AutoQA: From Databases To QA Semantic Parsers With Only Synthetic Training Data

Silei Xu∗ Sina J. Semnani∗ Giovanni Campagna Monica S. Lam
Computer Science Department
Stanford University
Stanford, CA, USA
{silei,sinaj,gcampagn,lm}@cs.stanford.edu

Abstract

We propose AutoQA, a methodology and toolkit to generate semantic parsers that answer questions on databases, with no manual effort. Given a database schema and its data, AutoQA automatically generates a large set of high-quality questions for training that covers different database operations. It uses automatic paraphrasing combined with template-based parsing to find alternative expressions of an attribute in different parts of speech. It also uses a novel filtered auto-paraphraser to generate correct paraphrases of entire sentences.

We apply AutoQA to the Schema2QA dataset and obtain an average logical form accuracy of 62.9% when tested on natural questions, which is only 6.4% lower than a model trained with expert natural language annotations and paraphrase data collected from crowdworkers. To demonstrate the generality of AutoQA, we also apply it to the Overnight dataset. AutoQA achieves 69.8% answer accuracy, 16.4% higher than the state-of-the-art zero-shot models and only 5.2% lower than the same model trained with human data.

1 Introduction

Semantic parsing is the task of mapping natural language sentences to executable logical forms. It has received significant attention in question answering systems for structured data (Wang et al., 2015; Zhong et al., 2017; Yu et al., 2018b; Xu et al., 2020). However, training a semantic parser with good accuracy requires a large amount of annotated data, which is expensive to acquire. The complexity of logical forms means annotating the data has to be done by an expert. This adds to the cost and hinders extending question answering to new databases and domains.

∗ Equal contribution

To eliminate the need for annotating data with logical forms, SEMPRE (Wang et al., 2015) proposed the new methodology of first synthesizing questions on the database, then manually paraphrasing them. Recently, the Schema2QA toolkit (Xu et al., 2020) demonstrated that it is possible to achieve high accuracy on realistic user inputs using this methodology with a comprehensive set of generic, domain-independent question templates. However, this approach requires a significant manual effort for each domain: the developers must supply how each attribute can be referred to using different parts of speech, and crowdworkers are needed to paraphrase the queries.
Our objective is to eliminate the need for manual effort in building semantic parsers, while achieving comparable accuracy. We hypothesize that, for common domains, the knowledge of how each attribute would be referred to in natural language is implicitly presented in large text corpora and can be captured by general-purpose paraphrasing models. With that insight, we developed AutoQA, a toolkit that (1) automatically annotates the database attributes using paraphrasing models, (2) uses generic templates to synthesize a large set of complex queries, and (3) uses a novel filtered auto-paraphraser to further increase the variety of the synthesized data. The resulting dataset is then used to train a BERT-LSTM model (Xu et al., 2020). The architecture of AutoQA is shown in Fig. 1.

The contributions of this paper are:

• AutoQA, a toolkit that automatically creates a semantic parser that answers questions about a given database. As the parser is trained only with automatically generated data, its cost is significantly lower than current approaches.

• A novel algorithm for annotating database attributes with phrases in different parts of speech. The algorithm is based on automatic paraphrasing combined with template-based parsing (Section 4).

• A new automatic paraphrasing model, based on BART (Lewis et al., 2019), that can generate natural paraphrases of sentences, with a filter trained with synthetic data to ensure the preservation of the original meaning expressed in a formal language (Section 5).

• The methodology has been tested on the Overnight dataset (Wang et al., 2015) and Schema.org web data (Xu et al., 2020) (Section 6). On Overnight, AutoQA achieves an average of 55.6% logical form accuracy and 69.8% denotation (answer) accuracy without using the human paraphrases for training, which are 18.6% and 16.4% higher than the state-of-the-art zero-shot models, respectively. On Schema.org, AutoQA achieves an average logical form accuracy of 62.9%, within 6.4% of models trained with manual annotations and human paraphrases.1

2 Related Work

Bootstrapping Semantic Parsers. Neural semantic parsing for question answering is a well-known research topic (Pasupat and Liang, 2015; Wang et al., 2015; Dong and Lapata, 2016; Jia and Liang, 2016; Krishnamurthy et al., 2017; Zhong et al., 2017; Yu et al., 2018b). State of the art methods use a sequence-to-sequence architecture with attention and copying mechanism (Dong and Lapata, 2016; Jia and Liang, 2016) and rely on large datasets. Acquiring such datasets is expensive, and the work must be replicated in every new domain.

Prior work proposed bootstrapping semantic parsers using paraphrasing (Wang et al., 2015), where a dataset is synthesized using a grammar of natural language, and then paraphrased by crowd-workers to form the training set. Paraphrasing has been applied to datasets for SQL (Zhong et al., 2017), as well as multi-turn dialogue datasets (Shaw et al., 2018; Rastogi et al., 2019).

Our previous work with Genie (Campagna et al., 2019) proposed training with large amounts of synthesized and smaller amounts of paraphrased data. Later, we developed Schema2QA (Xu et al., 2020), a synthesis tool based on a general grammar of English. Schema2QA was found to be effective for the question answering task on the Web. Both works rely on manual paraphrases and hand-tuned annotations on each database attribute. Training with synthetic data has also been explored to complement existed dataset (Weir et al., 2020) and in the few-shot setting (Campagna et al., 2020; Moradshahi et al., 2020).

A different line of work proposed training with a large multi-domain dataset, and then using transfer learning to generalize to new datasets, in a completely zero-shot fashion (Herzig and Berant, 2018a; Chang et al., 2019). Yet, such scenario requires acquiring the multi-domain dataset in the first place, and there is a significant gap between the accuracy of training with and without in-domain data (Yu et al., 2018b). Our approach instead is able to synthesize data for the new domain, so the model is exposed to in-domain data while retaining the zero-shot property of no human-annotated data.

Pre-trained Models for Data Augmentation. Previous work showed that pre-trained models are very effective at generalizing natural language knowledge in a zero- and few-shot fashion (Radford et al., 2019; Brown et al., 2020). These models

---

1The data and code can be downloaded from https://oval.cs.stanford.edu/releases/
Table 1: Example questions in the restaurant domain with their ThingTalk representations.

3 Schema2QA Data Synthesis Pipeline

AutoQA is based on Schema2QA (Xu et al., 2020), the state-of-the-art pipeline to generate high-quality training data for database QA at a low cost. Schema2QA first synthesizes utterance and formal representation pairs with a template-based algorithm, and then paraphrases utterances via crowdsourcing. The semantic parser is trained with both synthetic and paraphrased data, and tested on crowdsourced, manually annotated real questions.

Instead of relying on crowdsworkers to paraphrase and create variety from the synthesized canonical questions, Schema2QA uses a comprehensive set of 800 domain-independent templates, along with a few manual annotations for each attribute in each domain, to synthesize high-quality data. About 2% of the synthesized data are manually paraphrased.

Our previous work (Xu et al., 2020) shows that a parser trained on such dataset achieves 70% accuracy on natural complex questions. Table 1 shows a few questions that Schema2QA can parse and their representation in ThingTalk, which is a query language designed to support translation from natural language.

Schema2QA answers long-tail questions well because its synthesized data have good coverage of possible questions asked, while showing great linguistic variety. It synthesizes questions using generic question templates, which have placeholders to be substituted with domain-specific annotations that match the expected part-of-speech (POS) type. Table 2 shows how annotations of the 6 POS categories for the “AlumniOf” attribute are used in the example templates to synthesize example utterances. In total, six POS categories are identified: active verb phrase, passive verb phrase, adjective phrase, prepositional phrase, and two noun phrases: is-a noun phrase which describes what the subject is, has-a noun phrase which describes what the subject has. There is a wide variety in annotations for an attribute, and often only a sub-
425

Table 2: Annotations for “alumniOf” attribute with example templates and utterances in six POS categories, where table and value denote the placeholders for table canonical annotations and values, respectively.

<table>
<thead>
<tr>
<th>POS</th>
<th>Annotation</th>
<th>Example template</th>
<th>Example utterance</th>
</tr>
</thead>
<tbody>
<tr>
<td>is-a noun</td>
<td>alumni of value</td>
<td>table that[which][who is]are [noun phrase] value</td>
<td>people who are alumni of Stanford</td>
</tr>
<tr>
<td>has-a noun</td>
<td>a value degree</td>
<td>table with (a/an/the) value [noun phrase]</td>
<td>people with a Stanford degree</td>
</tr>
<tr>
<td>active verb</td>
<td>graduated from</td>
<td>table that[which][who] [verb phrase] value</td>
<td>people who graduated from Stanford</td>
</tr>
<tr>
<td>passive verb</td>
<td>educated at value</td>
<td>table [passive verb phrase] value</td>
<td>people educated at Stanford</td>
</tr>
<tr>
<td>adjective</td>
<td>value</td>
<td>value table</td>
<td>Stanford people</td>
</tr>
<tr>
<td>prepositional</td>
<td>from value</td>
<td>table [prepositional phrase] value</td>
<td>people from Stanford</td>
</tr>
</tbody>
</table>

set of POS types is relevant to an attribute. It is thus challenging, often requiring multiple rounds of error analysis, to come up with these different annotations manually.

4 Automatic Annotation

Our AutoQA toolkit automatically provides unambiguous attribute annotations for all parts of speech, with the help of a neural paraphrasing model.

4.1 Canonical Annotation

AutoQA first derives a canonical annotation for each table and its attributes. Where necessary, it splits the attribute name into multiple words (e.g. “alumniOf” turns into “alumni of”). It then uses a POS tagger to identify the category of the canonical annotation.

The canonical annotation is used both for training and as the starting point to identify alternative phrases for each attribute, hence it must be meaningful and unambiguous. When applying AutoQA to an existing ontology, developers can override the table or attribute names if they are not meaningful or they are ambiguous.

4.2 POS-based Annotation Extraction

As shown in Table 2, an attribute can be described in various ways in different parts of speech. It is not enough to retrieve synonyms of the canonical annotation, as all synonyms will have the same POS. Some synonyms may also be inappropriate for the domain, if generated without context.

Our goal is to automatically derive all the other POS annotations given a canonical annotation. For example, the canonical annotation for the “alumniOf” attribute is “alumni of value” of POS “is-a-noun”, as shown in the first row of Table 2. We wish to derive other “is-a-noun” annotations, as well as those in other POS categories in the table.

Our solution is to synthesize questions using the templates for the POS of the canonical annotation, get paraphrases from a neural model, parse the paraphrases using the templates as grammar rules, and turn successful parses into annotations.

AutoQA first generates short example sentences for each attribute using its canonical annotation. We generate questions that ask for objects with a given value of the attribute, using the grammar templates for the POS of the canonical annotation for the attribute. We generate up to 10 sentences for each alternative in the grammar template, using a different value for each one.

Second, AutoQA obtains paraphrases for the generated sentences using a neural paraphraser based on the BART sequence-to-sequence model (Section 6.1). To get more diverse paraphrases, we run 3 rounds of paraphrasing, where in each round we paraphrase the output of the previous round. All the words are tagged with their POS. For example, with “people who are alumni of Stanford” as an input, we can get paraphrases such as “people with a Stanford degree”, as shown in the last column of Table 2.

Third, AutoQA parses the paraphrases using the templates (third column in Table 2) as grammar rules. A phrase is considered a successful parse only if the “table” and the “value” match exactly and the POS of all placeholders match that of the corresponding words. Correctly parsed phrases are then turned into annotations.

Note that we generate only sentences that map to selection operations, such as “show me people who are alumni of Stanford”. Selection questions include a sample value, “Stanford”, for the attribute, which is useful to provide a better context for the paraphraser. The paraphraser can generate phrases like “find people from Stanford”, which is trivial to parse correctly. In contrast, values are missing in projection questions, such as “what institution are the people alumni of”, which makes paraphrasing and subsequent parsing harder. While we only paraphrase selection questions, the annotations identi-
4.3 Resolving Conflicts

Neural paraphrasing is imperfect and can generate incorrect annotations. Our priority is to eliminate ambiguity: we do not worry as much about including nonsensical sentences in the training, as such sentences are unlikely to appear at test time. Consider a movie domain with both “director” and “creator” attributes. The paraphrasing model might generate the annotation “creator” for “director”. To avoid generating such conflicted annotations within the domain, we detect annotations that appear in two or more attributes of the same type in the database. If such an annotation shares the same stem as one attribute name, it is assigned uniquely to that attribute. Otherwise, it is dropped entirely. As we train with data that is synthesized compositionally, we would rather lose a bit of variety than risk introducing ambiguity.

5 Automatic Paraphrasing

Synthetic training data is good for providing coverage with a large number of perfectly annotated sentences, and to teach the neural semantic parser compositionality. However, grammar-based synthesis often results in clunky sentences and grammatical errors. In addition, even with 800 generic templates, the synthesized sentences still lack naturalness and variety. In particular, people often compress multiple concepts into simpler constructions (sublexical compositionality (Wang et al., 2015)), e.g. “books with at least 1 award” can be simplified to “award-winning books”.

Capturing these linguistic phenomena in the training data is not possible with a finite set of templates. This is why paraphrasing is critical when training semantic parsers. Here we describe how we approximate manual paraphrases with a neural paraphrasing model.

5.1 Noise in Neural Paraphrasing

Using automatically generated paraphrases for training is challenging. First, paraphrasing models output noisy sentences, partially due to the noise in the existing paraphrasing datasets. We cannot accept paraphrases that change the meaning of the original sentence, which is represented by the logical form annotation. This noise problem exists even in human paraphrasing; Wang et al. (2015) reports that 17% of the human paraphrases they collected changed the logical form. Second, there is an inherent diversity-noise trade-off when using automatic generation. The more diverse we want to make the outputs, the noisier the model’s output will be. Third, the auto-paraphraser is fed with synthetic sentences, which have a different distribution compared to the paraphrase training set.

We have empirically found the following ways in which noise is manifested:

- The output is ungrammatical or meaningless.
- The output changes in meaning to a different but valid logical form, or rare words like numbers and proper nouns are changed.
- The model is “distracted” by the input sentence due to limited world knowledge. “I’m looking for the book the dark forest”, is very different from “I’m looking for the book in the dark forest”.
- The model outputs sentence pairs that can be used interchangeably in general, but not in the specific application. For example, “restaurants close to my home” and “restaurants near me” have different target logical forms.
- Automatically-generated annotations are not reviewed by a human to ensure their correctness. An example is the word “grade” instead of “stars” in the hotels domain. Further paraphrasing these noisy sentences amplifies the noise.

5.2 Paraphrase Filtering

How do we produce semantically correct paraphrases and yet obtain enough variety to boost the accuracy of the parser? Our approach is to generate high variety, and then filter out noisy sentences. More specifically, we feed auto-paraphrased sentences to a parser trained on only synthetic sentences. We accept the sentences as correct paraphrases only if this parser outputs a logical form equal to the original logical form.

Correct paraphrases are then used to train another parser from scratch, which will have a higher accuracy on the natural validation and test sets. The first parser can correctly parse the examples.
present in the synthetic set, e.g. “I am looking for the movies which have Tom Hanks in their actors with the largest count of actors.”. It also generalizes to paraphrased sentences like “I’m looking for Tom Hanks movies with the most actors in them.”. Paraphrased sentences like this are added to the training set to generate a second parser. This second parser can generalize to an even more natural sentence like “What is the Tom Hanks movie with the biggest cast?” This iterative process, as shown in Fig. 1, can be repeated multiple times.

This idea is borrowed from self-training (McClosky et al., 2006; He et al., 2019), where a model is used to label additional unlabeled data. Self-training requires an initial good-enough model to label data with, and optionally a filtering mechanism that is more likely to remove incorrect labels than correct labels (Yarowsky, 1995). We use a parser trained on a synthetic dataset as our initial good-enough model. The following two observations are the intuition behind this decision:

1. Paraphrases of a synthetic dataset are still relatively similar to that set. Thus, a parser trained on synthetic data, which delivers near perfect accuracy for the synthetic data, has a very high accuracy on the paraphrased data as well.

2. Unlike classification tasks, the set of valid logical forms in semantic parsing is so large that outputting the right logical form by chance is very unlikely.

Note that this filtering scheme might throw away a portion of correct paraphrases as well, but filtering out noisy examples is more important. The second observation ensures that the number of false positives is low.

5.3 Coupling Auto-Annotator with Auto-Paraphraser

Since both auto-annotation and auto-paraphrasing use a neural paraphraser, here we contrast them and show how they complement each other.

Auto-annotation provides alternative expressions with different POS for a single attribute at a time. The input sentences are simpler, so paraphrases are more likely to be correct, and they are filtered if they cannot be parsed correctly with the grammar rules. This makes it easier to coax more diverse expressions on the attribute from the paraphraser without having to worry about noisy outputs.

Annotations extracted by the auto-annotator are amplified as the synthesizer uses them to compose many full sentences, which are used to train the first parser with sufficient accuracy for self-training.

The auto-paraphraser, on the other hand, is applied on all synthesized data. It not only produces more natural alternative phrases for complex sentences, but also generates domain-specific and value-specific terminology and constructs. These two tasks complement each other, as supported by the empirical results in Section 6.2.2.

6 Experiments

In this section, we evaluate the effectiveness of our methodology: can a semantic parser created with AutoQA approach the performance of human-written annotations and paraphrases? We evaluate on two different benchmark datasets: the Schema2QA dataset (Xu et al., 2020) and the Overnight dataset (Wang et al., 2015).

6.1 AutoQA Implementation

Paraphrasing Model. We formulate paraphrasing as a sequence-to-sequence problem and use the pre-trained BART large model (Lewis et al., 2019). BART is a Transformer (Vaswani et al., 2017) neural network trained on a large unlabeled corpus with a sentence reconstruction loss. We fine-tune it for 4 epochs on sentence pairs from P ARABANK 2 (Hu et al., 2019a), which is a paraphrase dataset constructed by back- translating the Czech portion of an English-Czech parallel corpus. We use a subset of 5 million sentence pairs with the highest dual conditional cross-entropy score (Junczys-Dowmunt, 2018), and use only one of the five paraphrases provided for each sentence. We experimented with larger subsets of the dataset and found no significant difference. We use token-level cross-entropy loss calculated using the gold paraphrase sentence. To ensure the output of the model is grammatical, during training, we use the back-translated Czech sentence as the input and the human-written English phrase as the output. Training is done with mini-batches of 1280 examples where each mini-batch consists of sentences with similar lengths.

We use nucleus sampling (Holtzman et al., 2019) with top-$p=0.9$ and generate 5 paraphrases per sentence in each round of paraphrasing. We use greedy

---

3 This reduces the number of pad tokens needed, and makes training faster.
decoding and 4 temperatures (Ficler and Goldberg, 2017) of 0.3, 0.5, 0.7 and 1.0 to generate these paraphrases. Note that the input dataset to each paraphrasing round is the output of the previous round, and we have one round for Schema2QA and three rounds for Overnight experiments.

Semantic Parsing Model. We adopt our previously proposed BERT-LSTM model (Xu et al., 2020) as the semantic parsing model. The model is a sequence-to-sequence neural network that uses a BERT pre-trained encoder (Devlin et al., 2019), coupled with an LSTM decoder (Hochreiter and Schmidhuber, 1997) with attention (Bahdanau et al., 2014). The model uses a pointer-generator decoder (See et al., 2017) to better generalize to entities not seen during training. The model was implemented using the Huggingface Transformers library (Wolf et al., 2019). We use the same hyper-parameters as Xu et al. (2020) for all experiments. The model has approximately 128M parameters.

6.2 Applying AutoQA to Schema2QA

We first apply AutoQA to the Schema2QA dataset, a semantic parsing dataset that targets the ThingTalk query language, and uses Schema.org as the database schema. Queries are performed against structured data crawled from websites in 6 domains: restaurants (using data from Yelp), people (from LinkedIn), hotels (from the Hyatt hotel chain), books (from Goodreads), movies (from IMDB), and music (from Last.fm).

The Schema2QA training data set was created using synthesis based on manual field annotations and human paraphrasing, while its evaluation data was crowdsourced by showing the list of attributes to workers and asking them for natural questions. The evaluation data contains complex questions referring up to 6 attributes, with comparisons and relational algebra operators: join, selection, projection, sort, and aggregates.

In our experiments, we use the Schema2QA validation and test sets, but not the training data. We synthesize our own training data using the same 800 templates, and replace the manual annotations with our auto-annotation and the manual paraphrases with auto-paraphrases.

For auto-annotation to work, the table and attribute names must be meaningful and unambiguous as discussed in Section 4. We found it necessary to override the original names in only three cases. In the restaurants domain, “starRating” is renamed to “michelinStar” to avoid ambiguity with “aggregateRating”. In the people domain, “addressLocality” is renamed to “homeLocation” to avoid confusion with “workLocation”. In the music domain, “musicRecording” is renamed to “song” to better match natural language.

When applying auto-paraphrasing, we preprocess the questions to replace entity placeholders (e.g. TIME_0) with an equivalent token in natural language (e.g. 2pm), then postprocess the outputs to restore them. This way, the neural network does not have to deal with these tokens which it has not seen during its pre-training.

As shown in Table 3, AutoQA generates about 45% more attribute annotations, and produces 60 times larger paraphrase sets, compared with the original Schema2QA training set. Although AutoQA’s training set is larger than Schema2QA’s, we note that in our experiments, adding more synthetic data to Schema2QA did not improve its accuracy any further. We compare the diversity of the two datasets using distinct-1 and distinct-2 metrics (Li et al., 2016) which measure the ratio of distinct unigrams and bigrams in the datasets. AutoQA’s training sets have about 35% higher distinct-1 and 60% higher distinct-2.
Table 4: Test accuracy of AutoQA on the Schema2QA dataset. For the hotel domain, Xu et al. (2020) only report transfer learning accuracy, so we rerun the training with manual annotations and human paraphrases to obtain the accuracy for hotel questions.

<table>
<thead>
<tr>
<th>Model</th>
<th>Restaurants</th>
<th>People</th>
<th>Movies</th>
<th>Books</th>
<th>Music</th>
<th>Hotels</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema2QA (Xu et al., 2020)</td>
<td>69.7</td>
<td>75.2</td>
<td>70.0</td>
<td>70.0</td>
<td>63.9</td>
<td>67.0</td>
<td>69.3</td>
</tr>
<tr>
<td>Schema2QA w/o manual annotation &amp; paraphrase</td>
<td>30.0</td>
<td>30.4</td>
<td>36.6</td>
<td>34.9</td>
<td>33.7</td>
<td>59.7</td>
<td>37.6</td>
</tr>
<tr>
<td>AutoQA</td>
<td>65.3</td>
<td>64.6</td>
<td>66.1</td>
<td>54.1</td>
<td>57.3</td>
<td>70.1</td>
<td>62.9</td>
</tr>
</tbody>
</table>

Table 5: Ablation study on Schema2QA development sets. Each “–” line removes only that feature from AutoQA.

<table>
<thead>
<tr>
<th></th>
<th>Restaurants</th>
<th>People</th>
<th>Movies</th>
<th>Books</th>
<th>Music</th>
<th>Hotels</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema2QA (Xu et al., 2020)</td>
<td>70.8</td>
<td>74.9</td>
<td>75.3</td>
<td>80.7</td>
<td>71.8</td>
<td>69.3</td>
<td>73.8</td>
</tr>
<tr>
<td>Schema2QA (w/o manual annotation &amp; paraphrase)</td>
<td>33.9</td>
<td>32.7</td>
<td>35.7</td>
<td>39.9</td>
<td>37.1</td>
<td>37.1</td>
<td>40.2</td>
</tr>
<tr>
<td>AutoQA</td>
<td>69.5</td>
<td>66.1</td>
<td>68.0</td>
<td>67.6</td>
<td>66.9</td>
<td>66.6</td>
<td>67.4</td>
</tr>
<tr>
<td>– Auto-annotation</td>
<td>43.2</td>
<td>50.1</td>
<td>51.4</td>
<td>59.6</td>
<td>49.7</td>
<td>67.3</td>
<td>53.5</td>
</tr>
<tr>
<td>– Auto-paraphrase</td>
<td>62.1</td>
<td>50.5</td>
<td>62.7</td>
<td>61.5</td>
<td>58.6</td>
<td>59.1</td>
<td>59.1</td>
</tr>
<tr>
<td>– Paraphrase filtering</td>
<td>50.4</td>
<td>48.0</td>
<td>55.0</td>
<td>44.1</td>
<td>53.5</td>
<td>44.7</td>
<td>49.3</td>
</tr>
</tbody>
</table>

6.2.1 Evaluation

Our evaluation metric is logical form accuracy: the logical form produced by our parser must exactly match the one in the test set. As shown in Table 4, AutoQA achieves an average accuracy of 62.9% in six domains, only 6.4% lower compared to the models trained with manual attribute annotations and human paraphrases. The difference is mainly because paraphraser fails to generate a few common phrases in some cases. For example, it fails derive “employee” or “employed by” from the canonical annotation “works for”, which is quite common in the evaluation set. Compared with the baseline models trained with data generated by Schema2QA but without manual annotation and human paraphrase, AutoQA improves the accuracy by 25.3%. This result is obtained on naturally sourced test data, as opposed to paraphrases. This shows that AutoQA is effective for bootstrapping question answering systems for new domains, without any manual effort in creating or collecting training data.

6.2.2 Ablation Study

We conduct an ablation study on the development set to evaluate how each part of our methodology contributes to the accuracy. We subtract different components from AutoQA, generate the training data, and run the experiment with the same hyperparameters. When paraphrase filtering is removed, we still use simple string matching to remove erroneous paraphrases where entities and numbers in the utterance do not match the logical form.

As shown in Table 5, AutoQA reaches an overall accuracy of 67.4%, 6.4% lower than models trained with human annotations and human paraphrases. AutoQA outperforms the baseline trained on synthetic data generated from the canonical annotation by 27.2%. This indicates that AutoQA is an efficient and cost-effective replacement for manual annotation and paraphrasing.

On average, applying only auto-paraphrase on synthetic data based on canonical annotations without auto-annotation achieves 53.5%, which is 13.9% lower than the full AutoQA. Applying only auto-annotation without auto-paraphrase obtains 59.1%, and is 8.3% lower than AutoQA. This shows that the two components of AutoQA complement each other to achieve the best performance.

If auto-paraphrase is used without filtering, not only does it not improve the accuracy, but also the average accuracy drops by 18%. This shows that without filtering, even a paraphraser with a large pre-trained neural model like BART cannot be used for semantic parsing due to noisy outputs.

6.3 Applying AutoQA to Overnight

To evaluate if the AutoQA methodology generalizes to different types of databases, logical forms, and templates, we apply AutoQA on the well-known Overnight benchmark. Overnight is a semantic parsing dataset with questions over a knowledge base with very few entities across 8 domains. The dataset was constructed using paraphrasing; both training and test sets are paraphrased from the same set of synthetic sentences.

We train the BERT-LSTM model on data synthesized from Overnight templates with both auto-annotation and auto-paraphrase. Auto-annotation
is limited to two parts of speech, since Overnight uses a very simple template set to synthesize training examples, with only placeholders for active verb phrase and noun phrase. We use the standard train/test split and following previous work, use 20% of the human paraphrases from the original training set for validation, so that validation and test sets are from the same distribution.

We evaluate both logical form accuracy and answer accuracy, which checks whether the answer retrieved from the knowledge base matches the gold answer. The model outputs a ranked list of logical forms for each input question using beam search with 25 beams, and chooses the first output that is syntactically valid. Other than this, all models and hyperparameters are the same as Section 6.

In Table 6, we compare our technique to other approaches that do not use in-domain human data. They are either synthetic-only (Marzoev et al., 2020) or use human data from other Overnight domains (Herzig and Berant, 2018b). For reference, we also include two of the best-performing models that use in-domain human data (Cao et al., 2019; Chen et al., 2018).

Whereas Schema2QA dataset has naturally sourced evaluation and test data, Overnight evaluates on human paraphrase data. Evaluating with paraphrase data is not as meaningful, and makes the benchmark easier for models trained with human paraphrase data. Compared to other zero-shot models trained with no in-domain data, AutoQA outperforms the state of the art by 18.6% and 16.4% on logical form accuracy and answer accuracy, respectively. This shows that by generating diverse and natural paraphrases in domain, AutoQA can reach comparable performance with models with human training data, and is much more accurate compared to other zero-shot approaches.

7 Discussion

In this work, we propose AutoQA, a methodology and a toolkit to automatically create a semantic parser given a database. We test AutoQA on two different datasets with different target logical forms and data synthesis templates. On both datasets, AutoQA achieves comparable accuracy to state-of-the-art QA systems trained with manual attribute annotation and human paraphrases.

AutoQA relies on a neural paraphraser trained with an out-of-domain dataset to generate training data. We suspect the methodology to be less effective for domains full of jargon. Even for common domains, AutoQA sometimes failed to generate some common phrases. Further improvement on neural paraphraser is needed to generate more diverse outputs. Future work is also needed to handle attributes containing long free-form text, as AutoQA currently only supports database operations without reading comprehension.

Acknowledgements

This work is supported in part by the National Science Foundation under Grant No. 1900638 and the
Alfred P. Sloan Foundation under Grant No. G-2020-13938.

References


A The Cost of AutoQA

The only form of cost in AutoQA’s methodology is compute cost. Here we mention more details with regards to that. To use AutoQA for a new domain, the following steps will have to be executed to generate the final training set. Numbers are for the Schema2QA dataset, and batch sizes are set to maximize GPU utilization.

- Automatic annotation: This step runs inference using the BART paraphraser model as introduced in Section 6.1, it takes less than 10 minutes for each domain.
- Template-based data synthesizer: This step synthesize data with annotation generated by auto-annotator. Depending on the domain, it takes between 3 to 5 hours on an AWS m5.4xlarge machine (16 vCPU and 64 GiB of memory).
- Training a parser with the synthetic dataset to use as filter: We train the BERT-LSTM model for 4000 iterations only, as we empirically observed that training more than that does not improve the quality of the filter. This takes less than half an hour on an AWS p3.2xlarge machine (16GB V100 GPU, 8vCPUs, 61 GiB of memory).
- Automatic paraphrasing and filtering: This step uses the fine-tuned BART large model, which has about 400M parameters, to generate 5 paraphrases per input, and then the BERT-LSTM parser, which has 128M parameters, to filter those paraphrases. Note that no training is done in this step. In our experiments, this step takes less than 4 GPU-hours.
- Training of the semantic parser: Similar to training the filter, but we train for 60000 iterations, and it takes less than 6 GPU-hours.
The approximate total per-domain cost of Schema2QA experiments using Amazon Web Services is $36.