Conversational Query Understanding Using Sequence to Sequence Modeling

Gary Ren
Microsoft Bing Core Relevance
Sunnyvale, California
gren@microsoft.com

Xiaochuan Ni
Microsoft Bing Core Relevance
Sunnyvale, California
xiaon@microsoft.com

Manish Malik
Microsoft Bing Core Relevance
Sunnyvale, California
manisma@microsoft.com

Qifa Ke
Microsoft Bing Core Relevance
Sunnyvale, California
qke@microsoft.com

ABSTRACT

Understanding conversations is crucial to enabling conversational search in technologies such as chatbots, digital assistants, and smart home devices that are becoming increasingly popular. Conventional search engines are powerful at answering open domain queries but are mostly capable of stateless search. In this paper, we define a conversational query as a query that depends on the context of the current conversation, and we formulate the conversational query understanding problem as context-aware query reformulation, where the goal is to reformulate the conversational query into a search engine friendly query in order to satisfy users' information needs in conversational settings. Such context-aware query reformulation problem lends itself to sequence to sequence modeling. We present a large scale open domain dataset of conversational queries and various sequence to sequence models that are learned from this dataset. The best model correctly reformulates over half of all conversational queries, showing the potential of sequence to sequence modeling for this task.

CCS CONCEPTS

• **Computing methodologies** → **Discourse, dialogue and pragmatics**; *Artificial intelligence*; *Natural language processing*;

KEYWORDS

Search; deep learning; conversations; query understanding; sequence to sequence

ACM Reference Format:
Gary Ren, Xiaochuan Ni, Manish Malik, and Qifa Ke. 2018. Conversational Query Understanding Using Sequence to Sequence Modeling. In *WWW 2018: The 2018 Web Conference, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3178876.3186083

WWW 2018, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5639-8/18/04.
https://doi.org/10.1145/3178876.3186083

1 INTRODUCTION

The recent rise of technologies such as chatbots, digital personal assistants, and smart home devices [28] has led to much more conversational interactions between humans and machines than ever before. In conversations, humans naturally ask questions that depend on the context of the current conversation. A basic example is asking "When was California founded?" followed by "Who is its governor?" and "What is the population?", where both the follow up questions refer to California. For this paper, we define a conversational query to be a query that depends on the context of the current conversation, and the query can be either a natural language question or a traditional keyword query. Some more examples of conversational queries can be seen in Table 1.

Humans use this type of questions in conversations because it is tedious to continuously repeat the context and because we can maintain context and understand these questions. Therefore, users also naturally issue conversational queries when interacting with conversational technologies and expect these technologies to understand them.

However, information retrieval and question answering systems have traditionally been designed for stateless or standalone queries, and existing conversational query understanding capabilities are very limited. For even the simple aforementioned example about California, neither of two popular commercial search engines was able to correctly understand both the follow up queries (as of October 23, 2017).

Therefore, we want to improve query understanding for conversational search, by tackling the task of conversational query understanding (referred to as CQU from now on), which we formulate as a context-aware query reformulation task that consists of determining 1) whether or not a query depends on the previous context, and 2) if so, how to reformulate that query to include the necessary context. This will enable conversational queries to be reformulated to more search friendly standalone queries that can be understood by search engines or any other IR/QnA system.

Since we are targeting the open domain search scenario, our goal is CQU that is also open domain and can handle a wide variety of queries, both in terms of topic and structure. This is a major challenge of this open domain CQU task. Other challenges include: handling different types of context (Table 1 shows that the context to be maintained can be an entity, concept, question, etc.); knowing

Table 1: Examples of conversational queries

Previous query	Current conversational query	Current conversational query including context
when was California founded?	who is its governor?	who is California's governor?
California	population in 1990	population of California in 1990
Space Needle Seattle	its mayor	Seattle's mayor
how tall is Kobe Bryant?	what about Lebron James?	how tall is Lebron James?
when was the last summer Olympics?	and the winter one?	when was the last winter Olympics?
animals that live in Asia?	and are endangered?	animals that live in Asia and are endangered?
similarities between bacteria and viruses	differences	differences between bacteria and viruses

when to reformulate (e.g. "it" is not always referring to previous context); and knowing which part of the context to use (e.g. using "Seattle" instead of "Space Needle" for "its mayor"). It is infeasible for a rule based solution to address all of these challenges, therefore we sought to create a suitable dataset and apply machine learning. The main contributions of our work are:

(1) Defining and presenting the task of CQU.
(2) First open domain and large scale dataset of conversational queries.
(3) Novel sequence to sequence based model for CQU.
(4) Demonstrating the potential of deep learning for CQU.

2 RELATED WORK

For conversational queries that include an anaphora, coreference resolution is a related problem. Coreference resolution seeks to resolve an anaphora to the term(s) that it refers to. There has been lots of research done on coreference resolution and recent research with deep learning has achieved state of the art results [10] on this challenging task. However, existing coreference resolution systems have several limitations when applied to CQU. The following examples were tried with the coreference resolution service from the Stanford CoreNLP toolkit [24]:

(1) Multiple possible entities (e.g. for "Is Space Needle in Seattle? Who is its mayor?", the "its" is incorrectly resolved to "Space" instead of "Seattle")
(2) Knowing when a reformulation is actually needed (e.g. for "When was California founded? How long does it take bruised ribs to heal?", the "it" is incorrectly resolved to "California").
(3) When there isn't an explicit referring anaphora (e.g. for "When was California founded? What is the population?", there is no anaphora in the second query that explicitly refers to "California").

Example 3 shows that even the perfect coreference resolution system will not be able to handle the conversational queries that lack anaphoras, such as some of the examples in Table 1.

Paraphrase generation, where the goal is to generate a paraphrase for a given sentence, is also a related task. Research has been done on applying deep learning to paraphrase generation with state of the art results [27]; however, no existing research has been done on generating paraphrases that depend on context beyond just the input sentence.

There has also been research on dialogue agents using neural networks and reinforcement learning [13, 31, 33]. While these research show promising results, they are limited by being very domain specific, e.g. finding a movie.

Another related area is context-aware search or context-aware query suggestion for search. Bar-Yossef et al. [3], Li et al. [20, 21], Cao et al. [6], Sordoni et al. [32] and Dehghani et al. [12] used previous queries as context along with auto-completion logs and/or click logs to improve query suggestions for query auto-completion. Their work consists of either ranking or generating query suggestions. The query ranking task fundamentally differs from our query reformulation/generation task. For the solutions that generated query suggestions, they were focused on completing the query instead of modifying existing parts of the query. Also, these solutions are well suited for a traditional search engine interface where query suggestions can be shown to the user for them to select, but not as suited for the conversational technologies that we are targeting.

Research conducted by Cao et al. [5], Shen et al. [30] and Sun et al [34] focused on how context can be leveraged to improve document ranking accuracy. Grbovic et al. [15] proposed a context-aware query rewriting approach for sponsored search. He et al. [17] proposed using sequence to sequence for query rewriting. These research overlap with only individual aspects of our work, such as leveraging context or sequence to sequence modeling, and they are focused on the traditional query rewriting task of adding alternative queries to improve ranking performance by reducing mismatches.

For deep learning, the architecture that we built on top of is the sequence to sequence model [7, 8, 35]. More details about sequence to sequence can be found in section 4.2. We also used the technique of adding attention mechanism to sequence to sequence, which has been shown in existing research to improve performance [1, 23, 36]. Another deep learning technique that we used is multiple perspective matching, which has been shown to help with natural language sentence matching [37]. For our task of CQU, this matching was applied between the query and the context; more details can be found in section 4.3.4. For all these deep learning techniques, previous works have not yet explored applying them to the task of CQU.

3 DATASET

To apply deep learning for open domain CQU, we need a dataset that is large (at least tens of thousands samples), open domain, and, of course, containing conversational queries. Several existing datasets were considered, but unfortunately none of them met all the criteria, as shown in Figure 1.

The movie dialogue [2] and MSR twitter [33] datasets were both large and not domain specific, but they contained few conversational queries, and it would have been very difficult to filter for

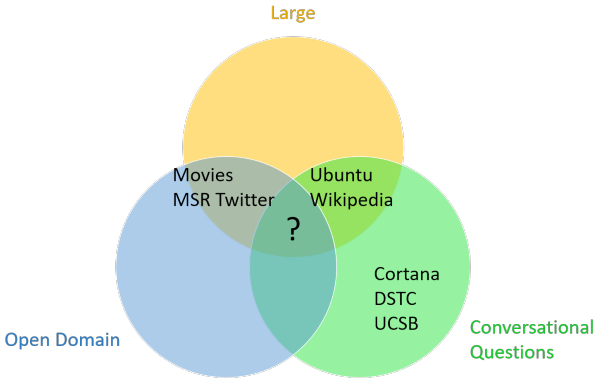

Figure 1: Existing datasets

those queries. The Ubuntu tech support [22] and Wikipedia editors [11] datasets were both large and contained good examples of conversational queries, but they were specific to Ubuntu tech support queries and Wikipedia editing queries, respectively. The Cortana (internal dataset), Dialog State Tracking Challenge [19], and UCSB datasets [14] were also domain specific, and not large enough.

Since there was no suitable existing dataset, data collection was the crucial first step. We hypothesized that there are users who interact with search engines as if they've having a conversation, and issue conversational queries, expecting the context of their previous queries to be maintained. From examining search engine logs, we did in fact find such queries, and what was particularly interesting is that because the search engine does not currently have great CQU, users themselves would often have to reformulate those queries to include the necessary context.

Therefore, we mined a commercial search engine's query logs for triplets of consecutive queries from the same user session, i.e. consisting of query 1, 2, and 3. The queries are already stored in sessions based on the search engine's definition of session. Then a filtering logic was applied to obtain the triplets where:

(1) query 1 can be any query
(2) query 2 is a conversational query that depends on context from query 1
(3) query 3 is the user's own reformulation of query 2 that includes context from query 1

For example: query 1 = "when was California founded", query 2 = "who is its governor", query 3 = "who is California's governor". Note that query 2 is conversational, and it was reformulated to query 3, which is non-conversational and can be a standalone query.

Some of the important criteria in the filtering logic include:

- query 2 doesn't result in any clicks (implies that the search engine did not understand the query and did not return any good results)
- query 3 does result in a satisfied user click (implies that the search engine did understand the query and did return a good result)
- query 3 consists of terms from query 2 and terms from query 1 that weren't in query 2 (implies that query 3 is a reformulation of query 2 to include context from query 1)
- query 3 was issued within 30 seconds of query 2 (implies that the user quickly noticed that the search engine did not understand query 2, and immediately reformulated it to query 3)

These criteria combined with the fact that the search engine currently has low coverage for conversational queries and shows unsatisfactory results, leads to the assumption that query 2 is a conversational query that depends on context from query 1.

Recall that the task of CQU involves reformulating the conversational query to include the correct context. Therefore, query 1 and query 2 (previous query and current query) can be treated as the inputs and query 3 (reformulation of current query) can be treated as the labels. This dataset currently does not include the previous answer, but we will explore adding it in the future.

These triplets are the positive samples, and we also mined for negative samples where no reformulation was needed. This was done by applying a filtering logic with the criteria that query 2 already resulted in a satisfied user click (implying that the search engine understood query 2 and returned a good result). In this case, no user reformulation was needed, so we can just set query 3 to be equal to query 2. For example: query 1 = "where is California", query 2 = "how to split string in Python", query 3 = "how to split string in Python". For these queries, we don't want to reformulate query 2, so the desired output query 3 is just the original query 2. These negative samples were added so that a model trained on this dataset can also learn whether or not a query depends on previous context.

For the mined search engine queries, only about 5% were natural language questions, with the rest being keyword queries. However, since users tend to issue more natural language questions for the conversational technologies that we are targeting, we upsampled the natural language questions to be 50% of our dataset. This upsampling was done for both positive and negative samples.

Note that the filtering logics for both positive and negative samples are not perfect and will result in some false positive/negative samples. This is not a surprise because if there is a perfect filtering logic for conversational queries, then the task of CQU can be solved by simply applying that logic. From manually checking 1000 samples, we found that 71% of positive samples actually contained conversational queries while 98% of negative samples actually contained standalone queries. Improving the filtering logic to generate cleaner data is something that we will continue to explore.

Even though the vast majority of search engine queries are not conversational, we were still able to create a large dataset because of the massive amount of queries in the search engine logs. The dataset that we used for our models consists of 3.6 million conversational query sessions, and also 3.6 million negative samples. This dataset was mined from only a subset of search engine logs, so additional samples can easily be obtained; and as new queries continue to come in daily, this dataset can continue to grow.

Another advantage of this dataset is that it includes human labels for free, without the need to hire crowdsourced judges to generate labels. Also, search engine data is very diverse and covers many domains, and this dataset reflects that. It contains a very wide range of context that are passed between queries, everything from various named entities to concepts/noun phrases to verbs. Therefore, this

dataset satisfies the criteria mentioned earlier, making it the first ever large scale and open domain dataset of conversational queries.

4 ALGORITHMS

4.1 Problem Formulation

As stated in the introduction, we formulate the problem as a context-aware query reformulation task. In a generic form, the inputs of the task are: context (conversation history), which includes previous queries and answers/results, and current input query; output of the task is a generated query which reformulates the input query by infusing information which exists in the context but is missing from the current query.

We use C to represent conversation history: $C = \{Q_t, A_t\}_{t=-1}^{-K}$ where K represents the window size of looking back at history, use Q_0 to represent current input query, and use Q' to represent the output after reformulation. The goal of the task can be formulated to find a function F:

$$F(C, Q_0) \rightarrow Q'$$

In addition, both query and answer are comprised of sequence of words:

$$Q = \{w_t^Q\}_{t=1}^M, A = \{w_t^A\}_{t=1}^N$$

We use w_t to represent a word at position or time step t in a sequence.

In the simplest form where we only use the previous query in a conversation history as context, C becomes: $C = Q_{-1}$. For simplicity, we remove the subscripts from C and Q_0 and represent them as:

$$C = \{w_t^C\}_{t=1}^N, Q = \{w_t^Q\}_{t=1}^M$$

The goal remains as:

$$F(\{w_t^C\}_{t=1}^N, \{w_t^Q\}_{t=1}^M) \rightarrow \{w_t^{Q'}\}_{t=1}^P$$

Words generated in Q' : $w_t^{Q'}$ can either be from the context: $\{w_t^C\}_{t=1}^N$ or current input query: $\{w_t^Q\}_{t=1}^M$, or a predefined vocabulary.

This problem setting fits very well as a sequence to sequence problem. We first briefly review the general sequence to sequence approach, and then introduce our approaches of applying the sequence to sequence technique to solve the context-aware query reformulation problem.

4.2 Sequence to Sequence Modeling

In the general sequence to sequence scenario, a collection of source-target sequence pairs is given, and the task is to learn to generate target sequences from source sequences. Let's use S and T to represent a source sequence and a target sequence, respectively:

$$S = \{w_t^S\}_{t=1}^M, T = \{w_t^T\}_{t=1}^N$$

Words in S and T can be from different vocabularies, like in machine translation, or the same vocabulary, like in text summarization. Sequences S and T can have different lengths and they represent a many-to-many relationship.

Sequence to sequence belongs to the broader class of encoder-decoder models [7], which has two stages: encoding and decoding. In the encoding stage, an encoder is used to transform the source sequence into an encoded representation. There are many different types of encoders targeting different source domains [9, 18]. Here we are agnostic to the form of the encoder and simply use a general recurrent neural network (RNN) to present the encoding process as:

$$\boldsymbol{u}_t^S = RNN^S(\boldsymbol{u}_{t-1}^S, \boldsymbol{e}_t^S) \tag{1}$$

Here $\boldsymbol{e}_t^S$ is the word embedding representation of word w_t^S in source sequence $S = \{\boldsymbol{e}_t^S\}_{t=1}^M$ and $\boldsymbol{u}_t^S$ represents the internal RNN state at time step t. After running the RNN through the entire source sequence, we obtain $\boldsymbol{u}^S = \{\boldsymbol{u}_t^S\}_{t=1}^M$, which is considered as the encoded representation of the source sequence. Instead of using the entire sequence $\boldsymbol{u}^S$ as the encoded representation of the source, usually the last RNN state, $\boldsymbol{u}_M^S$, is treated as the representation of the entire source sequence and is used for decoding.

Once the source sequence is encoded, sequence to sequence models generate a target sequence in the decoding stage. In this stage, a decoder, which is usually another RNN, generates the target sequence sequentially one word at a time by conditioning on the source sequence and the previously generated words.

$$\boldsymbol{s}_t = RNN^T(\boldsymbol{s}_{t-1}, h(\boldsymbol{y}_{t-1}, S)) \tag{2}$$

$$p(y_t|\{y_{<t}\}, S) = g(\boldsymbol{s}_t) \tag{3}$$

Here $\boldsymbol{s}_t$ represents the internal state of RNN at time t and y_t stands for the word generated at time t. We use bold font $\boldsymbol{y}_t$ to represent y_t's corresponding word embedding representation. g is usually an affine layer followed by a softmax. Usually dependence on S can be captured by setting $\boldsymbol{s}_0$ to be $\boldsymbol{u}_M^S$, which passes the source information to the target. Since the source information is already passed, we can set $h(\boldsymbol{y}_{t-1}, S) = \boldsymbol{y}_{t-1}$.

On top of above sequence to sequence framework, a technique called attention has been shown to significantly improve sequence to sequence models' performances and has become a default component in many sequence to sequence applications [1, 23, 36].

Instead of using a fixed vector, e.g. $\boldsymbol{u}^S$, to represent the source sequence S during decoding, attention mechanism introduces a dynamically changing attention vector $\boldsymbol{c}_t$ to the decoding process.

$$\boldsymbol{c}_t = \sum_{k=1}^{M} \alpha_{t,k} \boldsymbol{u}_k^S \tag{4}$$

$$\alpha_{t,k} = \frac{e^{f(\boldsymbol{s}_t, \boldsymbol{u}_k^S)}}{\sum_{k'} e^{f(\boldsymbol{s}_t, \boldsymbol{u}_{k'}^S)}} \tag{5}$$

Equations 4 and 5 give the computation of $\boldsymbol{c}_t$. Intuitively, $\alpha_{t,k}$ represents the strength of attention on the k^{th} word in source sequence at time step t during decoding. f is the attention function which is usually a multi-layer neural network with non-linear layers. In this paper, we use the Bahdanau mechanism introduced in [1]. $\boldsymbol{c}_t$ is computed by a weighted sum of the source words' representations based on their corresponding attention strengths. With the attention mechanism, the decoding process then becomes:

$$\boldsymbol{s}_t = RNN^T(\boldsymbol{s}_{t-1}, h(\boldsymbol{y}_{t-1}, \boldsymbol{c}_{t-1})) \tag{6}$$

In this paper, we simply consider concatenating $\boldsymbol{y}_{t-1}$ and $\boldsymbol{c}_{t-1}$: $h(\boldsymbol{y}_{t-1}, \boldsymbol{c}_{t-1}) = [\boldsymbol{y}_{t-1}, \boldsymbol{c}_{t-1}]$. Equation 3 then becomes:

$$p(y_t|\{y_{<t}\}, S) = g(\boldsymbol{s}_t, \boldsymbol{c}_t) \tag{7}$$

Here an attention layer is applied above the RNN cells, and $g(s_t, c_t)$ is set to be $g([s_t, c_t])$. Figure 2 illustrates the general sequence to sequence model with attention, where LSTM [18] is selected the as RNN cell and the <START> token is used to kick off decoding process.

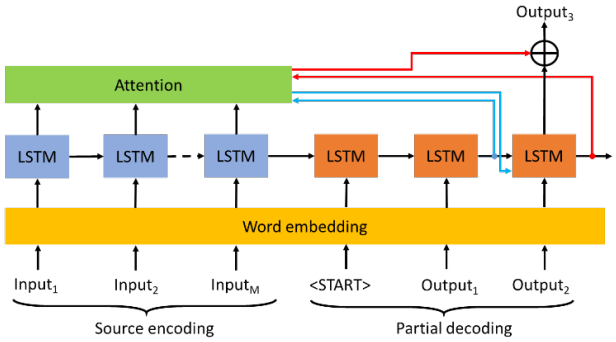

Figure 2: General sequence to sequence with attention.

4.3 Context-Aware Query Reformulation with Sequence to Sequence Modeling

This section will provide details on how we tackled the context-aware query reformulation problem with various sequence to sequence approaches. We proposed four approaches, with each one building on top of the previous.

4.3.1 Concatenated Sequence to Sequence (Concate_S2S, Baseline). The unique property of our query reformulation problem, which doesn't exist in general sequence to sequence settings, is that there are two source sequences: 1) context $C = \{w_t^C\}_{t=1}^N$ and 2) current query $Q = \{w_t^Q\}_{t=1}^M$. Our first approach is just to concatenate C and Q to form one source sequence, and then directly adopt general sequence to sequence:

$$S = [\{w_t^C\}_{t=1}^N, \_SEP, \{w_t^Q\}_{t=1}^M]$$

$\_SEP$ represents a special word used to be able to separate context and query sequences. This approach is considered as our baseline, and we derived more advanced approaches on top of it.

4.3.2 Pair Sequences to Sequence (Pair_S2S). Instead of concatenating context and query sequences, we keep them separate and use different RNNs to encode them:

$$\boldsymbol{u}_t^C = RNN^C(\boldsymbol{u}_{t-1}^C, \boldsymbol{e}_t^C)$$

$$\boldsymbol{u}_t^Q = RNN^Q(\boldsymbol{u}_{t-1}^Q, \boldsymbol{e}_t^Q)$$

$\boldsymbol{u}_0^Q$ is set to be $\boldsymbol{u}_N^C$ to pass information from context to current query, simulating natural conversation flow. With this encoding process, we obtain the encoded context representation $\boldsymbol{u}^C = \{\boldsymbol{u}_t^C\}_{t=1}^N$, and the encoded query representation $\boldsymbol{u}^Q = \{\boldsymbol{u}_t^Q\}_{t=1}^M$.

In decoding stage, $\boldsymbol{u}_M^Q$ is used to initialize RNN state $\boldsymbol{s}_0$. While for attention, we expand the traditional attention mechanism to a two-layer attention. First, attention is conducted on context and query sequences separately and independently:

$$\boldsymbol{c}_t^C = \sum_{k=1}^{N} \alpha_{t,k}^C \boldsymbol{u}_k^C \qquad \boldsymbol{c}_t^Q = \sum_{k=1}^{M} \alpha_{t,k}^Q \boldsymbol{u}_k^Q$$

$$\alpha_{t,k}^C = \frac{e^{f(\boldsymbol{s}_t, \boldsymbol{u}_k^C)}}{\sum_{k'} e^{f(\boldsymbol{s}_t, \boldsymbol{u}_{k'}^C)}} \qquad \alpha_{t,k}^Q = \frac{e^{f(\boldsymbol{s}_t, \boldsymbol{u}_k^Q)}}{\sum_{k'} e^{f(\boldsymbol{s}_t, \boldsymbol{u}_{k'}^Q)}}$$

Second, another attention is conducted to merge attention vectors $\boldsymbol{c}_t^C$ and $\boldsymbol{c}_t^Q$:

$$\boldsymbol{c}_t^{C+Q} = a\_\alpha_{t,k}^C \boldsymbol{c}_t^C + a\_\alpha_{t,k}^Q \boldsymbol{c}_t^Q$$

$$a\_\alpha_{t,k}^C = \frac{e^{f(\boldsymbol{s}_t, \boldsymbol{c}_t^C)}}{e^{f(\boldsymbol{s}_t, \boldsymbol{c}_t^C)} + e^{f(\boldsymbol{s}_t, \boldsymbol{c}_t^Q)}}$$

$$a\_\alpha_{t,k}^Q = \frac{e^{f(\boldsymbol{s}_t, \boldsymbol{c}_t^Q)}}{e^{f(\boldsymbol{s}_t, \boldsymbol{c}_t^C)} + e^{f(\boldsymbol{s}_t, \boldsymbol{c}_t^Q)}}$$

where $a\_\alpha_{t,k}^C$ and $a\_\alpha_{t,k}^Q$ can be considered as the attention strength at the sequence level on context and query, respectively. $\boldsymbol{c}_t^{C+Q}$, the weighted-sum vector of $\boldsymbol{c}_t^C$ and $\boldsymbol{c}_t^Q$, will be used as the final attention vector for decoding. The rest of the decoding is the same as general sequence to sequence using Equations 6 and 7.

Figure 3 illustrates the pair sequences to sequence approach with two layers of attentions. Figure 4 zooms in to the computation of attention vectors.

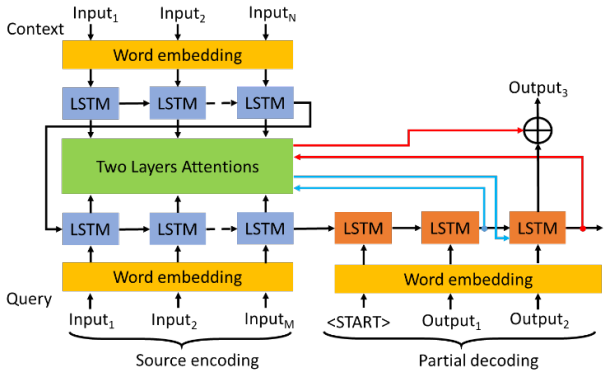

Figure 3: Pair sequences to sequence with two layers of attentions.

Keeping context and query separate when encoding makes the overall model structure more flexible. It provides better support for incorporating richer context in the future, such as including multiple previous turns from the conversation history with both queries and answers. Furthermore, it empowers the model to more efficiently and deeply capture the relationship between the context and query, instead of just concatenating them and treating them equally.

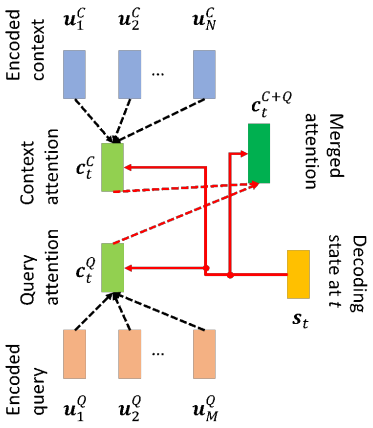

Figure 4: Computation of two layers of attentions.

4.3.3 Pair Sequences to Sequence With Context Embedding (Pair_S2S_Cxt_Embed). Based on Pair_S2S, we developed a new model which embeds context information into query sequence during encoding, with the purpose of capturing the semantic relationship between context and query.

In encoding stage, the context sequence is first encoded as done in model Pair_S2S. Then, the query sequence is encoded. When encoding the query, attention mechanism over the context is applied.

$$\boldsymbol{c}_t^{QC} = \sum_{k=1}^{N} \alpha_{t,k}^{QC} \boldsymbol{u}_k^C \tag{8}$$

$$\alpha_{t,k}^{QC} = \frac{e^{f(\boldsymbol{u}_t^Q, \boldsymbol{u}_k^C)}}{\sum_{k'} e^{f(\boldsymbol{u}_t^Q, \boldsymbol{u}_{k'}^C)}} \tag{9}$$

In Equations 8 and 9, $\alpha_{t,k}^{QC}$ represents the attention strength on the k^{th} word in the context sequence at time t while encoding the query. $\boldsymbol{c}_t^{QC}$ is the corresponding weighted-sum vector over the encoded context representation. Then the query encoder becomes:

$$\boldsymbol{u}_t^Q = RNN^Q(\boldsymbol{u}_{t-1}^Q, [\boldsymbol{e}_t^Q, \boldsymbol{c}_{t-1}^{QC}])$$

The decoding stage remains the same as in Pair_S2S.

You can see the unique property of this approach is the computation of $\boldsymbol{c}_t^{QC}$ and its embedding usage. The attention mechanism enables $\boldsymbol{c}_t^{QC}$ to capture matching information between each word in the query sequence to all the words in the context sequence. This additional information was expected to produce better source representations to be used for decoding. The effectiveness of this approach can be seen in our experiment results.

4.3.4 Pair Sequences to Sequence With Context Embedding From Multiple Perspective Matching (Pair_S2S_Cxt_Embed_MP). The effectiveness of Pair_S2S_Cxt_Embed inspired us to develop this new model, with the motivation of further improving the context embedding $\boldsymbol{c}_t^{QC}$.

[37] proposed a multiple perspective matching (MP-matching) approach to measure similarity between two natural language sentences. We borrowed that idea to compute a new context embedding $\boldsymbol{c}_t^{QC}$. Let's rephrase MP-matching for our reformulation scenario and describe how we're using it.

In MP-matching, a multiple perspective matching function f_m is proposed to compute similarity of two vectors:

$$\boldsymbol{m} = f_m(\boldsymbol{v}_1, \boldsymbol{v}_2; \boldsymbol{W})$$

where $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are two same sized vectors, e.g. with dimension d, and $\boldsymbol{W} \in \mathbb{R}^{l \times d}$ is a trainable parameter with l representing the number of perspectives. The returned value of $\boldsymbol{m}$ is a l-dimensional vector $\boldsymbol{m} = [m_1, ..., m_l]$, with each dimension $m_k \in \boldsymbol{m}$ representing the matching score from the k^{th} perspective. m_k is calculated by the following formula:

$$m_k = cosine(\boldsymbol{W}_k \odot \boldsymbol{v}_1, \boldsymbol{W}_k \odot \boldsymbol{v}_2)$$

where $\odot$ is the element-wise multiplication, and $\boldsymbol{W}_k$ is the k^{th} row of $\boldsymbol{W}$.

A few matching strategies based on f_m are proposed in [37]. In the following equations, we use *BiRNN* to represent a bi-directional RNN. Before matching the query against the context, we first encode them to new representations:

$$\overrightarrow{\boldsymbol{u}_t^C}, \overleftarrow{\boldsymbol{u}_t^C} = BiRNN(\boldsymbol{u}_{t-1}^C, \boldsymbol{e}_t^C)$$

$$\overrightarrow{\boldsymbol{u}_t^Q}, \overleftarrow{\boldsymbol{u}_t^Q} = BiRNN(\boldsymbol{u}_{t-1}^Q, \boldsymbol{e}_t^Q)$$

Note that context and query are encoded using the same RNN as proposed in [37].

Matching strategies:

1. Full matching. In this strategy, each time step of the query representation $\overrightarrow{\boldsymbol{u}_t^Q}$ (or $\overleftarrow{\boldsymbol{u}_t^Q}$) is compared with the final time step of the context representation $\overrightarrow{\boldsymbol{u}_N^C}$ (or $\overleftarrow{\boldsymbol{u}_N^C}$):

$$\overrightarrow{\boldsymbol{m}_t^{full}} = f_m(\overrightarrow{\boldsymbol{u}_t^Q}, \overrightarrow{\boldsymbol{u}_N^C}; \overrightarrow{\boldsymbol{W}^{full}})$$

2. Max pooling matching. In this strategy, each time step of the query representation $\overrightarrow{\boldsymbol{u}_t^Q}$ (or $\overleftarrow{\boldsymbol{u}_t^Q}$) is compared with every time step of the context representation $\overrightarrow{\boldsymbol{u}_i^C}$ (or $\overleftarrow{\boldsymbol{u}_i^C}$), and the maximum value of each dimension is selected:

$$\overrightarrow{\boldsymbol{m}_t^{max}} = \max_{i \in 1,...,N} f_m(\overrightarrow{\boldsymbol{u}_t^Q}, \overrightarrow{\boldsymbol{u}_i^C}; \overrightarrow{\boldsymbol{W}^{max}})$$

3. Attentive matching. In this strategy, first, at each time step of the query representation $\overrightarrow{\boldsymbol{u}_t^Q}$ (or $\overleftarrow{\boldsymbol{u}_t^Q}$), attentions over the context representation are computed. Attention weight is computed with cosine similarity:

$$\overrightarrow{a_{t,i}} = \frac{cosine(\overrightarrow{\boldsymbol{u}_t^Q}, \overrightarrow{\boldsymbol{u}_i^C})}{\sum_{j=1}^{N} cosine(\overrightarrow{\boldsymbol{u}_t^Q}, \overrightarrow{\boldsymbol{u}_j^C})} \quad i = 1, ..., N \tag{10}$$

Then an attention vector over the entire context representation $\overrightarrow{\boldsymbol{u}^C}$ (or $\overleftarrow{\boldsymbol{u}^C}$) is computed by weighted summing all time steps of the context representation:

$$\overrightarrow{\boldsymbol{u}_t^{C,mean}} = \sum_{i=1}^{N} \overrightarrow{a_{t,i}} \cdot \overrightarrow{\boldsymbol{u}_i^C}$$

Finally, each time step of the query representation is matched with its corresponding attention vector by the f_m function:

$$\overrightarrow{\boldsymbol{m}_t^{att}} = f_m(\overrightarrow{\boldsymbol{u}_t^Q}, \overrightarrow{\boldsymbol{u}_t^{C,mean}}; \overrightarrow{\boldsymbol{W}^{att}})$$

4. Max attentive matching. This strategy is similar to attentive matching. It picks the time step from the context representation which has the highest attention score (cosine similarity computed by Equation 10) as the attention vector, instead of taking the weighted sum of all the time steps as the attention vector. We use $\overrightarrow{\boldsymbol{m}_t^{max\_att}}$ to represent the max attentive matching vectors.

All these match strategies are applicable to the query's and context's word embedding representations as well by simply replacing $\boldsymbol{u}^Q$ and $\boldsymbol{u}^C$ with $\boldsymbol{e}^Q$ and $\boldsymbol{e}^C$ respectively and removing the direction. In our model, we expand max attentive matching with word embedding representations and represent it as: $\boldsymbol{m}_t^{e,max\_att}$. [37] has demonstrated that applying all strategies together works best. Therefore, we aggregate all the above matching vectors with an aggregation layer. The matchings at each time step in the query sequence are concatenated:

$$\boldsymbol{m}_t^{QC} = [\overrightarrow{\boldsymbol{m}_t^{full}}, \overleftarrow{\boldsymbol{m}_t^{full}}, \overrightarrow{\boldsymbol{m}_t^{max}}, \overleftarrow{\boldsymbol{m}_t^{max}}, \overrightarrow{\boldsymbol{m}_t^{att}}, \overleftarrow{\boldsymbol{m}_t^{att}}, \overrightarrow{\boldsymbol{m}_t^{max\_att}}, \overleftarrow{\boldsymbol{m}_t^{max\_att}}, \boldsymbol{m}_t^{e,max\_att}]$$

Then $\{\boldsymbol{m}_t^{QC}\}_{t=1}^M$ is fed into another RNN:

$$\overrightarrow{\boldsymbol{v}_t^{QC}}, \overleftarrow{\boldsymbol{v}_t^{QC}} = BiRNN^{Agg}(\boldsymbol{v}_{t-1}^{QC}, \boldsymbol{m}_t^{QC})$$

Finally, the context embedding is obtained as:

$$\boldsymbol{c}_t^{QC} = [\overrightarrow{\boldsymbol{v}_t^{QC}}, \overleftarrow{\boldsymbol{v}_t^{QC}}]$$

Figure 5 illustrates the Pair_S2S_Cxt_Embed_MP model.

5 EXPERIMENTS

For all the proposed models, the loss function that was optimized during training is the cross entropy loss, which is given by:

$$CE = -\sum_i \log(p_{y_i})$$

where p_{y_i} is the model's predicted probability of the correct target word y_i. Often the cross entropy loss is expressed as perplexity, which is just the exponential of cross entropy loss, exp(CE). Stochastic gradient descent was the chosen optimizer because of its popularity for other sequence to sequence models [35].

The final metrics that the models were evaluated on are exact match and BLEU score. Exact match is the percentage of predicted sequences that match word for word with the labels. BLEU score is the standard BLEU score [25]. BLEU score is included as a metric because the ultimate goal of this task is to allow IR/QnA systems to answer conversational query and an exact match is not always needed to correctly answer the query. For example, "what is the population of California" and "what is California population" should both result in the same answer for most QnA systems. Exact match is a very strict metric but it provides a precise measurement of our approaches. During evaluation, we removed stop-words from both target and generated sequences.

Table 2: Test set evaluation for different baseline (Concate_S2S) model setups.

	Conversational		Non-conversational	
Model	**EM**	**BLEU**	**EM**	**BLEU**
GRU, no att	20.5	52.1	40.7	54.7
LSTM, no att	23.8	53.1	40.6	54.1
LSTM, att	44.8	76.7	75.3	87.7
LSTM, att, bidir	43.7	75.6	73.6	86.5

The dataset described in Section 3 was split into train/dev/test sets by a 80/10/10 split. Train set was used for training, dev set was used for hyperparameter tuning, and test set was used for final evaluations. The test set was also split into the positive (conversational) and negative (non-conversational) examples.

A vocabulary size of 200k was chosen because 600k unique words were found in the inputs of the train set and the top 200k words covered 99% of all words. Batch size of 32 was chosen based on recommendation from [4]. Word embedding size of 300 was chosen since we hypothesized that the model would need as much expressivity at the word level as possible. The encoder and decoder share the same embeddings because for this task, the meaning of a word should be the same in both the input and output. The embeddings were trained from random initialization and using pretrained GloVe embeddings [26] did not help. Hidden size of 300 was chosen based on recommendation to have hidden size ≥ embedding size [4]. Larger hidden sizes of 400 and 600 were experimented with but did not result in improvements. Further tuning of these hyperparameters would be a worthwhile next step.

During training, a checkpoint was saved every 1/20th of an epoch, and the checkpoint is evaluated on the entire dev set. Training stopped when the perplexity on the dev set did not improve over 5 consecutive checkpoints.

We first studied the impact of different model architectures for the Concate_S2S model, including type of RNN cell, with or without attention, yes or no bidirectional encoder.

Table 2 shows the results of different settings of Concate_S2S model on the test set, which had about 700k samples total. LSTM provided an improvement over GRU and adding attention provided the biggest improvement, while using bidirectional encoders did not help.

Based on these results, subsequent experiments with the other models used single direction LSTM with attention.

6 RESULTS

The final results of the various models on the test set are shown in Table 3.

As mentioned earlier, Concate_S2S model is considered as our baseline due to its direct application of general sequence to sequence.

Pair_S2S model is better than the baseline in terms of EM on the conversational set but is worse in terms of EM on the non-conversational set. This might be because Pair_S2S tends to be more aggressive at reformulating the query. This is actually a good sign because reformulation on the conversational set is the more difficult

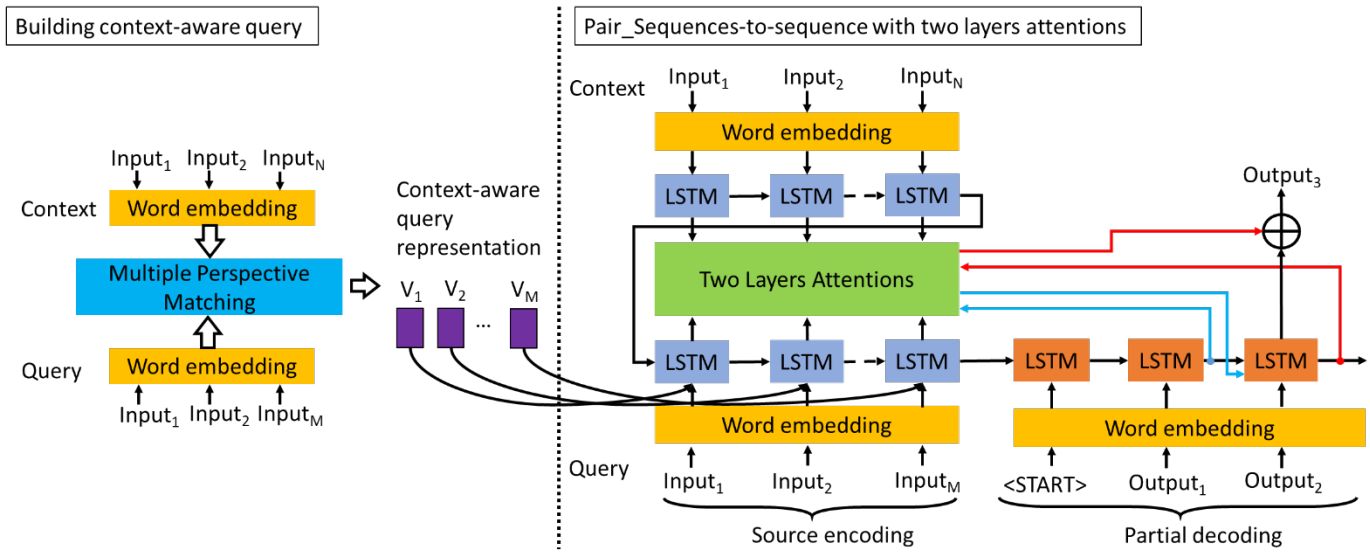

Figure 5: Sequences to sequence with context embedding from multiple perspective matching. There are two stages: 1) building context-aware query 2) feeding the context-aware query representation to pair sequences to sequence with two layers attentions.

Table 3: Test set evaluation for different models.

	Conv		Non-conv	
Model	**EM**	**BLEU**	**EM**	**BLEU**
Concate_S2S	44.8	76.7	75.3	87.7
Pair_S2S	48.5	75.6	74.5	82.9
Pair_S2S_Cxt_Embed	50.2	76.6	74.9	83.3
Pair_S2S_Cxt_Embed_MP	55.0	79.2	80.0	86.8
Pair_S2S_Cxt_Embed_MP + vocab truncate	**55.7**	**82.6**	**84.0**	**92.5**

task with much lower EM scores. Pair_S2S_Cxt_Embed model is better than Pair_S2S on all numbers. It indicates that embedding the context while encoding the current query can bring valuable information which is beneficial for reformulating the current query. Results of model Pair_S2S_Cxt_Embed_MP further verify that an advanced context embedding approach improves both EM and BLEU on both conversational and non-conversational sets.

For model Pair_S2S_Cxt_Embed_MP, the number of perspectives was set to be 50, and single layer bi-directional LSTM was used following the settings in [37]. The hidden size of LSTM is 200 and the final context embedding dimension is 400. To accommodate the 400 dimensions, the hidden size of LSTM in the encoder and decoder was set to be 700 (400 + 300). We also tried 300 hidden size, which performed slightly worse but still better than the other models.

We optimized model Pair_S2S_Cxt_Embed_MP further with the goal of improving latency during online prediction. The performance bottleneck for sequence to sequence model is at decoding. The last layer at each time step projects the LSTM output to a vocabulary-sized vector (200K here) and then runs a softmax on top of that, which involves a huge matrix multiplication and element summation. To mitigate the burden here, during prediction, we truncate the vocabulary to just the words that appear in the inputs (context + current query), plus some pre-selected stopwords (to maintain the language model information learned from training). This optimization improves latency by 10 times. Furthermore, accuracy is also improved, as you can see in the results for Pair_S2S_Cxt_Embed_MP + vocabulary truncation in Table 3.

This might be because users do not often use new words to reformulate their conversational queries, they mostly use words that they already used in the context. This might also expose a limitation of our current dataset: the lack of paraphrases in target outputs, due to the constrain that query 3 has to contain terms from query 2 and query 1, as described in Section 3. To obtain more paraphrase-style target reformulations, human labeling is required, which will be part of our future work.

6.1 Examples and Analysis

Here is a sample query session that was inputted to the best sequence to sequence model. The arrow indicates the output of the model. No arrow means that the model outputted the original query without any reformulation. The model and this paper is not concerned with answering the queries, so the answers were simply obtained by issuing these queries to a commercial search engine.

Q: When was California founded?
A: September 9, 1850
Q: Who is its governor? → Who is California governor?
A: Jerry Brown
Q: Where is Stanford?
A: Palo Alto, California
Q: Who founded it? → Who founded Stanford?
A: Leland and Jane Stanford

Q: Tuition costs → Tuition costs Stanford
A: $47,940 USD
Q: How to split string in Python?
A: split()
Q: How to read file? → How to read file in Python?
A: open()
Q: Kobe Bryant height
A: 6'6"
Q: His birth date → Kobe Bryant birth date
A: August 23, 1978
Q: What are the similarities between bacteria and viruses?
A: The similarities are...
Q: What are the differences? → What are the differences between bacteria and viruses?
A: The differences are...

This sample session shows that the model is able to perform reformulations that involve coreference resolution and also when there aren't any anaphoras. The model is able to pass different types of context, from "Kobe Bryant" to "between bacteria and viruses". The model is also able to determine when the topic changes and the query no longer depends on previous context, e.g. when the topic switched from Stanford to Python, the context of Stanford is no longer passed. Natural language questions and keyword queries are both handled.

To illustrate how the Pair_S2S_Cxt_Embed_MP + vocab truncate model improves on the baseline Concate_S2S model, Table 4 shows a few examples where this final model produced the correct output while the baseline was incorrect. For the first example, the baseline model added the wrong term from the context. For the second and third examples, it did not append any context. We suspect this is because both Q2s could be valid standalone queries, only by paying careful attention to Q1 can it be determined that context needs to be added.

In summary, the baseline model generated predictions that were not conditioned as strongly on Q1, a problem that was mitigated in the final model because of its context-aware query and pair sequences to sequence architecture, which provide greater dependence on the context, Q1, at both the encoding and decoding steps.

Table 4: Example predictions from different models.

	Concate_S2S	Pair_S2S_Cxt_Embed_MP + vocab truncate
Q1 = common math letter for scalar, Q2 = what is the integral		
Prediction	what is the integral in common	what is the integral in math
Q1 = who is andy from toy story, Q2 = what happened to sid		
Prediction	what happened to sid	what happened to sid on toy story
Q1 = what major studies multiculturalism in college Q2 = how to ask for a reference		
Prediction	how to ask for a reference	how to ask for a reference for college

However, this final model still has several weaknesses. It does well with common entities and concepts, but struggles with rare entities and concepts. For example, the model does not reformulate "His birth date" if the previous query was about Sasha Vujacic, a less popular basketball player. The model also still sometimes struggle with copying over the entire original query when it is necessary, especially for long queries. Some kind of pointer mechanism might help here [16]. Another weakness of the model is that it can't recognize some specific patterns of conversational queries, such as "How tall is Kobe Bryant? What about Lebron James?". This is likely because even among the millions of samples in the dataset, there is not enough samples of such patterns for the model to learn from.

7 CONCLUSION

In conclusion, using the first ever large scale and general dataset of conversational queries, a few sequence to sequence models were proposed, with the best one successfully reformulating more than half of all conversational queries across a very wide range of topics. There is still a ton of progress to be made, due to the many challenges of CQU presented earlier in this paper.

Immediate future work includes mining more data from the search engine logs to increase the size of the dataset, and retraining and further tuning the existing models. Incorporating the pointer mechanism might help as described above. Also based on the errors, augmenting the dataset with entity types, POS tags, and other NLP properties might help with generalization to uncommon entities. Another approach would be to artificially generating new samples by swapping entities for other entities of the same type, which has been shown to improve performance on other tasks [29]. We will also continue to explore new model architectures. For example, models proposed in Sordoni et al. [32] and Dehghani et al. [12] might be worth trying because they also modeled consecutive queries in a session using sequence to sequence, although they focused on the query suggestion scenario.

Despite the tremendous potential for further improvements on this task, the results in this paper already demonstrate a significant first step towards CQU, which can be incorporated into and improve any of the aforementioned conversational technologies, and have shown that deep learning is a promising approach for this task.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their helpful comments. We would aso like to thank our colleagues from Microsoft Research: Wei Wu, Can Xu and Ming Zhou for their constructive suggestion and feedbacks during the development of this work, particularly the vocabulary truncation idea for latency optimization; Wenhan Wang and Yuxiong He for their effort on optimizing inference latency.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).
[2] Rafael E Banchs. 2012. Movie-DiC: a movie dialogue corpus for research and development. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*. Association for Computational Linguistics, 203–207.
[3] Ziv Bar-Yossef and Naama Kraus. 2011. Context-sensitive query auto-completion. In *Proceedings of the 20th international conference on World wide web*. ACM, 107–116.

[4] Yoshua Bengio. 2012. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*. Springer, 437–478.

[5] Huanhuan Cao, Daxin Jiang, Jian Pei, Enhong Chen, and Hang Li. 2009. Towards context-aware search by learning a very large variable length hidden markov model from search logs. In *Proceedings of the 18th international conference on World wide web*. ACM, 191–200.

[6] Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. 2008. Context-aware query suggestion by mining click-through and session data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 875–883.

[7] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* (2014).

[8] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[9] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[10] Kevin Clark and Christopher D Manning. 2016. Deep reinforcement learning for mention-ranking coreference models. *arXiv preprint arXiv:1609.08667* (2016).

[11] Cristian Danescu-Niculescu-Mizil, Lillian Lee, Bo Pang, and Jon Kleinberg. 2012. Echoes of power: Language effects and power differences in social interaction. In *Proceedings of the 21st international conference on World Wide Web*. ACM, 699–708.

[12] Mostafa Dehghani, Sascha Rothe, Enrique Alfonseca, and Pascal Fleury. 2017. Learning to attend, copy, and generate for session-based query suggestion. *arXiv preprint arXiv:1708.03418* (2017).

[13] Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 484–495.

[14] John W Du Bois, Wallace L Chafe, Charles Meyer, Sandra A Thompson, and Nii Martey. 2000. Santa Barbara Corpus of Spoken American English. *CD-ROM. Philadelphia: Linguistic Data Consortium* (2000).

[15] Mihajlo Grbovic, Nemanja Djuric, Vladan Radosavljevic, Fabrizio Silvestri, and Narayan Bhamidipati. 2015. Context-and content-aware embeddings for query rewriting in sponsored search. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 383–392.

[16] Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *arXiv preprint arXiv:1603.08148* (2016).

[17] Yunlong He, Jiliang Tang, Hua Ouyang, Changsung Kang, Dawei Yin, and Yi Chang. 2016. Learning to Rewrite Queries. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 1443–1452.

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

[19] Seokhwan Kim, Luis Fernando DâĂŹHaro, Rafael E Banchs, Jason D Williams, and Matthew Henderson. 2017. The fourth dialog state tracking challenge. In *Dialogues with Social Robots*. Springer, 435–449.

[20] Liangda Li, Hongbo Deng, Jianhui Chen, and Yi Chang. 2017. Learning parametric models for context-aware query auto-completion via hawkes processes. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 131–139.

[21] Liangda Li, Hongbo Deng, Anlei Dong, Yi Chang, Ricardo Baeza-Yates, and Hongyuan Zha. 2017. Exploring Query Auto-Completion and Click Logs for Contextual-Aware Web Search and Query Suggestion. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 539–548.

[22] Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909* (2015).

[23] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* (2015).

[24] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit.. In *ACL (System Demonstrations)*. 55–60.

[25] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 311–318.

[26] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[27] Aaditya Prakash, Sadid A Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2016. Neural Paraphrase Generation with Stacked Residual LSTM Networks. *arXiv preprint arXiv:1610.03098* (2016).

[28] Gil Press. 2017. AI By The Numbers: 33 Facts And Forecasts About Chatbots And Voice Assistants. (2017). Retrieved October 23, 2017 from https://www.forbes.com/sites/gilpress/2017/05/15/ai-by-the-numbers-33-facts-and-forecasts-about-chatbots-and-voice-assistants

[29] Jonathan Raiman and John Miller. 2017. Globally normalized reader. *arXiv preprint arXiv:1709.02828* (2017).

[30] Xuehua Shen, Bin Tan, and ChengXiang Zhai. 2005. Context-sensitive information retrieval using implicit feedback. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 43–50.

[31] Satinder P Singh, Michael J Kearns, Diane J Litman, and Marilyn A Walker. 2000. Reinforcement learning for spoken dialogue systems. In *Advances in Neural Information Processing Systems*. 956–962.

[32] Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 553–562.

[33] Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714* (2015).

[34] Aixin Sun and Chii-Hian Lou. 2014. Towards context-aware search with right click. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 847–850.

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

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

[37] Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814* (2017).