Summarization Baselines on SParC Dataset

Shreya Dixit

May 2, 2019

Abstract

The Dialog to SQL and Summarization tasks are both hot-beds of NLP research, due to their difficulty and their vast real-life implications. In this paper, we discuss the dialog summarization task on the SParC dataset, which marks an intersection of the Dialog to SQL and text summarization problems. This task entails using the sub-questions from the SParC dataset as input and predicting the target question.

The SParC dataset was created to fill in the lack of public, complex, vast datasets for the Dialog to SQL task. A year ago, the LILY lab produced Spider, the first dataset to map natural language to multi-table databases of multiple domains and offer complex SQL translations, using "GROUP BY", "ORDER BY", and "JOIN" (Yu et. al., 2018). For the Dialog to SQL task, earlier this year, the LILY lab created the SParC dataset, which re-annotates the tables used in the Spider dataset. The new dataset not only retains the novel qualities of Spider, but also introduces thematic relations that are common in dialog. The smaller input size, variable input structure and thematic relations make it prime for approaching the Dialog to Summarization task, as it requires logic preservation to achieve high results for this dataset.

The Dialog to Summarization task is inherently difficult due to the challenge of creating a concise, factually correct, relevant and comprehensive summary. The goal of the LILY lab is to create a model that goes beyond basic pattern recognition and preserves the logic of its input. For this task, I adapted two baseline models: a pointer-generator model in the 2017 See et. Al paper and the word-graph with keyword extraction model in the 2013 Boudin et. Al Multi-Sentence Compression paper (See et. Al, 2017; Boudin et. Al, 2017). The error analysis conducted on these baseline models helps define the strengths and weaknesses of each approach, which can be used to guide the eventual construction of the summarization model used to satisfy this task. The results indicate the model construction will likely lean towards word-graph extraction, due to the smaller size input and little necessity for generation of words from a predefined vocabulary.

1 Introduction

This paper will 1) explain the contribution and features of SParC dataset; 2) describe the the goals of summarization on SParC task; and 3) delve into the baseline models I adapted for this task. To begin, the Lily Lab has dedicated resources to construct a Dialog System that can help facilitate natural language interactions with databases. Within this goal, first, I, in conjunction with 14 other college students under the direction of Tao Yu, annotated a new cross-domain, large scale dataset for conversational interactions, now called SParC. This dataset builds upon previous datasets WikiSQL and Spider, by capturing contextual dependencies between questions in the same example and a diversity of semantic content (Zhong et. Al, 2017; Yu et. Al, 2018).

As a next step, I also worked with Alex Fabbri, and Tao Yu on the summarization task on the SParC dataset with the final goal of summarizing the sub-questions in the dialog. This task entails creating a model that preserves the logical correctness. To do so, this project must explore a variety of baseline models for multi-sentence summarization, to find their strengths and weaknesses of each approach. The results can then be used to guide the construction of a superior model.

My final project helps create two baseline models. The first adapts the OpenNMT Pointer Generator Model from the 2017 See et. Al paper (See et. al, 2017). The second model, Takahe, is a multi-sentence comprehension model adapted from "Keyphrase Extraction for N-best Rerank-
ing in Multi-Sentence Compression” by Florian Boudin and Emmanuel Morin (Boudin and Morin, 2013). On both these models, I conducted error analysis to assess their ability to preserve the logic of the input.

2 Problem Description

Broadly, the goal of the Dialog to Summarization task on the SParC dataset is to create a model that preserves logical correctness. The SParC dataset, which is described in Section 4 below, consists of 4,298 target questions that have been each decomposed into a series of sub-questions (Yu et al., 2018). Please look at Figure 3 in that section below for context. The purpose of this task is to take those sub-questions as an input and compose the final question.

The difficulty of the general dialog summarization task stems from the fact that effective summaries must be concise, comprehensive, correct, and relevant. We can imagine a model that simply returns the concatenation of the sub-questions, but it would not satisfy the “concise” element of an effective summary. Moreover, the model must decipher what is relevant, and present the material in a concise manner. For summarization models that generate output from predefined vocabularies, the summaries must only present material that is correct. Reproduction of incorrect facts and repetitive behavior is an issue for many existing abstractive models (See et al., 2017).

The Summarization on SParC dataset problem encompasses all these general Dialog Summarization challenges described above, and additional ones due to the inherent qualities of the SParC dataset. The SParC dataset’s input crosses multiple domains and have variable lengths, which forces the summarization model to be flexible. Through thematic relations and diversity of domains, described in section 4, the SParC dataset forces models to follow logical and contextual dependencies in order to attain better results. Finally, the SParC dataset, composed of questions, naturally does not follow a singular structural pattern. To explain, news datasets, commonly used for dialog summarization, tend to follow a consistent lead sentence structure, in which much of the important information is contained in the first sentence. The lack of a consistent information presentation structure reinforces the requirement for the SParC summarization model to preserve logical dependencies.

3 Approach

3.1 OpenNMT Pointer-Generator Model with Coverage

For my senior project, I adapted the pointer-generator with coverage model used in “Get to the Point: Summarization with Pointer-Generator Networks” by See et al. This model uses an abstractive approach and develops the traditional encoder-decoder model with three main additions: 1) pointing, which allows the model to draw from the input words; 2) generating, which allows the model to generate words from the vocabulary outside of the input; and 3) coverage, which discourages the model repeatedly choosing the same words (See et al., 2017).

The model iterates on previous iterations of the traditional encoder-decoder model. First, it implements attention, which can be viewed as an additional step that calculates the magnitude of “attention” the decoder should pay to each source word. The model calculates the attention distribution or the probability distribution over the source words. This additional step addresses an issue of information loss with the traditional encoder decoder model due to the fact that only a single vector is outputted from the encoder to the decoder (See et al., 2017).

As Figure 1 shows, at each time step of the decoder, the model calculates the attention distribution, a function
of the previously decoded word’s embedding, the decoder state \( s_t \), and the encoder sequence of hidden states. The attention distribution is used to calculate a context vector, a fixed size vector representation of what has been read (See et. Al, 2017).

Next, the context vector and the decoder state are used to calculate \( P_{gen} \) and \( P_{vocab} \), which introduces the pointer-generator element. As described above, pointing allows for the model to account for rare words or proper names that are not included in the predefined vocabulary, while generation allows for the output of words found from the vocabulary. To implement this, the context vector and decoder state, and decoded input of the previous word are used to calculate \( P_{gen} \), where \( P_{gen} \) is the probability the decoded word must be generated from the predefined vocabulary. Meanwhile, the context vector and decoder state are used to calculate the probability distribution \( P_{vocab} \), which describes the probability of each word in the vocab being the decoded word. For the final distribution, \( P_{gen} \) is used to decide whether the decoded word should come from \( P_{vocab} \) (the probability distribution of the vocab), or \( a_t \), the attention distribution (the probability distribution of the input sequence) (See et. Al, 2017). The final distribution is given the equation below.

\[
P(w) = P_{gen}P_{vocab}(w) + (1 - P_{gen}) \sum_{i: w_i = w} d_i
\]

This process describes the diagram above.

The last addition of this model to the traditional encoder-decoder model is coverage. The model described, thus far, is liable to create a repetition of outputted decoded words at each time step. Thus, the model also includes a coverage mechanism, which is a vector that sums the attention distributions of all previous time steps. This coverage vector is used as an additional input to the attention mechanism to penalized attending to the same words (See et. Al, 2017).

### 3.2 Adaptation of OpenNMT Pointer Generator Model

The adaptation of this model simply included a transformation of the input from JSON files to the correct format. Moreover, we adjusted the minimum number of words to 5, which was lower than what was expected for original model trained on the CNN-daily news model.

### 3.3 Takahe: Multi-Sentence Compression with Keyphrase-based Re-ranking Method

For robustness, I also adapted an extractive approach: Florian Boudin and Emmanuel Morin’s Multi-Sentence Compression (MSC) Model with a Keyphrase-based Re-ranking Method. This model is an extractive summarization model, meaning it only summarizes using words from the source. Under the assumption that redundancy within this source sequence must be enough to identify important links between words for MSC, this model only requires as input, the input sequence and the POS tuples for each token in the sequence, in order to create predictions (Boudin and Morin, 2013).

![Figure 2: Example Word Graph adapted from (Filippova, 2010)](image)

This model builds upon Filippova’s 2010 graph-based multi-sentence comprehensive approach. As shown in Figure 2, the original approach takes the input sequence and creates a word graph, which is a directed graph containing all the words in the input sequence as nodes and edges representing adjacent words. The word graph also includes start and end nodes (Filippova, 2010).

The model parses through each sentence. It adds candidate nodes, if the token does not exist previously i.e. if a token representing the same word with the same part of speech does not exist. Edges are added to represent adjacent words and punctuation marks are not accounted for. Adjacent words are used if the mapping is unclear. The next step assigns weights to each edge. Filippova’s model’s weighting algorithm optimizes for word association and salient words. For word association, its weight system uses edge 1) frequency, rewarding higher fre-
frequency, and 2) distance between words, rewarding shorter distance. Meanwhile, the weighting algorithm uses node frequency, to encourage path with salient words (Filippova, 2010).

Finally, using K-shortest paths algorithm, the model finds 50 shortest paths using a weighting function. Of those, it only considers paths that have a verb node or of a predefined minimum length. After normalizing the total paths, it outputs the sequence on the path with the lightest average edge weight (Filippova, 2010).

Boudin and Morin iterate upon Filippova’s model by introducing a better re-ranking system than the normalization of the total paths in the last steps. This adjustment is to combat a near 48-60% failure to retain important information (Boudin and Morin, 2013). The motivation for keyphrase-based reranking method proposes to emphasize node saliency beyond using word frequency. This model uses the unsupervised method in which it creates a word-graph that connects two nodes if they occur in the same sentence. The word graph increments edge weights every two words coccur. The edge weights are then used to calculate a salience score. A keyword phrase score is then calculated by optimizing upon the sum of salience scores and the keyword-phrase length to emphasize longer word sequences. Boudin and Morin adjust the original MSC model such that the K-shortest path algorithm finds 200 shortest paths, where each path is re-ranked using normalization multiplied by the sum of keyword phrase scores in the path (Boudin and Morin, 2013).

3.4 Adaptation of Multi-Sentence Compression Model

To adapt this model, I used a Python Spacy token tagging algorithm to transform the input from the JSON files. To avoid the tagging of spaces, I preprocessed the input by reducing the number of spaces to 1. The script then iterated through each input example and outputted the highest ranked predicted sentence.

4 Data Sets used

The SParC Dataset is a cross-domain, large-scale dataset that encompasses many context dependencies. Prior to the SParC dataset, the most commonly used datasets for natural language to SQL include WikiSQL, and Spider (Zhong et al., 2017; Yu et al., 2018). The main additions of the Spider Dataset include a significant improvement in complexity by using databases with multiple tables and extending the SQL logic commands to include JOIN, GROUP BY, and ORDER BY (Yu et al., 2018). The SParC dataset iterates on the work of the Spider dataset to create a dataset for Sequential Question Answering, by reannotating the the same tables with sub-questions that capture assumed user behavior. The construction of this dataset stems from the hypothesis that users ask a series of questions to arrive at a target question. So, this database includes questions that may not be related to the target question (Yu et al., 2019).

The dataset also includes thematic relations to simulate user use of relative pronouns. These relations are summarized by the examples in Table 1. All these thematic relations describe a set of questions in which the subsequent question refers to a noun in a previous question without explicitly referring to it.

For clarity, below, we show an example of the SParC dataset input in Figure 3.

![Figure 3: Input Example from SParC Dataset](image-url)

For the purpose of summarization, we take the sub-questions as input and attempt to output the original question. The dataset includes 4,298 unique sequences (12,000+ annotated sub-questions), 200 complex datasets in 138 different domains (Yu et al., 2019).

5 Evaluation Method

For the OpenNMT model, I trained the model for 30,000 iterations but chose the model trained on 15,000 iterations due to signs of overfitting. For this decision...
<table>
<thead>
<tr>
<th>Thematic relation</th>
<th>Description</th>
<th>Example</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Refinement (con-</td>
<td></td>
<td>The current question asks for the same type</td>
<td><strong>Prev</strong>: Which major has the fewest students?&lt;br&gt;<strong>Cur</strong>: What is the most popular one?</td>
</tr>
<tr>
<td>straint refinement)</td>
<td></td>
<td>of entity as previous questions with a dif-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ferent constraint.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Theme-entity (top-</td>
<td></td>
<td>The current question asks for other proper-</td>
<td><strong>Prev</strong>: What is the capacity of Anonymous Donor&lt;br&gt;<strong>Cur</strong>: List all of the amenities it has.</td>
</tr>
<tr>
<td>ic exploration)</td>
<td>ties about the same entity as some previous</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>questions.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Theme-property (par-</td>
<td></td>
<td>The current question asks for the same</td>
<td><strong>Prev</strong>: Tell me the rating of the episode named&lt;br&gt;<strong>Cur</strong>: How about for &quot;Keepers&quot;?</td>
</tr>
<tr>
<td>ticipant shift)</td>
<td>property about another entity.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Theme/refinement-</td>
<td>If current question asks about (a subset of)</td>
<td><strong>Prev</strong>: Please list all the different department names.&lt;br&gt;<strong>Cur</strong>: What is the average salary of all instructors in the Statistics department?</td>
<td>8.1%</td>
</tr>
<tr>
<td>answer</td>
<td>the entity in the answer of previous question.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Thematic Relations between Questions in the SParC dataset on a manually classified 102 examples. Adapted from (Yu et. Al, 2019)

I used accuracy = 100 * number of correct / number of words as a measure of a stopping point. (Note, that the model itself was trained using log softmax). On the other hand, the Takahe dataset did not require any training with the word-graph method. I then conducted error analysis to classify the success of the model in preserving logic of the input.

After getting the predictions from the model with the best accuracy, we compared the predictions to the actual output. We looked at 100 samples and categorized them by SQL errors, to analyze the results. In other words, if both the actual and predicted questions were converted into SQL, the part of the SQL query in which they differed is the error they incurred.

Figure 4 shows examples of the different error classifications I adopted. Many of the predictions incurred more than one error. For example, the "select table error," describes a difference in the table name selected if both the actual and predicted were translated to SQL. The example shown in the table is read such that the actual sequence was "What are all the names of all sculptures in Gallary 226?” and the predicted sequence was "Find the names of the ships that are from the gallary activity.” Here if we were to translate both sequences to SQL, the model was able to output the right column "names,” but it was unable to output the right table, wrongly predicting "sculptures” instead of "ships.” Similarly, the same logic applies for the other errors. The category logical errors encompasses predictions that were either unable to output the correct SQL logic (i.e. if the actual asked for "how many” and

<table>
<thead>
<tr>
<th>Error Classification</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Table Error</td>
<td>What are the names of all sculptures in&lt;br&gt;gallery 226? Find the names of the ships that are from&lt;br&gt;in gallery activity.</td>
</tr>
<tr>
<td>Select Column Error</td>
<td>What is the average height and width of&lt;br&gt;paintings that are oil medium in gallery&lt;br&gt;241? What is the average height and&lt;br&gt;resolution of paintings whose oil ACT-&lt;br&gt;211?</td>
</tr>
<tr>
<td>Logical Error</td>
<td>What are the names of the movie&lt;br&gt;theaters that are playing 'G' or 'PG' rated&lt;br&gt;movies? Which list me a a a a which do</td>
</tr>
<tr>
<td>Group By Error</td>
<td>How many movies exist for each rating?&lt;br&gt;How many movies does each department</td>
</tr>
<tr>
<td>Filter Error</td>
<td>What are the first and last names of&lt;br&gt;artists who have painted using both oil&lt;br&gt;and lithographic mediums? What are the names and last names of&lt;br&gt;artists with medium race. at a&lt;br&gt;lithographic customers?</td>
</tr>
<tr>
<td>Order by Error</td>
<td>List the names of all distinct paintings&lt;br&gt;from shortest to longest in height. Give all the names of perpetrators</td>
</tr>
<tr>
<td>Select Table Error/ Filter Error</td>
<td>What are the full names of students living in MD?&lt;br&gt;What are the full names of all faculty&lt;br&gt;instructors who in football?</td>
</tr>
</tbody>
</table>

Figure 4: The left column describes the name of the error. The right column shows examples of the actual followed by the predicted that incurred this error.
the predicted output began the question “which”) OR if the predicted output just did not make sense.

Finally, for the error analysis of the Takahe dataset, I considered an additional error classification: relative pronoun, which takes into account how many times an error is caused by the adoption of a relative pronoun instead of the actual noun. This is to add information about the ability of the model to parse through the thematic relations.

6 Results

6.1 Open-NMT Pointer Generator Model Error Analysis

<table>
<thead>
<tr>
<th>Error</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>logic error</td>
<td>33</td>
</tr>
<tr>
<td>select table error</td>
<td>16</td>
</tr>
<tr>
<td>select column error</td>
<td>14</td>
</tr>
<tr>
<td>filter error</td>
<td>38</td>
</tr>
<tr>
<td>group by error</td>
<td>4</td>
</tr>
<tr>
<td>order by error</td>
<td>2</td>
</tr>
<tr>
<td>select column error</td>
<td>14</td>
</tr>
<tr>
<td>table+filter</td>
<td>15</td>
</tr>
<tr>
<td>Normal</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 5: OpenNMT Model Error Analysis Results

Figure 5 shows the results of the OpenNMT error analysis, conducted on 100 test samples. Of that, the table shows that 1/3 of the predicted output had logic errors. To reiterate, this means that the model was either unable to come up with the correct SQL logic, or outputted nonsensical output 1/3 of the time.

38 of the example outputs were marked as filter errors, which emphasizes issues with the model generation, as many of these errors were due to the model outputting unrelated filter outputs.

An important common trend was the table+filter error. Examples of these errors, as shown in the table in figure 4, preserved the correct structure of the sentences, but failed to generate the correct subject and objects on to which to filter. Thus this model, seems to do reasonably well in at least understanding what the intended type of question the output should be, but is unable to parse through the logic of finding the correct subject etc.

Finally, there were only two “normal” sentences that preserved the logic of the actual output. The two examples:

Example 1:
Actual: What are the names of every theater with at least one movie playing?
Predicted: Which theaters have at least 1 film listed?

Example 2:
Actual: What is the painting count of the artist with the longest life?
Predicted: How many paintings did does the oldest artist

The examples raise the discussion of the use of metrics of accuracy. The accuracy used to stop the iterations: number of correct words/ number of words would not rank this summary well, despite these summaries achieving the goal of being correct, concise, informative, and relevant. The most common alternative metrics of accuracy is Recall-Oriented Understudy for Gisting Evaluation (ROUGE). The two forms of ROUGE: recall and precision should be aimed to maximize. ROUGE(recall) is number of overlapping words/ total words in reference summary, while ROUGE(precision) is number of overlapping words/ total words in system summary. The precision ensures that the summary is not overly verbose, with many irrelelevant words. Forms of ROUGE include ROUGE-1, unigrams, and ROUGE-2, bigrams (“What is ROUGE and how it works for evaluation of summarization tasks?”, 2018). However, despite the apparent effectiveness of the ROUGE accuracy measure, the examples of the predicted output would not fair well on any ROUGE evaluation metric. The analysis of these two specific sentences raises doubt about the effectiveness of current automatic evaluation metrics, like ROUGE, for text summarization.

6.2 Takahe Model Error Analysis

I ran the Takahe model with both a minimum word count of 6 and a minimum word count of 5. On a minimum word count of 6, the model sometimes failed to produce any paths that were 6 words. So, I ran the model with a lower minimum count, however the resulting model failed to retain the same complexity of its results for the same
examples than the equivalent prediction by the previous model. For example,

**Actual**: What are the names of every movie that is not playing at the Odeon theater?

**Predicted**: nb 6: what are the codes of all the other movies playing at the odeon theater?

**Predicted**: nb 5: what are the corresponding names?

This shows one of many example where the higher minimum number model outputted a more relevant result to the actual. Thus, for the error analysis, I only considered the results for the model with minimum number 6, and only considered results for the alternate models for the examples that failed to return a result with nb: 6.

As shown in the Figure 6, logic errors are again most common. An examination of the logical errors shows that many result from difficulty predicting multiple columns, especially if they have been split in different input questions. For example,

**Actual**: What are the ids of paintings that are taller than 500 and shorter than 2000?

**Predicted**: of these which also are shorter than 2000?

The model also had trouble in outputting sequences that specify a question requiring a single output, like finding the maximum amount.

Third, the model also had many relative pronoun mistakes. Three examples are listed below:

**Actual**: How many movies are playing in theaters?

**Predicted**: how many are playing in theaters?

**Actual**: What are the movie names sorted by rating?

**Predicted**: order the movies by ratings

**Actual**: What is the painting count of the artist with the longest life?

**Predicted**: how many paintings did the oldest of them make?

As shown, many of the model’s answers were nearly correct, but often specified the relative pronoun instead of the column name or did not in specify the column name.

Overall, the model outputted one correct example:

**Actual**: Count the number of coutries.

**Predicted**: how many are the distinct countries?

The difference in sentence structure between the predicted and actual reiterates the discussion of accuracy in the section 6.1.

As a whole, the Takahe model outputs often relevant but not comprehensive summaries. Compared to the abstractive OpenNMT model, this model has substantially fewer filter errors. It escapes the error of generation that opens up a lot of room for error. While there was only one clearly correct example, many were very close to the logic desired. For many of the sequences that require such filters, however, it is often able to capture the filter part of the sequence but can not predict the desired column and table. The specificity of the column names and tables must be improved, and the logic of relative pronouns must be better parsed.

7 Conclusion

The two baseline models, I have looked at, both encouragingly have room for improvement. Analysis of the models show that each have different strengths and weaknesses in preserving logic. While the pointer-generator model was better at predicting logic commands, it had a large issue with generating false content. The Takahe model produces more relevant content, but faces many relative pronoun errors due to poor logical parsing of thematic relations. Given the small vocabulary size that caters to keyphrase extraction and an output that can be almost always constructed from the input, adapting an extractive word-graph based model like Takahe looks more promising for this model.

As for next steps, the LILY lab can look at other baseline models that have iterated on the extractive word-graph based model used in this paper, such as the model suggested by the 2018 Shang. et. al paper “Unsupervised Abstractive Meeting Summarization with Multi-Sentence Compression and Budgeted Submodular Maximization model.” Moreover, for each baseline model tested, error analysis can be extended to test the logic preservation
ability by checking whether the thematic relations for a test input sample are correctly parsed in the respective predicted output.

8 Acknowledgement

Thank you to Professor Dragomir Radev for giving me such a great learning experience and the opportunity to participate in the research of the Lily Lab. Thank you to Tao Yu for guiding me through annotation process. Thank you to Alex Fabbri and Tao Yu for meeting with me to implement the model and being wonderful, available, and patient in answering my questions.

9 Appendix-code and data

The code and dataset for this project can be found on Tangra.

References


[4] Tao Yu, Rui Zhang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In EMNLP.

