Learning from Unannotated QA Pairs to Analogically Disambiguate and Answer Questions

Maxwell Crouse, Clifton McFate, and Kenneth Forbus
Qualitative Reasoning Group, Northwestern University
mvcrouse@u.northwestern.edu; {c-mcfate, forbus}@northwestern.edu

Abstract

Creating systems that can learn to answer natural language questions has been a longstanding challenge for artificial intelligence. Most prior approaches focused on producing a specialized language system for a particular domain and dataset, and they required training on a large corpus manually annotated with logical forms. This paper introduces an analogy-based approach that instead adapts an existing general purpose semantic parser to answer questions in a novel domain by jointly learning disambiguation heuristics and query construction templates from purely textual question-answer pairs. Our technique uses possible semantic interpretations of the natural language questions and answers to constrain a query-generation procedure, producing cases during training that are subsequently reused via analogical retrieval and composed to answer test questions. Bootstrapping an existing semantic parser in this way significantly reduces the number of training examples needed to accurately answer questions. We demonstrate the efficacy of our technique using the Geoquery corpus, on which it approaches state of the art performance using 10-fold cross validation, shows little decrease in performance with 2-folds, and achieves above 50% accuracy with as few as 10 examples.

Introduction

The rise of software assistants and the need to move beyond web search has led to increased interest in natural language question-answering systems. The standard approach has been to learn, from a corpus annotated with queries in a domain-specific query language, a semantic parser that maps natural language questions to domain specific predicates (e.g. Zelle and Mooney 1996; Zettlemoyer and Collins 2005). Recent approaches have instead used natural language question answer pairs, which are easier to obtain, to obviate the need for annotated corpora. However, they train from scratch for each new predicate vocabulary and these vocabularies are often domain specific with limited expressive power (Liang *et al*. 2013; Berant *et al.* 2013).

While starting from scratch for each vocabulary is attractive from a machine learning perspective, we believe that doing so is both expensive and unnecessary. Our approach is instead to build on a robust general-purpose semantic parser, connected to the highly expressive ResearchCyc[1] ontology, and use analogy to tune its operation to a specific domain.

During training, the semantic parser produces competing sets of potential predicate calculus representations of a question and answer pair. Our approach uses ontological connections between this domain general representation and domain specific facts to guide a search for queries that evaluate to the correct answer and are supported by the semantic representations. The information learned is stored as two kinds of cases: *disambiguation cases* which are analogically retrieved to help disambiguate future text, and *query cases* which are then analogically retrieved and composed to form a query for unseen questions.

The contributions of this approach are two-fold: we provide a method for adapting a general purpose language system to a task in a new domain, and we also demonstrate that adapting an existing system in this way leads to learning with very few examples. We begin by summarizing the analogical processing used and the semantic parser. Then we discuss our technique, followed by an evaluation on the Geoquery benchmark which provides interesting challenges in terms of the compositionality of its language and queries. We close with related and future work.

Background

Analogical Processing

We use Forbus *et al*'s (2016) Structure Mapping Engine (SME), an implementation of Gentner's (1983) Structure Mapping Theory. Given two cases, a base and target consisting of structured descriptions (predicate calculus),

[1] http://www.cyc.com/platform/researchcyc/

SME produces one or more *mappings* that align entities and statements in the two descriptions. These mappings are constrained based on the principles of Structure Mapping Theory: that many to one mappings are forbidden, and that the arguments of matched statements must also align. A greedy algorithm constructs the mappings, preferring those that preserve higher-order structure (e.g. causality) via a trickle-down scoring measure.

Each mapping includes a *score* representing SME's estimate of the similarity of the cases, and *candidate inferences* that describe how structure might be projected from one to the other, based on the mapping. SME has been used to model several psychological phenomena, as well as being used in performance-oriented AI systems (Friedman, Barbella, and Forbus 2012; Klenk and Forbus 2009).

To retrieve cases, we use MAC/FAC (Forbus, Gentner, and Law 1995), which implements human-like similarity-based retrieval using a two-stage map/reduce process. The first stage uses a dot product on vectors constructed from structured descriptions to filter candidate cases, while the second stage uses SME to filter them further.

SAGE is a computational model of analogical generalization (McLure, Friedman, and Forbus 2015). *Generalization pools* contain accumulated examples and generalizations for a concept. When a new example is added, MAC/FAC is used to retrieve the most similar item from the pool. If their similarity is over a threshold, the new example is assimilated into that generalization, if that was what was retrieved, or a new generalization is constructed, if a prior example was retrieved. Frequencies of occurrence for each aligned statement are kept, thus over time, SAGE produces schema-like probabilistic representations.

Semantic Parsing & Disambiguation

Our semantic parser uses ResearchCyc, a broad-coverage domain-independent knowledge base. CycL is more expressive than other ontologies, allowing for higher order logic and efficient contextualization through microtheories (Ramachandran, Reagan, and Goolsbey 2005).

We use Tomai and Forbus' (2009) EA NLU semantic parser to produce potential predicate calculus interpretations of natural language. It uses a bottom-up chart parser with a head-driven feature-based grammar (Allen 1994), and the Comlex lexicon (Grishman, Macleod, and Meyers 1993), which provides lexical information such as part of speech, morphological form, and lexical features. EA NLU is domain independent and has been used to extract qualitative models from a strategy game manual (McFate and Forbus, 2016), for multimodal knowledge capture (Chang and Forbus 2015), and to interpret commonsense stories (Blass and Forbus 2017).

EA NLU makes use of existing mappings between lexical items in the lexicon and neo-Davidsonian semantic frames in the Cyc ontology. As an example, Figure 1 shows the ditransitive frame for the verb *give*, which identifies the role of the subject, object, and oblique-object when instantiated (as in "Joe gave Mary the ball").

```
(verbSemTrans Give-TheWord DitransitiveFrame
  (and (isa :ACTION GivingEvent)
       (giver :ACTION :SUBJECT)
       (givee :ACTION :OBJECT)
       (objectTransferred :ACTION :OBLIQUE)))
```

Figure 1: CycL Semantic Template

Ambiguity, due to multiple word senses and syntactic parses, is explicitly represented as choice sets between alternatives, as well as logical constraints between them. Disambiguation is performed by analogy to prior examples (Barbella and Forbus 2013). Analogical disambiguation uses SAGE to create generalization pools of cases for each word/sense pairing. Disambiguation cases contain both syntactic and semantic information from EA NLU's analysis of a choice that was made. To disambiguate a word, MAC/FAC retrieves the most analogically similar prior example from the union of generalization pools for a word. It is similar to a k-nearest-neighbor approach, but it uses an analogical (structural) measure of similarity and operates over generalizations as well as outliers.

Connection Subgraphs

Connection subgraphs were introduced by Faloutsos, McCurley, and Tomkins (2004) as a way of extracting a set of paths between two nodes in a large graph. The connection subgraph of two nodes is a subgraph that maintains the most relevant paths connecting the two nodes in a larger graph. Here we use connection subgraphs to connect two entities in a knowledge base. Paths of this graph can be translated to queries that evaluate to the entities connected.

Our Technique

We present an implemented technique that learns to answer questions given only unannotated natural language inputs. We first describe learning query and disambiguation cases. We then describe how we use analogical disambiguation to guide retrieval of query cases which are composed into a final query for new questions. Throughout, we use the following two Geoquery questions as examples:

QA Train: What states border Indiana?

QA Test: What states border states that border Colorado?

Learning Query and Disambiguation Cases

During training, the input consists of natural language question-answer pairs. The goal is to produce a query that evaluates to the answer, given the semantic constraints of the question. The initial stage of query-building is viewed as a path finding problem in the knowledge base. With any

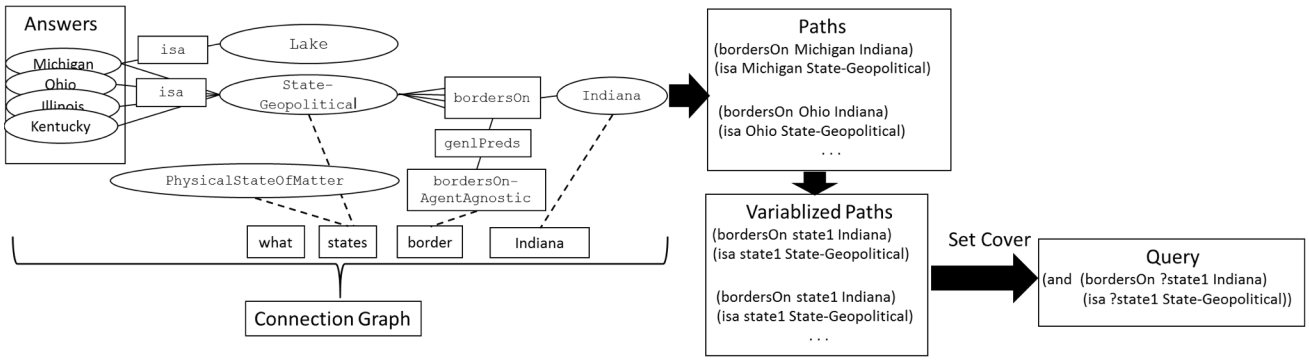

Figure 2: Query Learning Pipeline

sizable knowledge base, the graph is too large to attempt undirected search, so our approach leverages intermediate semantic forms produced by EA NLU. First EA NLU produces a semantic interpretation for both the question and answer. Ambiguities are represented as choice-sets, as noted above. For example, "state" might be either `State-Geopolitical` or `PhysicalStateOfMatter`.

Step 1: Building Paths

For each possible interpretation of each answer, a breadth-first expansion is performed through the knowledge base constrained by the semantic interpretations of the nouns and proper nouns of the question.

In Figure 2, paths branch from possible answer interpretations to the possible interpretations of constituents of the question. 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, we only require paths to align with the interpretations for nouns and proper nouns (e.g. *state* and *Indiana*). This facilitates path finding when there are missing links between the parser semantics and KB facts. In our 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 step 2. In Figure 2, solid lines indicate paths in the KB while dashed lines indicate connections to the EA semantic parse.

In some cases, questions rely on an implicit connection to an entity in the KB (e.g. "How many states?" = "How many states in the United States?"). Our 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). This heuristic is used during query learning and question answering.

When a link in a path aligns with a token in the semantic interpretation (e.g. `Michigan` → `State-Geopolitical` → *state*) it is replaced by the discourse variable (e.g. *state1*) from the parse. Thus, the path `(bordersOn Michigan Indiana)` becomes `(bordersOn state1 Indiana)`. This facilitates query building in step 2 because it maps the relevant KB entities to discourse variables in the question.

Step 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 state1 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 EA's interpretation. The path `(bordersOn state1 Indiana)` receives a higher weight because it connects to `bordersOn-AgentAgnostic` by predicate inheritance (`genlPreds`). Third, with the answer pairings and alignment weights, our approach 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 answers, `Indiana` and the `USA`. Cyc has a finer-grained representation of location that uses different predicates for statehood vs countryhood. This is taken into account by allowing the set-cover to combine differing representations with a disjunction (e.g. the set-cover would produce `(or (cityInState Indianapolis Indiana)(countryOfCity Indianapolis USA) )`). Combined queries that over-generate answers are pruned out, and the top 3 equal scoring queries are turned into cases.

Step 3: Creating Query and Disambiguation Cases

At this point, there is a set of queries and the semantic choices that they are consistent with. Each query is then deconstructed into evidence expressions (EEs). Each EE associates a subexpression of the query with all its linked parts of the semantic interpretation, its antecedents. EEs can be thought of as plausible inferences where the query-expression is the consequent of the choice-antecedents. For

example, Figure 3 shows an EE concluding that evidence for a query containing the `bordersOn` statement is when there are two states that are in a more generic bordering relationship in the EA interpretation.

```
(evidenceForExpression
  (bordersOn (TerritoryFn state211738)
                 (TerritoryFn Indiana-State))
  (and (isa Indiana-State State-UnitedStates)
       (isa state211738 State-Geopolitical)
       (bordersOn-AgentAgnostic Indiana-State
        state211738)))
```

Figure 3: Example Evidence Expression

An EE is generated for each element of the query. Thus, for something like, "What rivers flow through states that border Mississippi?", we would get an EE for each subpart of the question. This allows us to generate queries compositionally during question answering. Each EE is a query case that can be retrieved during question answering and used to compose queries.

Handling Queries Involving Additional Computation

To perform well on question-answering, we must handle questions involving additional computations (e.g. superlatives, cardinality, and summation). Cardinality and summation requires only a minor extension to the process presented above. At first, our method 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, we generate two new queries, 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. EEs are produced for these queries by adding the operator as a consequent to an EE and adding all semantic choices up to and including the first noun in the sentence as antecedents.

Superlatives require more subtlety. For each superlative in the question, our approach finds the word in the query that the superlative predicates (e.g. for "largest state", we are interested in the interpretation of "state"). One advantage of using EA NLU is that superlatives are marked with a feature in the grammar. 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. EEs are produced for superlatives by adding the method and direction of sorting as the consequent to their own EE with the relevant superlative semantics as antecedents.

Disambiguation Cases

In the previous step, our approach produces a set of queries linked to the semantic choices that support them. Choices consistent with the final query are selected and the generalization pool for each word/sense pairing is updated with a case consisting of the selected semantics, the semantics of other words in the sentence, and other lexical features (as in Barbella and Forbus 2013). Essentially, the choices that are consistent with the generated query are used to produce training cases for future disambiguation by analogical retrieval, using MAC/FAC.

Step 4: EE Case Refinement

Sometimes EEs are ambiguous, with different consequents following from the same antecedents. As an example, consider the question "What are the major rivers in Ohio?" In Geobase, every river in Ohio is a major river. Thus, two different queries generate the complete set of answers. As shown in Figure 4, the resulting EEs have the same antecedents since, in this case, neither consequent has a meaningful ontological link to the interpretation of "major" (`SignificantFn`) and it cannot be discounted or preferred in either EE. This erroneously creates an EE where `SignificantFn` is evidence for `flowsInRegion`.

Our solution is to exploit the statistical properties of the antecedent-consequent assignments across the whole case-library to determine incorrect pairings. This is posed as a hypergraph partitioning problem, wherein we partition sets of antecedents so that the number of times an antecedent matches with different consequents is minimized.

```
(evidenceForExpression
 (flowsInRegions river1124 (TerritoryFn Ohio-State))
 (and (isa Ohio-State State-UnitedStates)
      (isa river1124 River)
      (isa river1124 (SignificantFn River))))

(evidenceForExpression
 (majorRiverInRegions river112431
                      (TerritoryFn Ohio-State))
 (and (isa Ohio-State State-UnitedStates)
      (isa river1124 River)
      (isa river1124 (SignificantFn River))))
```

Figure 4: Ambiguous EEs generated for "What are the major rivers in Ohio?"

Hypergraph partitioning algorithms split vertices into disjoint sets such that the sum of hyperedges (sets of vertices) spanning multiple partitions is minimized (Papa and Markov 2007). Here we consider the antecedents of an EE to be vertices, EEs to be hyperedges, and define two vertices to be equivalent if they unify with one another (considering discourse variables and proper names to be variables).

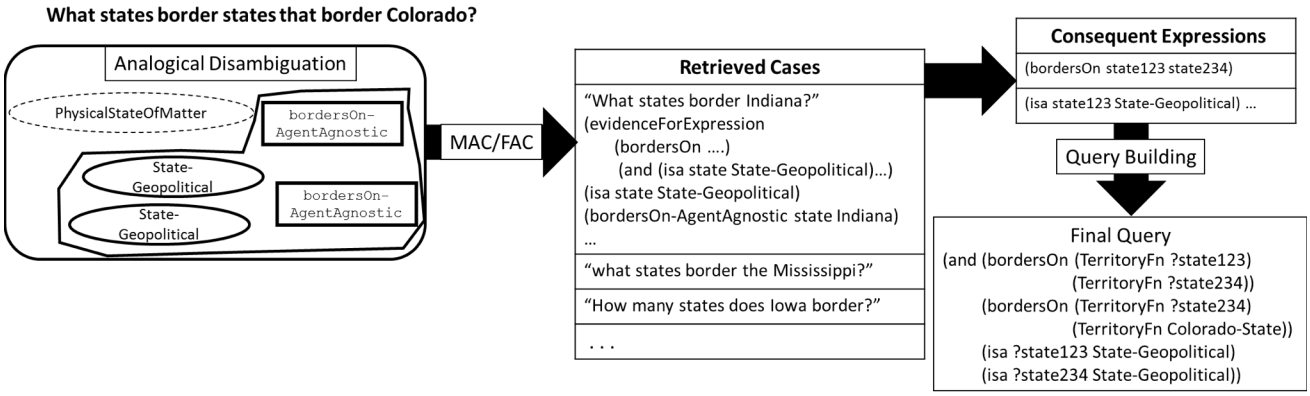

Figure 5: Question Answering Pipeline

We create partitions that minimize antecedent-overlap between EEs with different consequents. This optimization criterion allows us to require certain antecedents to be fixed with a particular consequent. We exploit this by fixing type constraints and ubiquitous antecedents (i.e. not allowing them to be removed from being paired with any consequent). Ubiquitous antecedents are those that appear in the context of at least 4 different type constraints (e.g. `State-Geopolitical, City, IndependentCountry`) since appearance in a variety of different contexts (e.g. a light verb like *have*) indicates that an antecedent shouldn't determine a partition.

EEs are grouped according to their type constraints. Partitioning occurs along those groups, rather than globally. We found this to be necessary so as not to have cases like the semantic interpretation for "largest" being assigned to "largest city" but not "largest state".

The partitioning algorithm simply moves antecedents (semantic choices) from one EE to another to minimize the number of times an antecedent is seen in two EEs with different consequents. In our example, this would result in the antecedent with `SignificantFn` being moved from the EE with consequent `flowsInRegions` to the EE with consequent `majorRiverInRegions` because it appears more frequently in EEs with this consequent. Once partitioning is complete, duplicate EEs are removed and the final set of partitioned EEs form the query case library.

Question Answering

Question answering involves a three-step process.

Step 1: Question Parsing and Disambiguation

The question is parsed by EA NLU and its semantic representations are partially disambiguated using analogical disambiguation with the disambiguation cases learned during training (Barbella and Forbus 2013). In Figure 5, analogical retrieval selects the *state*→`State-Geopolitical` generalization pool as containing the most similar prior usage. Thus, it is selected again. In the event of ties, all equally weighted choices are selected. The resulting semantics will be turned into probes for analogical retrieval of evidence expressions.

Step 2: Analogical Retrieval of Evidence Expressions

The partially disambiguated semantics are turned into a set of probes which MAC/FAC uses to retrieve a small set of EEs from the case library of previously seen examples (in our experiments, we retrieve up to 5 EEs per probe). The FAC stage of MAC/FAC aligns elements of the interpretation of the probe with elements in the retrieved case and produces as candidate inferences a new set of EEs instantiated with the aligned entities from the new question.

When the antecedents in the retrieved EE do not precisely align with the disambiguated probe case, SME introduces skolem entities. These skolems are resolved to entities in the question by matching antecedents containing a skolem to a choice in the current semantic discourse with the same predicate or a superordinate concept.

EEs can be considered abducible in that only a subset of the evidence set needs to be matched to justify the consequent. Returning to our example EE in Figure 3, one of the expressions, `(bordersOn-AgentAgnostic Indiana-State state211738)` would match with the `bordersOn-AgentAgnostic` statement in the semantic interpretation between *states* and *colorado*. If other antecedents were missing, they could be abduced. After being instantiated, the EEs are given an initial ranking according to the percent of their evidence set abduced and the prevalence of facts like the consequent in a domain microtheory.

Step 3: Query Generation

Each EE is associated with a set of choices from the semantic interpretation and a consequent query-expression. The query used to answer the given question will be some set of EE consequents combined together. The EEs selected will also produce a set of choices (the union of their antecedents), which can be considered a complete semantic interpretation of the question. When combining the consequents of selected EEs together, it is required that no two choices can contradict each other. Our query composition

algorithm, shown in Figure 6, finds a set of EEs that have overlapping choices and covers as many choices in the interpretation as is possible. In order to prefer cases which cover new elements the coverage score is cubed, thus ensuring EEs with worse coverage won't be used unless their overlap with prior choices is substantial. The selected EE consequents have their discourse variables transformed into logical variables (e.g. *state123* → *?state123*) and are conjoined to form the query.

```
1. With variables
 1.1. query & choices-made = empty
 1.2. choices-remaining = all semantic choices
2. While choices-remaining not empty
 2.1. For ee = <consequent, choice-antecedents>
         in set of retrieved EEs
  2.1.1.   coverage-score = 1 + length of the
           intersection between choices-remaining and
           choice-antecedents
  2.1.2.   overlap-score = 1 + length of the
           intersection between choices-made and
           choice-antecedents
  2.1.3.   conflict-score = 1 + number of choices in
           choices-remaining that conflict with some
           choice in choice-antecedents
  2.1.4.   if coverage-score == 1
           then: overall-score of ee = 0
           else: overall-score of ee =
                 coverage-score^3 * overlap-score
                        conflict-score
 2.2. max-ee = EE with maximum overall-score
 2.3. Add choices in max-ee to choices-made
 2.4. Remove choices in max-ee from choices-remaining
 2.5. Remove choices conflicting with choice in max-
      ee from choices-remaining
 2.6. Remove EEs with choices conflicting with
      choices in max-ee
 2.7. Add EE consequent to query
```

Figure 6: Query Composition Algorithm

Experiment

We 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. We scored our system's answer as correct only if it exactly matched the answer generated by Geoquery.

Cyc contains a substantial amount of pre-existing geographical knowledge. In order to ensure a fair comparison to other Geoquery systems, we restricted path search to exclude these geography-related microtheories. When we did not, the learned queries actually generated correct answers that Geobase did not account for. For example, the query our approach learns 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 Cyc's geography microtheories, the returned answer is a more complete set of cities in Indiana, because Cyc knows about more cities in Indiana *and* Cyc uses the same predicate, `cityInState`, to relate each city to its respective state.

During the experiment, we noticed that 22 of the Geoquery queries evaluated to incorrect answers when run in the Prolog interpreter. For example, both "What state has the longest river?" and "What state has the shortest river?" evaluate to all states with rivers. We were not sure how those were accounted for in the train/test splits of other approaches, so we ran experiments with and without those answers corrected.

Results

We present our results in comparison to other high performing Geoquery systems in Table 1. Consistent with several prior systems (e.g. Wong and Mooney 2007; Lu *et al*. 2008) we evaluate using 10-fold cross-validation over the entire set of 880 questions. We also evaluate using 2-fold cross-validation. On the corrected data-set, we achieve an average 10-fold performance of 88.8% accuracy. This only drops slightly, to 87.4% with far fewer training examples in the 2-fold experiment.

	Train/Test Split	Answer (%)
KZGS10 (Kwiatkowski *et al,* 2010)	680/200	88.9
LJK11 w/ base triggers (Liang *et al*, 2013)	250/250	84.0
LJK11 w/ base triggers (Liang *et al*, 2013)	680/200	87.9
LJK11 w/ augmented triggers (Liang *et al*, 2013)	250/250	87.6
LJK11 w/ augmented triggers (Liang *et al*, 2013)	680/200	91.4
Our System	2-fold (Uncorrected)	84.8
Our System	2-fold (Corrected)	87.4
Our System	10-fold (Uncorrected)	86.3
Our System	10-fold (Corrected)	88.8

Table 1: Geoquery Results

The first system in Table 1 (KZGS10) trains on annotated logical forms and, to our knowledge, is the top performing system that does so. On the corrected data-set, we perform comparably despite using unannotated training data.

Liang, Jordan, and Klein (2013) provides the current state of the art on Geoquery question answering and, like us, train using unannotated 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. They evaluate on Zettlemoyer and Collins (2005) 600/280 train-test split as well as on a 250/250 split.

A detailed comparison and contrast to prior systems is difficult, as our approach adapts an existing general purpose system and theirs learns a domain-specific parser, thus making fundamentally different assumptions. For instance, our approach must deal with distracting non-domain semantics (e.g. ***state*** as a state of matter). Still, our 2-fold and 10-fold corrected-set performance (87.4/88.8) is comparable to their 600/280 performance with base triggers (87.9) and only slightly out-performed by their system with manually augmented-triggers (91.4).

One theoretical benefit of our approach is that it only needs to see each part of a query once to apply those parts to a novel question. Given that there was not a large decrease between 10 and 2-fold performance we were interested in how few training examples were required. We thus evaluated on a set of training questions ranging from 10 to 100 QA-pairs automatically selected to maximize the number of questions covered. The questions were from the corrected set, and any questions not used for training were used for testing (e.g. we used 860 questions for testing after training with 20 questions).

In our minimum set of ten, performance decreased to 51.6%. We reach comparable performance to 10-fold training with only 100 QA-pairs (85.6%) and exceed 80% correct with only 60. The learning curve from 10-100 examples is shown below.

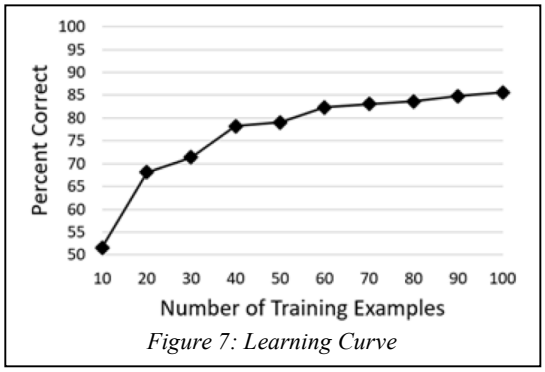

Figure 7: Learning Curve

The ability to extract large performance gains from very little data is a chief contribution of our approach. To our knowledge, none of the other comparable systems have evaluated their performance with so few examples.

Related Work

Our work dovetails nicely with related work in semantic parsing. For example, 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. Our connection-graph technique is like bridging in that it proposes expressions to connect entities in the question. Analogical retrieval could be used to infer logical forms when bridging fails.

IBM's Watson is an interesting hybrid model of question answering. Watson relies on the PRISMATIC KB which is constructed automatically via syntactic analysis of a massive corpus of text, but there is additional use of DBPedia and YAGO content for special purposes (Fan *et al*. 2012). Watson uses a plethora of methods to generate and evaluate answers. By contrast, our system requires a knowledge base with formally represented knowledge, from which it learns (with many fewer examples than Watson) to answer questions within a narrower domain.

Our work is perhaps most similar to Li and Clark (2015) which answered multiple choice questions from elementary science tests with connection subgraph techniques. Their system builds a connection subgraph that encompasses each answer and the entities of the question. The answer selected was the one that gave rise to the best connection subgraph. Our work differs in that we build connection subgraphs to train our system to produce answers to novel questions without the need for multiple-choice questions.

Sharma and Forbus (2013) produce horn axioms for the Cyc knowledge base with connection subgraph techniques. Given a fully instantiated 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, we start with natural language and use QA-pairs to learn query patterns rather than higher-order knowledge.

Our use of connection subgraphs is similar to recent applications of path ranking algorithms for relational learning (Lao, Mitchell, and Cohen 2011). With this approach, paths are found via random walks. By contrast, our method 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. Our constraint comes 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).

Finally, Liang *et al* (2016) also uses 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 that a

knowledge base can provide constraints for interpretation, but requires far more data than our approach.

Conclusions and Future Work

We have shown that analogical retrieval and connection subgraph techniques can be used to adapt a domain-general natural language understanding system to answer questions in a new domain. In doing so, our approach learns strategies for both future disambiguation and question answering. It trains using only natural language QA-pairs, rather than hand-annotated corpora, approaches state of the art performance with 10-fold cross-validation, and is able to reach high performance with relatively few examples.

Immediate future work will involve testing on other corpora such as WebQuestionsSP to ensure the generality of the approach. One downside of WebQuestions is that questions are limited to wh-questions with one entity, and thus they do not evaluate compositionality as well as Geoquery. We also plan to expand this approach to answer science questions similar to those of Li and Clark (2015).

Furthermore, adapting a symbolic language system has additional benefits. Symbolic representations tied to a large ontology are inspectable, both by people and machines, enabling self-explanation during learning and justifications for answers during interaction. In future work, we hope to elaborate upon these unique affordances of our approach.

We noted above that our approach would often over-generate correct answers when the queries it learned with respect to the Geobase microtheory were evaluated with respect to the Cyc knowledge base as a whole. This observation gives rise to an interesting question: Do we really need a complete and exact set of answers to learn the correct query for a given question? It may be possible to generate the correct query given only a handful of the answers to a question, thus alleviating the training data bottleneck even further. Consider a question where it would be difficult for a human to find the exact set of answers needed for existing semantic parsing approaches to train, "What states border states that border Texas?" If one were to reason through that question, they might come up with a few states like *Kansas*, *Arizona*, and *Mississippi*. Not many paths in our knowledge base can both reflect the semantics of the question *and* join those states to *Texas* in the same way. We plan to explore how few answers are needed to generate correct queries, and what modifications to our approach are needed to handle such question-answering with incomplete QA pairs.

Finally, this technique has applications in language learning and learning by reading. It discovers paths in the absence of linguistic support, thus the approach could use QA pairs to discover semantic templates for unknown words. For example, the word *abut* in "What states abut Texas?" may be unknown, but the path generated between the answers and Texas provides the meaning. The approach could also be used to create links between predicates in a knowledge base. Consider, "What rivers flow through states that border Mississippi?" we found that EA's representation of *flow* and the path did not align. However, the path's predicate, `flowsInRegion`, should be connected to EA's more general representation of motion. Our approach could generate such connections.

References

Allen, J. F. 1994. *Natural language understanding (2nd Ed.)* Benjamin/Cummings, Redwood City, CA.

Barbella, D. and Forbus, K. 2013. Analogical Word Sense Disambiguation. *In Advances in Cognitive Systems*, 2:297-315,

Blass, J. A. and Forbus, K. D. 2017. Analogical Chaining with Natural Language Instruction for Commonsense Reasoning. *In Proceedings of AAAI-2017*, San Francisco, CA

Berant, J.; Chou, A.; Frostig, R.; and Liang P. 2013. Semantic parsing on Freebase from question-answer pairs. *In Empirical Methods in Natural Language Processing (EMNLP)*, 2(5):6-18

Chang, M. D. and Forbus, K. D. 2015. Towards Interpretation Strategies for Multimodal Instructional Analogies. *In the Proceedings of the 28th International Workshop on Qualitative Reasoning (QR2015)*, Minneapolis, MN

Faloutsos, C.; McCurley, K. S.; and Tomkins A. 2004. Fast discovery of connection subgraphs. *In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 118-127, Chicago, Illinois

Fan, J., Kalyanpur, A., Gondek, D. C., & Ferrucci, D. A. (2012). Automatic knowledge extraction from documents. *IBM Journal of Research and Development*, 56(3.4), 5-1.

Forbus, K.; Ferguson, R. W.; Lovett, A.; and Gentner, D. 2016. Extending SME to handle large-scale cognitive modeling. *Cognitive Science*, pages 1-50

Forbus, K.D.; Gentner, D.; and Law, K. 1995. MAC/FAC: A model of similarity-based retrieval. *Cognitive Science*, 19(2), pp.141-205.

Friedman, S. E.; Barbella, D. M.; and Forbus, K. D. 2012. Revising domain knowledge with cross-domain analogy. *Advances in Cognitive Systems*, *2*, 13-24.

Gardner, M. and Mitchell, T. M. (2015). Efficient and Expressive Knowledge Base Completion Using Subgraph Feature Extraction. In *EMNLP* (pp. 1488-1498).

Grishman, R.; Macleod, C.; and Meyers A. 1994. COMLEX syntax: Building a computational lexicon. *In Proceedings of the 15th Conference on Computational Linguistics,* pages 268-272, Association for Computational Linguistics.

Klenk, M. and Forbus, K. 2009. Analogical model formulation for transfer learning in AP Physics. *Artificial intelligence*, *173*(18), 1615-1638.

Kwiatkowski, T.; Zettlemoyer, L.; Goldwater, S.; and Steedman, M. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. *In Proceedings of the 2010 conference on empirical methods in natural language processing.* (pp. 1223-1233). Association for Computational Linguistics.

Lao, N.; Mitchell, T.; and Cohen, W. W. (2011, July). Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (pp. 529-539). Association for Computational Linguistics.

Li, Y. and Clark, P. 2015. Answering Elementary Science Questions by Constructing Coherent Scenes using Background Knowledge. I*n Proceedings of EMNLP 2015*, pages 2007-2012, Lisbon, Portugal

Liang, P.; Jordan, M. I.; and Klein, D. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389-446

Liang, C.; Berant, J.; Le, Q.; Forbus, K. D.; and Lao, N. 2016. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *In Proceedings of the Neural Abstract Machines and Program Induction Workshop*, Barcelona, Spain

Lovett, A. and Forbus, K. 2012. Modeling multiple strategies for solving geometric analogy problems. *In Proceedings of the 34th Annual Conference of the Cognitive Science Society*. Sapporo, Japan

Lovett, A.; Forbus, K.; and Usher, J. 2010. A structure-mapping model of Raven's Progressive Matrices. *In Proceedings of CogSci-10*, pages 2761-2766

Lu, W.; Ng, H. T.; Lee, W. S.; and Zettlemoyer, L. S. 2008. A generative model for parsing natural language to meaning representations. *In Empirical Methods in Natural Language Processing(EMNLP)*, pages 783–792, Honolulu, HI.

McFate, C. J. and Forbus, K. 2016. Scaling up Linguistic Processing of Qualitative Process Interpretation. In *Proceedings of the Fourth Annual Conference on Advances in Cognitive Systems*, Evanston, IL

McLure, M. D.; Friedman, S. E.; and Forbus, K. D. 2015. Extending Analogical Generalization with Near-Misses. *In Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 565-571

Papa, D.A. and Markov, I.L., 2007. *Hypergraph Partitioning and Clustering.*

Ramachandran, D.; Reagan, P.; and Goolsbey, K. 2005. First-Orderized Research Cyc: Expressivity and Efficiency in a Common-Sense Ontology. *In Papers from the AAAI Workshop on Contexts and Ontologies: Theory, Practice and Applications,* Pittsburgh, PA

Sharma, A. and Forbus, K. 2013. Graph Traversal Methods for Reasoning in Large Knowledge-Based Systems. *In Proceedings of AAAI 2013*. Bellevue, WA

Tomai, E. and Forbus, K. 2009. EA NLU: Practical Language Understanding for Cognitive Modeling. *Proceedings of the 22nd International Florida Artificial Intelligence Research Society Conference*, Sanibel Island, Florida.

Wong, Y.W. and Mooney, R.J., 2007. Learning synchronous grammars for semantic parsing with lambda calculus. (Vol. 45, No. 1, p. 960).

Yih, W.; Richardson, M.; Meek, C.; Chang, M.; and Suh, J. 2016. The value of semantic parse labeling for knowledge base question answering. *In the Proceedings of ACL* 2016, pages 201-206

Zettlemoyer, L. S. and Collins, M. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *In Uncertainty in Artificial Intelligence (UAI), pages 658–666*

Zelle, J. M. and Mooney, R. J.. 1996. Learning to parse database queries using inductive logic programming. *In Proceedings of the national conference on artificial intelligence*, pages 105-1055