
Semantic Parsing and its Applications in Natural Language Programming

UNDERGRADUATE THESIS

*Submitted in partial fulfillment of the requirements of
BITS F421T Thesis*

By

Pratik JOSHI
ID No. 2015A7TS0019G

Under the supervision of:

Dr. Navin GOYAL
&
Dr. Monojit CHOUDHURY




BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI, GOA CAMPUS

December 2018

Declaration of Authorship

I, Pratik JOSHI, declare that this Undergraduate Thesis titled, 'Semantic Parsing and its Applications in Natural Language Programming' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: 

Date: 7/12/18

Certificate

This is to certify that the thesis entitled, "*Semantic Parsing and its Applications in Natural Language Programming*" and submitted by Pratik JOSHI ID No. 2015A7TS0019G in partial fulfillment of the requirements of BITS F421T Thesis embodies the work done by him under my supervision.



Supervisor

Dr. Navin GOYAL

Researcher,

Microsoft Research India

Date: 7/12/18

Co-Supervisor

Dr. Monojit CHOUDHURY

Researcher,

Microsoft Research India.

Date: '

“The limits of my language are the limits of my universe.”

Goethe

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI, GOA CAMPUS

Abstract

Bachelor of Engineering (Hons.) Computer Science

Semantic Parsing and its Applications in Natural Language Programming

by Pratik JOSHI

Semantic parsing is a broad but critical sub-domain in natural language processing/understanding which deals with the conversion of natural language into a structured form (called a logical form) which the machine better understands, and has a fixed representation which captures the semantics of a sentence or query. This problem overlaps with many specialized problems such as natural language programming, entity extraction, question-answering, as well as inference and reasoning in natural language. In this thesis, we focused our research efforts into three parallels. One: a specialized approach where we focus on improving previous state-of-the-art performance on the semantic parsing task of converting natural language descriptions of regular expressions into the regular expressions themselves. We were able to beat the state-of-the-art on this task by using multi-task learning networks, trained on 2-3 related tasks; Two: an investigation into previous semantic parsing approaches and state-of-the-art algorithms/models for well-known problems and datasets, as well as a break-down of where semantic parsing fails and various associated tradeoffs that you must control when approaching a certain problem in this field. Here, we calculated and plotted various quantitative factors and could identify subtle indicators for variation and quality of datasets; Three: A study into the data collection techniques employed in collecting past semantic parsing datasets, and the planning and design of the collection of a "NL2Regex" dataset by using these different techniques in pilots and analyzing the best methods. We are currently classifying different techniques by doing a cross-domain analysis, and attempting to resolve which methods would be the best for the NL2Regex task. This is currently being carried forward.

Acknowledgements

I would like to thank Dr. Navin Goyal and Dr. Monojit Choudhury, my managers at Microsoft Research India. Their knowledge and efforts have helped me drive this project forward and have taught me a lot about the field. I would like to thank Dr. Swati Agarwal, my on-campus supervisor for keeping me on track for the completion of this thesis. Finally, I would also like to thank all the research fellows and interns at MSR who have provided suggestions and tips to improve along the way.

Contents

Declaration of Authorship	i
Certificate	ii
Abstract	iv
Acknowledgements	v
Contents	vi
List of Figures	viii
Abbreviations	ix
1 Introduction	1
1.1 Overview of Semantic Parsing	1
1.2 History and Background	2
1.3 Applications of Semantic Parsing	3
1.3.1 Natural Language Interfaces to Databases	3
1.3.2 Natural Language Programming	3
1.3.3 Sequential Command Understanding	3
1.4 Challenges	4
1.4.1 Lack of Quality Data	4
1.4.2 Designing of Scalable and Generalizable Parsers	4
1.4.3 Lack of Partial Credit Metrics	5
2 Problem Outline	6
2.1 Problem Definition	6
2.2 Motivation	7
2.3 Related Work	7
2.3.1 Major Contributions	7
2.3.2 Drawbacks	8
3 NL2Regex Results Improvement	9

3.1	Sanity Check and Dataset Investigation	9
3.2	Paraphrase-to-Regex Parsing Task	10
3.3	Error Analysis and Data Analysis	10
4	Semantic Parsing Dataset Analysis	12
4.1	Literature Survey of Associated Datasets	12
4.2	Quantitative Analysis of Datasets	12
4.2.1	Quality of Natural Language	13
4.2.2	Quality of Logical Forms	15
4.2.3	Results	16
4.3	Qualitative Analysis of Datasets	18
5	Data Collection Analysis	19
5.1	Generalization of Existing Data Collection Methods	19
5.1.1	LF Phase	19
5.1.2	NL Phase	20
5.2	Classification of Existing Datasets	21
6	Conclusions	22
6.1	Observations and Inference	22
6.2	Future Work	22
6.3	Final Note	23
	Bibliography	24

List of Figures

3.1	Results of modifying models on NL-RX dataset	11
4.1	Datasets and their best-performing models	13
4.2	Top 20% 3-gram variations coverage	17
4.3	Qualitative analysis on the datasets. Each category can be classified as High, Medium, or Low	18
5.1	LF Phase Classification	21
5.2	NL Phase Classification	21

Abbreviations

NLP	Natural Language Processing
NL	Natural Language
LF	Logical Form
AI	Artificial Intelligence
SQL	Structured Query Language

I dedicate this to my family.

Chapter 1

Introduction

1.1 Overview of Semantic Parsing

Semantic parsing is formally defined as the process of mapping natural language utterances into formal, structured representations called logical forms. The logical forms are representations which are understandable by a machine.

Semantic parsing is an attempt to solve the larger, but less understood problem of natural language understanding. Natural language complexity can be defined into three layers: syntactic, semantic, and pragmatic. Syntax is a surface-level feature of language. It covers word-level features such as grammar, punctuation, and vocabulary. Essentially, it is the combination of words and phrases to create well-formed sentences. Semantics is a more complex study into language. It is concerned with the encapsulation of meaning in a sentence, or any body of text. How the reader perceives the intent and literal meaning of text is studied in semantics. Finally, pragmatics is the highest-level of abstraction in natural language, and is a field in linguistics which is still not well-understood in computational linguistics. It involves the study of language behaviour in different contexts, implicature, and other higher level phenomena which require the knowledge of the world and its use in the same. The issue with solving problems in these respective layers independently is that they aren't easily separable, apart from syntax. It can be considered as a bottom-up problem, where each level above involves the extensive use of the lower layers, along with existing world knowledge. NLP for syntax-level problems is a well-defined and developed field. State-of-the-art models on these problems are currently being used at production level. However, semantic-level problems are still an actively developing field. It is the next step towards approaching language understanding.

1.2 History and Background

Semantic parsing is not a new problem. It dates back to the 1970's, when it was approached in the context of a natural language understanding problem for interacting with an environment of blocks in a project called SHRDLU [Winograd, 1971]. It was a project which held historical significance, and set the ground for this field. In the early stages of semantic parsing research (1990s-2000s), semantic parsing was tackled in a so-called 'traditional' manner. It involved the manual creation of grammars, which were rules that decomposed natural language sentences into logical forms [Winograd, 1971] [Warren and Pereira, 1982], and the use of inductive logic programming [Zelle and Mooney, 1996a]. These parsers yielded excellent results in their restricted domain and demonstrated complete understanding, but could not scale well to other problems and did not handle large natural language variations well. In addition to this, the creation of grammars was cumbersome, and these processes were very sensitive to noise. However, at this point, researchers were already thinking about the importance of this problem, with the creation of datasets and grammars for problems such as natural language interfaces to databases, and robot instruction control.

In the 2000s, most of the work in this area involved the creation/learning and use of different grammars and lexicons on controlled tasks [Zettlemoyer and Collins, 2005][Kwiatkowski et al., 2010], particularly general grammars such as SCFGs [Wah Wong and J. Mooney, 2007] and CCGs [Zettlemoyer and Collins, 2005]. This improved upon manual grammars primarily because they leveraged the syntactical nature of the sentence, but they still couldn't cover enough variation and weren't robust enough to be used in the real world. However, following the development of advanced neural network techniques, especially the Seq2Seq model [Dong and Lapata, 2016], and the availability of powerful computational resources, neural semantic parsing started emerging. Not only was it providing competitive results on the existing datasets [Zhai et al., 2017] [Alvarez-Melis and Jaakkola, 2016], but it was robust to noise and did not require a lot of supervision and manual intervention.

The current transition of traditional parsing to neural semantic parsing has not been perfect though. Neural semantic parsing with its advantages, still fails to solve the problem at a deeper level. Neural models like Seq2Seq treat the parsing problem as a sequential translation problem, and the model learns patterns in a black-box manner, which means we cannot really predict whether the model is truly solving the problem. Intermediate efforts and modifications to the Seq2Seq to incorporate syntax and semantic meaning have been attempted [Yin and Neubig, 2017] [Dong and Lapata, 2018] [Shi et al., 2018] with a marked improvement in results, but there remains a lot of ambiguity to be taken care of.

1.3 Applications of Semantic Parsing

Language understanding has vast applications everywhere. It contributes to the ultimate goal of artificial intelligence: Having machines that can behave like humans. Currently, there are a large range of potential applications that can be roughly classified in the following sub-fields mentioned below.

1.3.1 Natural Language Interfaces to Databases

The large amount of data that resides on the internet can often be leveraged by technically-skilled experts who know how to query the database using a specific query language or a tool. However, in order to make this data easy to access for people with limited or no expertise in querying, it needs to be ensured that the barriers to access are lowered. The best way to do this is to allow them to communicate their queries in the best way they know how: using natural language. Plenty of datasets (WikiSQL, Spider, ATIS, Geo880) have been compiled for research, and models have been implemented on these datasets to yield promising performance. Google has started implementing these interfaces in Google Sheets, however it is far from complete. Salesforce’s Einstein AI has been hard at work to allow people to “talk to” their data and query large databases using only natural language. This sub-field has become the most prominent and product-oriented out of all the other fields, and we can expect production-level models to be implemented soon in this area.

1.3.2 Natural Language Programming

Natural language programming is the process of generating code snippets using descriptions of the intent via natural language. It has potential applications in automatic comment generation, language plug-ins and extensions for convenient programming, and more. It potentially makes it easier for people to interact with programs and coding. Although the work in this area is still very experimental and far from feasibility, there is a lot of intriguing progress being made for general-purpose code generation [Yin and Neubig, 2017], as well as bash command generation [Lin et al., 2018]. This work is crucial in realizing the lack of quality data in the domain, and where we stand in terms of perfect code generation.

1.3.3 Sequential Command Understanding

After being initiated by SHRDLU (Winograd 1972), there has been interesting work on parsing consecutive natural language commands, including navigational instructions for robots [Artzi and Zettlemoyer, 2013] and context dependent instructions [Long et al., 2016]. The parser

should be able to understand references to an original entity after a number of commands, a process called coreference resolution. The problem also stresses on context-dependence. Depending on the environment and what utterances have been parsed earlier, the model should be able to correctly convert the current utterance into an appropriate logical form. This has a wide scope in future robot design and task-oriented machines.

1.4 Challenges

There are a host of challenges and problems that modern-day semantic parsing faces. These need to be parallelly addressed before progress can be attained in the area.

1.4.1 Lack of Quality Data

Due to the largely data-driven nature of semantic parsers nowadays, there is a need to collect quality data that ensures that the semantic parser learns a deep, meaningful representation of the logical form. This means that the data needs to have the following properties:

- The natural language utterances need to have the appropriate level of variation.
- The data needs to be consistent and clean. Consistency means that there need to be enough datapoints that represent a wide range of permutations and nesting of logical forms.
- The distribution of logical forms in the dataset needs to accurately represent the real world.

Without the above properties, it is difficult for such models to incrementally learn mappings, and thus they tend to overfit on datapoints with complex logical forms, poorly generalizing to different complex queries. This has put an emphasis on efficient data collection methods, and work is being actively done here as well [Wang et al., 2015].

1.4.2 Designing of Scalable and Generalizable Parsers

There have been a range of excellent publications on semantic parsers on a certain domain, such as SQL and Python. However, these parsers are difficult to port into production due to their effectiveness only on the restricted problem. This includes their restriction in say, the database schemas they've been trained on (for SQL), or the particular codebase they've been designed for (such as on Hearthstone [Yin and Neubig, 2017]). This poses the challenge of creating semantic parsers that are generalizable to different environments and different tasks. This is being actively

worked on, involving the use of abstract syntax trees [Rabinovich et al., 2017] and training on table-aware representations for text-to-SQL problems [Yu et al., 2018a].

1.4.3 Lack of Partial Credit Metrics

Semantic parsers are often compared with a wide-range of metrics, such as execution accuracy (correctness of the execution of logical form), logical form matching accuracy (the correctness of the logical form), BLEU score, and F1 score. Due to the structured and executable nature of the logical forms, executing the forms and checking if they yield the desired result is a correct way to judge the effectiveness of the parser. However, even if the logical form is very close to being correct, or almost yields the desired result, it is still judged as inferior compared to another such model, due to the binary nature of the metrics. Therefore, there need to be better metrics which assign partial credit. This will result in a better comparison between models, and we will get a better idea on how these models have understood the task and how they'll perform in the real world. There have been publications which do more meaningful comparisons, such as evaluating generated SQL queries [Finegan-Dollak et al., 2018], but this has not been done for many sub-domains.

Chapter 2

Problem Outline

2.1 Problem Definition

Initially, at the start of the thesis, the general theme of the problem was to use semantic parsing to solve a certain problem in natural language programming. Work was done by a research intern prior to my joining, which involved going through a set of code generation tasks, such as text-to-SQL, generation of a query language called Kusto, and even the smaller problem of converting text to the appropriate Caesar ciphers. He had drawn a set of observations about the performance of sequence-to-sequence neural models on these tasks, and had attempted to improve upon the state-of-the-art on WikiSQL [Zhong et al., 2017], a text-to-SQL dataset. Then, the project was focused on a more defined problem of converting natural language descriptions to regular expressions. This is where I came in.

We started looking at the NL2Regex problem in more detail. There were only a few publications in this small sub-domain [Kushman and Barzilay, 2013] [Locascio et al., 2016]. Initially, our aim was to beat the state-of-the-art on this problem. However, it quickly became clear that just pushing up the accuracy was not a healthy way to approach the overall problem of semantic parsing. After multiple discussions and analyses over the course of one-and-a-half months, we slowly honed the problem into a three-pronged approach to the semantic parsing problem, which would help us understand the problem better and provide a platform to make better improvements.

Formally, we focused our research efforts into three parallels:

1. Designing a specialized approach where we focus on improving previous state-of-the-art performance on the semantic parsing task of converting natural language descriptions of regular expressions into the regular expressions themselves.

2. An investigation into previous semantic parsing approaches and state-of-the-art algorithms/models for well-known problems and datasets, as well as a break-down of where semantic parsing fails and various associated trade-offs that you must control when approaching a certain problem in this field.
3. Analyzing different data collection methods for the various semantic parsing datasets, and identifying good practices for collecting quality data by doing a quantitative analysis of various existing datasets, as well as quick qualitative checks for reinforcement.

2.2 Motivation

In this internship, I aimed to understand semantic parsing better. Due to the large gap between traditional and neural semantic parsing, we tried to break down the problem and understand why it is difficult to bridge that gap, and what existing techniques seem to do the job better. We investigated what makes the problem difficult, and why neural models seem to perform extremely poorly on real-world, uncontrolled data. We hope to leverage the knowledge gathered from this to continue to improve upon the NL2Regex task.

2.3 Related Work

Prominent work on the conversion of natural language to regular expressions problem has been limited. It can be classified as a code generation problem, with the aim to convert the descriptions of natural language into regular expressions. State-of-the-art research output on this problem has come from MIT [[Kushman and Barzilay, 2013](#)] [[Locascio et al., 2016](#)]. They have taken both a traditional approach and a neural approach to the problem in their respective papers, with the latter providing improved results and being the current state-of-the-art for this problem.

2.3.1 Major Contributions

Their work provides an excellent methodology to synthetically generate parallel data for this task by using a small hand-crafted grammar and jointly generating canonical descriptions(simple, structured natural language descriptions) and regular expressions by generating random syntax trees. A large amount of data is generated (10000 pairs), and the data is clean.

2.3.2 Drawbacks

Upon observation of their model performance, it seems to be a good improvement upon the previous paper. However, their paraphrased Turk dataset is extremely noisy. Many of the paraphrased descriptions are a direct copy of the synthetic descriptions. Also, there are many paraphrases which are unclear, and show that the crowdworkers misunderstood the paraphrasing task. This parsing task has been treated as a sequence-to-sequence generation problem, ignoring syntactical and structural components of the regular expressions, which results in many syntactically incorrect and incomplete generations. In addition to this, the data generated is still controlled in terms of length and variability, which makes it impractical to scale up into real-world systems. One primary component that their works missed out on is effective error analysis. They failed to investigate which data points tend to be misclassified, and what factors/characteristics of the data cause or correlate to a larger error rate.

Chapter 3

NL2Regex Results Improvement

3.1 Sanity Check and Dataset Investigation

In the state-of-the-art paper [Locascio et al., 2016] for text-to-regex, they demonstrated the effectiveness of sequence-to-sequence models on the task. Fueled by this, I used OpenNMT, an open-source neural machine translation framework, to build and run models on the task. We used the dataset from the state-of-the-art paper. We discussed possible ways we could improve upon the existing results, and decided to first examine the dataset and data generation methods before committing to the task.

The way the dataset was created was by first generating regular expressions and crude NL descriptions together in a tree-based synthetic manner. These crude NL descriptions (called synth-NL) were then provided to crowdsource workers for paraphrasing. This resulted in a parallel corpus of regular expressions (using Amazon Mechanical Turk) annotated with natural fluid paraphrases (called Turk-NL). Upon investigation of the dataset, we figured out that the way that the synth-NL was paraphrased was poor, and this resulted in a lot of ambiguity and misannotated examples, along with the fact that there were many examples which blatantly copied the synth-NL as the paraphrase. In addition to this, the regular expressions generated were not parseable. The grammar for creation for regex was hand-crafted, and used just as a proof of concept. This sparked off a deliberation on potentially collecting a dataset of our own, by generating parseable regex, and conducting efficient crowdsourcing to get quality paraphrases.

However, in order to also propose a model which beat the state-of-the-art before the collection of a new dataset, we needed to show marginal improvement over the existing dataset (called NL-RX). To kick this off, I ran a sanity check between the synth-NL and regex by running neural models between the two. This was to ensure that there was a deterministic mapping between the two, and we expected an accuracy of more than 95% on this. We easily achieved

this. In the meantime, I was also changing the generation script to generate parseable regex and corresponding synth-NL. I ran models on this new data as well, and achieved even better results.

3.2 Paraphrase-to-Regex Parsing Task

We then shifted our attention to the Turk-NL to regex conversion. To our dismay, we achieved results with slightly less accuracy than the state-of-the-art. There were two metrics we were using. One was accuracy of exact matching (The generated and gold regex were exactly the same). However, there was the possibility that two regexes could match the same text, even if they were not exactly the same. The paper used the concept of DFA-matching accuracy (the DFA's of the two regular expressions were equivalent), and we continued to use this. After extensive hyperparameter tuning, involving changing RNN size, global attention type, maximum gradient normalization, and word vector size, there was no significant change in the results, and we still achieved less than the state-of-the-art.

At this point, we started reading literature on neural semantic parsers for code generation problems in different domains. We gathered information on a range of neural modeling techniques, such as constrained decoding [Dong and Lapata, 2018], grammar models [Yin and Neubig, 2017], abstract syntax networks [Rabinovich et al., 2017], and multi-task learning models [McCann et al., 2018]. We decided to implement these techniques on the NL2Regex task.

The primary difficulty in implementing these models in a different sub-domain was that these works were very recent, and their codebases were not fully developed for push-button replication and general use. After a lot of modifications, I was able to implement most of these models on the NL2Regex task. The first exciting break came when the constrained decoding model ran and beat the state-of-the-art, albeit by 1-2% accuracy. Following this, I attempted to run the other models on the task, and observed the best improvement with the multi-task learning network called MQAN (Multi-Question Answering Network). The results of the different models are tabulated in Fig. 3.1.

3.3 Error Analysis and Data Analysis

We noticed that the improvement was very limited, despite using much more powerful neural techniques, and decided to manually check a random sample of the wrongly generated examples from the improved models. Out of the subsamples, we noticed that there were a large amount of datapoints in which the synth-NL and paraphrase had a different intent entirely. This was a result of poor paraphrasing and ambiguity of the synth-NL, which invariably made it difficult for crowdsource workers to accurately paraphrase. So, despite the model correctly converting

Model	Exact Matching Accuracy	DFA Equals Accuracy
SOTA [Locascio et al., 2016]	38.6%	58.2%
Seq2Seq, BiLSTM+CopyAttn	38.96%	55.24%
Coarse2Fine [Dong and Lapata, 2018]	42.52%	59.68%
MQAN [McCann et al., 2018]	44.96%	61.92%

FIGURE 3.1: Results of modifying models on NL-RX dataset

the paraphrase into the appropriate regex, the gold regex in the dataset was different. We then conducted a more thorough error analysis, and classified the errors as below:

At this point, I also wanted to attempt a semi-automatic evaluation of paraphrase quality. I took a small subsample of datapoints correctly and incorrectly generated by the above neural models, and manually extracted some surface-level features which could contribute to paraphrase quality. A few indicators I noted were:

- Homogeneity of prefix of natural language utterance, which would indicate poor paraphrasing.
- Use of the same terms in the paraphrase as those in the respective synth-NL.
- BLEU-4 score between synth-NL and turk-NL, which could be proportional to variation.

Following this, I used a decision tree regressor trained on these features, with labels being whether the neural parsers correctly generated the regex for the paraphrase or not. I then ran these results on a test set. The results were as below:

Although this was a good exercise in determining some surface-level factors, it did not produce the desired results that I was hoping for.

At this point, we understood that we needed to either collect quality data for this problem before we could note any more significant progress. The dataset was too faulty to be used any further. Taking away important observations from this parallel, we decided to focus on the other parallels.

Chapter 4

Semantic Parsing Dataset Analysis

4.1 Literature Survey of Associated Datasets

Before beginning work in this parallel, it was important to read the associated publications of the datasets, primarily the ones which outline the collection of these datasets. We identified around 10-15 datasets which we felt were consistently worked on by many papers, and were benchmarks for assessing performance of models. We also collected the state-of-the-art performance measure on these respective datasets to observe how close the problem was to being solved, and what margin of error there remained. This also gave us an idea of how much error could be attributed to the lack of or inconsistent data quality. The datasets are described in Fig. 4.1.

I took note of the general ways that data was collected, and noted the different methodologies of crowdsourcing, how effective paraphrases were generated, how logical forms were constructed. Particularly, I paid close attention to the qualitative checks that the creators did, and their intent for creating the dataset.

4.2 Quantitative Analysis of Datasets

I downloaded the datasets from various sources, and preprocessed them if they weren't already preprocessed. Following this, my managers and I had discussions on what quantitative features we could extract, from both the NL side and the logical form side, that could potentially correlate to natural language variation and reflect the quality of the datasets. As all of these datasets were from different domains (dealing with different logical forms, different environments and use cases), the hope was to be able to conduct a cross-domain analysis on factors like natural language variation and task complexity.

Dataset	State-of-the-art	Metric	Performance
ATIS [Dahl et al., 1994]	WKZ14 [Wang et al., 2014]	Accuracy	91.3%
Geo880 [Zelle and Mooney, 1996b]	WKZ14 [Wang et al., 2014]	Accuracy	90.4%
Jobs640 [Tang and Mooney, 2001]	ASN [Rabinovich et al., 2017]	Accuracy	92.9%
WikiSQL [Zhong et al., 2017]	IncSQL+Exec-Guided Decoding [Shi et al., 2018]	Execution Accuracy	87.2%
WebQuestions [Berant and Liang, 2014]	[Xu et al., 2016]	F1-score	52.5
IFTTT [Quirk et al., 2015]	LSTM-DRNN [Alvarez-Melis and Jaakkola, 2016]	F1-score	77.4
Django [Oda et al., 2015]	[Yin and Neubig, 2017]	Accuracy	71.6%
NL2Bash [Lin et al., 2018]	ST-CopyNet [Lin et al., 2018]	Token-level Accuracy	36%
Hearthstone [Ling et al., 2016]	ASN [Rabinovich et al., 2017]	Accuracy	22.7%
CodeNN [Iyer et al., 2016]	CodeNN [Iyer et al., 2016]	BLEU-4	20.4 (C#), 17.0 (SQL)
WikiTableQuestions [Pasupat and Liang, 2015]	[Krishnamurthy et al., 2017]	Accuracy	45.9%
Spider [Yu et al., 2018b]	SyntaxSQLNet [Yu et al., 2018a]	Accuracy	24.8%
SCONE [Long et al., 2016]	RandoMer [Guu et al., 2017]	Accuracy	52.9%,37.1%,46.2% (Al,Ta,Sc)
Overnight [Wang et al., 2015]	[Wang et al., 2015]	Accuracy	58.8%

FIGURE 4.1: Datasets and their best-performing models

4.2.1 Quality of Natural Language

The quality of natural language in the dataset was important for ensuring that models properly understood the intent of the question, and were able to learn how to derive meaning from them. In order to train a model which was robust to differently worded, yet semantically similar statements, such examples would have to be included in the dataset. In other words, there needed to be sufficient amount of variation in the dataset. But variation is not easily measurable. There were different kind of variations: lexical, phrasal, and syntactic variations. Lexical variations involved words with the same meaning being present in similar contexts. For example, "beginning" and "starting" are synonyms or similar words, and we can classify their presence in the dataset as lexical variation. Phrasal variations define small consecutive occurrence of words which are semantically equivalent, yet differently worded. For example, a possible phrasal variation of "beginning" is "that begins", where even though the base lexical unit (begin) is unchanged, the way it is represented is different. Finally, syntactic variations are much larger variations, which cannot simply be swapped with one another. Syntactic variations

alter the structure of the sentence, yet maintain semantic equivalence. An example is "lines that start with a capital letter but not a vowel" and "lines not starting with a vowel, but need to be a capital letter". In addition to the above variations, we also needed to capture the complexity of natural language, for which there also wasn't a clear way to measure.

In order to capture a sense of the above variations, we decided to compute the following metrics:

1. **Size of vocabulary:** The larger the vocabulary (unique words) of words used in the dataset, the more variation there tends to be. Although this is not an excellent metric, since larger vocabulary can also be attributed to a larger dataset or more difficult task, it still gives a brief idea.
2. **Common Word Analysis:** I computed the number of common words per datapoint, and uncommon words per datapoint. I had a scraped list of most commonly used words from Wikipedia, and the intuition was that this metric could come in handy when deciding which dataset would perform the best on transfer learning models, since more common words indicates more common shared knowledge.
3. **Anaphora detection:** One of the phenomena in linguistics which machines tend to struggle with is anaphora detection, that is, the association of a pronoun to an earlier entity in the sentence. For example, in "Jack didn't like James because he was very annoying", the "he" is an anaphor referring to "Jack". Machines tend to confuse this, and work is being done in this field (Lee.et.al 2017). Therefore, this would be a good way to assess how much complexity anaphora contributed to the dataset.
4. **Zipf distribution:** It is a commonly known observation in NLP that word distributions in well-made datasets tend to follow the Zipf distribution. That is, the best fit line between the natural log of the rank of the words (ranked in decreasing order of frequency) vs. the natural log of the frequency of the associated word has a slope equal to or near -1. Therefore, to hold these datasets to the same comparison, I plotted these values and noted the slopes.
5. **3-gram variations:** I extracted the top 20% most frequent unique 3-grams in the dataset, and computed in how many datapoints those 3-grams existed. This is a well known way to get a quick idea of variation in the dataset. The higher the percentage of representation of the 20% in the data, the less variation there was in the dataset.
6. **Perplexity of N-gram model:** I ran a 4-gram model on a train split of the dataset, and then obtained the perplexity of that model on a test split. The perplexity is a measure of how well a probabilistic model predicts a given sample. The smaller the perplexity, the better the ability to predict. This number also can indicate how much variation there is in a dataset, more importantly how well represented variation is in the dataset. For

example, a dataset with a large amount of variation being represented in a small number of datapoints will no doubt be of no use to the model, since it doesn't have enough data to truly learn the semantic equivalences. I used the SRILM framework for running the 4-gram models.

4.2.2 Quality of Logical Forms

The quality of logical forms in the dataset is generally governed by how correct it is (is it parseable/executable), how meaningful it is (does the execution result make any sense?), and how consistent and complex it is (is there sufficient permutation of variables, sufficient nesting variations?). Fortunately, there were statistics of some metrics contributing to logical form quality in some existing publications [Finegan-Dollak et al., 2018] [Yin and Neubig, 2017], making our choices of the below indicators more concrete:

1. **3-gram variation:** I tokenized the logical forms appropriately (using parsers for the code, and manual grammars for the domain-specific languages), and then did the same 3-gram variation experiments as in the above natural language experiments. Although this metric wasn't perfect, and didn't make immediate intuitive sense, it was a way to detect variations in structure of logical form to some extent.
2. **Average AST size:** Most general-purpose programming languages can be represented as an abstract syntax tree (AST). The deeper or larger this tree, the more operators, operands and functions are often involved. Therefore, the average size of the AST, represented by average number of nodes in the tree can indicate the complexity of a code snippet. The average size of ASTs over the whole dataset was therefore measured by using various lexical parsers and libraries (antlr-4,bashlex,ast).
3. **Number of keywords:** This is particularly for the text-to-SQL datasets. Generally, the presence of complex keywords such as GROUPBY, HAVING, ORDERBY, JOIN indicate that the query is not a simple query, and does more table manipulations and lookups than those that don't have those keywords. Therefore this was a good indicator of complexity of SQL queries.
4. **Number of Tables, Schemas, Unique queries:** Again for text-to-SQL, the number of tables the queries query over, as well as the number of schemas involved also indicate the level of complexity that the dataset is dealing with. The number of unique queries defines the number of queries which, after anonymization of column names, entity names, and table names, have the same format. This also indicates how complex the functionalities for which the queries operate on are.

In addition to the above, I spent a lot of time manually observing datapoints from all these datasets. I was able to get a good idea, from my observations as well as from the general opinion of the datasets from existing publications, which datasets were of good quality.

4.2.3 Results

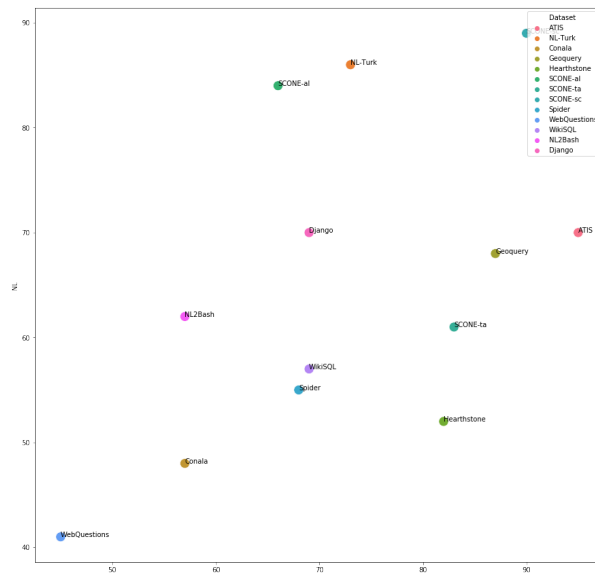
After getting the numbers for the above indicators, we were able to justify the quality of datasets to some extent. For example, for the WikiSQL dataset, which covered a large set of tables and domains, we noted that it had high NL variation due to its coverage because of the large size of vocabulary (47000 words). We could look at these figures qualitatively and make observations about the quality of dataset. However, we needed to find definite correlations in order to solidify the quantitative analysis

I plotted visualizations of the Zipf-ian distributions of word frequencies, and made some observations. For example, the less consistent datasets tended to have a very thick tail in the plots, and gaps in the middle, indicating that the distribution of words was skewed heavily.

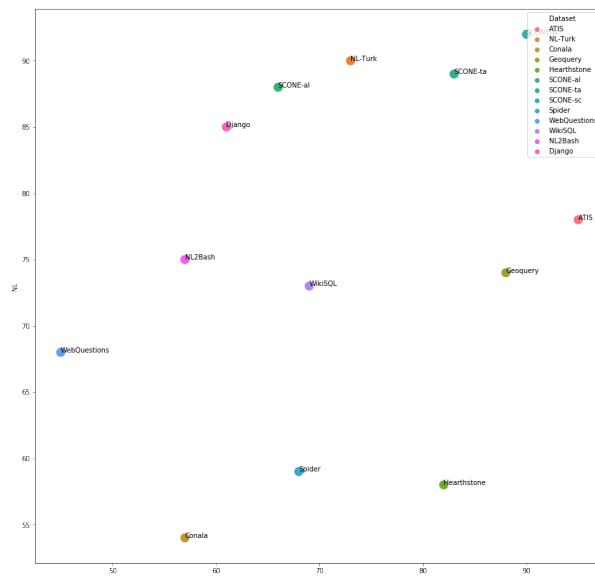
In order to carry out a comparison between datasets of both NL variation and logical form complexity, I plotted the coverage of top 20% 3-gram variations in the dataset between natural language and code on y and x axes respectively. I could not detect any definite boundaries that separated the good and bad quality datasets, the datasets with less versus more variation. I decided to conduct more data cleaning techniques to improve the representation, namely the following:

- **Keyword Anonymization:** Here, I took the frequencies of all words, and replaced all the words, which weren't common words (by referencing the common words list) and had a frequency greater than threshold 3, by the placeholder "KW". Here I assumed that keywords of a particular domain would be used often in the NL, therefore setting a threshold.
- **Named entity anonymization:** I used the state-of-the-art Stanford Named Entity Recognition (StanfordNER) software to recognize named entities in the datasets and replaced them with placeholder "NE". In addition to this, I used the frequency counts from keyword anonymizations, and replaced all words with frequency lower than or equal to 3 as "NE". I assumed here that any words with such low frequency would be constant values assigned to some keyword and hence would be replaceable with any other constant values.

After doing so, I plotted the same 3-gram visualizations. Finally, we noticed linear boundaries which separated datasets in terms of quality and variation. We also compared the plots before



(A) W/o cleaning



(B) W/ cleaning

FIGURE 4.2: Top 20% 3-gram variations coverage

and after cleaning, and noted that datasets such as Spider and WikiSQL, which are largely different in terms of code complexity and natural language variation became much farther apart. The plots are as shown in Fig. 4.2.

In order to investigate these models as well as verify their results, I took their codebases and replicated their results. This also familiarized me with the implementations, thus helping me use these models for the NL2Regex parallel.

Dataset	Variation of NL	Complexity of LF	Complexity of Task	Quality of NL	Quality of LF
ATIS	Low	Low	Low	High	Medium
Geo880	Medium	Low	Low	High	High
Jobs640	Low	Low	Low	High	Medium
WikiSQL	High	Low	Medium	High	Medium
WebQuestions	High	Medium	Medium	High	High
IFTTT	High	Medium	Medium	High	Medium
Django	High	High	Medium	High	High
NL2Bash	High	High	High	High	Medium
Hearthstone	Low	High	High	Medium	High
CodeNN	High	High	High	Medium	Low
WikiTableQuestions	High	High	High	High	High
Spider	High	High	High	High	High
SCONE	Low	Low	High	High	High
Overnight	Low	Medium	Medium	High	High

FIGURE 4.3: Qualitative analysis on the datasets. Each category can be classified as High, Medium, or Low

4.3 Qualitative Analysis of Datasets

This analysis involved observing the different datasets by taking a random subsample of the dataset and observing both the NL and the LF. The results are mentioned in Fig. 4.3. The factors I looked at were:

1. The level of observed variation that exists in the natural language.
2. The complexity of the observed logical forms, which includes length of the LF, number of observed operators and operands, as well as depth.
3. How difficult the task actually is. I answered this by judging how difficult it would be for a layman to find the logical form for a given question.
4. The quality of the NL, which involves less spelling errors, grammatical errors, semantic meaningfulness of the NL queries.
5. The quality of the LF, which involves diversity in LFs, consistency of LFs, syntactical accuracy of LFs.

Chapter 5

Data Collection Analysis

This parallel was primarily designed to ensure that we identify and investigate the best data collection methodologies in semantic parsing scenarios, so that this would guide collection of a bigger, high-quality dataset in the future, be it for NL2Regex or any other task. It was also an interesting analysis to carry out to better understand how certain methods yield better results.

5.1 Generalization of Existing Data Collection Methods

We revised the literature of the existing semantic parsing datasets, and read supplementary materials on how exactly the data collection process was carried out. However, since each collection method was always slightly different than the other, we needed to classify these methods into a smaller number of generalized techniques. We could then state that specific methods within these generalized techniques brought about higher quality. Also, we generalized in a way that other unique data collection methods would be a mix of such techniques, and could be easily classified.

We assumed the data collection process to consist of two phases: natural language data collection and logical form data collection. For each of these phases, there is an *input* and a certain *process* which converts the input into either NL or LF.

5.1.1 LF Phase

In the LF collection phase, we classify the inputs as:

- **Web/Internet:** This refers to any knowledge or content which exists on the internet.

- **Grammar+Lexicon:** This refers to an existing grammar and lexicon which defines the logical form. It consists of operators and operands which make up the LF, as well as a syntax which governs the structure of the LF.
- **World States:** This refers to an existing environment or simulation which defines the states which can be transitioned through by applying certain actions, which can be represented by logical forms.

We classify the processes as:

- **Scrape:** This refers to any action of taking any corpus of existing logical forms as input and extracting them.
- **Generative Model:** This refers to any model which, given some input, is able to create logical forms on its own.
- **Manual:** This refers to humans themselves writing out the logical forms, based on the knowledge of some input.

5.1.2 NL Phase

In the NL collection phase, we classify the inputs as:

- **NL-description:** This refers to any input which is a natural language utterance.
- **LF-description:** This refers to any input which is a logical form.
- **World description:** This refers to any input which conveys a description of the domain and environment in which the semantic parsing task is dealing with.

We classify the processes as:

- **Generate:** This refers to the action of using the input to generate NL, which is influenced by the information transferred by the input.
- **Extract:** This refers to the process of taking natural language utterances out of the given input. This is conditioned on the fact that there is no alteration, just extraction of the NL.
- **Paraphrase:** This refers to the action of taking a given input and providing a NL which is semantically equivalent to the given input. This is different from "Generate" in that here the input and output have to be semantically equivalent, unlike "Generate".

		Process		
		Scrape	Generative Model	Manual
Input	Web/Internet	CodeNN,Conala,Geo880, Hearthstone,MTG, IFTTT,NL2Bash	WebQuestions	
	Grammar+Lexicon	Invalid	NL-Turk,Overnight,WikiSQL	
	World State	Invalid	SCONE	ATIS,Spider

FIGURE 5.1: LF Phase Classification

		Process		
		Generate	Extract	Paraphrase
Input	NL-desc	WebQuestions	CodeNN,Conala, Geo880,Hearthstone, MTG,NL2Bash,IFTTT	NL-Turk,WikiSQL, Overnight
	LF-desc		Conala,NL2Bash	Invalid
	World-desc	ATIS,SCONE,Spider	Invalid	Invalid

FIGURE 5.2: NL Phase Classification

Note that some of the inputs are incompatible with the processes. Also note that some datasets could be a mix of two processes. We felt that this classification generalized well, and wasn't too abstract either.

5.2 Classification of Existing Datasets

The classification of the datasets are as in Fig. 5.1 and 5.2. Those marked "Invalid" are those combinations which are incompatible. Further analysis is still ongoing work.

Chapter 6

Conclusions

6.1 Observations and Inference

With regards to the NL2Regex parallel, we managed to beat the state-of-the-art by approximately 3%, with the use of more advanced neural architectures. We inferred that given a better dataset for NL2Regex (which at the moment doesn't exist), the margin will only increase. However, looking at the problem on a whole, the best accuracy for a small-domain problem like regex is just 61%, which means that there is a lot of room for improvement before a system can be created for general-use. With regards to the dataset analysis, although there are some promising indicators which do provide some distinction between bad and good quality datasets, we still haven't been able to devise comparable quantitative metrics for which the values will be more useful and differences between these values will be significant. With regards to the data collection, we were able to distinctive and exhaustive generalized classes for data collection.

6.2 Future Work

For improving the existing NL2Regex performance, new neural models can be implemented. Since the field of semantic parsing is being very actively worked upon, we can expect to see more architectures to come up. I still haven't implemented neural syntax networks for the problem, and it will be interesting to see the results. However, this parallel will make more headway once a new, better quality dataset is created for this task. Once that happens, we will not only be able to observe the change in improvement margins between the current state-of-the-art and our models, but we'll be able to do a more effective error analysis which points out the mistakes with the model, and not the dataset. There is a large amount of work remaining for the second parallel, the dataset analysis. A much more fine-grained qualitative analysis is required. A possible analysis could be done by taking a small subsample of each dataset, and

asking workers (from crowdsourcing) to evaluate their quality with a set of carefully decided metrics. More research has to be done on possible quantitative metrics which better represent quality. There needs to be a better way to represent the information at hand, perhaps with different visualizations. Finally, for the data collection method analysis, we still need to make observations about which general methods contribute more to NL variation and quality. We also need to reinforce this by collecting data using all these methods for a particular domain, say regex. We are planning to collect small datasets for regex with different collection methods, and then doing an analysis on these datasets to support our general observations.

6.3 Final Note

The most exciting part of this project is that the more you uncover about the domain of semantic parsing, the more questions you uncover, causing you to meticulously re-evaluate your existing experiments. This makes the problem of semantic parsing a challenging one, and one with great scope to make an impact in the larger problem of language understanding.

Bibliography

- [Alvarez-Melis and Jaakkola, 2016] Alvarez-Melis, D. and Jaakkola, T. S. (2016). Tree-structured decoding with doubly-recurrent neural networks.
- [Artzi and Zettlemoyer, 2013] Artzi, Y. and Zettlemoyer, L. (2013). Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association of Computational Linguistics*, 1:49–62.
- [Berant and Liang, 2014] Berant, J. and Liang, P. (2014). Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1415–1425.
- [Dahl et al., 1994] Dahl, D. A., Bates, M., Brown, M., Fisher, W., Hunicke-Smith, K., Pallett, D., Pao, C., Rudnicky, A., and Shriberg, E. (1994). Expanding the scope of the atis task: The atis-3 corpus. In *Proceedings of the workshop on Human Language Technology*, pages 43–48. Association for Computational Linguistics.
- [Dong and Lapata, 2016] Dong, L. and Lapata, M. (2016). Language to logical form with neural attention. *arXiv preprint arXiv:1601.01280*.
- [Dong and Lapata, 2018] Dong, L. and Lapata, M. (2018). Coarse-to-fine decoding for neural semantic parsing. *arXiv preprint arXiv:1805.04793*.
- [Finegan-Dollak et al., 2018] Finegan-Dollak, C., Kummerfeld, J. K., Zhang, L., Ramanathan, K., Sadasivam, S., Zhang, R., and Radev, D. (2018). Improving text-to-sql evaluation methodology. *arXiv preprint arXiv:1806.09029*.
- [Guu et al., 2017] Guu, K., Pasupat, P., Liu, E. Z., and Liang, P. (2017). From language to programs: Bridging reinforcement learning and maximum marginal likelihood. *arXiv preprint arXiv:1704.07926*.
- [Iyer et al., 2016] Iyer, S., Konstas, I., Cheung, A., and Zettlemoyer, L. (2016). Summarizing source code using a neural attention model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2073–2083.

- [Krishnamurthy et al., 2017] Krishnamurthy, J., Dasigi, P., and Gardner, M. (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.
- [Kushman and Barzilay, 2013] Kushman, N. and Barzilay, R. (2013). Using semantic unification to generate regular expressions from natural language. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 826–836.
- [Kwiatkowski et al., 2010] Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., and Steedman, M. (2010). Inducing probabilistic ccg grammars from logical form with higher-order unification. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 1223–1233. Association for Computational Linguistics.
- [Lin et al., 2018] Lin, X. V., Wang, C., Zettlemoyer, L., and Ernst, M. D. (2018). Nl2bash: A corpus and semantic parser for natural language interface to the linux operating system. *arXiv preprint arXiv:1802.08979*.
- [Ling et al., 2016] Ling, W., Grefenstette, E., Hermann, K. M., Kočiský, T., Senior, A., Wang, F., and Blunsom, P. (2016). Latent predictor networks for code generation. *arXiv preprint arXiv:1603.06744*.
- [Locascio et al., 2016] Locascio, N., Narasimhan, K., DeLeon, E., Kushman, N., and Barzilay, R. (2016). Neural generation of regular expressions from natural language with minimal domain knowledge. *arXiv preprint arXiv:1608.03000*.
- [Long et al., 2016] Long, R., Pasupat, P., and Liang, P. (2016). Simpler context-dependent logical forms via model projections. *arXiv preprint arXiv:1606.05378*.
- [McCann et al., 2018] McCann, B., Keskar, N. S., Xiong, C., and Socher, R. (2018). The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.
- [Oda et al., 2015] Oda, Y., Fudaba, H., Neubig, G., Hata, H., Sakti, S., Toda, T., and Nakamura, S. (2015). Learning to generate pseudo-code from source code using statistical machine translation (t). In *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on*, pages 574–584. IEEE.
- [Pasupat and Liang, 2015] Pasupat, P. and Liang, P. (2015). Compositional semantic parsing on semi-structured tables. *arXiv preprint arXiv:1508.00305*.
- [Quirk et al., 2015] Quirk, C., Mooney, R., and Galley, M. (2015). Language to code: Learning semantic parsers for if-this-then-that recipes. 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)*, volume 1, pages 878–888.
- [Rabinovich et al., 2017] Rabinovich, M., Stern, M., and Klein, D. (2017). Abstract syntax networks for code generation and semantic parsing. *arXiv preprint arXiv:1704.07535*.
- [Shi et al., 2018] Shi, T., Tatwawadi, K., Chakrabarti, K., Mao, Y., Polozov, O., and Chen, W. (2018). Incsql: Training incremental text-to-sql parsers with non-deterministic oracles. *arXiv preprint arXiv:1809.05054*.
- [Tang and Mooney, 2001] Tang, L. R. and Mooney, R. J. (2001). Using multiple clause constructors in inductive logic programming for semantic parsing. In *European Conference on Machine Learning*, pages 466–477. Springer.
- [Wah Wong and J. Mooney, 2007] Wah Wong, Y. and J. Mooney, R. (2007). Learning synchronous grammars for semantic parsing with lambda calculus. volume 960-967.
- [Wang et al., 2014] Wang, A., Kwiatkowski, T., and Zettlemoyer, L. (2014). Morpho-syntactic lexical generalization for ccg semantic parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1284–1295.
- [Wang et al., 2015] Wang, Y., Berant, J., and Liang, P. (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)*, volume 1, pages 1332–1342.
- [Warren and Pereira, 1982] Warren, D. H. D. and Pereira, F. C. N. (1982). An efficient easily adaptable system for interpreting natural language queries. *Comput. Linguist.*, 8(3-4):110–122.
- [Winograd, 1971] Winograd, T. (1971). Procedures as a representation for data in a computer program for understanding natural language. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE PROJECT MAC.
- [Xu et al., 2016] Xu, K., Reddy, S., Feng, Y., Huang, S., and Zhao, D. (2016). Question answering on freebase via relation extraction and textual evidence. *arXiv preprint arXiv:1603.00957*.
- [Yin and Neubig, 2017] Yin, P. and Neubig, G. (2017). A syntactic neural model for general-purpose code generation. *arXiv preprint arXiv:1704.01696*.
- [Yu et al., 2018a] Yu, T., Yasunaga, M., Yang, K., Zhang, R., Wang, D., Li, Z., and Radev, D. (2018a). Syntaxsqlnet: Syntax tree networks for complex and cross-domain text-to-sql task. *arXiv preprint arXiv:1810.05237*.
- [Yu et al., 2018b] Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., Ma, J., Li, I., Yao, Q., Roman, S., et al. (2018b). Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*.

- [Zelle and Mooney, 1996a] Zelle, J. M. and Mooney, R. J. (1996a). Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, pages 1050–1055. AAAI Press.
- [Zelle and Mooney, 1996b] Zelle, J. M. and Mooney, R. J. (1996b). Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055.
- [Zettlemoyer and Collins, 2005] Zettlemoyer, L. S. and Collins, M. (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, UAI'05, pages 658–666, Arlington, Virginia, United States. AUAI Press.
- [Zhai et al., 2017] Zhai, F., Potdar, S., Xiang, B., and Zhou, B. (2017). Neural models for sequence chunking. In *AAAI*, pages 3365–3371.
- [Zhong et al., 2017] Zhong, V., Xiong, C., and Socher, R. (2017). Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.