Building a Semantic Parser Over a Very Long Period of Time: Parsing Queries About Song Data with SEMPRE

Arushi Himatsingka, Brina Seidel, Derek Yen, Marina Zavalina
NYU Center for Data Science
{ah3243, bs3743, dy1078, mz2476}@nyu.edu

Abstract

This paper presents a new semantic parser for natural language queries about a song data set. It is built on the SEMPRE (Wang et al., 2015) framework and trained using natural language paraphrases of logical queries. Our final parser correctly parses 63.1% of the test data, while our baseline model correctly parsed just 50.8%. We find that our model is very effective at parsing different paraphrases of types of queries it has seen before, but less effective at parsing new types of queries. From our experiments, we conclude that this type of semantic parser can be trained on remarkably few training examples so long as those training examples include a sufficiently diverse range of queries.

1 Introduction

In this paper, we apply semantic parsing techniques to a new knowledge domain: music.

Semantic parsers allow individuals to query a database using language that makes sense to them; it removes technological barriers between interested users and the data itself. For this reason, semantic parsing has long been a topic of inquiry. Basic programs that parse based on a strict grammar have existed for almost four decades.

Similar grammars form the basis of many modern semantic parsers, which layer a machine learning model and other innovations atop a domain-specific grammar. We implement one such framework, Semantic Parsing with Execution v2.4 (SEMPRE), to answer basic queries about a subset of the Million Song Dataset (Wang et al., 2015). In doing so, we sought to uncover the pros and cons of using a parser with a domain-specific grammar that is trained on natural language paraphrases of database queries.

In particular, our experiments explore the oft-cited advantage of such parsers: they learn very directly from examples and thus can be trained on minimal data. In doing so, we encounter a major disadvantage: such parsers have difficulty parsing types of queries that were not seen during training.

These issues are particularly relevant in light of recent neural network approaches to semantic parsing. While such models are able to answer a wider range of queries, it is difficult to understand why the model decides upon any given parse. Neural nets are not yet used in many applications because, unlike the model we use, they cannot easily be "taught" to parse new types of queries by simply adding training data (Labutov, 2018). Understanding the strengths and weaknesses of a paraphrase model with domain-specific grammar is becoming increasingly important as the alternatives become more viable.

Our final model has an accuracy score of 63.1%, meaning it correctly parses 63.1% of the test queries, and an oracle score of 75.8%, meaning that the correct parse was one of the top 20 candidates for 75.8% of the test data. For comparison, our baseline model has an accuracy score of 50.3% and an oracle score of 68.6%. We further conclude that this type of semantic parser can be trained on remarkably few training examples so long as those training examples include a sufficiently diverse range of queries.

2 Prior Literature

The SEMPRE Framework. Our parser relies heavily on the methods outlined in Wang et al. (2015) and the accompanying SEMPRE code. In this paper, Wang et. al. describe a method for applying SEMPRE to a new knowledge domain that yields a semantic parser overnight.

SEMPRE contains a generalizable grammar that represents many - though by no means all - relationships between arbitrary entities in a knowledge graph. The rules in this grammar use lambda dependency-based compositional semantics (Lambda DCS), as presented in Liang (2013). Though it is more limited than traditional lambda calculus, Lambda DCS is better suited for describing database queries because succinctly describes relationships on a knowledge graph.

The first step that Wang et al. (2015) suggest is seeding this general grammar with domain-specific entities and relationships. For example, to parse queries about song data, it is necessary to explicitly state that "sung by" is a potential relationship between song entities and artist entities. The parser can only answer questions about relationships that are fully described in the grammar.

This seeded grammar can be used to generate human-interpretable representations of Lambda DCS formulas that Wang et. al. refer to as "canonical utterances." For example, the formula

$$\text{count}(\mathbf{R}(\text{songs})).(\text{artist.TaylorSwift} \sqcap \text{year.2017})$$

would translate to the canonical utterance

number of songs whose artist is Taylor Swift and whose year is 2017

Such canonical utterances are awkwardly phrased and should not be considered natural language. However, they are useful because they are both easily interpretable and logically unambiguous.

By contrast, queries that qualify as natural language have a high degree of ambiguity and may not follow the rules specified in the grammar at all. For example, a natural language query such as "How many songs are sung by Taylor Swift from 2017?" would not have an immediately clear mapping to such a logical representation without a specific grammar. In theory, the phrase "Taylor Swift from 2017" could represent an artist and the relationship between song and year would be lost. Ultimately, it would not be feasible to write grammar rules which cover every such circumstance.

To allow the parser to handle such queries, Berant and Liang (2014) present a paraphrase model which constructs a set of candidate canonical utterances and produces probabilistic scores for each one. The model learns associations using word co-occurrence and syntactic elements as features to predict if a canonical utterance is the correct paraphrase of the natural language query.

To actually construct the set of candidate canonical utterances, the model uses the floating parser presented by Pasupat and Liang (2015). Unlike the commonly used CKY (Younger, 1967) algorithm, the floating parser is less anchored to lexicon; it allows to parse the query more freely by introducing floating cells. As a result, the parser can skip some words in the sentence and combine logical forms in different ways.

Other Methods of Semantic Parsing. However, all these methods of training a parser via paraphrases differ greatly from numerous newer neural network models. One example is the encoder-decoder model described by Dong and Lapata (2016). Contrary to our method, the authors use a neural network for semantic parsing. They encode natural language utterances into vectors and generate corresponding logical forms as sequences or trees using recurrent neural networks with LSTM units. Unlike our method, which is very domain-specific, this method does not rely on domain-or representation-specific features and does not have a grammar generating utterances.

Jia and Liang (2016) also train a sequence-to-sequence RNN model to pair natural language queries and logical forms; however, as an additional step, they implement a data enhancement strategy based on grammar. Essentially, the authors start with natural language and logical form pairs, similar to those used by SEMPRE, and then use different methods to induce new grammatical rules which can then generate more training examples for their neural network. Other neural semantic parsers that use sequence-to-sequence learning problems were explored by Xiao et al. (2016), Krishnamurthy et al. (2017) and Cheng et al. (2019).

3 Data & Methods

Data. Our parser queries a subset of the Million Song Dataset (Bertin-Mahieux et al., 2011). Song data presents a novel challenge for semantic parsing because of the enormous variety in the entity names; a phrase of any length or grammatical structure could be a song or band name.

We used a small subset which represents relationships between songs, artists, track years, and genres. For demonstration purposes, we extract 20 artists and at most 20 songs per artist. We reformat the table representation as a knowledge graph where each line lists a directional relationship and two entities. Even with such a small subset, there are more than 1,000 potential relationships represented.

To train our parser, we use a dataset of 765 pairs of canonical utterances and corresponding natural language paraphrases. The canonical utterances are generated from the seeded grammar that we created. As a result, they represent only the sorts of questions that can be parsed using our grammar, not all the sorts of questions that could be answered using our songs dataset. We generate 255 unique canonical utterances and write three paraphrases per utterance by hand.

We use 60% of these pairs of canonical utterances and paraphrases as training data, 20% as validation data, and 20% as testing data.

[1]We implemented the steps outlined in the literature review above with the aid of a tutorial for SEMPRE (Labutov, 2018) provided by the company laer.ai.

Methods.[1] We first seed the general grammar by declaring entities of type song, artist, year, and genre. We also add all relationships between these types, such as the artist of a song and the genre of an artist. Examples are shown in Table 1.

We then generate canonical utterances based on the rules in this grammar, and write natural language paraphrases of each canonical utterance.

Finally, we train a model to predict a canonical utterance (and thus the corresponding lambda DCS logical form) based on a natural language input query. The model first performs a beam search to iteratively identify the top 20 candidate parses that are plausible given the rules in the grammar and the weights learned thus far. We use a maximum depth of 22 to terminate the beam search.

Then, a log-linear model assigns probabilities to each of the candidate parses based on a set of features computed for each paired natural language query q and candidate parse p. Because SEMPRE calls the Stanford CoreNLP package, it can calculate a wide array of linguistic features for each pair. These include:

- *baseline:*
  - count of uncommon tokens in q and p
  - part of speech of tokens in q and p
- *token:* count of common tokens in q and p
- *skip-bigram:* count of common (not necessarily consecutive) pairs of tokens in q and p
- *unalign:* count of unaligned tokens in q and p
- *align:* scores on the alignment between tokens in q and p according to the Berkeley Aligner (Liang et al., 2006)
- *ppdb:* count of paraphrase pairs in q and p according to a Paraphrase Database (Ganitkevitch et al., 2013) [2]
- *candidate:* complexity of p, measured by the number of nodes in the parse tree
- *db-relationships:* counts of various database relationships (comparatives, superlatives, and domain-specific relationships between types) in p.

We train the model using two iterations of stochastic gradient descent through the training data as recommended by Wang et al. (2015).

4 Results

Baseline. Our baseline model is trained on only *baseline* features described above. Using these features, the baseline model has an accuracy score of 50.3% and an oracle score of 68.6% when evaluated on the test data.

[2]We used the small English version to compute features.

Entity	Type	Database representation
song	**TypeNP**	*en.song*
artist	**TypeNP**	*en.artist*
song	**RelNP**	*by_artist_song*
artist	**RelNP**	*!by_artist_song*
Ameno	**EntityNP**	*en.song.ameno*
Andromenda	**EntityNP**	*en.artist.andromeda*

Table 1: Examples of relationships and internal types, as represented in the seed lexicon part of the grammar.

Experiments with Additional Features. We next train a model with all of the features, which correctly parses 64.1% of the validation data and has an oracle score of 75.8% on the validation data.

To understand the importance of each individual feature, we conduct a leave-one-out experiment, removing one category of features in each trial. Figure 1 shows the results. Oracle accuracy on the validation set is fairly static across trials, which suggests that the marginal gain from using any additional features comes from improved ranking of the candidate parses.

The model which we train without using *token* features has the lowest accuracy at 58.2%, which suggests that common tokens between the canonical utterance and the paraphrase is a relatively useful feature for choosing the correct utterance. While removing the *align* features produces a relatively large drop in accuracy, it also improves oracle accuracy by a marginal amount.

Two features did not affect the performance at all when left out of the model: the *ppdb* and *db-relationships*. Interestingly, the model trained without the *unalign* performed better than the model trained using all features, with a 65.4% accuracy and 76.5% oracle accuracy on validation data. We treat this as our final model. Evaluated on the reserved test data, it achieved an accuracy of 63.1% and an oracle score of 75.8%.

Experiments with Sample Size One of the benefits of using a paraphrase model with a domain-specific grammar is that it requires little training data. We therefore explore the marginal contribution of additional training data to the accuracy score.

We train our model on five different sizes of training data using the best set of features identified above, taking 10 random samples of each size from our training data. To ensure that larger samples represent a greater number of *types* of queries - instead of simply a greater number of paraphrases of the *same* queries - we sample unique

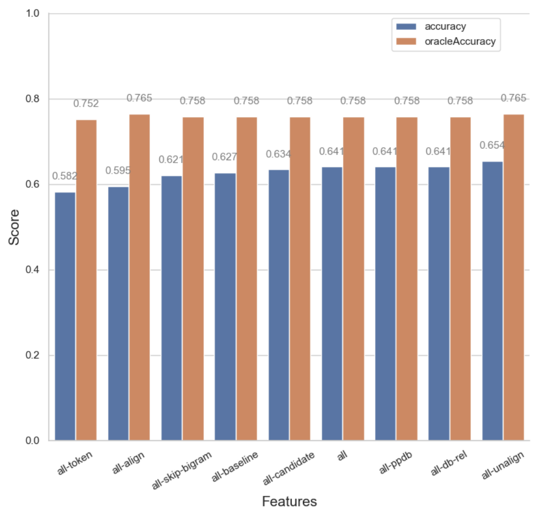

Figure 1: Percentage of phrases parsed correctly on validation set using different sets of features.

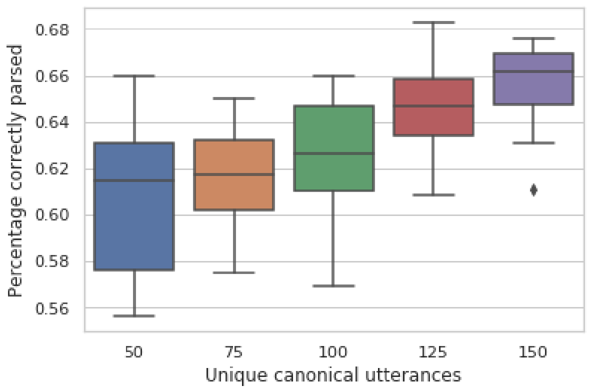

Figure 2: Percentage of phrases parsed correctly for different size of the training data.

canonical utterances and use all three paraphrases of each selected canonical utterance.

The results of this experiment are displayed in Fig 2. We find that larger sample sizes tend to improve accuracy, but much of the learning can be accomplished with just 50 unique canonical utterances paraphrased three different ways. There is also quite a spread between the performance of different samples, pointing to the impact of the specific composition of the sample. [3]

Experiments with Paraphrases per Utterance. While the amount of training data can be reduced by using fewer unique canonical utterances, it can also be reduced by using fewer paraphrases per canonical utterance.

We therefore run our best performing model specification with different number of paraphrases per utterance - one, two or three paraphrases per utterance. We find that for all these three iterations, our model has very similar results. The accuracy scores on the validation data for one, two, and three utterances are 63.4%, 65.4%, and 64.1% respectively – and all three trials have the exact same oracle score.

From this, we conclude that increasing the number of paraphrases per utterance does not significantly improve the performance of our model. Each additional paraphrase appears to encode relatively little new linguistic information; the presence of one example of each type of query is sufficient.

[3]We note that for larger sample sizes, we expect a smaller spread because the samples are quite similar; we only had 159 unique utterances in the training data to choose from.

Error Analysis. We examine cases that were incorrectly parsed in the test data to better understand the limitations of our model. We find that the parser has difficulty with numerical relationships, such as “Show songs with more than two artists.” and “Which songs came out after 2006?”.

For some specific numerical relationships, we can attribute these mistakes to the fact that no similar relationships were modeled in the training data. For example, due to our particular random train/validation/test split, there was not a single training example demonstrating relationships like ”after” or ”later than” - and every single test example with such a relationship was parsed incorrectly.

This illustrates a general principle: our model is pretty good at parsing any paraphrase of types of queries that it has seen before, but very bad at understanding types of queries that it has not seen. Our error analysis thus confirms this common characterization of paraphrase models with domain-specific grammar.

5 Conclusion

The semantic parser that performs best on our validation data achieves an accuracy score of 63.1% and an oracle score of 75.8%, which represents a large improvement over baseline performance.

We find that *token* and *align* features provide the largest marginal gains compared to using every other feature, while *unalign* features do not help parse this dataset. Furthermore, taken together, our experiments on sample size indicate that it is more important to train a semantic parser on many unique queries than to train it on many paraphrases of the same query.

There remains room for further development of a parser that can, with limited training data, learn to flexibly parse a wide range of queries.

6 Collaboration Statement

All team members contributed equally to building the model and writing the paper.

7 Reproducibility

All code and data for this paper is available at https://github.com/brinaseidel/sempre.

References

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425, Baltimore, Maryland. Association for Computational Linguistics.

Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. 2011. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*.

Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2019. Learning an executable neural semantic parser. *Computational Linguistics*, 45(1):59–94.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. *CoRR*, abs/1601.01280.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764, Atlanta, Georgia. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany. Association for Computational Linguistics.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526, Copenhagen, Denmark. Association for Computational Linguistics.

Igor Labutov. 2018. Semantic parsing tutorial with sempre.

Percy Liang. 2013. Lambda dependency-based compositional semantics. *CoRR*, abs/1309.4408.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 104–111, Stroudsburg, PA, USA. Association for Computational Linguistics.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. *CoRR*, abs/1508.00305.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1332–1342. Association for Computational Linguistics.

Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1341–1350, Berlin, Germany. Association for Computational Linguistics.

Daniel Younger. 1967. Recognition and parsing of context free languages in time n3. *Information and Control*, 10:189–208.