

$$e(y_t) = [w(y_t); k_j],$$  \hspace{1cm} (8)

where $e(y_t)$ is the concatenation of the word vector $w(y_t)$ and the previous knowledge triple vector $k_j$ from which the previous word $y_t$ is selected.

$c_t$ is the context vector as used in Eq. 2, $c^g_t$ and $c^K_t$ are context vectors attended on knowledge graph vectors $\{g_1, g_2, \ldots, g_{NC}\}$ and knowledge triple vectors $\{K(g_1), K(g_2), \ldots, K(g_{NC})\}$ respectively.

**Dynamic Graph Attention**

The dynamic graph attention mechanism is a hierarchical, top-down process. It first attentively reads all the knowledge graphs and then attentively reads all the triples in each graph for final word generation. Given the decoder state $s_t$, it first attends on the knowledge graph vectors $\{g_1, g_2, \ldots, g_{NC}\}$ to compute the probability of using of each graph $g_i$, which is defined as below:

$$c^i_t = \sum_{i=1}^{NC} \alpha^g_i g_i,$$  \hspace{1cm} (9)

$$\alpha^g_i = \frac{\exp(\beta^g_i)}{\sum_{j=1}^{NC} \exp(\beta^g_j)},$$  \hspace{1cm} (10)

$$\beta^g_i = (W_r r_n + W_t t_n),$$  \hspace{1cm} (11)

where $(h_n, r_n, t_n) = k_n$, $W_h, W_r, W_t$ are weight matrices for head entities, relations, and tail entities, respectively.

The attention weight measures the association of a relation $r_n$ to a head entity $h_n$ and a tail entity $t_n$.

Essentially, a graph vector $g_i$ is a weighted sum of the head and tail vectors $[h_n; t_n]$ of the triples contained in the graph.
training corpus. Additionally, we apply supervised signals on the knowledge aware generator layer to teacher-force the selection of an entity or a generic word. The loss on one sample $<X, Y> (X = x_1x_2\cdots x_n, Y = y_1y_2\cdots y_m)$ is defined as:

$$L(\theta) = -\sum_{t=1}^{m} p_t \log(\alpha_t) - \sum_{t=1}^{m} (q_t \log(\gamma_t) + (1-q_t) \log(1-\gamma_t)),$$

where $\gamma_t$ is the probability of selecting an entity word or a generic word, and $q_t \in \{0, 1\}$ is the true choice of an entity word or a generic word in $Y$. The second term is used to supervise the probability of selecting an entity word or a generic word.

4 Experiments

4.1 Dataset

Commonsense Knowledge Base

ConceptNet$^4$ is used as the commonsense knowledge base. It contains not only world facts such as “Paris is the capital of France” that are constantly true, but also informal relations between common concepts that are part of daily knowledge such as “A dog is a pet”. This feature is desirable in our experiments, because the ability to recognize the informal relations between common concepts is necessary in the open-domain conversation setting. For simplicity, we removed triples containing multi-word entities, and 120,850 triples were retained with 21,471 entities and 44 relations.

Commonsense Conversation Dataset

We adopted 10M reddit single-round dialogs from the site$^5$. Since we target at using commonsense knowledge to facilitate language understanding and generation, we filtered the original corpus with the knowledge triples. If a post-response pair can not be connected by any triple (that is, one entity appears in the post and the other in the response), the pair will be removed. The statistics can be seen in Table 1.

We randomly sampled 10,000 pairs for validation. To test how commonsense knowledge can help understand common or rare concepts in a post, we constructed four test sets: high-frequency pairs in which each post has all top 25% frequent words, medium-frequency pairs where each post contains at least one word whose frequency is within the range of 25%-75%, low-frequency pairs within the range of 75%-100%, and OOV pairs where each post contains out-of-vocabulary words. Each test set has 5,000 pairs randomly sampled from the dataset$^6$.

4.2 Implementation Details

Our model was implemented with Tensorflow$^7$.

<table>
<thead>
<tr>
<th>Conversational Pairs</th>
<th>Commonsense KB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training 3,384,185</td>
<td>Entity 21,471</td>
</tr>
<tr>
<td>Validation 10,000</td>
<td>Relation 44</td>
</tr>
<tr>
<td>Test 20,000</td>
<td>Triple 120,850</td>
</tr>
</tbody>
</table>

Table 1: Statistics of the dataset and the knowledge base.

cells for each layer and they do not share parameters. The word embedding size is set to 300. The vocabulary size is limited to 30,000. We used TransE [Bordes et al., 2013] to obtain entity and relation representations. The embedding size of entities and relations is set to 100.

We used the Adam optimizer with a mini-batch size of 100. The learning rate is 0.0001. The models were ran at most 20 epochs, and the training stage of each model took about a week on a Titan X GPU machine. Our code is available at: https://github.com/tuxchow/ccm.

4.3 Baselines

We chose several suitable baselines:

- A seq2seq model (Seq2Seq) [Sutskever et al., 2014], which is widely used in open-domain conversational systems.
- A knowledge-grounded model (MemNet) adapted from [Ghazvininejad et al., 2017], where the memory units store TransE embeddings of knowledge triples.
- A copy network (CopyNet) model [Zhu et al., 2017], which copies a word from knowledge triples or generates a word from the vocabulary.

4.4 Automatic Evaluation

Metrics: We adopted perplexity [Serban et al., 2015] to evaluate the model at the content level (whether the content is grammatical and relevant in topic). We also calculated the number of entities per response to measure the model’s ability to select the concepts from the commonsense knowledge base in generation. This metric is denoted by entity score.

Results: As shown in Table 2, CCM obtains the lowest perplexity on all the test sets, indicating that CCM can understand users’ posts better and generate more grammatical responses. Moreover, CCM selects the most entities from the commonsense knowledge among the models during generation, demonstrating that commonsense knowledge can truly facilitate response generation.

More interestingly, commonsense knowledge is more frequently used by CCM in low-frequency posts than in high-frequency ones (entity score: 1.196 vs. 1.156). This is in line with our intuition that rare concepts need more background knowledge to understand and respond. For the perplexity, the score for high-frequency posts is lower than low-frequency ones (35.36 vs. 40.67) since the probabilities for the common words can be more sufficiently trained.

4.5 Manual Evaluation

We resorted to a crowdsourcing service, Amazon Mechanical Turk, for manual annotation. 400 posts were randomly sampled for manual annotation. We conducted pair-wise comparison between the response generated by CCM and the one by...
a baseline for the same post. In total, there are 1,200 pairs since we have three baselines. For each response pair, seven judges were hired to give a preference between the two responses, in terms of the following two metrics. Tie was allowed. Notice that system identifiers were masked during annotation.

**Metrics:** We defined two metrics: *appropriateness* at the content level (whether the response is appropriate in grammar, topic, and logic); and *informativeness* at the knowledge level (whether the response provides new information and knowledge in addition to the post).

**Annotation Statistics:** We calculated the agreements to measure inter-rater consistency. For *appropriateness*, the percentage of the pairs that at least 4 judges gave the same label (4/7 agreement) amounts to 96.3%, and the percentage for at least 5/7 agreement is 60.2%. For *informativeness*, the percentage for at least 4/7 agreement is 90.4% and that for at least 5/7 agreement is 55.1%.

**Results:** The results are shown in Table 3. The score is the percentage that CCM wins its competitor after removing “Tie” pairs. CCM is significantly better (sign test, p-value < 0.005) than all the baselines on all the test sets.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Seq2Seq</td>
<td>47.02</td>
<td>0.717</td>
<td>42.41</td>
<td>0.713</td>
<td>47.25</td>
<td>0.740</td>
<td>48.61</td>
<td>0.721</td>
<td>49.96</td>
<td>0.669</td>
</tr>
<tr>
<td>MemNet</td>
<td>46.85</td>
<td>0.761</td>
<td>41.93</td>
<td>0.764</td>
<td>47.32</td>
<td>0.788</td>
<td>48.86</td>
<td>0.760</td>
<td>49.52</td>
<td>0.706</td>
</tr>
<tr>
<td>CopyNet</td>
<td>40.27</td>
<td>0.96</td>
<td>36.26</td>
<td>0.91</td>
<td>40.99</td>
<td>0.97</td>
<td>42.09</td>
<td>0.96</td>
<td>42.24</td>
<td>0.96</td>
</tr>
<tr>
<td>CCM</td>
<td><strong>39.18</strong></td>
<td><strong>1.180</strong></td>
<td><strong>35.36</strong></td>
<td><strong>1.156</strong></td>
<td><strong>39.64</strong></td>
<td><strong>1.191</strong></td>
<td><strong>40.67</strong></td>
<td><strong>1.196</strong></td>
<td><strong>40.87</strong></td>
<td><strong>1.162</strong></td>
</tr>
</tbody>
</table>

Table 2: Automatic evaluation with perplexity (ppx.), and entity score (ent.).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CCM vs. Seq2Seq</td>
<td>0.616</td>
<td>0.662</td>
<td>0.605</td>
<td>0.656</td>
<td>0.549</td>
<td>0.624</td>
<td>0.636</td>
<td>0.650</td>
<td>0.673</td>
<td>0.716</td>
</tr>
<tr>
<td>CCM vs. MemNet</td>
<td>0.602</td>
<td>0.647</td>
<td>0.593</td>
<td>0.656</td>
<td>0.566</td>
<td>0.640</td>
<td>0.622</td>
<td>0.635</td>
<td>0.626</td>
<td>0.657</td>
</tr>
<tr>
<td>CCM vs. CopyNet</td>
<td>0.600</td>
<td>0.640</td>
<td>0.606</td>
<td>0.669</td>
<td>0.586</td>
<td>0.619</td>
<td>0.610</td>
<td>0.633</td>
<td>0.596</td>
<td>0.640</td>
</tr>
</tbody>
</table>

Table 3: Manual evaluation with appropriateness (app.), and informativeness (inf.). The score is the percentage that CCM wins its competitor after removing “Tie” pairs. CCM is significantly better (sign test, p-value < 0.005) than all the baselines on all the test sets.

Noticeably, the probabilities that CCM wins Seq2Seq on the OOV dataset are remarkably higher than those on the high-frequency dataset (0.673 vs. 0.605 in appropriateness, and 0.716 vs. 0.656 in informativeness). This further indicates that commonsense knowledge is more useful in understanding rare concepts in post since Seq2Seq has no ability to use such knowledge. For MemNet and CopyNet, we did not observe such differences because the two baselines have the ability of using knowledge more or less.

**4.5 Case Study**

A sample conversation is shown in Table 4. The red-colored word “breakable” in the post is an entity word in the knowledge base as well as an out-of-vocabulary word for all the models. Without the access to commonsense knowledge, the Seq2Seq model is unable to understand the post because of the out-of-vocabulary word, “breakable”, thereby generating OOV words. MemNet can generate some meaningful words as it reads the triple embeddings in its memory, but still outputs OOV. CopyNet can read and copy words from knowledge triples. However, CopyNet generated fewer entity words than ours (see Table 2), as it only deals with separate knowledge triples. Instead, CCM treats the knowledge graph as a whole and encodes more structured, connected information via linked entities and relations. It thus generates more reasonable responses through better use of knowledge. This simple example shows that CCM can generate more appropriate and informative responses than the baselines.

<table>
<thead>
<tr>
<th>Post</th>
<th>Why are you so breakable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knowledge</td>
<td>(glass, RelatedTo, breakable), (brittle, RelatedTo, breakable), (rule, RelatedTo, breakable)</td>
</tr>
<tr>
<td>Seq2Seq</td>
<td>I’m not a OOV, I’m just a OOV.</td>
</tr>
<tr>
<td>MemNet</td>
<td>I’m not OOV, I’m just a really nice person.</td>
</tr>
<tr>
<td>CopyNet</td>
<td>I’m not. I’m just a lurker.</td>
</tr>
<tr>
<td>CCM</td>
<td>Because I’m a brittle man.</td>
</tr>
</tbody>
</table>

Table 4: Sample responses generated by all the models.

5 Conclusion and Future Work

In this paper, we present a commonsense knowledge aware conversational model (CCM) to demonstrate how commonsense knowledge can facilitate language understanding and generation in open-domain conversational systems. Automatic and manual evaluation show that CCM can generate more appropriate and informative responses than state-of-the-art baselines.

As future work, our graph attention mechanisms may inspire other tasks to use commonsense knowledge.
Acknowledgments
This work was partly supported by the National Science Foundation of China under grant No.61272227/61332007 and the National Basic Research Program (973 Program) under grant No. 2013CB329403.

References