NORTHWESTERN UNIVERSITY

Question-Answering with Structural Analogy

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Computer Science

By

Maxwell Crouse

EVANSTON, ILLINOIS

September 2021

Abstract

Designing intelligent systems that can answer questions has been an ongoing and active challenge for the artificial intelligence community. In the past, researchers were focused on producing specialized language systems for particular domains and datasets. Such approaches would require deeper-than-ideal amounts of expertise to design, and often necessitated the expensive manual annotation of datasets with logical forms. Modern methods have since shifted to being deep learning-based, which has allowed for effective and flexible question-answering systems that can be constructed in a more hands-off approach.

Both paradigms have their advantages and disadvantages. The earlier systems were far more interpretable as they often involved learning explicit grammar rules. However, they were less performant and had to start from scratch for each new domain to which they were applied. Mainstream deep learning-based methods are very effective, easier to train, and do exhibit some degree of transferability (largely due to their use of techniques like word embeddings), but their internal reasoning processes are opaque, and they generally require a significant amount of data to train on to achieve good performance.

In this thesis, we approach question-answering from an analogical perspective. In particular, we introduce an approach that uses analogy to adapt an existing general-purpose semantic parser to answer questions in novel domains. The adaptation is learned automatically and performs well when given either natural-language question-answer pairs or questions annotated with logical forms. The incorporation of a general-purpose semantic parser allows the system to avoid having to learn from scratch for each new domain to which it is applied, while also making the question-answering task simpler, which allows for better performance and data efficiency.

We demonstrate the effectiveness and generality of our approach by applying it to three different datasets that each require a distinct type of reasoning. We show that the method is competitive with modern neural approaches to question-answering while maintaining interpretability and explainability.

Acknowledgements

First and foremost, I would like to thank my advisor, Ken Forbus. It is easy for me to point to the numerous individual actions and ways in which he helped me grow (e.g., this entire thesis is based on conversations he networked me into early in my career). However, I think what I have appreciated most of all in my time at Northwestern is the culture that Ken fosters in the QRG lab.

In the lab, we learn to pay attention to what the methods *du jour* of the research community are, but to never blindly follow a trend just because it's popular. As a graduate student, it can be easy to lose sight of what meaningful science is when in the pursuit of the metrics by which we are often judged (e.g., conference papers and citations). These metrics are, of course, important to keep in mind, but they should never be the sole reason for tackling a particular problem. In the QRG lab, we are taught to look for interesting problems to solve and to solve such problems in a principled, general-purpose way. This is more challenging than the all-too-common practice of taking some existing neural model and trivially modifying it such that it achieves state-of-the-art on *some* dataset. But, as Ken has taught me, "problems worthy of attack, prove their worth by fighting back."

I would like to thank the Air Force Office of Scientific Research, the Office of Naval Research, and Northwestern University for their generous funding and support of my work through the years.

A graduate student is shaped by their mentors, and for that reason I would like to thank Michael Witbrock and Achille Fokoue for their mentorship at IBM Research. The two internships I had working with them were some of my most valuable learning experiences as a graduate student, and I cannot thank them enough for providing those opportunities for me. Similarly,

Ibrahim Abdelaziz and Maria Chang at IBM and Tom Hinrichs at Northwestern were wonderful to work with, and I appreciate their patience with me as I bounced no fewer than ten million ideas off of them.

I would like to thank Ian Horswill for his guidance on my thesis and for serving on my thesis committee. The class I took from Ian on Logic Programming was one of the most entertaining courses I have taken, and it served to partially inspire some of the methods in this thesis.

I would like to thank CJ McFate, Irina Rabkina, Joe Blass, Chen Liang, Constantine Nakos, and Kezhen Chen for the frequent debates and interesting conversations in the early years that made starting out in the QRG lab so entertaining. To the newer graduate students: Sam Hill, Will Hancock, Danilo Ribeiro, Cathy Lin, and Taylor Olson, thank you for bringing new insights and ideas into the lab. I would particularly like to thank my frequent co-authors CJ and Constantine for making the papers we worked on together so much better, and my cohort-buddy Irina Rabkina for going through the struggles of graduate student life with me.

To my friends Aabesh De, Prathik Kini, and Jonathan Leganza, thank you for keeping life constantly entertaining through this degree with all of the gaming, debates, poker, and time together. To my partner Carolyn Wahlen, thank you for being my rock through the past few years, keeping me steady and moving forward through both the good times and the bad. I could not be more excited to see what lies ahead for the two of us together.

To Mimi, Popi, Grandpa, Granddad, and Grandma, though you all could not be here to see this, I am sure you all would be proud. The memories I have with each of you I cherish every day.

To my sister Claire, thank you for being the competitive force through my life that always drove me to be better. I always looked to you to see what person I should strive to be.

This thesis is dedicated to my parents, Jane and Eric Crouse. They inspired my love of science from an early age while also encouraging me to live a well-balanced and well-rounded life beyond just academic performance. They are my role models, and I could not ask for better or more supportive parents.

Table of Contents

Table of Figures

Table of Tables

1 Introduction

Creating systems that can learn to answer natural language questions has been a longstanding challenge for artificial intelligence research. The most recent approaches tackling this challenge have focused on engineering systems via machine learning over massive amounts of data, often using neural networks. With enough data and craft in the dataset construction and training process, these systems can produce good results. However, such methods are generally uninspectable (a problem made more concerning by their susceptibility to adversarial attacks, e.g., (Jia and Liang 2017, Marcus 2018), and are known to require tremendous amounts of data to successfully train. Symbolic approaches seem like they would handle these issues, since they are highly inspectable and have been demonstrated to yield data efficiency for many tasks (Chen et al., 2019). Unfortunately, their application to natural language question answering is far from a simple problem. The messiness, breadth, and ambiguity of language pose significant challenges to purely symbolic approaches that are typically considered inflexible in how they reason and what they can reason over. So where does this leave us? What would an approach that takes the strengths of both machine learning-based techniques (e.g., their flexibility in handling a breadth of language) and symbolic methods (e.g., their innate inspectability) look like?

In this thesis, I propose a new approach to question-answering over structured knowledge that is driven by analogy, which I refer to as AQA (*analogical question-answering*). The design of AQA is motivated by the concerns above, namely that a question-answering system should be flexible and powerful enough to well handle question-answering over varied natural language while maintaining inspectability into both what and how it learns. Analogy provides AQA the

flexibility needed to handle the diverse language it encounters, and its purely symbolic nature gives it the inspectability missing from deep learning-based approaches.

In addition to flexibility and transparency, AQA also exhibits remarkable data efficiency. A key part of how AQA achieves efficiency is that it approaches answering a question more compositionally than standard machine learning-based methods. In particular, most prior methods would view question-answering as a one-step process where text is mapped directly to a logical form. In contrast, AQA considers question-answering as a two-step process. First, a domain-general, broad-coverage semantic parser is applied to a given question. Then, the outputs of the parser are adapted via analogy to domain-specific logical forms needed to answer the question at hand. The general-purpose semantic parser is carried across domains, meaning that AQA avoids starting from scratch for each new task to which it is applied. Learning how to adapt an existing domain-general semantic parser to domain-specific question-answering tasks is a simpler problem than learning to map from language directly to task-specific logical forms, and thus AQA is capable of far more efficient learning than traditional machine learning methods.

In this work, the use of analogy is ubiquitous. During training, AQA uses analogy to identify the most salient subparts of a question that provide evidence for the use of a particular logical form. When AQA is applied to test questions, it uses analogy to propose candidate solutions that have been adapted from training examples. The centrality of analogy can be ascribed to a key fundamental assumption; namely, that questions can be understood (and answered) through analogies with solved questions. Going even further, AQA was designed with the assumption that the Structure Mapping Theory (SMT) of analogy (Gentner, 1983) provides a useful set of rules for constraining how questions should be compared against one another for the purposes of question-

answering. But is analogy the right tool for the job? In this thesis, we will show that in many cases it is, i.e., that analogy (in the form of SMT) is a powerful enough tool to successfully handle question-answering for a variety of domains and question types.

1.1 Claims and Contributions

The claims of this dissertation are as follows:

1. It is possible to design an approach to question-answering with analogy as its core operation that performs effectively on a variety of question-answering domains and tasks.
2. Adapting an existing domain-general semantic parser to domain-specific question-answering tasks leads to far more efficient learning on those tasks than would starting from scratch.
3. Combining machine learning (in the form of inductive logic programming) with traditional symbolic reasoning methods can produce an approach to semantic parsing that is completely transparent as to what it is learns and how it applies what it learns, that also performs competitively with black-box neural methods.

The contributions of this dissertation are as follows:

1. It provides a method, Analogical Question-Answering (AQA), for adapting a general-purpose semantic parser to question-answering tasks in multiple domains that each require different types of reasoning.
2. It demonstrates that the method can perform on both annotated and unannotated question-answering datasets.
3. It characterizes the conditions that allow AQA to learn in data-sparse situations.

1.2 Organization

Section 2 of this thesis provides terminology definitions for the basic operations and algorithms seen throughout the thesis. It also includes a detailed description of analogy as it relates to this thesis. Section 3 provides the relevant background needed to understand the rest of the thesis. Summaries of connection subgraphs, comparative analysis (Weld, 1990) and Qualitative Process Theory (Forbus, 1984) are given, along with aspects of the domain-general semantic parsers that AQA relies upon. Section 4 provides a high-level overview of AQA, to provide the context for Section 5 and 6. Section 5 describes how the adaptation between a domain-general semantic parser and domain-specific logical forms is learned, i.e., both what AQA learns and how AQA learns. Section 6 describes how the adaptation is carried out to answer novel questions, i.e., how AQA applies what has been learned. Sections 7, 8, and 9 provide experimental results for AQA on three different question-answering domains. The first domain is GeoQuery (Zelle and Mooney, 1996), a standard benchmark for question-answering approaches. The second domain, introduced for this thesis, consists of science test questions given to elementary school students. The last domain is QuaRel (Tafjord et al., 2019), which is a larger dataset designed to test question-answering approaches for an important subset of qualitative reasoning. Section 10 provides related work for previous question-answering approaches. Lastly, Section 11 summarizes the claims and contributions of this thesis, ending with conclusions, open questions, and future work.

2 Preliminaries

2.1 Symbolic Representations

AQA reasons over sets of logical expressions. In this thesis, logical expressions are constructed from symbols drawn from the OpenCyc ontology (Matuszek et al, 2006). Expressions may contain constants, variables, functions, collections, relations, and other expressions. Abstractly, constants represent objects drawn from some universe. Typically, constants will be the entities of questions given to AQA, e.g., people or places. Functions are mappings from a tuple of objects to another object. Collections are used to represent concepts, and relations are used to represent relationships. Collections and relations are unary and binary predicates, respectively, that take arguments and express a truth assertion. As an example, we write (bordersOn (TerritoryFn Indiana-State) (TerritoryFn Michigan-State)) to indicate that the relation bordersOn holds between the territories of Indiana and Michigan, with the territories denoted by the function TerritoryFn separately applied to both Indiana-State and Michigan-State. Collections are most often found in isa statements, e.g., (isa Indiana-State State-UnitedStates), which denotes that Indiana is an instance of a U.S. state. This isa statement is equivalent to (State-UnitedStates Indiana-State). We use camel case notation when writing collections, functions, and relations (with upper case initially for collections and functions and lower case initially for relations).

Throughout this thesis, the representations being manipulated will be given as lisp-style s-expressions. When describing algorithms, we use infix notation with standard symbols for representing logical connectives. This is done for both stylistic reasons (i.e., to use notation

consistent with prior work in this space) and to maintain a separation between the logical operations used in each algorithm from the representations being reasoned over.

As they will be used in this thesis, logical connectives take s-expressions as arguments and maintain their usual definitions and notations (e.g., "¬" for negation, "∧" for conjunction, "→" for implication, etc.). We will mostly deal with negation and conjunction, where a typical setting will be describing operations over sets of logical expressions subject to pairwise "nogoods", which are inconsistent combinations of expressions (e.g., $\neg (s_1 \wedge s_2)$ where s_1 and s_2 are logical expressions). Unless otherwise specified, a set of logical expressions will be given an uppercase italicized variable name and its constituent elements will be lowercase italicized, e.g., $S = \{s_1, s_2, s_3\}$.

A key operation within AQA is the determination of how to fit the entities of a test question to those seen in training questions. For instance, if asked "How big is Indiana?", it may analogize to the training question "How big is Texas?". To do this, it determines how entities from the test question ("Indiana") match with entities of the training question ("Texas"). These matchable entities must be explicitly separated from other entities found in the logical expressions given to AQA. To see why, consider one of the logical expressions used to represent "How big is Texas": (possessiveRelation Texas-State (HighAmountFn Size)). Clearly, when matching a statement like this to an expression specific to "How big is Indiana?", the only entity we would want to be substitutable is Texas-State and not Size or (HighAmountFn Size). To denote the set of matchable entities, we define a function *m-ents*(*S*), which returns the set of such matchable entities from a set *S* of logical expressions.

For sets of expressions *S* and *T*, we use σ to denote a substitution between *m-ents*(*S*) and *m-ents*(*T*). A substitution σ between *S* and *T* is a set of pairs $\sigma \subseteq$ *m-ents*(*S*)×*m-ents*(*T*), where each

pair is of the form $<s_{ent}, t_{ent}>$, with $s_{ent} \in m\text{-}ents(S)$ and $t_{ent} \in m\text{-}ents(T)$. We write $S\sigma$ (or $\sigma(S)$ if otherwise ambiguous) to denote the expression that results from simultaneously substituting every occurrence of a left-hand-side entity in σ with its corresponding right-hand-side entity. In our previous example, we may have σ = { <Texas-State, Indiana-State> }, with the result of applying σ to our expression then being (possessiveRelation Indiana-State (HighAmountFn Size)).

2.2 The Structure-Mapping Theory of Analogy

The Structure-Mapping Theory (Gentner, 1983) of analogy is central to AQA's design. From an operational perspective, it drives the key pattern matching operation within AQA that allows it to reason about new test questions with respect to older training questions. It also has more subtle influence in heuristics that AQA uses during both training and testing, as described in Sections 5 and 6 before going into the specifics for how analogy is applied with AQA, we will first give a high-level overview of SMT.

Structure-Mapping Theory (SMT) centers around the structural alignment of logical expressions (see Figure 1). Structural alignment is the process of producing a mapping between two relational representations (referred to as the base and target). Each mapping is a triple <*M*, *C*, *S*>, where *M* is a set of correspondences between the base and target, *C* is a set of candidate inferences (i.e., inferences about the target that can be made from the structure of the base), and *S* is a structural evaluation score that measures the quality of *M*. In this work, we will only consider the set of correspondences *M* and the score *S*.

Correspondences are pairs of elements between the base and target (i.e., expressions or entities) that are identified as matching with one another. While entities can be matched together

irrespective of their labels, there are more rigorous criteria for matching expressions. SMT asserts that matches should satisfy the following properties:

1. One-to-One: Each element of the base and target can be a part of at most one correspondence.
2. Parallel Connectivity: Two expressions can be in a correspondence with each other only if their arguments are also in correspondences with each other.
3. Strict / Tiered Identicality: Relations of expressions in a correspondence must match identically by default, however if two relations are sufficiently close then they may be allowed to match (where closeness is typically defined by whether or not they share a common superordinate in a predicate hierarchy). Functions need not be identical if their correspondence would support structural connectivity.
4. Systematicity: Preference should be given to mappings with more deeply nested expressions.

To understand these properties, we use a classic analogy (see Figure 1) from (Gentner, 1983) which draws an analogy between the Solar System and the Rutherford model of the atom. A set of correspondences *M* between the base (Solar System) and target (Rutherford atom) is a set of pairs of elements from both sets, e.g., {<[1], [8]>, <[2], [9]>}. The one-to-one constraint restricts each element to be a member of at most one correspondence. Thus, if <[7], [15]> was a member of *M*, then <[7], [16]> could not be added to *M*. Parallel connectivity enforces correspondence between arguments if the parents are in correspondence. In this example, if <[7], [15]> was a member of *M*, then both <[3], [10]> and <[4], [11]> would need to be members of *M*. Parallel connectivity also respects argument order when dealing with ordered relations. Tiered

[1] nucleus [2] electron [3] (mass [1]) [4] (mass [2]) [5] (attracts [1] [2]) [6] (revolves-around [2] [1]) [7] (greater [3] [4])	[8] sun [9] planet [10] (mass [8]) [11] (mass [9]) [12] (temperature [8]) [13] (temperature [9]) [14] (revolves-around [9] [8]) [15] (greater [10] [11]) [16] (greater [12] [13]) [17] (attracts [9] [8]) [18] (causes (and [15] [17]) [14]) [19] (yellow [8])

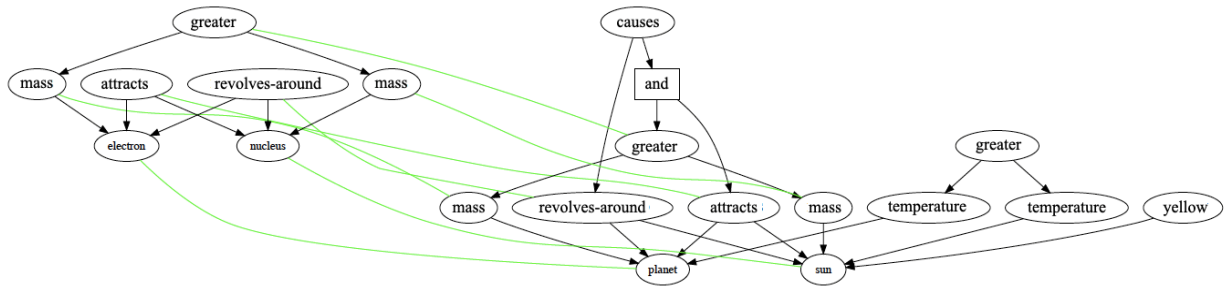

Figure 1. An example of an SMT-satisfying match between two sets of expressions, with green edges representing correspondences

identicality is not relevant in this example; however, if [10] used the label WEIGHT instead of MASS, tiered identicality could be used to match [3] and [10], since such a correspondence would allow for a match between their parents. The last property, systematicity, results in larger correspondence sets being preferred over smaller ones. Note that the singleton set {<[1], [8]>} satisfies SMT's constraints, but it is clearly not useful by itself. Systematicity captures the natural preference for larger, more interesting matches.

It is well known that, in the general case, finding an SMT-satisfying mapping between two expressions is an NP-Hard problem (Veale and Keane, 1997). The most effective computational method for solving the general case has thus far been the Structure-Mapping Engine (SME) (Falkenheiner et al., 1989; Forbus et al., 2017). It solves the SMT problem with a middle-out matching procedure that operates in roughly quadratic time (the most complex step of SME

operates in time $O(n^2 \log(n))$, with n being the number of elements in the larger of the two representations being matched). Though not guaranteed to yield the most optimal mapping, it has empirically been observed to produce mappings that are optimal or near optimal in many benchmark tests of analogical reasoning.

Recall that for sets of logical expressions S and T, a substitution σ is a mapping between *m-ents*(S) and *m-ents*(T). Within AQA, SMT is used as a hard constraint on which substitutions are allowable. Letting M be the set of correspondences in an SMT-satisfying mapping between S and T, the substitution σ will be a subset of M that involves only entities from *m-ents*(S) and *m-ents*(T). As mentioned in the previous subsection, not all entities from S and T are matchable, and thus we restrict M to only allow non-identical matches for those correspondences that form σ, i.e., only those involving entities from *m-ents*(S) and *m-ents*(T). We further restrict M to be from an SMT-satisfying mapping between full logical expressions, rather than between subexpressions as is shown in Figure 1 (i.e., for two subexpressions to match, their full parent expressions must match as well). In the context of this work, these added constraints mean that the full power of SME in most cases is not needed. In particular, without arbitrary entity and subgraph matching, finding a constrained SMT-satisfying mapping between two expressions can be done in linear time. For the remainder of this thesis, a substitution σ will be assumed to meet the criteria listed above.

2.3 Finding Substitutions Between Individual Expressions

In this section, we will detail the procedure used to compute the substitution for the matchable entities of two individual expressions. For our purposes, the matching of two expressions to produce an entity substitution can be performed by a simple linear time algorithm that walks down

```
   With variables
        B = base s-expression
        T = target s-expression
        binds = {}
   With functions
        is-list? = returns True if argument is list
        car = returns first element of list
        cdr = returns the rest of the list beyond the first element
        zip = returns list of pairs where each of the i = 1,...,n elements of
              both lists are paired together
        either-imm-genls? =   returns True if one argument is an immediate genls
                              of the other
        m-ents = returns the matchable entities of its input argument expression

1. Define Match(B, T, binds):
2.      If is-list?(B) and is-list?(T):
3.              If car(B) == car(T) == isa:
4.                      B-col = car(cdr(cdr(B)))
5.                      B-ent = car(cdr(B))
6.                      T-col = car(cdr(cdr(T)))
7.                      T-ent = car(cdr(T))
8.                      If B-col == T-col or either-imm-genls?(B-col, T-col):
9.                              return Match(B-ent, T-ent, binds)
10.                     Else:
11.                             return False
12.             If car(B) == car(T) and length(B) == length(T):
13.                     for b_el, t_el in zip(cdr(B), cdr(T)):
14.                                     binds = Match(b_el, t_el, binds)
15.                             If not binds:
16.                                     return False
17.                     return binds
18.             Else:
19.                     return False
20.     If not is-list?(B) and not is-list?(T):
21.             If B in binds and binds[B] == T and binds[T] == B:
22.                     return binds
23.             If B in binds or T in binds:
24.                     return False
25.             If B in m-ents(B) and T in m-ents(T):
26.                     binds[B] = T
27.                     binds[T] = B
38.                     return binds
29.             If B == T:
30.                     return binds
31.             Else:
32.                     return False
33.     Else:
34.             return False
```

Figure 2. Expression matching algorithm

two expressions simultaneously and checks for variable / symbol mismatches as it goes along. We present this algorithm, `Match`, in Figure 2, where the algorithm given returns a substitution

between the matchable entities of two expressions or indicates a failure to match. AQA uses this algorithm whenever two individual expressions are matched together. Important to note is the condition specifying isa matches (line 3). In particular, when matching two isa statements, the standard strict equality check for symbols is relaxed to allow collections (i.e., the third argument to an isa) to match if one is an immediate superordinate to the other (i.e., a direct genls statement holds between the two collections). This is a simplified form of minimal ascension (Falkenhainer, 1988), a common way to implement tiered identicality.

3 Background

3.1 Semantic Parsing

The work in this dissertation uses the Companion NLU semantic parser (Tomai and Forbus, 2009). Companion NLU is a bottom-up rule-based chart parser that uses a feature-based grammar and Baker et al's (1998) FrameNet. FrameNet ties words to a semantic schema and describes how semantic roles are bound to arguments in syntactic patterns. To illustrate how CNLU operates, we will use as our running example the simple sentence "The snow changes to water."

To build up a semantic parse for its input sentence, CNLU first tokenizes the input. For our example sentence, this would produce the list (the snow changes to water punc-period). Each token is assigned a discourse variable, which is a unique identifier specific to each token (e.g., snow18587 for the token snow and change18419 for the token changes). The tokens are matched against entries in the Nulex lexicon (McFate and Forbus, 2011), which maintains information about parts of speech and other syntactic features. Figure 3 shows the two entries for the word "changes". As shown in the figure, morphology is accounted for through separate entries for each form of the word (e.g., for different parts of speech, different tenses, etc.).

```
(definitionInDictionary Nulex changes Change-TheWord Noun
  (TheSet (root change) (agr (TheSet 3p)) (countable +)))

(definitionInDictionary Nulex changes Change-TheWord Verb
  (TheSet (root change) (vform (TheSet pres)) (agr (TheSet 3s))
    (subcat (TheSet adv-middle np np-pp np-pp-agent np-pp-pp np-pp-theme pp pp-pp))))
```

Figure 3. Nulex entries for "changes"

Because CNLU is a bottom-up parser, it begins its parse with terminal constituents (i.e., words or groups of words treated as a single unit) at the leaves of a parse tree. The different lexical entries form the basis for terminal constituents. Phrasal constituents are formed by taking sequences of lower-level constituents (either terminal or other phrasal constituents) and matching them against grammar rules. Importantly, as phrasal constituents are being built, they maintain key syntactic information (e.g., the subject and object of a verb along with their associated discourse variables) that will be used to build up possible semantic representations of their underlying phrase.

Each semantic representation for a word or phrase originates from a terminal constituent. The mappings between terminal constituents and semantic meaning representation are stored as Semtrans statements. In Figure 4, we provide an example of a verb Semtrans statement for the word "change". In the figure, the first and second arguments indicate the underlying word / phrase (in this case, only a single word). The third argument indicates the constituent's associated concept from the OpenCyc ontology (Matuszek et al., 2006), here being an IntrinsicStateChangeEvent. The fourth argument provides the source FrameNet frame from which all role relations for this Semtrans are derived (e.g., FN_Undergo_change). The bindingTemplate argument provides a discrimination tree which specifies triple consisting of grammatical function (e.g., :OBJECT), part of speech, and individual role relation / frame element (e.g., fromState). The groupPatterns argument specifies a set of *valence patterns*, which are valid combinations of syntactic patterns that are used to instantiate the frame-derived semantic representations from a phrasal constituent. These are stored as sets of offsets into the discrimination tree, where each offset is considered a path in the tree. For instance, the list (0 0 1) maps into the bindingTemplate discrimination tree to return the triple (:SUBJECT NP objectActedOn). The valence pattern can be combined with the

```
(FNVerbSemtrans (TheList) Change-TheWord
 (and (isa :ACTION IntrinsicStateChangeEvent))
 (frame FN_Undergo_change)
 (bindingTemplate
  (TheList (:SUBJECT (NP fe_attribute objectActedOn fromState))
   (:OBJECT (NP fe_attribute))
   (:OBLIQUE-OBJECT
    ((PPFn with) (InverseBinaryPredicateFn causes-Underspecified))
    ((PPFn to) toState)
    ((PPFn in) toState)
    ((PPFn from) fromState)
    (AVP fe_degree frequencyOfEventType mannerOfAction)
    ((PPFn before) mannerOfAction)
    (Sub situationConstituents temporallyIntersects)
    ((PPFn during) temporallyIntersects)
    ((PPFn over) temporallyIntersects)
    ((PPFn throughout) temporallyIntersects))
   (:GAP (2nd objectActedOn) (DNI toState fromState)
       (INI fe_attribute toState toState fromState fromState
           fe_value_range)) (:NOUN (N objectActedOn))))
 (groupPatterns
  (TheSet ((4 0 0)) ((0 0 0)) ((0 0 1) (2 5 0)) ((0 0 1) (2 4 2))
   ((0 0 1) (2 4 0)) ((0 0 0) (2 4 2)) ((0 0 1) (2 6 1))
   ((0 0 1) (2 1 0) (2 3 0)) ((0 0 1) (2 4 0) (2 8 0))
   ((0 0 2) (2 2 0) (2 6 0)) ((0 0 1) (2 0 0) (2 4 0))
   ((0 0 1) (1 0 0) (2 7 0)) ((0 0 1) (0 0 1) (2 4 1) (2 4 2) (2 9 0)))))
```

Figure 4. Semtrans for verb form of "change"

meaning of the terminal constituent to yield a semantic representation, e.g., a phrasal constituent that matches with the valence pattern ((0 0 1)) would result in (and (isa :ACTION IntrinsicStateChangeEvent) (objectActedOn :ACTION :SUBJECT)). Each phrasal constituent maintains syntactic information that can be used to instantiate these abstract forms. In this case, :ACTION would be bound to the discourse variable change18419 (associated with the token changes) and :SUBJECT would be bound to the discourse variable snow18587 (associated with the

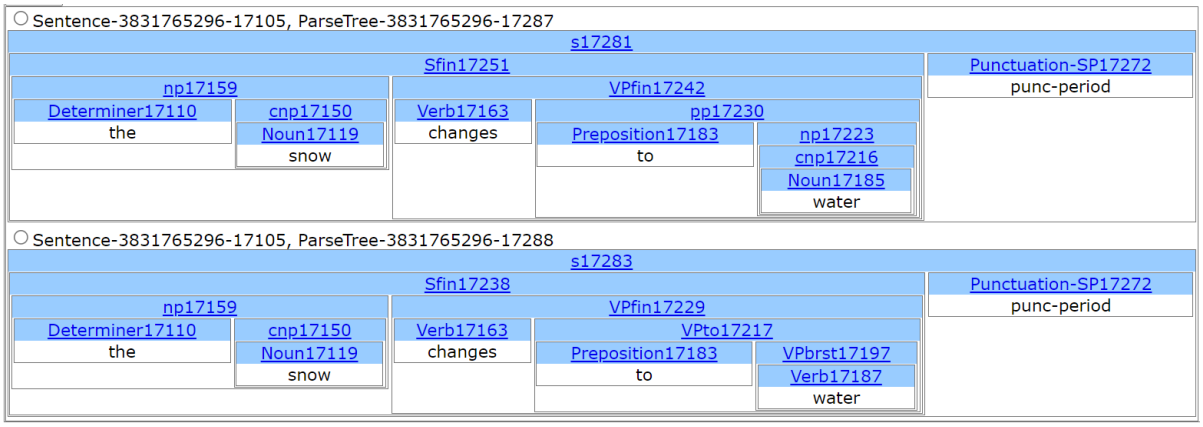

Figure 5. Alternative syntactic parse trees for "The snow changes to water."

token snow), leading to the instantiated representation (and (isa change18419 IntrinsicStateChangeEvent) (objectActedOn change18419 snow18587)).

CNLU continues to build constituents until no more can be generated, at which point it returns the highest-level constituents that span the most input tokens (in our case, returning sentence-level constituents that cover the entirety of the input). These highest-level constituents are the possible syntactic parse trees for the input, as shown in Figure 5.

Figure 6 shows the semantic representation produced for the example input. The set of semantic representations associated with each word is referred to as a *choice set*. In the context of this work, a choice set is a set of mutually exclusive alternative meanings for a particular terminal constituent. These alternative meanings are referred to as *choices*, and the mutual exclusion constraint prohibits more than one choice per choice set from being believed to be true at a time. Choice sets arise when there are multiple applicable Semtrans statements for a given token (e.g., "water" as an instance of (LiquidFn Water) or as an instance of BodyOfWater). In addition to constraints between the members of an individual choice set, there can also be constraints between

1. “snow”
    1a. (isa snow18587 SnowMob)
    1b. (isa snow18587 SnowProcess)

2. “changes”
    2a. (and (isa change18631 (CausingFn IntrinsicStateChangeEvent))
        (fe_final_value change18631 water18653)
        (doneBy change18631 snow18587))
    2b. (and (isa change18631 IntrinsicStateChangeEvent)
        (objectActedOn change18631 snow18587))

3. “to”
    3a. (to-Generic change18631 water18653)
    3b. (toLocation change18631 water18653)

4. “water”
    4a. (isa water18653 (LiquidFn Water))
    4b. (isa water18653 BodyOfWater)
    4c. (isa water18655 Irrigation)

Figure 6. An example of choice sets for “The snow changes to water”

the choices of distinct choice sets. For instance, there are constraints between choices that originated from different parse trees (in our example, all choices derived from the first parse tree conflict with all choices derived from the second parse tree).

CNLU and its outputs have many favorable qualities for our purposes. First, as CNLU defers ambiguity resolution (through the use of choice sets), the decision for how best to interpret a sentence is always left to AQA. This is ideal, as disambiguation can then be incorporated into the objectives explicitly optimized by AQA for question-answering. Second, FrameNet (and thus CNLU) has a reasonably sized set of frames and frame elements (around 1,200 combined) with which it can represent language (for reference, a popular alternative such as AMR has only around 100 relations (Banarescu et al. 2013)). We thus believe that FrameNet achieves a reasonable level

of resolution, providing a level of representation that is neither too general nor too specific. Last, CNLU maintains the actual spans in text from which a particular semantic choice was generated. This is not the case for the current best neural-based domain general semantic parsers that map to AMR (Bevilacqua et al. 2021; Zhang et al. 2019a; Zhang et al. 2019b). By leveraging the link between a semantic choice and text, AQA can produce natural language explanations for its outputs (thus making it more understandable to a novice user).

3.2 Connection Subgraphs

Connection subgraphs have a long history in computer science (e.g., see (Kowalski, 1975) for an early example). Generally, they are considered a set of paths that each connect two nodes in some larger graph. Their use in this thesis was most motivated by the work of Faloutsos, McCurley, and Tomkins (2004). In their work, the connection subgraph of two nodes is a subgraph that maintains only the most relevant paths connecting the two nodes in a larger graph. For instance, in a social network the connection subgraph between two people may include their shared friends or family, but may exclude extremely common relationships (e.g., that they both follow a famous celebrity). AQA uses connection subgraphs to connect two entities in a knowledge base. They provide heuristics for selecting semantic choices, as well as forming the basis for a procedure that allows for the self-annotation of training data.

3.3 Qualitative Process Theory

Qualitative Process Theory (Forbus, 1984) formalizes continuous processes as the mechanism underlying continuous change. The direct effects of a process (e.g., liquid flow into a tub) are called *direct influence*s (represented as i+ and i-), and their indirect effects are called *indirect*

```
1. (isa ?self NaiveMeltingProcess)
2. (mfTypeParticipant NaiveMeltingProcess ?thing-melting SolidTangibleThing focusOf)
3. (mfTypeParticipant NaiveMeltingProcess ?sub ChemicalCompoundTypeByChemicalSpecies substOf)
4. (mfTypeParticipantConstraint NaiveMeltingProcess (substanceOfType ?thing-melting ?sub))
5. (mfTypeParticipantConstraint NaiveMeltingProcess (relationAllInstance freezingPoint ?sub ?m-temp))
6. (mfTypeCondition NaiveMeltingProcess (qGreaterThan (TemperatureFn ?thing-melting) ?m-temp))
7. (mfTypeConsequence NaiveMeltingProcess (qGreaterThan (LiquidGenerationRateFn ?self) 0))
8. (mfTypeConsequence NaiveMeltingProcess
     (qprop (LiquidGenerationRateFn ?self) (TemperatureFn ?thing-melting)))
9. (mfTypeConsequence NaiveMeltingProcess
     (i- (AmountOfFn ?sub Solid-StateOfMatter ?thing-melting) (LiquidGenerationRateFn ?self)))
10. (mfTypeConsequence NaiveMeltingProcess
     (i+ (AmountOfFn ?sub Liquid-StateOfMatter ?thing-melting) (LiquidGenerationRateFn ?self)))
```

Figure 7. A qualitative model of melting

influences (represented as qprop / qprop-), e.g., the level of the water in the tub. *Model fragments* are compositional schemas that define types of entities and relationships in the world. They have *participants* which are related by the model fragment, constraints among participants that determine when the model fragment should be considered, and conditions of activation. When a model fragment is active, its consequences hold. These are frequently influences, though other relationships can be consequences as well. Consider the model fragment in Figure 4, which describes the process of melting. Lines 2 and 3 define the participants, entities that are instances of the collections SolidTangibleThing and ChemicalCompoundTypeByChemicalSpecies. They play the role of focusOf and substOf respectively (the thing melting and the chemical substance of the thing melting) in an instantiated model fragment. Lines 4 and 5 define constraints, enforcing that this model fragment will only be considered when the thing melting is composed of a chemical substance for which one knows the freezing point. Line 6 defines the condition that must hold for the model fragment to be considered active, which is that the thing melting must have a temperature greater than that of its freezing point. The remaining lines provide the consequences,

including things like an indirect influence that holds between the amount of liquid accumulating in the object melting and the rate of liquid generation of the melting process.

3.4 Comparative Analysis

Comparative analysis (Weld, 1990) uses qualitative representations to ascertain the causal consequences of differences. These differences may be a hypothetical change to a system (e.g., using a stiffer spring in a mechanical design) or the differences between two physical systems (e.g., the difference in the periods of two pendulums based on differences in their lengths). An example from the QuaRel (Tafjord et al., 2019) dataset (one of the three datasets explored in this thesis) is shown in Figure 5. In the example, because distance is inversely proportional to resistance, the change in the distance of rolling can be attributed to the added resistance of the carpet as compared to the wood floor.

Two techniques have been developed to solve comparative analysis problems. The first technique is differential qualitative analysis (DQA) (Weld, 1990), which uses a set of rules to compute relative values across the descriptions of two systems, based on assumed differences between them. For example, the duration rule says that if a rate is lower in one system versus another, then the time required to reach a limit point will be longer in that system. In (Weld, 1990) the problem of aligning the two systems to be compared was simplified by only considering

"Alan noticed that his toy car rolls further on a wood floor than on a thick carpet. This suggests that:"

(A) "The carpet has more resistance"
(B) "The floor has more resistance"

Figure 8. A multiple-choice comparative analysis question from the QuaRel dataset

changes in parameters, but as (Klenk et al., 2005) showed, the same techniques can be generalized by using analogical mappings to automatically align two systems under analysis.

The second technique is exaggeration (Weld, 1990), which uses qualitative simulation with extreme values substituted to reason about perturbations. Given a proposed perturbation to a system, exaggeration first transforms the problem with the perturbation being an extreme value, i.e., infinite if increased, zero if decreased. For example, when reasoning through how a car would roll on a carpet if the carpet had more resistance, the reformulated model would have the resistance of the carpet being infinite, and thus the rolling speed would decrease to zero. This result would then be rescaled, to respond that there would be a decrease in speed.

3.5 Knowledge Base and Ontology

This work leverages NextKB as its underlying knowledge base (KB). NextKB integrates the OpenCyc ontology (Matuszek et al., 2006) with FrameNet (Baker et al., 1998) and richer support for qualitative and analogical reasoning. This thesis draws heavily upon the large set of *structural relations* present in the OpenCyc ontology. These relations define structural, meta-level relationships between entities, collections, predicates, and functions in the knowledge base. Important examples from the OpenCyc ontology in NextKB include type argument constraints (e.g., (arg1Isa cityInState City)), instance relations (e.g., (isa Indiana-State State-UnitedStates)), type hierarchies (e.g., (genls State-UnitedStates State-Geopolitical)), and predicate hierarchies (e.g., (genlsPreds majorCityInState cityInState)).

Knowledge in the NextKB knowledge base uses the Cyc concept of *microtheories* (Guha, 1991). A microtheory is a collection of facts that is treated as a unit for reasoning. For example, we use a Geobase microtheory which contains geographical knowledge of the US. Microtheories

enable a knowledge base to contain multiple perspectives which, taken together, might be contradictory (e.g., Newtonian versus Relativistic mechanics). Microtheories can inherit from other microtheories through the genlMt relationship. Every reasoning operation is done with respect to some microtheory and those it inherits from, which constitute its *logical environment*. Each microtheory on its own is assumed to be consistent.

3.6 Ontological Similarity

We often use the conceptual similarity of logical expressions as a heuristic during reasoning. Conceptual similarity is determined as in (Crouse et al. 2018; Ribeiro et al., 2019; Wilson et al., 2019), which we briefly describe here. At a high level, the similarity of two expressions is given as a function of the connectedness of their constituent elements in our underlying knowledge base. Given expressions E_A and E_B, the constituent elements of both expressions (i.e., predicates, functions, collections, and entities) are extracted into sets A and B. Following that, a connection subgraph (Faloutsos et al., 2004) is found between each pair of elements in the cross product of A and B (i.e., each p in $A \times B$) through the set of all structural facts in the knowledge base.

A structural fact is treated as a labeled hyperedge between two or more concepts in the KB (where the structural fact's predicate is the edge's label). In this context, a connection subgraph then corresponds to a set of paths that connect two entities, where each path is constructed from only structural facts. We compute the ontological similarity score between a pair of expressions as the sum of weights between each of their constituent elements. Let $\Pi = \{P_1, \ldots, P_n\}$ be the set of paths between two nodes X and Y with $P_i = (v_{i,1}, \ldots, v_{i,m})$ being the vertices in each of these paths. Let $deg(v_{i,j})$ be the out-degree of a vertex $v_{i,j}$.The ontological connection weight (ocw) between nodes X and Y is then

$$ocw(X,Y) = \sum_{P_i \in \Pi} \sum_{v_{i,j} \in P_i} \frac{1}{deg\ (v_{i,j})}$$

The intuition behind this equation is to favor those pairs of nodes which are connected through less dense regions of the knowledge base (with the assumption that less dense regions of the KB will yield less common, more informative connections between entities).

3.6 Analogical Processing

AQA uses the Structure Mapping Engine (SME) (Forbus et al, 2017) during training. SME is a computational implementation of Gentner's (1983) Structure Mapping Theory that aligns hierarchical structured representations (predicate calculus) according to the principles of SMT. It provides a means of determining how entities can be aligned between two sets of logical expressions.

4 Analogical Question-Answering Overview

By design, the NLU system's outputs are task independent. That is, they provide a translation of an input utterance into logical forms, but those logical forms are sometimes too high-level for domain specific reasoning tasks. Analogical Q/A provides a data-efficient and inspectable means of learning how to adapt such outputs to the representations needed for specific tasks. Training produces *query cases* (QCs), which are rule-like constructs that treat semantic choices as antecedents and task specific logical forms as consequents. To apply a query case, the semantics for a question are aligned to the query case's antecedents to find an SMT-satisfying substitution which can be used to produce an instantiation of its consequent logical form with the entities of the question at hand. The instantiated consequents can then be passed to a task-specific reasoner (e.g., QR module, KB fact-lookup, etc.) to carry out the actions needed to respond to the input.

Consider the choice sets from the semantic parse for the question in Figure 6. The question can be answered with the facts from the KB (Figure 7). As can be seen, there is a stark difference

```
1. "states"
    1a. (isa state5288 State-Geopolitical)
    1b. (isa state5288 PhysicalStateOfMatter)
2. "border"
    2a. (bordersOn-AgentAgnostic state5288 Texas-State)
    2a. (and (isa border5311 BorderingSomething)
             (fe_ground border5311 Texas-State)
             (focalSubject border5311 state5288))
    2b. (isa border5311 RelativeLocationalPredicate)
3. "Texas"
    3a. (isa Texas-State State-UnitedStates)
```

Figure 9. Choice sets for the NLU semantic parse for the question "What states border Texas?"

between the representation used for the semantic parse and the more traditional KB representations. AQA bridges between these two representations with query cases.

An example QC is shown in Figure 8, with the consequent being the first argument to queryCaseFor and the non-abducible and abducible antecedents being the second and third arguments, respectively. A simple matching procedure (described in Section 2.3) is used to apply QCs to parses. In this case, it would match the semantic choices of Figure 6 to the antecedents of the QC in Figure 8 to produce a substitution like {<state123, state5288>, <border123, border5311>, <Indiana-State, Texas-State>}. This would then be used to instantiate the consequent as (bordersOn (TerritoryFn state5288) (TerritoryFn Texas-State)), which is then variablized (i.e., its discourse variables replaced with universally quantified variables) and passed to a KB fact lookup operation.

Training consists of automatically learning a mapping between domain-general semantic choices (antecedents) and task-specific logical forms (consequents). The input to AQA's training procedure is a set of questions and either natural-language answers or logical forms (referred to as unannotated / annotated training settings). If operating in the unannotated setting, AQA first takes the natural-language question-answer pair and generates a target logical form for the question from

```
(bordersOn (TerritoryFn Arkansas-State) (TerritoryFn Texas-State))
(bordersOn (TerritoryFn Louisiana-State) (TerritoryFn Texas-State))
(bordersOn (TerritoryFn NewMexico-State) (TerritoryFn Texas-State))
                    …          …          …
(isa Arkansas-State State-UnitedStates)
(isa Louisiana-State State-UnitedStates)
(isa NewMexico-State State-UnitedStates)
                    …          …          …
```

Figure 10. A relevant subset of the KB for answering the question in Figure 9

```
(queryCaseFor
 (and (bordersOn (TerritoryFn state123) (TerritoryFn Indiana-State)))
 (and (isa state123 State-Geopolitical)
      (isa Indiana-State State-UnitedStates)
 (and (isa border123 BorderingSomething)
      (fe_ground border123 Indiana-State)
      (focalSubject border123 state123))))
```

Figure 11. A query case that bridges between semantic parser outputs and KB facts

a connection-graph-based procedure operating on the KB. With a question and target logical form in hand, AQA must then determine a mapping between the entities from the semantic parse of the question to the entities within the target logical form. Once these entities have been determined, all that remains is to select a consistent set of choices from the semantic parse of the question (pertaining to the entities of the mapping) that will be used to form the returned QC.

Given a new question to answer, AQA applies the query cases it learned during training. First, AQA uses CNLU to produce a semantic parse of the question. Then, it retrieves relevant prior QCs based on their antecedents, aligning their antecedents with the semantic parse via analogy to produce a substitution between prior and current entities that can be used to instantiate the retrieved QC's consequent. A special query construction algorithm operates over instantiated QCs to determine the set of consequents to return. Once selected, the set of consequents is given to a task-specific reasoner (e.g., a fact-lookup operator) to produce the answer to the question. The next section describes AQA's training phase, where QCs are learned from a set of training examples.

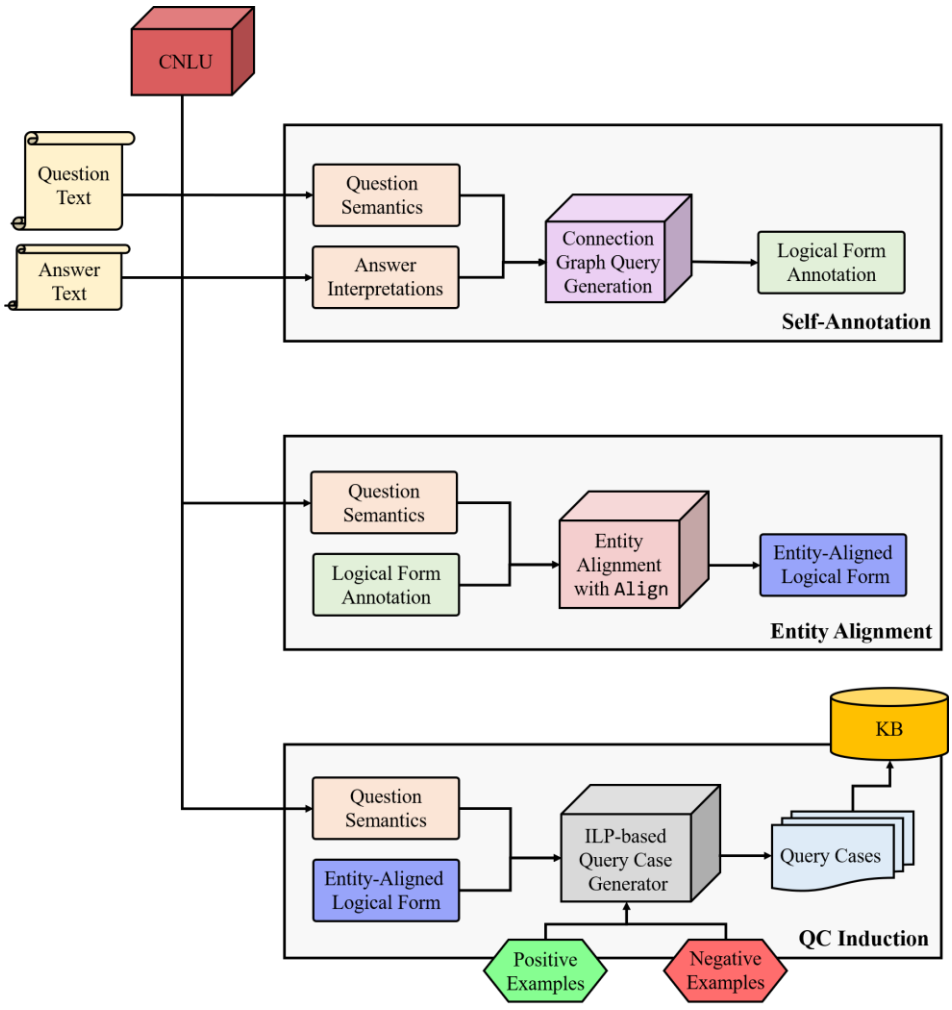

Figure 12. The three main processes for training AQA

5 Analogical Question-Answering Training

The input to training consists of a set of natural-language questions, each of which is paired with either natural-language answers or a target logical form. A high-level overview of the different steps involved in training AQA is given in Figure 9. We first describe how AQA handles being given natural-language answers using a question from GeoQuery as our running example.

5.1 Self-Annotation of Training Data

In the unannotated training setting questions are *not* hand-annotated with logical forms, but instead natural language answers are provided. Given a question and a set of such answers, the training procedure first builds a query that evaluates to only the answers given, consistent with the semantic constraints of the question. This is posed as a connection graph problem, wherein the objective is to find a connection-subgraph in the knowledge base that connects each answer to the entities of the question. With any sizable knowledge base, the graph is too large to attempt undirected search, so AQA leverages the semantic forms produced by a CNLU semantic parse of the input question to guide the search procedure. This approach assumes that the semantic parse of the question is (mostly) complete and is appropriately linked to the facts in the knowledge base that are needed to answer the given question. To understand this method, we use as a running example the following question, "What states border Indiana?", with answers being "Michigan, Ohio, Illinois, Kentucky".

5.1.1 Connection Graph Query Generation

For each possible interpretation of every answer, a breadth-first expansion is performed through the knowledge base constrained by the semantic choices for the nouns and proper nouns of the question ("states" and "Indiana").

In Figure 10, paths branch from possible answer interpretations to the possible interpretations of constituents of the question. For instance, Illinois can be a state that bordersOn Indiana. Answer ambiguity is possible (e.g., "Michigan" can be a lake) and so separate paths are maintained for each interpretation. Each path links to the elements of the semantic interpretation used in their construction. Initially, the only requirement is that paths align with the interpretations

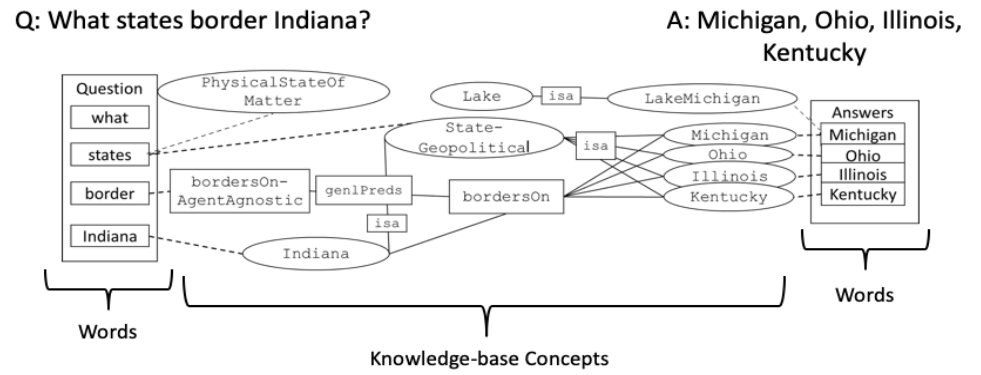

Figure 13. A connection graph between the entities of the question and answer

for nouns and proper nouns (in this case, "state" and "Indiana"). This facilitates path finding when there are missing links between the parser semantics and KB facts. In this example, the predicate bordersOn does align with an interpretation for the verb "border" via inheritance, and this information is used to prefer this path in the next step. In Figure 10, solid lines indicate paths in the KB while dashed lines indicate connections to the CNLU semantic parse.

In some cases, questions rely on an implicit connection to an entity in the KB (e.g., "How many states?" can be interpreted as "How many states in the United States?"). The approach proposes connecting entities by looking for KB entities that are most often associated with instances of the last observed type in the sentence (e.g., "states"). To do this, all KB facts containing an instance of the last type in the sentence are retrieved and the most commonly re-occurring entity from those facts is selected.

When a link in a path aligns with a token in the semantic interpretation (e.g., "Michigan" to State-Geopolitical to "state") it is replaced by the discourse variable (e.g., state123) from the parse. Thus, the path (bordersOn (TerritoryFn Michigan) (TerritoryFn Indiana)) becomes

(bordersOn (TerritoryFn state123) (TerritoryFn Indiana)). This facilitates query building in the next step because it maps the relevant KB entities to discourse variables in the question.

5.1.2 Combining Paths to Build Queries

A subset of these variablized paths are combined into a query. First, each path is paired with all the answers the path accounts for. For example, (bordersOn (TerritoryFn state123) (TerritoryFn Indiana)) accounts for "Michigan", "Ohio", "Illinois", and "Kentucky" which are all states that border "Indiana". Second, each path is assigned a score based on how well its expressions align with CNLU's parse. The path (bordersOn (TerritoryFn state123) (TerritoryFn Indiana)) receives a higher weight because it connects to bordersOn-AgentAgnostic by predicate inheritance (i.e., genlPreds). Third, with the answer pairings and alignment weights, the training algorithm finds the minimum combination of paths producing at least the target set of answers using a weighted set cover algorithm. It prohibits combining paths paired with conflicting choices (e.g., "state" as both State-Geopolitical and PhysicalStateofMatter).

In any large knowledge base concepts can be represented with different levels of granularity. For example, the question "Where is Indianapolis?" has two explicit answers, Indiana and the USA. Cyc has a finer-grained representation of location that uses different predicates for statehood versus countryhood. This is accounted for by allowing differing representations to be combined with a disjunction (e.g., set cover would produce (or (cityInState Indianapolis Indiana) (countryOfCity Indianapolis USA))). Combined queries that over-generate answers are pruned out, and the best scored query is returned.

```
(and (evaluate ?resp-state1
        (TheClosedRetrievalSetOf (TheList ?state1 ?NUM-VAR-1)
           (and (areaOfRegion (TerritoryFn ?state1) (SquareMile ?NUM-VAR-1))
                (bordersOn (TerritoryFn Texas-State) (TerritoryFn ?state1))
                   (isa ?state1 State-Geopolitical))))
(evaluate ?ans
        (FirstInListFn
           (FirstInListFn
                (SortFn ?resp-state1 lessThan
                     (FunctionToArg 2 (Kappa (?a ?b) (evaluate ?b (SecondInListFn ?a)))))))))
```

Figure 14. An example of a query for "What is the largest state that borders Texas?"

5.1.3 Handling Queries Involving Additional Computation

To perform well on question-answering, AQAT must handle questions involving additional computations (e.g., superlatives, cardinality, and summation). Cardinality and summation require only a minor extension to the process presented above. At first, the query generator builds a path with respect to the nouns and proper nouns of the question while ignoring the answer (a number). These paths are each paired with the answer and passed into the set cover algorithm as before. For each query generated by the set cover, two new queries are generated, one that includes a cardinality operator and one that includes a summation operator. The new set of queries is then filtered as usual, removing those that produce the incorrect answer.

Superlatives require more subtlety. For each superlative in the question, the word in the query that the superlative predicates is first identified (e.g. for “largest state”, this is the interpretation of “state”). To evaluate the superlative, it finds all methods by which each path instance of the word predicated by the superlative can be sorted (e.g., for “state” interpreted as State-Geopolitical, there are assertions in the knowledge base about the area and population of states, thus both alternatives are explored). It evaluates whether sorting by greater-than or less-

than returns the correct answer, and thus discovers both the method of sorting and its direction. An example of a generated superlative logical form is shown in Figure 11.

5.2 Training from Annotated Question-Answer Pairs

Using the logical form annotation process described above, AQA can convert unannotated data into annotated data. From annotated data (either through self-annotation or if it is provided by the dataset), AQA must then determine which part of the semantic parse justifies which parts of the logical form. For this to occur, AQA requires a correspondence between the entities of the semantic parse for a question and the entities for the target logical form of the question. Referring back to Figure 8, notice how both the antecedents and consequent of the QC share entities (`state123` and `Indiana-State`) in common. When QCs are used to answer questions, this allows for matches between a new set of semantic choices and the antecedents of a QC to produce a substitution that can be applied to the consequent of the QC. In the case of GeoQuery, this entity substitution is given to us, as we are constructing the target logical form from the semantic parse of our question. However, this does not hold generally (e.g., the science test process identification experiment in Section 8), and when it is not given, AQA must determine the most appropriate substitution of entities to make. We next explain how AQA determines this substitution, and then explain how QCs are induced.

5.2.1 Analogical Entity Matching

We pose the problem of finding the most appropriate entity substitution in terms of analogy. Though this procedure is not used in the GeoQuery domain, for simplicity, we explain its workings with the GeoQuery example shown in Figure 12. The figure includes a set of semantic choices

(isa Texas-State State-UnitedStates)	(flowsInRegions ?river (TerritoryFn ?state))
(isa state91893 State-Geopolitical)	(isa ?river River)
(isa state91893 MatterTypeByPhysicalState)	(bordersOn (TerritoryFn ?state) (TerritoryFn Texas-State))
(isa border91960 RelativeLocationalPredicate)	(isa ?state State-Geopolitical)
(focalSubject border91960 Texas-State)	(isa Texas-State State-Geopolitical)
(fe_ground border91960 state91893)	
(situationLocation flow91807 state91893)	
(primaryObjectMoving flow91807 river91800)	
(isa river91800 River)	

Figure 15. A subset of the semantic parse (left) and the expressions of the target query (right) for the question "What rivers flow through states that borders Texas?"

(left) and a set of query expressions (right). The system must determine a mapping from the entities of the semantic choices (e.g., Texas-State, state91893, etc.) to the entities of the target query (e.g., ?river, ?state, Texas-State). For this problem, the ideal substitution would be σ = {<state91893,?state>, <river91880,?river>, <Texas-State,Texas-State>}.

We first define some notation. Let *S* be the set of semantic choices, *T* be the target set of logical expressions, and *C* be a set of conflict pairs derived from our semantic parser's choice constraints) between elements of *S*. That is, if $(s_k, s_l) \in C$, this means that s_k and s_l cannot be asserted as true simultaneously. Within *S* and *T* we have a set of substitutable entities E_S and E_T, with the goal being to find a substitution σ between the elements of both sets. E_S is defined as the set of noun discourse variables from *S*. In Figure 12, this is E_S = {state91893, Texas-State, river91800}. For this example, E_T is simply the set of all entities in *T*, however, this will not always be the case, as examples in Section 8 illustrate.

A naïve method might try only matching entities that have similar type constraints (e.g., state91893 to ?state or Texas-State); however, such a scheme would face problems with

(isa Texas-State State-UnitedStates)	(isa ?river River)
(isa state91893 State-Geopolitical)	(isa ?state State-Geopolitical)
(isa state91893 MatterTypeByPhysicalState)	(isa Texas-State State-Geopolitical)
(isa river91800 River)	(closeConnection ?river ?state)
(closeConnection state91893 Texas-State)	(closeConnection ?state Texas-State)
(closeConnection state91893 river91800)	

Figure 16. Rerepresented semantic parse (left) and target query expressions (right)

ambiguous matches (e.g., that state91893 can match to two entities in the target expressions). Our method instead looks at both type constraints as well as structural features to determine how best to match entities. Intuitively, since both sets of expressions are alternative representations for the same question (one set of expressions as the semantic parse and the other as the appropriate KB-level query), we might expect them to share a sort of structural parallelism. That is, entities close together in one set of expressions would likely correspond to entities closer together in the other set of expressions. To express this notion of closeness, we introduce the symmetric predicate closeConnection. This holds between two elements of E_S / E_T when there is a path between them in S / T that does not go through another element of E_S / E_T. For instance, (closeConnection state91893 Texas-State) holds because there is a path between those two variables that avoids river91800. However, no such path exists between river91800 and Texas-State and thus a closeConnection statement between those two entities does not hold. Figure 13 shows a rerepresentation of both sets of expressions that incorporates only the relevant type constraints on E_S and E_T with closeConnection statements.

With the rerepresented sets of expressions, the problem of finding the most appropriate substitution can now be treated as the problem of finding an SMT-satisfying set of

correspondences between the two sets of expressions. This is almost solvable with SME alone, however, the presence of *C* (choice set constraints) makes the problem more challenging. Choice set constraints in this context preclude sets of correspondences *M* from simultaneously including pairs $(s_i, t_j) \in M$ and $(s_k, t_l) \in M$ where $(s_i, s_k) \in C$. This negative-disjointness constraint makes even the standard bipartite matching problem NP-Hard (Darmann et al., 2011). For efficiency, we utilize a hill-climbing local search procedure that starts from a promising candidate solution produced by SME and moves amongst better neighboring solutions until it can no longer improve. For a conflict-free correspondence set *M*, it expands outwards to all conflict-free sets that differ by at most two edges from *M*. The score of *M* is determined by two properties: KB relatedness and structural parallelism.

Calculating and Using the Score

For the KB relatedness score, we use the connection graph-based score introduced in Section 3.5, which computes the relatedness between two concepts as a function of how they are connected in the underlying KB. In this case, this is the relatedness of the type constraints (which may, for instance, match through genls, isa). We define *ocw* as a function that takes in two choices and returns this relatedness score. For structural parallelism, we simply compute the number of aligned closeConnection statements. We define a function *sr* which takes a set of correspondences *M* and simply counts the number of such statements in *M*. Then, the overall score of a correspondence set *M* between *S* and *T* is given as

$$sr(M) + \sum_{(s_i,t_j)\in M} \alpha * ocw(s_i, t_j)$$

where α is a preference weight for the KB relatedness score.

```
   With variables
        S = rerepresented set of semantic choices
        T = rerepresented set of logical expressions
        M = a set of correspondences between two sets of expressions

1. Define Neighbors(M, S, T):
2.      one-neigh & two-neigh = empty
3.      For edge1 in M:
4.          M' = M - {edge1}
5.          For (s, t) in S×T:
6.              If s and t are not used in M' and s does not conflict with a choice in M':
7.                  new-M = M' ∪ {(s, t)}
8.                  one-neigh = one-neigh ∪ {new-M}
9.      For edge1, edge2 in M×M:
10.         M' = M - {edge1, edge2}
11.         For (s₁, t₁), (s₂, t₂) in (S×T)×(S×T):
12.             If none of s₁, t₁, s₂, t₂ are used in M' and
13.                neither s₁ nor s₂ conflicts with a choice in M' or each other:
14.                 new-M = M' ∪ {(s₁, t₁), (s₂, t₂)}
15.                 two-neigh = two-neigh ∪ {new-M}
16.     return one-neigh ∪ two-neigh

1. Define Align(S, T)
2.      best-M = SME(S, T)
3.      While True:
4.              curr-M = best-M
5.              For matching M in Neighbors(curr-M, S, T)
6.                      score = sr(M) + Σ_(s,t)∈M α * ocw(s,t)
7.                      If score > best-score:
8.                              best-score = score
9.                              best-M = M
10.             If curr-M == best-M
11.                     return best-M
```

Figure 17. Structure-aware alignment algorithm

Figure 14 gives the local-search algorithm. The local-search algorithm obtains a first-pass candidate solution by using SME to find a correspondence set *M*. From that initial *M*, the algorithm begins to iteratively improve its solution by exploring the conflict-free alterations that differ by at most two edges from *M* (this is determined by the `Neighbors` function) until it can no longer find a match with a higher score as determined by the equation above. When no new set is found with a higher score, it returns *M*.

The final set contains pairings of expressions in *S* with expressions in *T* which can be used to extract entity mappings. Though this matching procedure can readily align entities in the

(isa Texas-State State-UnitedStates)	(flowsInRegions river91800 (TerritoryFn state91893))
(isa state91893 State-Geopolitical)	(isa river91800 River)
(isa state91893 MatterTypeByPhysicalState)	(bordersOn (TerritoryFn state91893) (TerritoryFn Texas-State)
(isa border91960 RelativeLocationalPredicate)	(isa state91893 State-Geopolitical)
(focalSubject border91960 Texas-State)	(isa Texas-State State-Geopolitical)
(fe_ground border91960 state91893)	
(situationLocation flow91807 state91893)	
(primaryObjectMoving flow91807 river91800)	
(isa river91800 River)	

Figure 18. Semantic parse (left) and set of target logical expressions (right) after substitution has been applied

absence of their type constraints matching (i.e., SME could produce a set of entity correspondences from closeConnection statements alone), we extract only those entity substitutions where the pair of entities had matching type constraints as well. In the example from Figure 12, our procedure yields the substitution σ = {<state91893,?state>, <river91880,?river>, <Texas-State,Texas-State>}. The substitution is applied to *T* (i.e., *T*σ), producing the target set of expressions shown in Figure 15.

The analogical entity substitution procedure is applied to every training question, producing a new set of training examples that consist of pairs of the form <*S*, *T*> where both *S* and *T* share entities in common. This is utilized during the QC induction procedure, described next.

5.2.2 Inducing Query Cases

Once the target logical form for each training question has had its relevant entities substituted for entities in the semantic parse, all that remains is to determine which semantic choices justify which expressions from the target form. This is formulated as an inductive logic programming (ILP) (Muggleton, 1992) problem.

Consider again the filtered semantic parse for "What rivers flow through states that border Texas?" shown in Figure 12. AQA assumes that each question is paired with a *set* of logical expressions. We refer to the question as *Q* and to its target set of logical forms as *T*. In general, there are no restrictions on the size of the set *T* other than that it is non-empty. It is often the case that it contains only one element (e.g., in the science test process identification experiments, *T* is always a singleton set). Each expression in *T* will become the consequent of its own separate query case, which allows for a greater degree of flexibility when answering novel questions. How a given logical form for a question (e.g., a conjunct of expressions in GeoQuery or a qualitative model for answering a science test question) is broken into a set of logical forms *T* varies by domain (this will be described in the relevant sections of the experiments). In the GeoQuery example of Figure 12, *T* would contain the two non-isa expressions (flowsInRegions river91800 (TerritoryFn state91893)) and (bordersOn (TerritoryFn state91893) (TerritoryFn Texas-State)).

To distill query cases, we start by pairing *Q* with a set of base statements $S = \{s_1, ..., s_i\}$ and a set of pairwise nogood (i.e., inconsistency) constraints $C = \{c_1, ..., c_k\}$ between pairs of elements in *S* (with constraints taking the form $\neg(s_j \wedge s_k)$). The elements of *S* include the semantic choices produced for *Q* (i.e., the NLU outputs), but can include additional statements as needed (e.g., in the comparative analysis experiment in Section 9, it includes statements for the root forms of words in *Q*). In our example, *S* would be the set of statements shown in Figure 12, and *C* would be nogood constraints asserted between semantic choices. For instance, two choices for "state" are (isa state91893 State-Geopolitical) and (isa state91893 MatterTypeByPhysicalState), representing different word senses. *C* will include a constraint that prevents these from being simultaneously

true. With *S* and *C* fixed, the ILP procedure is run for each expression *L* in the target set of forms *T*. The ILP procedure selects a subset of *S* to be considered the antecedents to the consequent *L*.

Given a set of positive examples *Pos* (this is often the set of all questions for which the consequent *L* is a member of its set of target logical forms) and negative examples *Neg* (this is often the set of all questions for which *L* is not a member of its set of target logical forms), our approach selects the expressions to add to the antecedent set *A* of a query case incrementally (with $A = \{\}$ initially). At each step, it selects an element s_i from *S* to add to *A*, i.e., $A \cup \{s_i\}$.

The choice of which elements from *S* to add to *A* made using the information gain heuristic of FOIL (Quinlan, 1990). Let E_1 and E_2 be sets of expressions and let C_2 be a conjunction of pairwise nogoods between the elements of E_2 (i.e., a set of mutual exclusivity constraints between elements of E_2) and define the pair $p = (E_2, C_2)$. E_1 is said to *cover* the pair *p* if there exists an SMT-satisfying substitution σ (as described in Section 2.2) between the entities of E_1 and E_2 such that the following holds

$$cov(E_1, p) = (E_1\sigma \subseteq E_2) \wedge (E_1\sigma \wedge N_2 \vDash \top)$$

Informally, this means that there exists a substitution that can be applied to the set of expressions E_1 such that they can be found within E_2 and the subset of E_2 found is non-conflicting. We write that E_1 covers *p* rather than the other way around because E_1 is an abstraction that will be used to draw inferences from multiple other sets of expressions. We write the coverage score of a set of expressions to be *E*

$$E^+ = \{\, p \in Pos : \; cov(E, p) \,\}$$

$$E^- = \{\, p \in Neg : \; cov(E, p) \,\}$$

$$cs(E) = \; -\log_2 \frac{|E^+|}{|E^+| + |E^-|}$$

With these definitions, we can define the value of adding a statement s_i to A as

$$gain(s_i, A) = |A^+| * (cs(A \cup \{ s_i \}) - cs(A))$$

which can be thought of as a coverage-weighted information gain heuristic. At each iteration, the element from *S* maximizing the gain value is added to the query case.

Intuitively, this can be viewed as adding statements that maximize the number of positive examples with matching constituent statements while minimizing the number of negative examples with matching constituent statements. It is often the case that two statements will have the same gain because they cover the same numbers of positive and negative examples. When such a tie occurs, AQA picks the statement that is most closely related to the target logical form *L* as measured by the connectedness weight from Section 5.2.1 (i.e., the *ocw* function).

Choices from *S* are added to *A* until no elements from *Neg* are covered. The result is a set of expressions that can be interpreted as the antecedents to a query case. Additionally, we store with each query case its particular coverage of positive questions. This gives our approach a rough estimate of quality / confidence in the query case, with query cases that cover a substantial number of questions being of greater value than those covering only a few questions.

QCs allow for two separate sets of antecedents (i.e., *A* is partitioned into two disjoint sets) to be specified, which correspond to the sets of non-abducible and abducible antecedents. The non-abducible should be considered the necessary conditions for a QC to hold, i.e., it specifies the smallest set of conditions needed to confidently apply this QC to novel inputs. The abducible set of antecedents should be considered additional evidence for a QC. These antecedents are those not considered necessary but provide further indication when satisfied that a particular QC should hold in some new scenario. Following the ILP procedure, the resultant set of expressions is partitioned

into those two sets, paired with the consequent expression *L*, and stored in the KB for later use. How an expression is selected to be a member of the non-abducible set versus the abducible set varies by domain, and thus we will describe the particular selection mechanism used in the relevant sections of the experiments.

ILP with Constraints

While the ILP procedure allows AQA to capitalize on structural regularities across questions, it is not entirely obvious that it would avoid the data-dependence pitfalls of standard machine learning approaches to question-answering. In particular, we would like AQA to still operate in the most data-sparse cases where it would be given a single positive example and zero negative examples. Fortunately, this can be addressed in the AQA framework through the use of constraints on the set of valid QCs.

AQA currently implements two general-purpose types of constraints. The first are inclusion-exclusion constraints that force the ILP procedure to add or exclude particular choices from the final QC returned. An example of this is the forced inclusion of type-constraining semantic choices, i.e., any isa choices that were used in the connection graph procedure to annotate a question with a logical form are forced to be included in the QC for a given question (e.g., the State-Geopolitical, Texas-State, and River semantic choices in the example of Figure 12). The second are termination constraints that force the ILP procedure to continue selecting expressions to add to a QC until some output criteria is met. An example of this for GeoQuery are connectivity constraints, i.e., that every entity in a returned QC must be connected through semantic choices (in the example of Figure 12, if both state91893 and Texas-State were a part of a QC then they would need to be connected through some set of choices, e.g., { (focalSubject border91960 Texas-State),

(fe_ground border91960 state91893) }. For any constraint given to the ILP procedure, if the conditions specified cannot be met, then QC induction is considered a failure, and nothing is returned. As with the determination of the non-abducible and abducible sets of expressions, the specific constraints used will vary by domain, and thus we will describe which constraints are used in the relevant experiment sections (as well as our intuitions guiding why we use a particular constraint for a given domain).

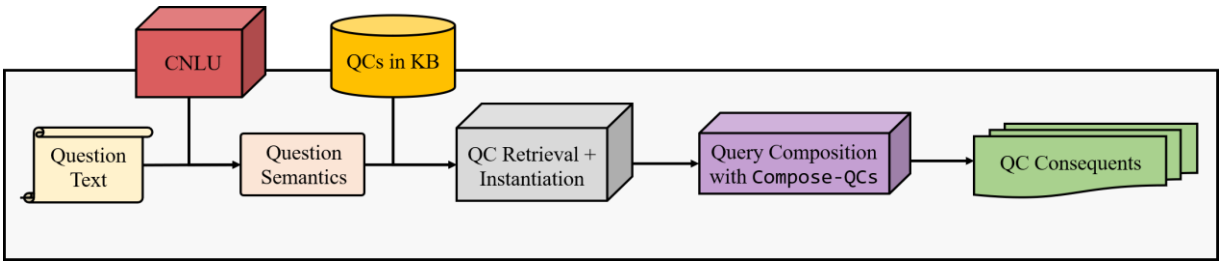

Figure 19. Overview of AQA's process for applying QCs

6 Applying Query Cases

Once training has completed, its output is a set of QCs that can be applied to new, unseen questions. A high-level depiction of the testing process is given in

Figure 16. We will explain AQA's process for applying QCs with an example question from the GeoQuery domain.

6.1 Instantiation of Query Cases

Given a question, AQA first parses the question with CNLU to produce a set of semantic choices. We use the same notation from Section 5.2.2, with the question as Q, base statements (produced from Q) as $S = \{s_1, ..., s_i\}$, a set of pairwise nogood (i.e., inconsistency) constraints $C = \{c_1, ..., c_k\}$ between pairs of elements in S, and the pair $p = (S, C)$.

For larger datasets, many QCs will be generated during training (e.g., over 2000 QCs are generated for the QuaRel experiment in Section 9). To quickly filter out inapplicable QCs, AQA removes all QCs without at least one antecedent that can be matched with a choice from the semantic parse. This is made very efficient by capitalizing on the fact that, for two expressions to successfully match together they must be syntactically equivalent with respect to a renaming of all

```
   With variables
    QC-index = a hash table that has key-value pairs of the form
               <renamed-antecedent, (list of QCs sharing that renamed antecedent)>
    choices = the set of semantic choices

1. Define Retrieve-QCs(QC-index, choices)
2.     rel-qcs = empty
3.     For choice in choices:
4.         renamed-choice = swap entities in choice with generic entities
5.         rel-qcs = rel-qcs ∪ QC-index[renamed-choice]
6.         return rel-qcs
```

Figure 20. QC retrieval algorithm

their matchable entities with indices representing their occurrence within the expression (this is analogous to the concept of De Bruijn (De Bruijn, 1972) indices from lambda calculus). More concretely, given an arbitrary expression like (focalSubject border123 state123), the renaming procedure walks down the expression and replaces each matchable entity with an index representing the point at which it was encountered (e.g., border123 is the first entity encountered, and thus it would be renamed as matchable-ent1). In this case, that would yield the more generic expression (focalSubject matchable-ent1 matchable-ent2). Checking for matchable expressions then involves only checking for equality, which allows for an efficient indexing scheme where a hash table is constructed that keeps generic expressions as keys and associated query cases as values. For a particular problem, each semantic choice is renamed and used as a key to retrieve applicable QCs, with the final set retrieved being the union of such query cases across choices. We provide pseudo-code in Figure 17.

For a particular QC, individual antecedents from $A = A_N \cup A_A$ (where A_N is the set of non-abducible antecedents and A_A is the set of abducible antecedents) of the QC are matched against elements of S using the `Match` procedure of Section 2.3 to find every maximal subset M of A for

which *cov*(*M*, *p*) holds (using the same definition of *cov* from Section 5.2.2). More concretely, *every* subset *M* of *A* will be found that meets the following conditions:

```
    With variables
         A_N = the set of non-abducible QC antecedents
         A_A = the set of abducible QC antecedents
         choices = the set of semantic choices

 1. Define Match-QC-Antecedents(A_N, A_A, choices)
 2.      all-insts = empty
 3.      For expr in A_N:
 4.          expr-insts = empty
 5.          For choice in choices:
 6.              For <existing-binds, used-choices> in all-insts:
 7.                  new-binds = Match(expr, choice, existing-binds)
 8.                  If new-binds and choice does not conflict with used-choices:
 9.                      expr-insts = expr-insts ∪ {<new-binds, used-choices ∪ {choice}>}
10.          If expr-insts:
11.              all-insts = expr-insts
12.          Else:
13.              return False
14.      For expr in A_A:
15.          expr-insts = empty
16.          For choice in choices:
17.              For <existing-binds, used-choices> in all-insts:
18.                  new-binds = Match(expr, choice, existing-binds)
19.                  If new-binds and choice does not conflict with used-choices:
20.                      expr-insts = expr-insts ∪ {<new-binds, used-choices ∪ {choice}>}
21.          If expr-insts:
22.              all-insts = all-insts ∪ expr-insts
23.      max-insts = remove pairs from all-insts whose bindings are subsets of other pairs
24.      return max-insts
```

Figure 21. Antecedent matching algorithm

1. $M \subseteq A$

2. $A_N \subseteq M$

3. $M\sigma \subseteq S$ (where σ is an SMT-satisfying substitution between the entities of *A* and *S*)

4. $M\sigma \wedge C \vDash \top$

i.e., that *M* is a subset of *A* that covers the non-abducible antecedents and can be matched to a consistent subset of *S* to yield an SMT-satisfying substitution σ (as defined in Section 2.2). We present the algorithm `Match-QC-Antecedents` in Figure 18, which returns a list of substitutions

1. “states”
    1a. (isa state5288 State-Geopolitical)
    1b. (isa state5288 PhysicalStateOfMatter)
2. “border”
    2a. (bordersOn-AgentAgnostic state5288 state3259)
    2a. (and (isa border5311 BorderingSomething)
        (fe_ground border5311 state3259)
        (focalSubject border5311 state5288))
    2b. (isa border5311 RelativeLocationalPredicate)
3. “states”
    3a. (isa state3259 State-Geopolitical)
    3b. (isa state3259 PhysicalStateOfMatter)
4. “border”
    4a. (bordersOn-AgentAgnostic state3259 Texas-State)
    4a. (and (isa border3115 BorderingSomething)
        (fe_ground border3115 Texas-State)
        (focalSubject border3115 state3259))
    4b. (isa border3115 RelativeLocationalPredicate)
5. “Texas”
    5a. (isa Texas-State State-UnitedStates)

Figure 22. Choice sets from the semantic parse for the question "What states border states that border Texas?"

paired with the choices that justified them. At a high-level, the procedure iteratively combines the substitutions found between the current iteration’s antecedent and each semantic choice (as determined by the `Match` algorithm) with non-conflicting substitutions found in previous iterations (i.e., it joins substitutions together when the combined substitution would not violate one-to-one constraints between entities and the semantic choices underlying both substitutions are non-conflicting). In the figure, lines 3-13 and 14-22 show the matching processes for the non-abducible and abducible sets of antecedents, respectively. The key differences between how both sets of antecedents are handled can be observed in lines 10-13 and 21-22. For non-abducible antecedents (lines 10-13), the set of all possible bindings is either overwritten with the new bindings discovered for the current iteration or the procedure terminates because no valid bindings could be found (line

```
(queryCaseFor
 (and (bordersOn (TerritoryFn state123) (TerritoryFn Indiana-State)))
 (and (isa state123 State-Geopolitical)
      (isa Indiana-State State-UnitedStates)
 (and (isa border123 BorderingSomething)
      (fe_ground border123 Indiana-State)
      (focalSubject border123 state123))))
```

Figure 23. Example of a query case available to AQA

13). In contrast, for abducible antecedents, any valid bindings found are added to the list of all possible bindings and there is no early termination. With all valid substitutions determined, the returned list is then the list of all pairs of substitutions with the semantic choices used in their construction.

For a given pair in the returned list of `Match-QC-Antecedents`, the substitution σ of the pair is used to instantiate the consequent with the exchangeable entities of the given question, and the pairing of the semantic choices with the instantiated consequent is passed to the recombination procedure, which we describe next.

6.2 Composing Query Cases

Each instantiated QC is associated with a set of choices from the semantic interpretation and a consequent logical expression. In Figure 21, we provide examples of QCs that would be instantiated from the choice sets in Figure 19 given the training query case shown in Figure 20. Recall that the question was “What states border states that border Texas?” In the figure, the top two instantiated query cases are clearly the two query cases that would be best to return (due to their structural connectedness and large numbers of matched antecedents).

```
1. (queryCaseFor
    (and (bordersOn (TerritoryFn state3259) (TerritoryFn Texas-State)))
    (and (isa state3259 State-Geopolitical)
         (isa Texas-State State-UnitedStates)
    (and (isa border3115 BorderingSomething)
         (fe_ground border3115 Texas-State)
         (focalSubject border3115 state3259))))

2. (queryCaseFor
    (and (bordersOn (TerritoryFn state5288) (TerritoryFn state3259)))
    (and (isa state5288 State-Geopolitical)
         (isa state3259 State-UnitedStates)
    (and (isa border5311 BorderingSomething)
        (fe_ground border5311 state3259)
        (focalSubject border5311 state5288))))

3. (queryCaseFor
    (and (bordersOn (TerritoryFn state5288) (TerritoryFn Texas-State)))
    (and (isa state5288 State-Geopolitical)
         (isa Texas-State State-UnitedStates)
    (and ))
```

Figure 24. Examples of instantiated query cases

The final logical form used to answer the given question will be some set of QC consequents combined together. The QCs selected will also produce a set of choices (the union of their antecedents), which can be considered a complete semantic interpretation of the question. AQA treats the selection of QCs as a coverage problem, where the objective is to select the minimal set of QCs whose combined antecedents cover *S*. At a high level, what we are asserting is that the two outputs (the combined antecedents drawn from *S* and the combined consequents) are *equivalent* (i.e., that both are alternative ways of characterizing the same thing). This is a stronger assertion than that of implication, where implication would instead yield a procedure that forward-chains exhaustively (*not* terminating when it has found a set of QCs whose antecedents cover *S*).

In this sense, QCs can loosely be viewed as rewrite rules, where the antecedents for a QC are

```
    With variables
        choices-remaining = all semantic choices

1.  Define Compose-QCs(choices-remaining)
2.      ret-expressions & choices-made = empty
3.      While choices-remaining not empty
4.          qc-options = empty
5.          For qc = <consequent, antecedents> in set of instantiated QCs
6.              cov_ct = length of intersection between choices-remaining and antecedents
7.              If cov_ct > 0:
8.                  Add <score(qc), qc> to qc-options
9.          best-qc = qc from qc-options with maximum score
10.         Add antecedents in best-qc to choices-made
11.         Remove choices in max-qc from choices-remaining
12.         Remove choices conflicting with choice in best-qc from choices-remaining
13.         Add consequent of best-qc to ret-expressions
14.     return ret-expressions and choices-made
```

Figure 25. Query case composition algorithm

transformed into its corresponding consequent.

Selecting QCs is made challenging by the constraint that the final set of returned elements from *S* be non-conflicting (i.e., two QCs cannot both be chosen if any of their antecedents conflict with one another). The composition algorithm, shown in Figure 22, finds such a set of non-conflicting QCs. It ranks each QC based on a function `score` which takes an argument `qc` and scores it based on the following properties:

1. Question Coverage – How many uncovered elements from *S* are present in `qc`
2. Overlap – How many antecedents are shared with the already-chosen elements of *S*
3. Conflict – How many elements from *S* are ruled out by selecting `qc`
4. Abduced Ratio – The fraction of antecedents for `qc` matched to elements of *S*
5. Positive Training Coverage – How many positive training examples did `qc` cover
6. Negative Training Coverage – How many negative training examples did `qc` cover

Once each QC is scored, it selects the highest scoring QC and adds the antecedents and consequent of that QC to the final sets of returned antecedents and consequent expressions. When there are no more QCs that can be selected (either because no remaining QC covers new elements of *S*, is non-conflicting with the selected QCs thus far, or there are no elements of *S* left to cover), the selected elements of *S* and the combined set of consequent expressions are returned.

Of the three instantiated QCs in Figure 21, the first QC selected by `Compose-QCs` would either be the first or second QCs (i.e., 1 or 2 in the figure) given that they cover the same number of elements from the semantic parse and have the same number of conflicts. Supposing the first QC were to be selected, its antecedents would be removed from the uncovered set of semantics (i.e., the `choices-remaining` variable in the algorithm), along with any other semantic choices that conflict with those antecedents. The consequent of that QC (i.e., (bordersOn (TerritoryFn state3259) (TerritoryFn Texas-State))) would then be added to the return variable `ret-expressions`. The next QC selected would be the second QC in the figure. Again, its antecedents (and the choices conflicting with those antecedents) would be removed from the set of uncovered semantics and its consequent would be added to `ret-expressions`. Once those two QCs have been selected, there are no more uncovered semantic choices, and thus the last QC (whose two isa antecedents were accounted for by the selected QCs) is simply discarded. When there are no choices remaining, the list of expressions (`ret-expressions`) as well as the list of choices (`choices-made`) are returned. The expressions are transformed into a format acceptable to the domain-specific reasoner being used for the current task (e.g., for GeoQuery this includes translating discourse variables into logical variables).

7 GeoQuery

The first domain is GeoQuery (Zelle and Mooney, 1996). It consists of 880 questions paired with logical forms, intended to be evaluated against the Geobase knowledge base.

To work with GeoQuery, we first translated Geobase into CycL and imported it into a Cyc microtheory. The GeoQuery answers were generated by running the GeoQuery gold-standard queries in a Prolog interpreter loaded with the Geobase rules and KB. The system's answer was scored as correct only if it exactly matched the answer generated by GeoQuery.

The NextKB knowledge base contained a substantial amount of pre-existing geographical knowledge. In order to ensure a fair comparison to other GeoQuery systems, path search was restricted to exclude these geography-related microtheories. Otherwise, the learned queries actually generated correct answers that Geobase did not account for. For example, the query learned for a question like, "What cities are in Indiana?" uses the predicate cityInState to retrieve all cities in the state of Indiana. When the query is evaluated with respect to the Geobase microtheory, the returned answer is a subset of the full list of cities in Indiana because Geobase does not include all cities in Indiana. However, when the same query is evaluated with respect to other geography microtheories, the returned answer is a more complete set of cities in Indiana.

To lessen the effect of broken parses (i.e., sentences for which the CNLU syntactic parse fails to produce a singular parse tree, and instead produces parse tree fragments for the sentence's constituent phrases) on AQA's performance, the semantic parse for each sentence is automatically augmented with a series of connectivity statements. Each connectivity statement connects two entities identified in the syntactic parse as either a noun or proper name and takes the form (closeConnectionBetween ent123 ent234). The entities connected are adjacent within the sentence

(e.g.., for “cities in states bordering Indiana”, there would be a connectivity statement between “cities” and “states” and a statement between “states” and “Indiana”), which allows for some structural information to be available even when the syntactic parse fails.

7.1 Training for GeoQuery

As GeoQuery consists of natural-language question-answer pairs, AQA must first self-annotate each training example. This replaces the natural language answer with a formal query as described in Section 5.1. Following self-annotation, AQA applies the ILP approach of Section 5.2.2 which then learns a set of QCs that can be used to answer the test questions. Notably, the entity matching step outlined in Section 5.2.1 is not used for this dataset because the self-annotation process produces logical forms that use discourse variables and entities from the semantic parse of each question. Questions in GeoQuery are frequently compositional in that they have target logical forms that are conjunctive queries. To aid with generalization, when AQA is given a question to train on, it produces query cases for each of the elements of the provided conjunct (this is the target set of logical forms *T* referred to in Section 5.2.2). In addition, AQA employs two constraints for the ILP component of training (mentioned at the end of Section 5.2.2), which are inclusion constraints forcing any type-constraining semantic choices to be included in generated QCs and termination constraints forcing the entities of the set of semantic choices selected by the ILP procedure to be a member of the same connected component.

The subtlety with training AQA for different datasets lies mostly in the ILP procedure of Section 5.2.2. Specifically, within that procedure there is an important choice to be made as to how to partition the set of all training examples into positive and negative sets for a particular question. Suppose for a particular question with a target set of logical expressions E_1, ..., E_n, we are trying

(isa California-State State-UnitedStates)	(isa California-State State-UnitedStates)
(objectFoundInLocation city3810 California-State)	(objectFoundInLocation city4405 California-State)
(groupMembers California-State city3810)	(groupMembers California-State city4405)
(isa city3810 City)	(isa city4405 City)
(isa city3810 UrbanArea)	(isa city4405 UrbanArea)
	(significanceOfSituation city4405 highAmountOf)

Figure 26. Semantic parses for "What are the cities in California?" (left) and "What are the major cities in California?" (right)

to induce a QC for the expression E_1. For instance, given a "cities in" question, this might mean that we are trying to induce a QC with the consequent expression (cityInState city123 California). A naïve proposal to generating the positive and negative examples (i.e., those examples that our QC should cover and should *not* cover) would be to use as positive and negative examples all those training questions where a cityInState expression is or *is not* one of the associated expressions for the target logical form. Though appealing for its simplicity, this scheme ultimately does not work when there are expressions that can be related through the specPreds relation. Consider the two questions "What are the cities in California?" and "What are the major cities in California?". The predicate for the target expression of the first question is cityInState while the predicate for the second question's target expression is majorCityInState. Looking at the associated semantic parses for each question (provided in Figure 23), it can be seen that the "cities in" question has a semantic parse that is a subset to the "major cities in" question. Thus, if the majorCityInState question was used as a negative example when inducing a QC for the cityInState question, AQA would not be able to successfully induce a QC because there does not exist a choice in the semantic parse of the cityInState question that would allow it to *not* cover the choices of the majorCityInState question.

The solution we use for this is quite simple. When training for a particular question with expressions E_1, ..., E_n, AQA simply does not include in the negative examples any question that involves an expression that can be linked via specPreds to any of E_1, ..., E_n. Notably, the converse is not enforced, i.e., the negative examples can include those questions with expressions linked via genlPreds to any of E_1, ..., E_n. This distinction is made under the assumption that more specific questions (e.g., "major cities in" is more specific than "cities in") will involve additional semantic choices that are relevant to the QC, which at least appears to hold for GeoQuery.

There are two other training considerations to note. First, for superlative questions that involve a quantity (e.g., population, area, etc.), we also include in their positive examples all training questions that involve that quantity (with negative examples being otherwise unchanged). Second, for a particular question's induced QC, the non-abducible antecedents of the QC are chosen to be any adjective, superlative, and noun choices that were chosen by the ILP procedure, and the abducible antecedents are all other choices. In the "major cities" example, this would mean that the semantic choices for both "major" and "cities" would be included in the non-abducible antecedents of the induced QC.

7.2 Testing for GeoQuery

To answer a question in this dataset, AQA is applied to the semantic parse for the question (see Section 6 for the application process) to produce a set of query expressions. The output expressions of AQA are combined with a conjunction and have their discourse variables converted to universally quantified variables. For instance, if AQA output the set of expressions { (bordersOn state123 Indiana-State), (isa state123 State-Geopolitical) } the query evaluated against the knowledge base would be (and (bordersOn ?state123 Indiana-State) (isa ?state123 State-

System	*Acc.*
Zettlemoyer, 2005	79.3%
Kwiatkowski, 2010	88.6%
Liang, 2013	91.4%
Jia, 2016	89.3%
Cheng, 2017	86.7%
Dong, 2018	88.2%
AQA	87.5%

Table 1. GeoQuery Main Results

Geopolitical)). The result is compared against the gold-standard answer and only if it exactly matches is it considered correct.

7.3 Experiments and Results

The test set for GeoQuery consists of 280 questions annotated with logical forms. As with training, we execute the logical forms to produce a set of answers for each question. Success is measured by the number of questions for which AQA generates the exact same answers as the gold-standard.

Our main results for this domain are presented in comparison to other high performing GeoQuery systems in Table 1. Liang, Jordan, and Klein (2013) provide the current state of the art on GeoQuery question answering and, like us, train using natural-language QA pairs. They introduce dependency based compositional semantics, a formalism that encodes logical forms in trees that are generated from a fixed set of domain predicates tied to lexical triggers. They evaluate their system with base triggers, where domain predicates are attached to parts-of-speech for the words that trigger them, and augmented triggers, where the predicates are manually mapped to specific prototypical words.

One theoretical benefit of AQA is that it only needs to see each part of a query once to apply those parts to a novel question. To test this, the system was evaluated on a set of training

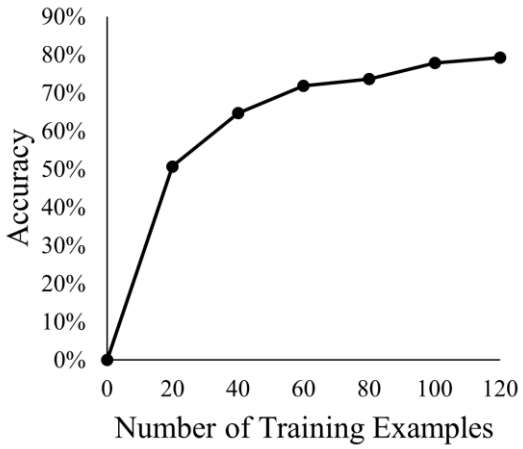

Figure 27. Learning curve experiment

questions ranging from 20 to 120 QA-pairs automatically selected with a max-coverage optimization algorithm to maximize the number of questions covered. The training questions were taken from the training set (600 questions) and the learned QCs were then tested on the standard GeoQuery test set (280 questions).

The learning curve from 20-100 examples is shown in Figure 24. With only twenty questions, performance reached 51%. Comparable performance to full training is reached around 120 QA-pairs, where AQA achieves about 80% correct. The ability to extract large performance gains (50% from 20 questions) from very little data is a chief contribution of AQA. To our knowledge, none of the other comparable systems have evaluated their performance with so few examples.

Assessing Performance

We performed an error analysis of the 35 missed questions and found that there were four main types of errors. The first type of error affected 15 questions and involved the test question having some key component being too dissimilar from a training question. For instance, the phrase “total length” in the question “What is the total length of all rivers in the USA?” was never seen in the

training set, and thus AQA did not have a query case that could be used to map the semantics for this phrase to the appropriate query pattern. Another 10 were missed simply because the query composition algorithm, `Compose-QCs`, produced the incorrect query, even though there were correct query cases that could be utilized. As an example, for the question "What is the population of Washington?", AQA generated a query for the population of Washington D.C., rather than the correct interpretation of Washington the state. Three questions were missed due to parsing failures that resulted in AQA not being given the inputs needed to answer the question (e.g., the semantics for "long" were not produced in "How long is the longest river in California?" due to a broken parse). Lastly, 7 were missed due to the logical form being annotated incorrectly in the test set, e.g., the question "What state has the longest river?" has a logical form that evaluates to all states in the U.S. with rivers.

The error analysis suggests a few directions for improvement. The first is to learn query cases that better generalize to unseen words and phrases. An example of where this could be explored is in the selection of non-abducible antecedents which restrict a query case's applicability. The current implementation of AQA has `isa` constraints on the entities of a query case in the non-abducible set of antecedents which restricts the applicability of query cases to those that involve the same types of entities. Finding a way to relax such constraints without resulting in the overgeneration of incorrect query cases may be one way of improving generalizability of query cases (in preliminary experiments, overgeneration was found to be the main difficulty with such a strategy). Another clear area for improvement is in the composition algorithm. The algorithm is currently designed to be general-purpose enough to handle the three drastically different tasks of this thesis. It is possible that there is some level of output-type-specific (but not task-specific)

tuning that may improve performance for this domain. That is, it may be that for factoid question-answering, there are useful strategies that could be implemented within AQA that are reasonably general and could improve performance but are not useful for answering scenario questions (as explored in the following experiments of this thesis).

We believe that a key source of AQA's data efficiency lies in the use of termination constraints within the ILP query case generation procedure. Termination constraints force AQA to produce query cases that have properties known to be desirable for a particular domain. For instance, because GeoQuery is highly compositional, the particulars of how the many entities within a question relate to one another provides useful information for constructing well-formed queries. To explore the utility of termination constraints, we performed a follow-up ablation experiment where we measured AQA's performance when the connectivity termination constraint was removed (i.e., we removed the constraint noted at the beginning of Section 7.1 that stops AQA from producing query cases with disconnected entities). For this experiment, we used the same minimal set of 10 questions from the learning curve experiments which provided us with the most extreme data sparse setting. The effect of this change was dramatic, resulting in a drop in performance of 30% (from 50% down to 19%). Without the connectivity termination constraint, AQA learned query cases that were extremely overfit to the training set (e.g., a query case was learned that would assert a bordersOn relation between two states because there were no questions in the negative set of examples that involved two states). These query cases would regularly lead to overgeneration of query expressions during testing (i.e., queries with too many constituent subexpressions) because their antecedents were far less restrictive than before.

7.4 GeoQuery Related Work

Sharma and Forbus (2013) produced horn axioms for the Cyc knowledge base with connection subgraph techniques. Given a query to justify, their algorithm built a subgraph between the entities of that query, with paths limited to particular predicate types. These types were learned via reinforcement learning over examples. In contrast, this work starts with natural language and uses QA-pairs to learn query patterns rather than higher-order knowledge.

The use of connection subgraphs is similar to recent applications of path ranking algorithms for relational learning (Lao et al., 2011). With this approach, paths are found via random walks. By contrast, the method in this work relies on finding the same path between many answers and a question entity. Random walks could lead to missing overlapping paths, leading set cover to produce inflated queries. The key constraints here come from the linkage between question and answer interpretations; whereas path ranking techniques that replace random walk with a more exhaustive search need to add more heuristic restrictions, e.g. avoid expanding nodes with large out-degrees (Gardner and Mitchell, 2015).

Berant et al.'s (2013) SEMPRE maps phrases to predicates using a lexicon and composes across the sentence to over-generate semantic derivations. They also use a bridging operation which injects new predicates based on the types of neighboring predicates in the question. They then train a log-linear model to select good derivations using QA pairs. The connection-graph technique presented here is like bridging in that it proposes expressions to connect entities in the question.

Dong and Lapata (2018) introduced a two-stage neural semantic parsing model that operated by mapped language into a coarse-grained logical form that would subsequently be

transformed into a final formal logical form. Conceptually this is quite similar to our two-stage approach to question-answering; however, unlike our approach (which uses a domain-general semantic parser to produce the intermediate representation), their intermediate representations are derived from the annotated logical forms themselves (i.e., their approach abstracts out details of the provided target logical forms). This means that their intermediate representations are task-specific and must be provided as annotations for the dataset.

8 Science Test Process Identification

While GeoQuery presented interesting challenges due to the highly compositional nature of its questions, it came with a number of simplifications that do not hold for general question-answering. For instance, the questions, while compositional, were decomposable into simple logical form primitives (e.g., "What states border states through which the Mississippi flows?", can be decomposed into two simple fact lookups). Additionally, there was very little in the way of distracting or irrelevant information in each question (i.e., each part of a question would be a part of the activation conditions for some query case). The second domain of this thesis uses process identification in science test questions to demonstrate how to extend AQA to domains for which those advantages do not hold. Science tests often involve questions with complicated scenarios concerning multiple entities, relationships, and processes. Naturally, the logical form representations of such questions are more complicated than the case of GeoQuery, where the questions were relatively compact and straightforward.

Consider the simple example question in Figure 25. For a system to understand this question, it would require a conceptual model of evaporation and a means of fitting the elements of that model to the specifics of the question. In Figure 26, we present one possible simple

"Which of the following processes is responsible for changing liquid water into water vapor?"

A. "photosynthesis"
B. "condensation"
C. "evaporation"
D. "precipitation"

Figure 28. Example science test question

```
1. (mfTypeParticipant NaiveEvaporationProcess ?liquid LiquidTangibleThing liquidOf)
2. (mfTypeParticipant NaiveEvaporationProcess ?gas GaseousTangibleThing gasOf)
3. (mfTypeParticipant NaiveEvaporationProcess ?sub ChemicalCompoundTypeByChemicalSpecies substanceOf)
4. (mfTypeParticipantConstraint NaiveEvaporationProcess (substanceOfType ?liquid ?sub))
5. (mfTypeCondition NaiveEvaporationProcess (touches-Directly ?liquid ?gas))
6. (mfTypeConsequence NaiveEvaporationProcess (qprop (EvaporationRateFn ?self) (TemperatureFn ?liquid)))
7. (mfTypeBiconditionalConsequence NaiveEvaporationProcess (hasQuantity ?self (EvaporationRateFn ?self)))
8. (mfTypeConsequence NaiveEvaporationProcess (i+ (AmountOfFn ?sub Gaseous-StateOfMatter ?gas)
                                                  (EvaporationRateFn ?self)))
9. (mfTypeConsequence NaiveEvaporationProcess (i- (AmountOfFn ?sub Liquid-StateOfMatter ?liquid)
                                                  (EvaporationRateFn ?self)))
```

Figure 29. Model fragments for the process of evaporation

qualitative model for evaporation. Informally, the model specifies that there is a liquid and a gas involved (lines 1 and 2), and that liquid evaporates into the gas (lines 5, 8, and 9).

8.1 Training for Process Identification

For science test process identification, the system is given as inputs a natural-language scenario concatenated with the correct multiple-choice answer paired with a process name. For the problem in Figure 25, it would be given the scenario text concatenated with answer C along with the model fragment name NaiveEvaporationProcess. AQA then retrieves the specified initial set of model fragments (in this case, for NaiveEvaporationProcess), any model fragment participants, as well as any fragments that they depend on. This retrieved set will be referred to as the target set of expressions.

In GeoQuery the alignment between entities in the logical form and entities in the semantic parse was determined by the query generation procedure. In this domain, this is not explicitly provided and thus we must determine the alignment through other means. This is done with the `Align` algorithm (presented in Section 5.2.1) that determines which entities from our semantic parse correspond with which entities of our target model fragment. When the algorithm is given

“changing”
- (isa change98547 IntrinsicStateChangeEvent)
- (isa change98547 (CausingFn IntrinsicStateChangeEvent))
... ...
“liquid”
- (isa liquid65646 Liquid-StateOfMatter)
... ...
“water”
- (isa water52719 (LiquidFn Water))
- (isa water52719 Water-Ingestible)
... ...
“water vapor”
- (isa water-vapor65365 (VaporFn Water))
... ...

Figure 30. Choice sets from the semantic parse for the question in Figure 28

the semantic choices from Figure 28 along with the model fragment shown in Figure 26, it determines the best entity matches to be <liquid65646, ?liquid>, <water-vapor65365, ?gas>, and <water52719, ?sub> due to the relations between their constraining collections (in this case, Water from (isa water4250 (LiquidFn Water)) is an instance of ChemicalCompoundTypeByChemicalSpecies, LiquidTangibleThing matches identically, and VaporFn is related to GaseousTangibleThing through a resultGenl relation). The matched entities from the semantic parse are then substituted into the model fragments to allow the logical form to share variables with the semantic parse. Following the substitution, the ILP procedure learns a single query case for the entire set of model fragments at once. That is, unlike in GeoQuery where separate query cases were learned for each expression of a question’s conjunct, the ILP procedure will learn only one query case for each question that incorporates the entire process involved in the question (i.e., *T* in Section 5.2.2. is a singleton set consisting of a conjunction of the target process’s model fragment statements). For this dataset, AQA partitions positive and negative

```
(queryCase
  (and (isa evaporation46201 NaiveEvaporationProcess)
       (mfTypeConsequence NaiveEvaporationProcess
                          (i+ (AmountOfFn water52719 Gaseous-StateOfMatter water-vapor65365)
                          (EvaporationRateFn ?self)))
       (mfTypeConsequence NaiveEvaporationProcess
                          (i- (AmountOfFn water52719 Liquid-StateOfMatter liquid65646)
                          (EvaporationRateFn ?self)))
       (mfTypeConsequence NaiveEvaporationProcess (qprop (EvaporationRateFn ?self)
                                                         (TemperatureFn liquid65646)))
       (mfTypeBiconditionalConsequence NaiveEvaporationProcess (hasQuantity ?self (EvaporationRateFn ?self)))
       (mfTypeCondition NaiveEvaporationProcess (touches-Directly liquid65646 water-vapor65365))
       (mfTypeParticipantConstraint NaiveEvaporationProcess (substanceOfType liquid65646 water52719))
       (mfTypeParticipant NaiveEvaporationProcess water-vapor65365 GaseousTangibleThing gasOf)
       (mfTypeParticipant NaiveEvaporationProcess water52719
                          ChemicalCompoundTypeByChemicalSpecies substanceOf)
       (mfTypeParticipant NaiveEvaporationProcess liquid65646 LiquidTangibleThing liquidOf))
  (and (isa water-vapor65365 (VaporFn Water))
       (isa liquid65646 LiquidTangibleThing)
       (isa water52719 (LiquidFn Water)))
  (and (isa evaporation46201 NaiveEvaporationProcess)
       (isa process18654 MethodType)
       (to-Generic change98547 water-vapor65365)
       (isa change98547 Transaction)
       (fe_themes change98547 liquid65646)))
```

Figure 31. The query case learned for the evaporation question of Figure 28

examples based on whether they share the same underlying process (e.g., for our running example, the positive and negative sets would be those that do and do not, respectively, involve evaporation as their target process). Figure 28 shows the query case learned for the evaporation question. For this dataset, the non-abducible antecedents are those semantic choices used in the matching procedure and the abducible antecedents are all other choices.

8.2 Testing for Process Identification

A question in this dataset comes paired with natural language answer options (typically four per question). For a particular question, each answer option is processed separately, with the question being concatenated with each answer and fed to CNLU to produce distinct semantic parses for the different options. For a particular answer option, its corresponding semantic parse was passed to

System	*Acc.*
Random	25.0%
AQA	77.0%

Table 2. Science Test Process Identification Main Results

the QC application component of AQA (Section 6) to determine the single best matching QC and its instantiation with the entities of the question-answer pair. The answer option selected would be the one with the best matching QC as measured by coverage (i.e., the QC that matched with the largest number of semantic choices overall) that also involved at least one semantic choice from the parse of the answer option.

Though not necessary for this task (due to its purely natural-language multiple-choice evaluation), the output of the composition algorithm is a set of statements that can be used to instantiate model fragments relevant to the scenario at hand (i.e., the model fragment participants and conditions). A model formulation algorithm could thus be used to instantiate applicable model fragments for subsequent reasoning.

8.3 Experiments and Results

The result in Table 2 describes the performance of AQA on a set of 30 process-identification questions from 4th and 5th grade elementary science tests. This set of questions was extracted by a script that searched across a large set of science test questions (collected by the Allen Institute for Artificial Intelligence). The questions the script returned had keywords associated with the model fragments outlined in Crouse and Forbus' (2016) science test analysis. The original script returned 45 questions, however the 15 questions we did not include required reasoning beyond process identification, and thus were out of scope for the methods presented above. Furthermore, the

questions were restricted to those involving reasoning about a scenario, not questions involving definitions or taxonomies.

Due to the small size of the dataset, leave-one-out cross validation was used to measure performance. That is, at each test question the system had access to all QCs learned for each question *except* the question currently being tested (meaning 29 out of 30 questions were usable for each test question). Performance for this dataset was measured in terms of multiple-choice answer accuracy as there are no pre-specified gold-standard logical forms to compare against. Random guessing on this dataset would lead to 25% correct.

Assessing Performance

We performed an error analysis to better determine the source of AQA's failures in this domain and found that the most impactful source of errors was overfitting. In particular, while the higher-level components of a particular scenario type would be consistent across questions (e.g., 3 out of 4 evaporation questions involved a liquid disappearing outside on a hot day), the specific scenario elements would differ. For instance, though most evaporation questions involved a liquid disappearing outside on a hot day, the particular liquid disappearing differed between questions. The evaporating liquids were given as "puddle", "rainwater", and "water" in three different questions, which were parsed into isa semantic choices involving the collections Puddle, Rainwater, and (LiquidFn Water), respectively. Though these collections are very closely related (both Puddle and Rainwater can be linked through (LiquidFn Water-Fresh) to (LiquidFn Water)), they do not match to one another during QC instantiation because there is not an *immediate* genls statement that can link them.

Though we leave the resolution of this issue to future work, two possible solutions seem like viable options to explore. The first is to match semantic choices more loosely to QC antecedents during instantiation through a connection graph search (rather than the current system that allows collections to match only with immediate genls statements). The difficulty with this approach, however, will be in finding a principled matching criterion that does not lead to overgeneration of allowable matches. Another possible option is to generalize learned QCs after training such that they become more broadly applicable. For instance, one might use SAGE (McLure, 2015) to automatically determine the best collections to generalize between query cases and then use concept learning (Mitchell, 1978) to explicitly determine the best generalized collection to substitute into a new, abstracted query case.

8.4 Process Identification Related Work

The alignment method used during the training portion of AQA is intended to find a mapping between the outputs of a semantic parser to some task-specific logical form. This is similar to the work of Fan and Porter (2004), which introduced Loose-speak, an algorithm that would take in a (possibly malformed) novice user's query and map it to a query more likely to return the results the user was expecting. As part of the mapping process, their algorithm employed a range of heuristics, some of which are analogous to the ontological alignment heuristics of our approach (e.g., type hierarchy similarities).

Khot et al. (2017) introduced a method for answering complex, compositional science test questions from OpenIE (Schmitz et al., 2012) extracted knowledge bases. They posed the problem of multiple-choice question-answering as a search for an optimal subgraph connecting a question and answer through the knowledge base. While their approach is similar to the training procedure

presented for the GeoQuery domain, it is unclear how they would adapt their system to perform for non-multiple-choice questions.

Other works have explored reading comprehension for science textbooks specific to biological processes (Berant et al., 2014; Rajagopal et al., 2019). Those works differ in that they focus entirely on biological processes (e.g., photosynthesis) and involve less formal reasoning; however, their high-level approach of matching the entities of a paragraph into a structured model of a process to be able to reason about the outcomes of the process shares much in common with ours.

9 QuaRel

QuaRel (Tafjord et al. 2019) is a dataset consisting of 2,771 questions that tests the ability to answer comparative analysis questions about 19 different quantities. Figure 29 provides an example from this dataset. In the example, because the distance of an episode of rolling is qualitatively inversely proportional to the friction of the floor, rolling further on the wood floor implies that (A) is the correct choice. Here the qualitative reasoning is straightforward, the complexity is formulating a qualitative model from language.

Each question in QuaRel comes annotated with multiple choice answers and a representative logical form. In this work, we consider only the semantic parsing task (i.e., we do *not* attempt the multiple-choice portion) where our objective is to produce logical forms that match the gold-standard logical forms provided. Within their logical forms, QuaRel denotes situations being compared as *worlds*. Each problem includes exactly two worlds. The comparison between worlds is made with respect to their entities, which can be either different entities (e.g., a "rough ball" versus a "smooth ball") or the same entity at two different points in time. In the prior example of a car rolling on wood versus the same car rolling on carpet, the worlds would be the two rolling events. The associated logical form for the question is given as the implication qrel(distance,

"Alan noticed that his toy car rolls further on a wood floor than on a thick carpet. This suggests that ..."

(A) "The carpet has more resistance"
(B) "The floor has more resistance"

Figure 32. A question from the QuaRel dataset

higher, world1) → qrel(friction, higher, world2), where the antecedent represents the question setup and the consequent represents the correct answer option.

We use automatically translated variants of their logical forms to have representations that are compatible with our off-the-shelf comparative analysis reasoner. In particular, we map their quantity representations to our hasQuantity statements (e.g., (hasQuantity world1 ((QPQuantityFn Strength) world1))), their direct comparative qrel to our qGreaterThan statements (e.g., (qGreaterThan ((QPQuantityFn Friction) world1) ((QPQuantityFn Friction) world2))) and their indirect comparative qval to our valueOf statements (e.g., (and (valueOf ((QPQuantityFn Friction) world1) (LowAmountFn Friction)) (valueOf ((QPQuantityFn Friction) world2) (HighAmountFn Friction))).

The system automatically constructs three microtheories per problem, one for each world and one for the problem as a whole, which inherits from the two world microtheories. These are populated with facts automatically constructed by the NLU system (see Figure 30). For example, the world-specific microtheories describe which quantities are relevant (e.g., (hasQuantity world1 ((QPQuantityFn Friction) world1))) and information about values (e.g., (valueOf ((QPQuantityFn Friction) world2) (HighAmountFn Friction))). The problem-level microtheory contains facts involving both worlds (e.g., (qLessThan ((QPQuantityFn Friction) world1) ((QPQuantityFn Friction) world2))). In our system, microtheories can be treated as cases for analogical reasoning, thereby simplifying the DQA alignment process.

QuaRel includes a hand-generated domain theory describing qualitative proportionalities (Forbus, 1984) between quantity types. For example, q-(speed, friction) represents that if friction goes up, speed goes down. These are translated into our knowledge base into relations between

World 1 Microtheory

(hasQuantity world1 ((QPQuantityFn Distance) world1))

(qprop- ((QPQuantityFn Distance) world1) ((QPQuantityFn Friction) world1))

... ...

(qprop ((QPQuantityFn Sound) world1) ((QPQuantityFn Distance) world1))

World 2 Microtheory

(hasQuantity world2 ((QPQuantityFn Distance) world2))

(qprop- ((QPQuantityFn Distance) world2) ((QPQuantityFn Friction) world2))

... ...

(qprop ((QPQuantityFn Sound) world2) ((QPQuantityFn Distance) world2))

Problem Microtheory

(qGreaterThan ((QPQuantityFn Distance) world1) ((QPQuantityFn Distance) world2))

Figure 33. Partial inputs to DQA for the example question

fluents, e.g., (qprop- ((QPQuantityFn Speed) world1) ((QPQuantityFn Friction) world1)), which are added automatically when a relevant quantity is mentioned.

The baseline methods provided for QuaRel do not handle problems compositionally. That is, they take as input the concatenation of the question setup with an answer option and generate a single logical form that represents both problem components jointly. For instance, given the example of Figure 29, their approach would take as input the concatenation of the question and answer, i.e., "Alan noticed that his toy car rolls further on a wood floor than on a thick carpet. This suggests that the carpet has more resistance", and produce as output qrel(distance, higher, world1) → qrel(friction, higher, world2). When selecting between the two answer options, they take the answer for which the generated logical form has highest probability (as predicted by the neural model).

Our approach handles QuaRel questions differently. In particular, our technique processes each QuaRel problem in two stages. First, AQA is used to predict a logical form for *only* the question setup (e.g., "Alan noticed that his toy car rolls further on a wood floor than on a thick carpet."). Then, the predicted form for the question setup is passed to an off-the-shelf DQA reasoner to produce a set of possible answers. Finally, AQA predicts a logical form for the consequent component of the problem (e.g., "This suggests that the carpet has more resistance"), which is used to select the best DQA-inferred output to return.

9.1 Training for QuaRel

QuaRel consists of natural-language questions paired with logical forms, and thus self-annotation is not necessary for this dataset. In addition, the dataset provides world-annotations which specify the entities within each question that should be mapped to either world1 or world2. We use these annotations so that our generated logical forms will be comparable to the gold-standard forms given for each question. We also extend the representations produced by the semantic parser with additional statements indicating the root forms of words found in the question text (we found

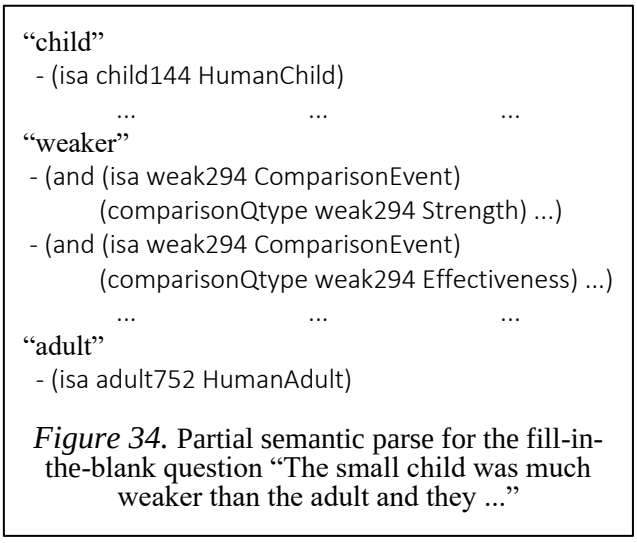

Figure 34. Partial semantic parse for the fill-in-the-blank question "The small child was much weaker than the adult and they ..."

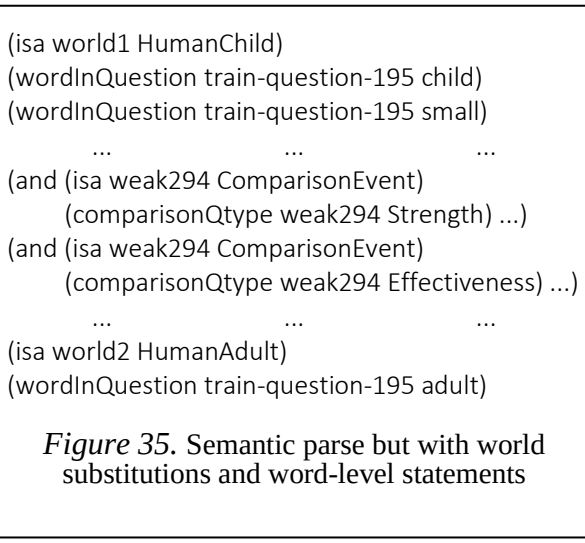

Figure 35. Semantic parse but with world substitutions and word-level statements

```
(queryCaseFor
 (and (hasQuantity world1 ((QPQuantityFn Strength) world1))
      (hasQuantity world2 ((QPQuantityFn Strength) world2))
 (and )
 (and (comparer strong623 world1)
      (isa strong623 ComparisonEvent)
      (comparisonQtype strong623 Strength)
      (comparee strong623 world2)))
```

Figure 36. A query case that would apply to the semantics shown in Figure 34

empirically that such information is helpful, as describe later). Figure 31 and Figure 32 show the semantic parse for a question both before and after these world substitutions and extensions are applied. The world substitutions mean that both the target form and the semantic parse for a question are utilizing a shared set of entities, and thus entity alignment is not necessary for this dataset.

As with the other datasets, AQA applies the ILP approach of Section 5.2.2 which then learns a set of QCs that can be used to answer the test questions. The system must learn two types of query cases for QuaRel: 1) quantity prediction query cases, which propose quantity assertions (e.g., (hasQuantity world1 ((QPQuantityFn Strength) world1))), and 2) ordinal prediction query cases, which have consequent statements either direct comparatives (e.g., (qGreaterThan ((QPQuantityFn Friction) world1) ((QPQuantityFn Friction) world2))) or indirect comparatives (e.g., (and (valueOf ((QPQuantityFn Friction) world1) (LowAmountFn Friction)) (valueOf ((QPQuantityFn Friction) world2) (HighAmountFn Friction))). AQA is used separately on each type of query case, since this was found to improve the learning process. The ILP procedure constructs its positive and negative sets of examples for learning quantities from the entire dataset. However, when learning ordinals, it only draws positive and negative examples from questions involving the same

quantity as the target form. For instance, when learning a QC with consequent (qGreaterThan ((QPQuantityFn Friction) world1) ((QPQuantityFn Friction) world2)), it constructs its sets of positive and negative examples *only* from other Friction questions.

The quantity and ordinal information provide all that is needed for our off-the-shelf DQA algorithms to generate answers. An example for a quantity query case is provided in Figure 33. Key to note, there are no non-abducible antecedents for QuaRel query cases (represented by the empty conjunction in the QC). The decision was made to have all antecedents considered as abducible because, for this task, being more precise (i.e., not returning a QC for every question) resulted in worse performance. That is, it was always better to simply guess a QC that *might* apply to a given question than to not answer (due to overly restrictive QC antecedents). Within code, this was achieved by shifting all ILP output expressions for a question to the abducible antecedent set of the question's stored query case.

In this larger dataset, the value of the ILP procedure in selecting the best choices becomes more obvious. In our example from Figure 31, the semantic choice (isa world1 HumanChild), while a seemingly relevant choice, is actually quite useless for predicting the correct quantity. Such a statement appears in 5 questions regarding Strength and 22 questions involving other quantities (e.g., "A child slips more easily on ice ...", which involves Friction). Alternatively, the two choices for "weaker", being (comparisonQtype weak294962 Strength) and (comparisonQtype weak294962 Effectiveness), both appear in 17 questions regarding Strength and only 3 questions involving other quantities (e.g., "Earth has stronger gravity than Mars because ...", which involves Gravity). In this case, both "weaker" choices have the same gain score because they cover the same

numbers of positive and negative examples, which means the tie is broken by relatedness to the target logical form.

Learning Quantities and Ordinals

The only difference between the process for constructing quantity versus ordinal query cases is in how positive and negative examples are partitioned. With quantity query cases, the positive and negative examples are those that would be expected, i.e., positive examples are questions with the same target quantity and negative examples are those questions with a different target quantity. This does not suffice for query cases predicting ordinals, because all questions involve essentially the same logical form.

For ordinal query cases, the system uses the same set of training questions for both the positive and negative examples. It frames positive examples in terms of a qGreaterThan prediction and all negative examples are framed in terms of a qLessThanOrEqualTo prediction. That is, for a given question, its positive form is coerced into (qGreaterThan ((?GenericQuantity) world1) ((?GenericQuantity) world2)) and its negative form is coerced into (qLessThanOrEqualTo ((?GenericQuantity) world1) ((?GenericQuantity) world2)). Because antecedents are added to *QC* until no negative examples are covered, this means that antecedents are added until the negation of the correct logical form is no longer covered. For training ordinals, the positive and negative sets of examples for a given question are then the same set, with positive examples being the positive forms of questions and negative examples being the negative forms of questions.

9.2 Testing for QuaRel

The result of training is a case library of quantity and ordinal query cases. To predict the setup for a question (i.e., the set of statements to hand off to the off-the-shelf DQA reasoner), AQA first predicts a single quantity query case (taken as the best scoring QC from the `Compose-QCs` procedure of Section 6), and then uses that predicted quantity to filter out irrelevant ordinals (i.e., those ordinal QCs that were generated for different quantities). From the pruned down set of ordinal QCs, AQA again uses the `Compose-QCs` procedure to determine the best matching ordinal prediction for the question. The predicted statements are added to the appropriate microtheories along with any necessary qprop statements, and DQA is used to generate a set of *possible* answers to the given question. For instance, given the statements in Figure 30, this may include (qGreaterThan ((QPQuantityFn Distance) world2) ((QPQuantityFn Distance) world1)), because distance is qualitatively proportional to strength. The consequent component to the question (which has also been parsed by CNLU) is then passed to the QC application procedure, which yields a quantity prediction that is then used to select a statement from the outputs of the DQA reasoner to return to the user.

9.3 Experiments and Results

We measure the performance of our approach in terms of three metrics: 1) Consequent generation accuracy (i.e., do we produce the same form for the logical form's consequent as provided by QuaRel), 2) Question setup accuracy (i.e., do we predict the correct ordinal and quantity values for the setup to a given question), and 3) Both consequent and question setup accuracy (i.e., do we correctly predict the gold-standard form in its entirety).

System	*Consequent Acc.*	*Setup Acc.*	*Both Acc.*
AQAT	36.1%	44.7%	32.2%
QuaSP	N/A	N/A	32.2%
QuaSP+	N/A	N/A	43.8%

Table 3. QuaRel Main Results

The original QuaRel paper introduced two Elmo-based (Peters, 2018) neural approaches: 1) QuaSP, a neural semantic parser intended to be a strong baseline that is representative of current state-of-the-art neural semantic parsing approaches and 2) QuaSP+, an extension of QuaSP that better handles the world formalism. Their reported performances for the QuaSP and QuaSP+ systems were 32.2% and 43.8% on the held-out test set. These percentages were measured for the correct prediction of both the answer and question setup (they did not measure consequent or question setup generation in isolation). While our system was capable of matching the lower performing neural model, the top performing system specifically designed for QuaRel outperformed ours, although we note that no system is yet close to the human results of 96.4% (produced via Mechanical Turk).

Assessing Performance

An error analysis indicates that problems in mapping the generic world tokens onto entities in the semantic parse accounted for a substantial number of the missed problems. Of the questions in the test set, there were only 331 out of 552 questions (i.e., a little more than half) where both generic world tokens (world1 or world2) could be mapped to entities in the corresponding semantic parse. Similarly, only 1092 out 1947 training questions had mappings from both world entities to semantic parse entities. This likely caused issues during both training and testing, as our approach

would thus typically be reasoning or learning with half of the relevant information for a given question.

We also examined the value of adding word-level statements to the semantics output by our semantic parser. As Table 3 shows, with word-level statements, the accuracy of logical form prediction was 32.4%. Without word-level statements, performance for logical form prediction dropped to 19.7%. Interestingly, word-level statements only had an impact at the level of quantity prediction. For ordinal prediction, the utility of word-level statements was nonexistent. Recall that the partitioning of positive and negative examples differs between the learning of quantity and ordinal query cases. When learning quantities, the positive and negative examples form a disjoint partition of the set of training questions, but when learning ordinals, the positive and negative examples are *both* defined from the same set of training questions. This means that word-level statements provide no information gain for ordinal prediction, as they appear in identical numbers between the positive and negative examples. Thus, despite having full access to word-level statements, none of the learned ordinal query cases had a word-level statement as an antecedent.

Interpretability and Explainability

Our main advantage over neural models lies in the difference in transparency of learned representations and the learning process. In contrast to high-dimensional vector space representations, our learned query cases are comprehensible to those familiar with predicate calculus. In addition, the learning process itself is quite transparent. Each query case maintains the list of questions it covered, which simplifies drawing dataset-level insights. For instance, word-level statements are frequently sufficient to infer that Friction is the quantity of interest, e.g., (wordInQuestion train-question-29 resistance) is the sole antecedent to a query case that covers

Generated quantity + ordinal: Distance("car on wood floor") is greater than Distance("car on thick carpet")	**Relevant question context:** - “his toy car rolls further on a wood floor than on a thick carpet” - “on” **Supporting training questions:** - "... likes to push his toy car around the house ... notices his car rolls slower on the carpet ... than on the hardwood floor ..."
Used qualitative proportionality (from KB): Distance is inversely qualitatively proportional to Friction	**Relevant question context:** - “far” - “floor” - “carpet” **Supporting training questions:** - “... when he pushes his book on the carpet, it doesn't go as far as when he pushes the book on the kitchen floor.”
Selected answer from DQA outputs: Friction("car on thick carpet") is greater than Friction("car on wood floor")	**Relevant answer context:** - “resistance” **Supporting training questions:** - “A door mat has more resistance then a microfiber cloth.” - “Rolling a marble over dirt creates less resistance then rolling it over sand.”

Figure 37. Automatically generated natural language outputs for the question in Figure 1. The left column is produced from generated logical forms (see Figure 2), while the right column is drawn from all underlying justifications (i.e., semantic choices used to instantiate selected query cases and the training questions those query cases were learned from).

74 questions. Furthermore, just the word “heat” was sufficient to cover 61 questions where Temperature was the target quantity. Insights like these could help explain why the QuaSP system was noted as performing particularly well at predicting quantities.

Because its representations are logic-based, our system can automatically generate explanations for its answers in terms of natural language, using relationships stored during language analysis and reasoning. The text can then be output to the user along with additional information like the training questions used to generate the selected query cases. Figure 34 shows an example of an automatically generated natural language explanation for how our approach analyzed each part of the original QuaRel problem from Figure 29. In the figure, we can see that the correct phrase is identified for generating the inputs to DQA. In addition, relevant qualitative proportionalities are extracted based on the quantity of the generated logical form, which allows

our approach to justify them with the text used to instantiate the quantity prediction query case. Lastly, the text associated with query cases during training helps explain which answer is selected.

We emphasize that these natural language outputs are automatically produced by our approach. The presence of shallow, word-level statements can be seen (e.g., “resistance” was one such case) and the outputs are not flawless. For instance, in the explanation for the generated quantity and ordinal predictions, the word “on” by itself would seem out of place to an end-user (though we note that the semantic choice underlying this justification was quite important, as it was used to tie key elements of the semantic parse together). Overall, the coherence of the natural language outputs highlights a key strength of our approach, namely that it allows for explanations to be provided for even novice end-users.

9.4 QuaRel Related Work

The original QuaRel work (Tafjord et al., 2019) provided two neural-based semantic parsing models that followed an encoder-decoder framework for generating logical forms. Given a question-and-answer option, the concatenation of the question and answer would be fed to an LSTM encoder which would produce a vector-space representation of the input text. A subsequent decoder architecture would take as input the vector representation and sequentially decode production rules from a formal grammar to build up an abstract syntax tree that would be considered the logical form. Their method is completely neural, generating a logical form for both the question and answer simultaneously (i.e., no qualitative reasoner is used to generate the answer from the logical form of the question).

Subsequent efforts on QuaRel have instead focused on only the multiple-choice portion of the dataset. For instance, (Mitra et al., 2019) proposed translating the logical forms to text such

that a BERT-based (Devlin et al., 2018) textual entailment model could be used to improve multiple-choice performance. Similarly, (Asai and Hajishirzi, 2020) used logic-based rules to extend the training data for QuaRel and enhance a RoBERTa-based (Liu, et al., 2019) multiple-choice selection model.

In a similar shift from the use of formal representations, another dataset for question-answering about qualitative relationships, QuaRTz (Tafjord et al., 2018), was released as a follow-up to QuaRel. Unlike QuaRel which involves reasoning over formal qualitative knowledge specified by a small ontology, QuaRTz involves only textual reasoning. However, despite adopting a purely textual question-answering setting, it was still shown in the benchmark that state-of-the-art language models had difficulty with questions that required qualitative knowledge.

10 Related Work

The high-level approach of using stored solved questions to formulate new answers is conceptually quite similar to prior case-based question-answering techniques, e.g., (Burke, et al. 1997; Lenz, et al. 1998; Weis, K. 2015). Where analogy-based methods (such as ours) typically distinguish themselves from traditional case-based reasoning approaches is their emphasis on structural similarity. That is, case-based reasoning methods do not uniformly hold themselves to enforcing analogical constraints between the comparison of two cases (e.g., structural consistency or systematicity) and instead focus only on representations and similarities that help solve whatever task they have been given (Burstein, 1989).

Inductive logic programming has been used previously to generate rules for question-answering approaches (Calif and Mooney 1999; Zelle and Mooney 1996; Mitra and Baral 2016). Such prior approaches have operated very similarly to our framework, where methods learn rules that map from an initial general parse of a question to logical forms that can be passed to some reasoner. AQA differs from these methods in one critical way, namely, that analogy is front-and-center to our approach. For our work, analogy provides a theoretical commitment to the question of how induced mappings between the initial parse (in our case, the NLU semantic parse) and task-specific logical forms should be applied.

The use of intermediate representations for factoid question-answering with semantic parsing has been explored previously (Cheng et al., 2017; Choi et al., 2015; Kwiatkowski et al., 2013; Reddy et al. 2016); however, such works have generally been subject to two main limitations. First, they used very lightweight intermediate representations constructed on-the-fly from natural language predicates, e.g., dependency parses (Reddy et al., 2016), FunQL (Kate et al.

2005; Cheng et al., 2017), and lambda calculus with Wikitionary-derived (Zesch et al. 2008) features in (Kwiatkowski et al., 2013; Choi et al., 2015). Second, they assume that the correct query for a given question will be isomorphic to the generated intermediate representation. That is, they assume that there will be a structure-preserving one-to-one correspondence between each element of their intermediate representation and their final query form. This assumption clearly does not hold for non-factoid question-answering domains (e.g., with QuaRel or science tests).

Barbella and Forbus (2011) introduced *analogical dialogue acts* (ADAs), which formalize the roles played by individual utterances in instructional analogies. Their approach used the ADAs recognized from the semantic parse of an instructional text to build structured cases that were then compared with SME. Their system used inferences from these analogies to interpret and answer questions. The approach of this work also uses analogical inferences to construct an interpretation of text, however it goes a step further in that QCs are learned from natural language while ADAs were recognized with manually constructed rules.

Chang (2016) combined natural language understanding, spatial reasoning, and analogical reasoning to interpret instructional analogies. These analogies could be used to learn qualitative knowledge. Of particular relevance to AQA was the use of visual representations to disambiguate natural language. Their work used the CogSketch sketch understanding system (Forbus et al., 2011) to represent sketches with the Cyc ontology. CNLU semantic choice sets were disambiguated by selecting those choices that were most related to the outputs of the sketch understanding system.

Khashabi et al. (2017) introduced the notion of essential question terms, which were terms absolutely critical to the understanding of a particular question. They showed that without those

terms, human performance on science test questions dropped significantly. This is conceptually related to the ILP procedure of Section 5.2.2, which learns the essential components of a question needed to infer a particular task-specific form.

Li and Clark (2015) introduced a system that answered multiple choice questions from elementary science tests with connection subgraph techniques. Their system built a connection subgraph that encompassed each answer and the entities of a question. The answer selected was the one that gave rise to the best connection subgraph. Like with Khot et al. (2017), the work here differs in that it builds connection subgraphs to train the system to produce answers to novel questions without the need for multiple-choice questions.

Liang et al., (2016) used unannotated natural language corpora to learn a semantic parser for Yih et al.'s (2016) WebQuestionsSP dataset, a curated subset of Berant et al.'s (2013) WebQuestions corpus answerable using Freebase. Their approach leverages the same insight, i.e., that a knowledge base can provide constraints for interpretation, but requires far more data than our approach.

11 Conclusions

We now revisit the claims and contributions of this thesis and summarize how we provided evidence for them. Following that, we will end with open questions and future work.

11.1 Claims Revisited

1. It is possible to design an approach to question-answering with analogy as its core operation that performs effectively on a variety of question-answering domains and tasks.

In this work, we began by defining an analogical matching procedure based on the Structure Mapping Theory of analogy. As detailed in Sections 4, 5, and 6, this procedure was a critical component of both the training process (where it was used to identify structural regularities across questions that could be used to build query cases) and the testing process (where it was used to adapt learned query cases and propose candidate logical forms for a given question). Thus, it is apparent that analogy should be considered a core operation of AQA. In addition, we showed that AQA could perform well in a variety of different settings. With GeoQuery, we demonstrated the ability of AQA to handle questions involving highly complex, compositional reasoning. With Science Test Process Identification and QuaRel, we demonstrated the ability of AQA to handle longer scenario questions that used broad and varied language.

2. Adapting an existing domain-general semantic parser to domain-specific question-answering tasks leads to far more efficient learning on those tasks than would starting from scratch.

In Section 7, we explored AQA's data efficiency with a learning curve experiment on GeoQuery. We showed that with only 20 questions (out of 600 available) to train on, it could achieve 50%

accuracy, and with only 100 questions its performance became comparable to other systems that used all 600 questions for training.

3. Combining machine learning (in the form of inductive logic programming) with symbolic reasoning methods can produce an approach to semantic parsing that is completely transparent as to what it is learns and how it applies what it learns, while also performing competitively with black-box neural methods.

Because its representations are symbolic s-expressions, the mappings AQA learns (shown in Sections 7, 8, and 9) are readily interpretable by those familiar with predicate calculus. In addition, the means by which AQA learns query cases allows it to maintain the provenance for each query case, i.e., the exact training questions from which a particular query case was induced. We also demonstrated that it is trivial to go beyond formal, logic-based representations. In Section 9, we showed how the representations that AQA learns can be automatically turned into human-understandable natural language explanations by linking the semantic choices used to instantiate a query case to the natural language words and phrases that those semantic choices originated from.

11.2 Contributions Revisited

1. It provides a method, Analogical Question-Answering (AQA), for adapting a general-purpose semantic parser to question-answering tasks in multiple domains that each require different types of reasoning.

This thesis provided a method, AQA, for adapting a general-purpose semantic parser to question-answering tasks. During training, AQA combines analogical reasoning with inductive logic programming to learn query cases, which are rule-like constructs that have as antecedents the

outputs of a domain-general semantic parser and as consequents the domain-specific logical forms needed for the task at hand. During testing, AQA applies the query cases it has learned during training to novel questions through an abductive analogy-based coverage algorithm, which determines the best domain-specific logical forms to return for a given question.

2. It demonstrates that the method can perform on both annotated and unannotated question-answering datasets.

Using two benchmark datasets, we have shown that the method performs competitively with entirely machine learning-based approaches in both unannotated (GeoQuery) and annotated (QuaRel) settings. In addition, with the Science Test Process Identification experiment, we went further and showed how AQA could be applied to the setting where training questions were paired with generic logical forms.

3. It characterizes the conditions that allow AQA to learn in data-sparse situations.

In Section 7, we explored the conditions allowing AQA to achieve data efficiency, with experiments demonstrating that strong constraints imposed on the learning process can be made to force AQA to produce more generalizable query cases from fewer amounts of data.

11.3 Open Questions and Future Work

In this thesis we demonstrated that AQA can map questions involving relatively limited language to complex, compositional logical forms (GeoQuery). We also demonstrated that AQA can handle varied language involving lengthy scenarios when those scenarios are mapped to simple logical forms. It remains to be demonstrated that AQA can handle both of those challenges at once, i.e., map varied language to complex logical forms, while retaining competitiveness with state-of-the-

art methods (e.g., applying AQA to a dataset like WikiTables (Pasupat and Liang, 2015) or WikiSQL (Zhong et al., 2017)).

We believe AQA is strongest when the task-specific logical forms are limited in terms of how varied they are. That is, domains like GeoQuery and QuaRel, though complex in terms of the language they use, involve only a handful of predicates and collections. There are many settings for which this limitation is not an issue (e.g., personal assistants that may only need to map language to one of a few dozen commands), and it is in those settings that AQA should first be applied. However, what would it take to let AQA scale to handling hundreds or even thousands of relations? Certainly, self-annotation would become much more challenging, as the connection graph procedure used to generate queries would face a much more expensive search. When provided annotations, is it possible that the ILP procedure AQA uses during training would be sufficient for handling such tasks?

Lastly, AQA bridges the gap between semantic parser outputs and task-specific logical forms, but should that gap be bridged in a single step? If query cases were hierarchical (i.e., a query case could have as its antecedents the consequents of other query cases), there would effectively be multiple layers of translation between the semantic parser and the final logical form. Prior work has investigated learning higher order relations with ILP (Muggleton et al., 2015) that can concisely capture more complex concepts. In addition to the challenge with learning compositional query cases, the coverage-based nature of the query case composition procedure also does not immediately lend itself to inference involving chaining.

References

Asai, Akari, and Hannaneh Hajishirzi. "Logic-guided data augmentation and regularization for consistent question answering." arXiv preprint arXiv:2004.10157 (2020).

Baker, Collin F., Charles J. Fillmore, and John B. Lowe. "The berkeley framenet project." In 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1, pp. 86-90. 1998.

Banarescu, Laura, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, U. Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer and Nathan Schneider. “Abstract Meaning Representation for Sembanking.” LAW@ACL (2013).

Barbella, David, and Kenneth Forbus. "Analogical dialogue acts: Supporting learning by reading analogies in instructional texts." In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 25, no. 1. 2011.

Berant, Jonathan, Andrew Chou, Roy Frostig, and Percy Liang. "Semantic parsing on freebase from question-answer pairs." In Proceedings of the 2013 conference on empirical methods in natural language processing, pp. 1533-1544. 2013.

Berant, Jonathan, Vivek Srikumar, P. Chen, A. V. Linden, Brittany Harding, Brad Huang, Peter Clark and Christopher D. Manning. “Modeling Biological Processes for Reading Comprehension.” EMNLP (2014).

Bevilacqua, Michele, Rexhina Blloshmi, and Roberto Navigli. "One SPRING to Rule Them Both: Symmetric AMR Semantic Parsing and Generation without a Complex Pipeline." AAAI (2021).

Burke, Robin D., Kristian J. Hammond, Vladimir Kulyukin, Steven L. Lytinen, Noriko Tomuro, and Scott Schoenberg. "Question answering from frequently asked question files: Experiences with the faq finder system." AI magazine 18, no. 2 (1997): 57-57.

Burstein, Mark H. "Analogy vs. CBR: The purpose of mapping." In Proc. of the 2nd Workshop on Case-Based Reasoning, pp. 133-136. 1989.

Califf, Mary Elaine, and Raymond Mooney. "Relational Learning of Pattern-Match Rules for Information Extraction." In CoNLL97: Computational Natural Language Learning. 1997.

Chang, Maria. "Capturing qualitative science knowledge with multimodal instructional analogies." PhD diss., Northwestern University, 2016.

Chen, Kezhen, Irina Rabkina, Matthew D. McLure, and Kenneth D. Forbus. "Human-like sketch object recognition via analogical learning." In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, no. 01, pp. 1336-1343. 2019.

Cheng, Jianpeng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. "Learning Structured Natural Language Representations for Semantic Parsing." In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 44-55. 2017.

Choi, Eunsol, Tom Kwiatkowski, and Luke Zettlemoyer. "Scalable semantic parsing with partial ontologies." 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), pp. 1311-1320. 2015.

Crouse, Maxwell and Forbus, Kenneth D. "Elementary school science as a cognitive system domain: how much qualitative reasoning is required?." Advances in Cognitive Systems 19 (2016).

Crouse, Maxwell, C. McFate and Kenneth D. Forbus. “Learning From Unannotated QA Pairs to Analogically Disambiguate and Answer Questions.” AAAI (2018).

Crouse, Maxwell, C. McFate and Kenneth D. Forbus. “Learning to Build Qualitative Scenario Models From Natural Language.” In Proc. of the 31st Workshop on Qualitative Reasoning (2018).

Darmann, Andreas, Ulrich Pferschy, Joachim Schauer, and Gerhard J. Woeginger. "Paths, trees and matchings under disjunctive constraints." Discrete Applied Mathematics 159, no. 16 (2011): 1726-1735.

De Bruijn, Nicolaas Govert. "Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem." In Indagationes Mathematicae (Proceedings), vol. 75, no. 5, pp. 381-392. North-Holland, 1972.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

Dong, Li, and Mirella Lapata. "Coarse-to-fine decoding for neural semantic parsing." ACL (2018).

Falkenhainer, Brian, Kenneth D. Forbus, and Dedre Gentner. "The structure-mapping engine: Algorithm and examples." Artificial intelligence 41, no. 1 (1989): 1-63.

Faloutsos, Christos, Kevin S. McCurley, and Andrew Tomkins. "Fast discovery of connection subgraphs." In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 118-127. 2004.

Fan, James, and Bruce Porter. "Interpreting loosely encoded questions." In AAAI, pp. 399-405. 2004.

Forbus, Kenneth D. "Qualitative process theory." Artificial intelligence 24, no. 1-3 (1984): 85-168.

Forbus, Kenneth D., Ronald W. Ferguson, Andrew Lovett, and Dedre Gentner. "Extending SME to handle large-scale cognitive modeling." Cognitive Science 41, no. 5 (2017): 1152-1201.

Forbus, Kenneth, Jeffrey Usher, Andrew Lovett, Kate Lockwood, and Jon Wetzel. "CogSketch: Sketch understanding for cognitive science research and for education." Topics in Cognitive Science 3, no. 4 (2011): 648-666.

Gardner, Matt, and Tom Mitchell. "Efficient and expressive knowledge base completion using subgraph feature extraction." In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1488-1498. 2015.

Gentner, Dedre. "Structure-mapping: A theoretical framework for analogy." Cognitive science 7, no. 2 (1983): 155-170.

Guha, Ramanathan V. Contexts: a formalization and some applications. Vol. 101. Stanford, CA: Stanford University, 1991.

Jia, Robin, and Percy Liang. "Data recombination for neural semantic parsing." arXiv preprint arXiv:1606.03622 (2016).

Jia, Robin, and Percy Liang. 2017. "Adversarial Examples for Evaluating Reading Comprehension Systems." Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing.

Rohit J. Kate, Yuk Wah Wong, and Raymond J. Mooney. 2005. Learning to Transform Natural to Formal Languages. In Proceedings for the 20th National Conference on Artificial Intelligence. Pittsburgh, Pennsylvania, pages 1062–1068.

Khashabi, Daniel, Tushar Khot, Ashish Sabharwal, and Dan Roth. "Learning what is essential in questions." In Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017), pp. 80-89. 2017.

Khot, Tushar, Ashish Sabharwal, and Peter Clark. "Answering complex questions using open information extraction." arXiv preprint arXiv:1704.05572 (2017).

Klenk, Matthew, Kenneth Forbus, Emmett Tomai, Hyeonkyeong Kim, and Brian Kyckelhahn. 2005. "Solving Everyday Physical Reasoning Problems by Analogy using Sketches." Proceedings of the AAAI Conference on Artificial Intelligence.

Kowalski, Robert. "A proof procedure using connection graphs." Journal of the ACM (JACM) 22, no. 4 (1975): 572-595.

Kwiatkowski, T., Eunsol Choi, Yoav Artzi and Luke Zettlemoyer. “Scaling Semantic Parsers with On-the-Fly Ontology Matching.” EMNLP (2013).

Kwiatkowksi, Tom, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. "Inducing probabilistic CCG grammars from logical form with higher-order unification." In Proceedings of the 2010 conference on empirical methods in natural language processing, pp. 1223-1233. 2010.

Lao, Ni, Tom Mitchell, and William Cohen. "Random walk inference and learning in a large scale knowledge base." In Proceedings of the 2011 conference on empirical methods in natural language processing, pp. 529-539. 2011.

Lenz, Mario, Andre Hübner, and Mirjam Kunze. "Question answering with textual CBR." In International Conference on Flexible Query Answering Systems, pp. 236-247. Springer, Berlin, Heidelberg, 1998.

Li, Yang, and Peter Clark. "Answering elementary science questions by constructing coherent scenes using background knowledge." In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 2007-2012. 2015.

Liang, Chen, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. "Neural symbolic machines: Learning semantic parsers on freebase with weak supervision." arXiv preprint arXiv:1611.00020 (2016).

Liang, Percy, Michael I. Jordan, and Dan Klein. "Learning dependency-based compositional semantics." Computational Linguistics 39, no. 2 (2013): 389-446.

Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. "Roberta: A robustly optimized bert pretraining approach." arXiv preprint arXiv:1907.11692 (2019).

Marcus, Gary. 2018. "Deep learning: A critical appraisal." arXiv preprint arXiv:1801.00631.

Matuszek, Cynthia, Michael Witbrock, John Cabral, and John DeOliveira. "An introduction to the syntax and content of Cyc." UMBC Computer Science and Electrical Engineering Department Collection (2006).

McFate, Clifton. and Kenneth D. Forbus. “NULEX: An Open-License Broad Coverage Lexicon.” ACL (2011).

McLure, Matthew, Scott Friedman, and Kenneth Forbus. "Extending analogical generalization with near-misses." In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 29, no. 1. 2015.

Mitchell, Tom Michael. Version spaces: an approach to concept learning. STANFORD UNIV CALIF DEPT OF COMPUTER SCIENCE, 1978.

Mitra, Arindam, and Chitta Baral. "Addressing a question answering challenge by combining statistical methods with inductive rule learning and reasoning." In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30, no. 1. 2016.

Mitra, Arindam, Chitta Baral, Aurgho Bhattacharjee, and Ishan Shrivastava. "A generate-validate approach to answering questions about qualitative relationships." arXiv preprint arXiv:1908.03645 (2019).

Muggleton, Stephen, and Luc De Raedt. "Inductive logic programming: Theory and methods." The Journal of Logic Programming 19 (1994): 629-679.

Muggleton, Stephen H., Dianhuan Lin, and Alireza Tamaddoni-Nezhad. "Meta-interpretive learning of higher-order dyadic datalog: Predicate invention revisited." Machine Learning 100, no. 1 (2015): 49-73.

Pasupat, Panupong, and Percy Liang. "Compositional semantic parsing on semi-structured tables." arXiv preprint arXiv:1508.00305 (2015).

Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. "Deep contextualized word representations." arXiv preprint arXiv:1802.05365 (2018).

Quinlan, J. Ross. "Learning logical definitions from relations." Machine learning 5, no. 3 (1990): 239-266.

Rajagopal, Dheeraj, Nidhi Vyas, Aditya Siddhant, Anirudha Rayasam, Niket Tandon and E. Hovy. “Domain Adaptation of SRL Systems for Biological Processes.” BioNLP@ACL (2019).

Reddy, Siva, Oscar Täckström, Michael Collins, T. Kwiatkowski, Dipanjan Das, Mark Steedman and Mirella Lapata. “Transforming Dependency Structures to Logical Forms for Semantic Parsing.” Transactions of the Association for Computational Linguistics 4 (2016): 127-140.

Ribeiro, Danilo, Thomas Hinrichs, Maxwell Crouse, Kenneth Forbus, Maria Chang, and Michael Witbrock. "Predicting state changes in procedural text using analogical question answering." In 7th Annual Conference on Advances in Cognitive Systems. 2019.

Schmitz, Michael, Stephen Soderland, Robert Bart, and Oren Etzioni. "Open language learning for information extraction." In Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning, pp. 523-534. 2012.

Sharma, Abhishek, and Kenneth Forbus. "Graph traversal methods for reasoning in large knowledge-based systems." In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 27, no. 1. 2013.

Tafjord, Oyvind, Peter Clark, Matt Gardner, Wen-tau Yih, and Ashish Sabharwal. 2019. "Quarel: A dataset and models for answering questions about qualitative relationships." Proceedings of the AAAI Conference on Artificial Intelligence.

Tafjord, Oyvind, Matt Gardner, Kevin Lin and P. Clark. “QuaRTz: An Open-Domain Dataset of Qualitative Relationship Questions.” ArXiv abs/1909.03553 (2019): n. pag.

Tomai, Emmett, and Kenneth D. Forbus. "EA NLU: Practical Language Understanding for Cognitive Modeling." In FLAIRS Conference. 2009.

Veale, Tony, and Mark T. Keane. "The competence of sub-optimal theories of structure mapping on hard analogies." In IJCAI (1), pp. 232-237. 1997.

Weis, Karl-Heinz. "A case based reasoning approach for answer reranking in question answering." arXiv preprint arXiv:1503.02917 (2015).

Weld, Daniel. 1990. "Theories of comparative analysis." MIT Press.

Wilson, Jason R., Kezhen Chen, Maxwell Crouse, Constantine Nakos, Danilo Neves Ribeiro, Irina Rabkina, and Kenneth D. Forbus. "Analogical Question Answering in a Multimodal Information Kiosk." In Proceedings of the Seventh Annual Conference on Advances in Cognitive Systems. 2019.

Yih, Wen-tau, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. "The value of semantic parse labeling for knowledge base question answering." In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp. 201-206. 2016.

Zelle, John M., and Raymond J. Mooney. "Learning to parse database queries using inductive logic programming." In Proceedings of the national conference on artificial intelligence, pp. 1050-1055. 1996.

Zesch, Torsten, Christof Müller, and Iryna Gurevych. "Extracting Lexical Semantic Knowledge from Wikipedia and Wiktionary." In LREC, vol. 8, no. 2008, pp. 1646-1652. 2008.

Zettlemoyer, Luke S., and Michael Collins. "Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars." arXiv preprint arXiv:1207.1420 (2012).

Zhang, Sheng, Xutai Ma, Kevin Duh and Benjamin Van Durme. “AMR Parsing as Sequence-to-Graph Transduction.” ACL (2019a).

Zhang, Sheng, Xutai Ma, Kevin Duh and Benjamin Van Durme. “Broad-Coverage Semantic Parsing as Transduction.” ArXiv abs/1909.02607 (2019b)

Zhong, Victor, Caiming Xiong, and Richard Socher. "Seq2sql: Generating structured queries from natural language using reinforcement learning." arXiv preprint arXiv:1709.00103 (2017).