Paraphrase-Driven Learning for Open Question Answering

Anthony Fader **Luke Zettlemoyer** **Oren Etzioni**
Computer Science & Engineering
University of Washington
Seattle, WA 98195
{afader, lsz, etzioni}@cs.washington.edu

Abstract

We study question answering as a machine learning problem, and induce a function that maps open-domain questions to queries over a database of web extractions. Given a large, community-authored, question-paraphrase corpus, we demonstrate that it is possible to learn a semantic lexicon and linear ranking function without manually annotating questions. Our approach automatically generalizes a seed lexicon and includes a scalable, parallelized perceptron parameter estimation scheme. Experiments show that our approach more than quadruples the recall of the seed lexicon, with only an 8% loss in precision.

1 Introduction

Open-domain question answering (QA) is a long-standing, unsolved problem. The central challenge is to automate every step of QA system construction, including gathering large databases and answering questions against these databases. While there has been significant work on large-scale information extraction (IE) from unstructured text (Banko et al., 2007; Hoffmann et al., 2010; Riedel et al., 2010), the problem of answering questions with the noisy knowledge bases that IE systems produce has received less attention. In this paper, we present an approach for learning to map questions to formal queries over a large, open-domain database of extracted facts (Fader et al., 2011).

Our system learns from a large, noisy, question-paraphrase corpus, where question clusters have a common but unknown query, and can span a diverse set of topics. Table 1 shows example paraphrase clusters for a set of factual questions. Such data provides strong signal for learning about lexical variation, but there are a number of challenges. Given that the data is community-authored, it will inevitably be incomplete, contain incorrectly tagged paraphrases, non-factual questions, and other sources of noise.

Who wrote the Winnie the Pooh books? Who is the author of winnie the pooh? What was the name of the authur of winnie the pooh? Who wrote the series of books for Winnie the poo? Who wrote the children's storybook 'Winnie the Pooh'? Who is poohs creator?
What relieves a hangover? What is the best cure for a hangover? The best way to recover from a hangover? Best remedy for a hangover? What takes away a hangover? How do you lose a hangover? What helps hangover symptoms?
What are social networking sites used for? Why do people use social networking sites worldwide? Advantages of using social network sites? Why do people use social networks a lot? Why do people communicate on social networking sites? What are the pros and cons of social networking sites?
How do you say Santa Claus in Sweden? Say santa clause in sweden? How do you say santa clause in swedish? How do they say santa in Sweden? In Sweden what is santa called? Who is sweden santa?

Table 1: Examples of paraphrase clusters from the WikiAnswers corpus. Within each cluster, there is a wide range of syntactic and lexical variations.

Our core contribution is a new learning approach that scalably sifts through this paraphrase noise, learning to answer a broad class of factual questions. We focus on answering open-domain questions that can be answered with single-relation queries, *e.g.* all of the paraphrases of "Who wrote Winnie the Pooh?" and "What cures a hangover?" in Table 1. The algorithm answers such questions by mapping them to executable queries over a tuple store containing relations such as `authored(milne, winnie-the-pooh)` and `treat(bloody-mary, hangover-symptoms)`.

The approach automatically induces lexical structures, which are combined to build queries for unseen questions. It learns lexical equivalences for relations (*e.g.*, *wrote*, *authored*, and *creator*), entities (*e.g.*, *Winnie the Pooh* or *Pooh Bear*), and question templates (*e.g.*, *Who* `r` *the* `e` *books?* and *Who is the* `r` *of* `e`*?*). Crucially, the approach does not require any explicit labeling of the questions in our paraphrase corpus. Instead, we use 16 seed question templates and string-matching to find high-quality queries for a small subset of the questions. The algorithm uses learned word alignments to aggressively generalize the seeds, producing a large set of possible lexical equivalences. We then learn a linear ranking model to filter the learned lexical equivalences, keeping only those that are likely to answer questions well in practice.

Experimental results on 18 million paraphrase pairs gathered from WikiAnswers[1] demonstrate the effectiveness of the overall approach. We performed an end-to-end evaluation against a database of 15 million facts automatically extracted from general web text (Fader et al., 2011). On known-answerable questions, the approach achieved 42% recall, with 77% precision, more than quadrupling the recall over a baseline system.

In sum, we make the following contributions:

- We introduce PARALEX, an end-to-end open-domain question answering system.
- We describe scalable learning algorithms that induce general question templates and lexical variants of entities and relations. These algorithms require no manual annotation and can be applied to large, noisy databases of relational triples.
- We evaluate PARALEX on the end-task of answering questions from WikiAnswers using a database of web extractions, and show that it outperforms baseline systems.
- We release our learned lexicon and question-paraphrase dataset to the research community, available at `http://openie.cs.washington.edu`.

2 Related Work

Our work builds upon two major threads of research in natural language processing: information extraction (IE), and natural language interfaces to databases (NLIDB).

Research in IE has been moving towards the goal of extracting facts from large text corpora, across many domains, with minimal supervision (Mintz et al., 2009; Hoffmann et al., 2010; Riedel et al., 2010; Hoffmann et al., 2011; Banko et al., 2007; Yao et al., 2012). While much progress has been made in converting text into structured knowledge, there has been little work on answering natural language questions over these databases. There has been some work on QA over web text (Kwok et al., 2001; Brill et al., 2002), but these systems do not operate over extracted relational data.

The NLIDB problem has been studied for decades (Grosz et al., 1987; Katz, 1997). More recently, researchers have created systems that use machine learning techniques to automatically construct question answering systems from data (Zelle and Mooney, 1996; Popescu et al., 2004; Zettlemoyer and Collins, 2005; Clarke et al., 2010; Liang et al., 2011). These systems have the ability to handle questions with complex semantics on small domain-specific databases like GeoQuery (Tang and Mooney, 2001) or subsets of Freebase (Cai and Yates, 2013), but have yet to scale to the task of general, open-domain question answering. In contrast, our system answers questions with more limited semantics, but does so at a very large scale in an open-domain manner. Some work has been made towards more general databases like DBpedia (Yahya et al., 2012; Unger et al., 2012), but these systems rely on hand-written templates for question interpretation.

The learning algorithms presented in this paper are similar to algorithms used for paraphrase extraction from sentence-aligned corpora (Barzilay and McKeown, 2001; Barzilay and Lee, 2003; Quirk et al., 2004; Bannard and Callison-Burch, 2005; Callison-Burch, 2008; Marton et al., 2009). However, we use a paraphrase corpus for extracting lexical items relating natural language patterns to database concepts, as opposed to relationships between pairs of natural language utterances.

3 Overview of the Approach

In this section, we give a high-level overview of the rest of the paper.

Problem Our goal is to learn a function that will map a natural language question x to a query z over a database D. The database D is a collection of assertions in the form $r(e_1, e_2)$ where r is a bi-

[1] `http://wiki.answers.com/`

nary relation from a vocabulary R, and e_1 and e_2 are entities from a vocabulary E. We assume that the elements of R and E are human-interpretable strings like `population` or `new-york`. In our experiments, R and E contain millions of entries representing ambiguous and overlapping concepts. The database is equipped with a simple interface that accepts queries in the form $r(?, e_2)$ or $r(e_1, ?)$. When executed, these queries return all entities e that satisfy the given relationship. Thus, our task is to find the query z that best captures the semantics of the question x.

Model The question answering model includes a lexicon and a linear ranking function. The lexicon L associates natural language patterns to database concepts, thereby defining the space of queries that can be derived from the input question (see Table 2). Lexical entries can pair strings with database entities (*nyc* and `new-york`), strings with database relations (*big* and `population`), or question patterns with templated database queries (*how* `r` *is* `e`*?* and `r(?,e)`). We describe this model in more detail in Section 4.

Learning The learning algorithm induces a lexicon L and estimates the parameters θ of the linear ranking function. We learn L by bootstrapping from an initial seed lexicon L_0 over a corpus of question paraphrases $\mathcal{C} = \{(x, x') : x' \text{ is a paraphrase of } x\}$, like the examples in Table 1. We estimate θ by using the initial lexicon to automatically label queries in the paraphrase corpus, as described in Section 5.2. The final result is a scalable learning algorithm that requires no manual annotation of questions.

Evaluation In Section 8, we evaluate our system against various baselines on the end-task of question answering against a large database of facts extracted from the web. We use held-out known-answerable questions from WikiAnswers as a test set.

4 Question Answering Model

To answer questions, we must find the best query for a given natural language question.

4.1 Lexicon and Derivations

To define the space of possible queries, PARALEX uses a lexicon L that encodes mappings from natural language to database concepts (entities, relations, and queries). Each entry in L is a pair (p, d) where p is a pattern and d is an associated database concept. Table 2 gives examples of the entry types in L: entity, relation, and question patterns.

Entry Type	NL Pattern	DB Concept
Entity	*nyc*	`new-york`
Relation	*big*	`population`
Question (1-Arg.)	*how big is* `e`	`population(?, e)`
Question (2-Arg.)	*how* `r` *is* `e`	`r(?, e)`

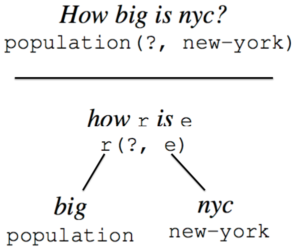
Table 2: Example lexical entries.

Entity patterns match a contiguous string of words and are associated with some database entity $e \in E$.

Relation patterns match a contiguous string of words and are associated with a relation $r \in R$ and an argument ordering (*e.g.* the string *child* could be modeled as either `parent-of` or `child-of` with opposite argument ordering).

Question patterns match an entire question string, with gaps that recursively match an entity or relation patterns. Question patterns are associated with a templated database query, where the values of the variables are determined by the matched entity and relation patterns. A question pattern may be **1-Argument**, with a variable for an entity pattern, or **2-Argument**, with variables for an entity pattern and a relation pattern. A 2-argument question pattern may also invert the argument order of the matched relation pattern, *e.g.* *who* `r` `e`*?* may have the opposite argument order of *who did* `e` `r`*?*

The lexicon is used to generate a derivation y from an input question x to a database query z. For example, the entries in Table 2 can be used to make the following derivation from the question *How big is nyc?* to the query `population(?, new-york)`:



This derivation proceeds in two steps: first matching a question form like *How* `r` *is* `e`*?* and then mapping *big* to `population` and *nyc* to `new-york`. Factoring the derivation this way allows the lexical entries for *big* and *nyc* to be reused in semantically equivalent variants like *nyc how big is it?* or *approximately how big is nyc?* This factorization helps the system generalize to novel questions that do not appear in the training set.

We model a derivation as a set of (p_i, d_i) pairs, where each p_i matches a substring of x, the substrings cover all words in x, and the database concepts d_i compose to form z. Derivations are rooted at either a 1-argument or 2-argument question entry and have entity or relation entries as leaves.

4.2 Linear Ranking Function

In general, multiple queries may be derived from a single input question x using a lexicon L. Many of these derivations may be incorrect due to noise in L. Given a question x, we consider all derivations y and score them with $\theta \cdot \phi(x, y)$, where $\phi(x, y)$ is a n-dimensional feature representation and θ is a n-dimensional parameter vector. Let $\text{GEN}(x; L)$ be the set of all derivations y that can be generated from x using L. The best derivation $y^*(x)$ according to the model (θ, L) is given by:

$$y^*(x) = \underset{y \in \text{GEN}(x;L)}{\arg\max} \; \theta \cdot \phi(x, y)$$

The best query $z^*(x)$ can be computed directly from the derivation $y^*(x)$.

Computing the set $\text{GEN}(x; L)$ involves finding all 1-Argument and 2-Argument question patterns that match x, and then enumerating all possible database concepts that match entity and relation strings. When the database and lexicon are large, this becomes intractable. We prune $\text{GEN}(x; L)$ using the model parameters θ by only considering the N-best question patterns that match x, before additionally enumerating any relations or entities.

For the end-to-end QA task, we return a ranked list of answers from the k highest scoring queries. We score an answer a with the highest score of all derivations that generate a query with answer a.

5 Learning

PARALEX uses a two-part learning algorithm; it first induces an overly general lexicon (Section 5.1) and then learns to score derivations to increase accuracy (Section 5.2). Both algorithms rely on an initial seed lexicon, which we describe in Section 7.4.

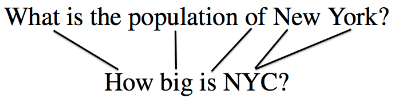
5.1 Lexical Learning

The lexical learning algorithm constructs a lexicon L from a corpus of question paraphrases $\mathcal{C} = \{(x, x') : x' \text{ is a paraphrase of } x\}$, where we assume that all paraphrased questions (x, x') can be answered with a single, initially unknown, query (Table 1 shows example paraphrases). This assumption allows the algorithm to generalize from the initial seed lexicon L_0, greatly increasing the lexical coverage.

As an example, consider the paraphrase pair x = *What is the population of New York?* and x' = *How big is NYC?* Suppose x can be mapped to a query under L_0 using the following derivation y:

what is the `r` *of* `e`	=	`r(?, e)`
population	=	`population`
new york	=	`new-york`

We can induce new lexical items by aligning the patterns used in y to substrings in x'. For example, suppose we know that the words in (x, x') align in the following way:



Using this information, we can hypothesize that *how* `r` *is* `e`, *big*, and *nyc* should have the same interpretations as *what is the* `r` *of* `e`, *population*, and *new york*, respectively, and create the new entries:

how `r` *is* `e`	=	`r(?, e)`
big	=	`population`
nyc	=	`new-york`

We call this procedure $\text{InduceLex}(x, x', y, A)$, which takes a paraphrase pair (x, x'), a derivation y of x, and a word alignment A, and returns a new set of lexical entries. Before formally describing InduceLex we need to introduce some definitions.

Let n and n' be the number of words in x and x'. Let $[k]$ denote the set of integers $\{1, \ldots, k\}$. A word alignment A between x and x' is a subset of $[n] \times [n']$. A phrase alignment is a pair of index sets (I, I') where $I \subseteq [n]$ and $I' \subseteq [n']$. A phrase alignment (I, I') is consistent with a word alignment A if for all $(i, i') \in A$, $i \in I$ if and only if $i' \in I'$. In other words, a phrase alignment is consistent with a word alignment if the words in the phrases are aligned only with each other, and not with any outside words.

We will now define $\text{InduceLex}(x, x', y, A)$ for the case where the derivation y consists of a 2-argument question entry (p_q, d_q), a relation entry

```
function LearnLexicon
  Inputs:
  - A corpus $\mathcal{C}$ of paraphrases $(x, x')$. (Table 1)
  - An initial lexicon $L_0$ of (pattern, concept) pairs.
  - A word alignment function $\text{WordAlign}(x, x')$. (Section 6)
  - Initial parameters $\theta_0$.
  - A function $\text{GEN}(x; L)$ that derives queries from a question $x$ using lexicon $L$. (Section 4)
  - A function $\text{InduceLex}(x, x', y, A)$ that induces new lexical items from the paraphrases $(x, x')$ using their word alignment $A$ and a derivation $y$ of $x$. (Section 5.1)
  Output: A learned lexicon $L$.
  $L = \{\}$
  for all $x, x' \in \mathcal{C}$ do
    if $\text{GEN}(x; L_0)$ is not empty then
      $A \leftarrow \text{WordAlign}(x, x')$
      $y^* \leftarrow \arg\max_{y \in \text{GEN}(x; L_0)} \theta_0 \cdot \phi(x, y)$
      $L \leftarrow L \cup \text{InduceLex}(x, x', y^*, A)$
  return $L$
```

Figure 1: Our lexicon learning algorithm.

(p_r, d_r), and an entity entry (p_e, d_e), as shown in the example above.[2] InduceLex returns the set of all triples $(p'_q, d_q), (p'_r, d_r), (p'_e, d_e)$ such that for all p'_q, p'_r, p'_e such that

1. p'_q, p'_r, p'_e are a partition of the words in x'.
2. The phrase pairs $(p_q, p'_q), (p_r, p'_r), (p_e, p'_e)$ are consistent with the word alignment A.
3. The p'_r and p'_e are contiguous spans of words in x'.

Figure 1 shows the complete lexical learning algorithm. In practice, for a given paraphrase pair (x, x') and alignment A, InduceLex will generate multiple sets of new lexical entries, resulting in a lexicon with millions of entries. We use an existing statistical word alignment algorithm for WordAlign (see Section 6). In the next section, we will introduce a scalable approach for learning to score derivations to filter out lexical items that generalize poorly.

5.2 Parameter Learning

Parameter learning is necessary for filtering out derivations that use incorrect lexical entries like *new mexico* = mexico, which arise from noise in the paraphrases and noise in the word alignment. We use the hidden variable structured perceptron algorithm to learn θ from a list of (question x, query z) training examples. We adopt the iterative parameter mixing variation of the perceptron (McDonald et al., 2010) to scale to a large number of training examples.

Figure 2 shows the parameter learning algorithm. The parameter learning algorithm operates in two stages. First, we use the initial lexicon L_0 to automatically generate (question x, query z) training examples from the paraphrase corpus $\mathcal{C}$. Then we feed the training examples into the learning algorithm, which estimates parameters for the learned lexicon L.

Because the number of training examples is large, we adopt a parallel perceptron approach. We first randomly partition the training data $\mathcal{T}$ into K equally-sized subsets $\mathcal{T}_1, \ldots, \mathcal{T}_K$. We then perform perceptron learning on each partition in parallel. Finally, the learned weights from each parallel run are aggregated by taking a uniformly weighted average of each partition's parameter vector. This procedure is repeated for T iterations.

The training data consists of (question x, query z) pairs, but our scoring model is over (question x, derivation y) pairs, which are unobserved in the training data. We use a hidden variable version of the perceptron algorithm (Collins, 2002), where the model parameters are updated using the highest scoring derivation y^* that will generate the correct query z using the learned lexicon L.

6 Data

For our database D, we use the publicly available set of 15 million ReVerb extractions (Fader et al., 2011).[3] The database consists of a set of triples $r(e_1, e_2)$ over a vocabulary of approximately 600K relations and 2M entities, extracted from the ClueWeb09 corpus.[4] The ReVerb database contains a large cross-section of general world-knowledge, and thus is a good testbed for developing an open-domain QA system. However, the extractions are noisy, unnormalized (*e.g.*, the strings obama, barack-obama, and president-obama all appear as distinct entities), and ambiguous (*e.g.*, the relation born-in contains facts about both dates and locations).

[2] InduceLex has similar behavior for the other type of derivation, which consists of a 1-argument question entry (p_q, d_q) and an entity (p_e, d_e).

[3] We used version 1.1, downloaded from http://reverb.cs.washington.edu/.

[4] The full set of ReVerb extractions from ClueWeb09 contains over six billion triples. We used the smaller subset of triples to simplify our experiments.

```
function LEARNPARAMETERS
    Inputs:
    - A corpus $\mathcal{C}$ of paraphrases $(x, x')$. (Table 1)
    - An initial lexicon $L_0$ of (pattern, db concept) pairs.
    - A learned lexicon $L$ of (pattern, db concept) pairs.
    - Initial parameters $\theta_0$.
    - Number of perceptron epochs $T$.
    - Number of training-data shards $K$.
    - A function $\mathrm{GEN}(x; L)$ that derives queries from a question $x$ using lexicon $L$. (Section 4)
    - A function $\mathrm{PerceptronEpoch}(\mathcal{T}, \theta, L)$ that runs a single epoch of the hidden-variable structured perceptron algorithm on training set $\mathcal{T}$ with initial parameters $\theta$, returning a new parameter vector $\theta'$. (Section 5.2)
    Output: A learned parameter vector $\theta$.
    // Step 1: Generate Training Examples $\mathcal{T}$
    $\mathcal{T} = \{\}$
    for all $x, x' \in \mathcal{C}$ do
        if $\mathrm{GEN}(x; L_0)$ is not empty then
            $y^* \leftarrow \arg\max_{y \in \mathrm{GEN}(x; L_0)} \theta_0 \cdot \phi(x, y)$
            $z^* \leftarrow$ query of $y^*$
            Add $(x', z^*)$ to $\mathcal{T}$
    // Step 2: Learn Parameters from $\mathcal{T}$
    Randomly partition $\mathcal{T}$ into shards $\mathcal{T}_1, \ldots, \mathcal{T}_K$
    for $t = 1 \ldots T$ do
        // Executed on $k$ processors
        $\theta_{k,t} = \mathrm{PerceptronEpoch}(\mathcal{T}_k, \theta_{t-1}, L)$
        // Average the weights
        $\theta_t = \frac{1}{K} \sum_k \theta_{k,t}$
    return $\theta_T$
```

Figure 2: Our parameter learning algorithm.

Our paraphrase corpus $\mathcal{C}$ was constructed from the collaboratively edited QA site WikiAnswers. WikiAnswers users can tag pairs of questions as alternate wordings of each other. We harvested a set of 18M of these question-paraphrase pairs, with 2.4M distinct questions in the corpus.

To estimate the precision of the paraphrase corpus, we randomly sampled a set of 100 pairs and manually tagged them as 'paraphrase' or 'not-paraphrase.' We found that 55% of the sampled pairs are valid paraphrased. Most of the incorrect paraphrases were questions that were related, but not paraphrased *e.g. How big is the biggest mall?* and *Most expensive mall in the world?*

We word-aligned each paraphrase pair using the MGIZA++ implementation of IBM Model 4 (Och and Ney, 2000; Gao and Vogel, 2008). The word-alignment algorithm was run in each direction (x, x') and (x', x) and then combined using the `grow-diag-final-and` heuristic (Koehn et al., 2003).

7 Experimental Setup

We compare the following systems:

- PARALEX: the full system, using the lexical learning and parameter learning algorithms from Section 5.
- **NoParam:** PARALEX without the learned parameters.
- **InitOnly:** PARALEX using only the initial seed lexicon.

We evaluate the systems' performance on the end-task of QA on WikiAnswers questions.

7.1 Test Set

A major challenge for evaluation is that the REVERB database is incomplete. A system may correctly map a test question to a valid query, only to return 0 results when executed against the incomplete database. We factor out this source of error by semi-automatically constructing a sample of questions that are known to be answerable using the REVERB database, and thus allows for a meaningful comparison on the task of question understanding.

To create the evaluation set, we identified questions x in a held out portion of the WikiAnswers corpus such that (1) x can be mapped to some query z using an initial lexicon (described in Section 7.4), and (2) when z is executed against the database, it returns at least one answer. We then add x and all of its paraphrases as our evaluation set. For example, the question *What is the language of Hong-Kong* satisfies these requirements, so we added these questions to the evaluation set:

What is the language of Hong-Kong?
What language do people in hong kong use?
How many languages are spoken in hong kong?
How many languages hong kong people use?
In Hong Kong what language is spoken?
Language of Hong-kong?

This methodology allows us to evaluate the systems' ability to handle syntactic and lexical variations of questions that should have the same answers. We created 37 question clusters, resulting in a total of 698 questions. We removed all of these questions and their paraphrases from the training set. We also manually filtered out any incorrect paraphrases that appeared in the test clusters.

We then created a gold-standard set of (x, a, l) triples, where x is a question, a is an answer, and l

Question Pattern	Database Query
who r e	r(?, e)
what r e	r(?, e)
who does e r	r(e, ?)
what does e r	r(e, ?)
what is the r *of* e	r(?, e)
who is the r *of* e	r(?, e)
what is r *by* e	r(e, ?)
who is e*'s* r	r(?, e)
what is e*'s* r	r(?, e)
who is r *by* e	r(e, ?)
when did e r	r-in(e, ?)
when did e r	r-on(e, ?)
when was e r	r-in(e, ?)
when was e r	r-on(e, ?)
where was e r	r-in(e, ?)
where did e r	r-in(e, ?)

Table 3: The question patterns used in the initial lexicon L_0.

	F1	Precision	Recall	MAP
PARALEX	**0.54**	0.77	**0.42**	**0.22**
NoParam	0.30	0.53	0.20	0.08
InitOnly	0.18	**0.84**	0.10	0.04

Table 4: Performance on WikiAnswers questions known to be answerable using REVERB.

	F1	Precision	Recall	MAP
PARALEX	**0.54**	0.77	**0.42**	**0.22**
No 2-Arg.	0.40	**0.86**	0.26	0.12
No 1-Arg	0.35	0.81	0.22	0.11
No Relations	0.18	0.84	0.10	0.03
No Entity	0.36	0.55	0.27	0.15

Table 5: Ablation of the learned lexical items.

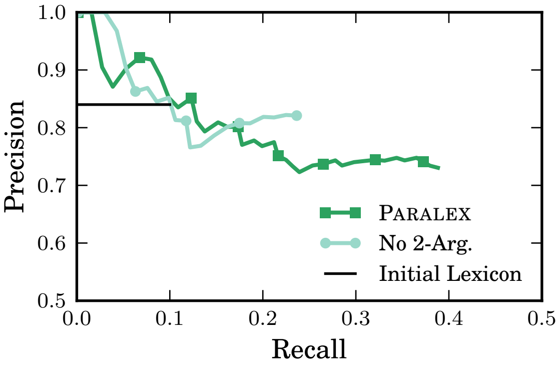


Figure 3: Precision-recall curves for PARALEX with and without 2-argument question patterns.

is a label (correct or incorrect). To create the gold-standard, we first ran each system on the evaluation questions to generate (x, a) pairs. Then we manually tagged each pair with a label l. This resulted in a set of approximately $2,000$ human judgments. If (x, a) was tagged with label l and x' is a paraphrase of x, we automatically added the labeling (x', a, l), since questions in the same cluster should have the same answer sets. This process resulted in a gold standard set of approximately $48,000$ (x, a, l) triples.

7.2 Metrics

We use two types of metrics to score the systems. The first metric measures the precision and recall of each system's highest ranked answer. Precision is the fraction of predicted answers that are correct and recall is the fraction of questions where a correct answer was predicted. The second metric measures the accuracy of the entire ranked answer set returned for a question. We compute the mean average precision (MAP) of each systems' output, which measures the average precision over all levels of recall.

7.3 Features and Settings

The feature representation $\phi(x, y)$ consists of indicator functions for each lexical entry $(p, d) \in L$ used in the derivation y. For parameter learning, we use an initial weight vector $\theta_0 = 0$, use $T = 20$ iterations and shard the training data into $K = 10$ pieces. We limit each system to return the top 100 database queries for each test sentence. All input words are lowercased and lemmatized.

7.4 Initial Lexicon

Both the lexical learning and parameter learning algorithms rely on an initial seed lexicon L_0. The initial lexicon allows the learning algorithms to bootstrap from the paraphrase corpus.

We construct L_0 from a set of 16 hand-written 2-argument question patterns and the output of the identity transformation on the entity and relation strings in the database. Table 3 shows the question patterns that were used in L_0.

8 Results

Table 4 shows the performance of PARALEX on the test questions. PARALEX outperforms the baseline systems in terms of both F1 and MAP. The lexicon-learning algorithm boosts the recall by a factor of 4 over the initial lexicon, showing the utility of the InduceLex algorithm. The

String	Learned Database Relations for String
get rid of	treatment-for, cause, get-rid-of, cure-for, easiest-way-to-get-rid-of
word	word-for, slang-term-for, definition-of, meaning-of, synonym-of
speak	speak-language-in, language-speak-in, principal-language-of, dialect-of
useful	main-use-of, purpose-of, importance-of, property-of, usefulness-of

String	Learned Database Entities for String
smoking	smoking, tobacco-smoking, cigarette, smoking-cigar, smoke, quit-smoking
radiation	radiation, electromagnetic-radiation, nuclear-radiation
vancouver	vancouver, vancouver-city, vancouver-island, vancouver-british-columbia
protein	protein, protein-synthesis, plasma-protein, monomer, dna

Table 6: Examples of relation and entity synonyms learned from the WikiAnswers paraphrase corpus.

parameter-learning algorithm also results in a large gain in both precision and recall: InduceLex generates a noisy set of patterns, so selecting the best query for a question is more challenging.

Table 5 shows an ablation of the different types of lexical items learned by PARALEX. For each row, we removed the learned lexical items from each of the types described in Section 4, keeping only the initial seed lexical items. The learned 2-argument question templates significantly increase the recall of the system. This increased recall came at a cost, lowering precision from 0.86 to 0.77. Thresholding the query score allows us to trade precision for recall, as shown in Figure 3. Table 6 shows some examples of the learned entity and relation synonyms.

The 2-argument question templates help PARALEX generalize over different variations of the same question, like the test questions shown in Table 7. For each question, PARALEX combines a 2-argument question template (shown below the questions) with the rules *celebrate* = holiday-of and *christians* = christians to derive a full query. Factoring the problem this way allows PARALEX to reuse the same rules in different syntactic configurations. Note that the imperfect training data can lead to overly-specific templates like *what are the religious* r *of* e, which can lower accuracy.

Celebrations for Christians?
r *for* e*?*

Celebrations of Christians?
r *of* e*?*

What are some celebrations for Christians?
what are some r *for* e*?*

What are some celebrations of the Christians?
what are some r *of* e*?*

What are some of Christians celebrations?
what are some of e r*?*

What celebrations do Christians do?
what r *do* e *do?*

What did Christians celebrate?
what did e r*?*

What are the religious celebrations of Christians?
what are the religious r *of* e*?*

What celebration do Christians celebrate?
what r *do* e *celebrate?*

Table 7: Questions from the test set with 2-argument question patterns that PARALEX used to derive a correct query.

9 Error Analysis

To understand how close we are to the goal of open-domain QA, we ran PARALEX on an unrestricted sample of questions from WikiAnswers. We used the same methodology as described in the previous section, where PARALEX returns the top answer for each question using REVERB.

We found that PARALEX performs significantly worse on this dataset, with recall maxing out at approximately 6% of the questions answered at precision 0.4. This is not surprising, since the test questions are not restricted to topics covered by the REVERB database, and may be too complex to be answered by any database of relational triples.

We performed an error analysis on a sample of 100 questions that were either incorrectly answered or unanswered. We examined the candidate queries that PARALEX generated for each question and tagged each query as correct (would return a valid answer given a correct and complete database) or incorrect. Because the input questions are unrestricted, we also judged whether the questions could be faithfully represented as a $r(?, e)$ or $r(e, ?)$ query over the database vocabulary. Table 8 shows the distribution of errors.

The largest source of error (36%) were on complex questions that could not be represented as a query for various reasons. We categorized these questions into groups. The largest group (14%) were questions that need n-ary or higher-order database relations, for example *How long does it take to drive from Sacramento to Cancun?* or *What do cats and dogs have in common?* Approximately 13% of the questions were how-to questions like *How do you make axes in minecraft?* whose answers are a sequence of steps, instead of a database entity. Lastly, 9% of the questions require database operators like joins, for example *When were Bobby Orr's children born?*

The second largest source of error (32%) were questions that could be represented as a query, but where Paralex was unable to derive any correct queries. For example, the question *Things grown on Nigerian farms?* was not mapped to any queries, even though the ReVerb database contains the relation `grown-in` and the entity `nigeria`. We found that 13% of the incorrect questions were cases where the entity was not recognized, 12% were cases where the relation was not recognized, and 6% were cases where both the entity and relation were not recognized.

We found that 28% of the errors were cases where Paralex derived a query that we judged to be correct, but returned no answers when executed against the database. For example, given the question *How much can a dietician earn?* Paralex derived the query `salary-of(?, dietician)` but this returned no answers in the ReVerb database.

Finally, approximately 4% of the questions included typos or were judged to be inscrutable, for example *Barovier hiriacy of evidence based for pressure sore?*

Discussion Our experiments show that the learning algorithms described in Section 5 allow Paralex to generalize beyond an initial lexicon and answer questions with significantly higher accuracy. Our error analysis on an unrestricted set of WikiAnswers questions shows that Paralex is still far from the goal of truly high-recall, open-domain QA. We found that many questions asked on WikiAnswers are either too complex to be mapped to a simple relational query, or are not covered by the ReVerb database. Further, approximately one third of the missing recall is due to entity and relation recognition errors.

Incorrectly Answered/Unanswered Questions	
36%	Complex Questions
	Need n-ary or higher-order relations (14%)
	Answer is a set of instructions (13%)
	Need database operators e.g. joins (9%)
32%	Entity or Relation Recognition Errors
	Entity recognition errors (13%)
	Relation recognition errors (12%)
	Entity & relation recognition errors (7%)
28%	Incomplete Database
	Derived a correct query, but no answers
4%	Typos/Inscrutable Questions

Table 8: Error distribution of Paralex on an unrestricted sample of questions from the WikiAnswers dataset.

10 Conclusion

We introduced a new learning approach that induces a complete question-answering system from a large corpus of noisy question-paraphrases. Using only a seed lexicon, the approach automatically learns a lexicon and linear ranking function that demonstrated high accuracy on a held-out evaluation set.

A number of open challenges remain. First, precision could likely be improved by adding new features to the ranking function. Second, we would like to generalize the question understanding framework to produce more complex queries, constructed within a compositional semantic framework, but without sacrificing scalability. Third, we would also like to extend the system with other large databases like Freebase or DBpedia. Lastly, we believe that it would be possible to leverage the user-provided answers from WikiAnswers as a source of supervision.

Acknowledgments

This research was supported in part by ONR grant N00014-11-1-0294, DARPA contract FA8750-09-C-0179, a gift from Google, a gift from Vulcan Inc., and carried out at the University of Washington's Turing Center. We would like to thank Yoav Artzi, Tom Kwiatkowski, Yuval Marton, Mausam, Dan Weld, and the anonymous reviewers for their helpful comments.

References

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open Information Extraction from the Web. In *Proceedings of the 20th international joint conference on Artifical intelligence*.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with Bilingual Parallel Corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*.

Regina Barzilay and Lillian Lee. 2003. Learning to Paraphrase: An Unsupervised Approach Using Multiple-Sequence Alignment. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics*.

Regina Barzilay and Kathleen R. McKeown. 2001. Extracting Paraphrases from a Parallel Corpus. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*.

Eric Brill, Susan Dumais, and Michele Banko. 2002. An Analysis of the AskMSR Question-Answering System. In *Proceedings of Empirical Methods in Natural Language Processing*.

Qingqing Cai and Alexander Yates. 2013. Large-scale Semantic Parsing via Schema Matching and Lexicon Extension. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Chris Callison-Burch. 2008. Syntactic Constraints on Paraphrases Extracted from Parallel Corpora. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving Semantic Parsing from the World's Response. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*.

Michael Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying Relations for Open Information Extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*.

Qin Gao and Stephan Vogel. 2008. Parallel Implementations of Word Alignment Tool. In *Proc. of the ACL 2008 Software Engineering, Testing, and Quality Assurance Workshop*.

Barbara J. Grosz, Douglas E. Appelt, Paul A. Martin, and Fernando C. N. Pereira. 1987. TEAM: An Experiment in the Design of Transportable Natural-Language Interfaces. *Artificial Intelligence*, 32(2):173–243.

Raphael Hoffmann, Congle Zhang, and Daniel S. Weld. 2010. Learning 5000 relational extractors. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-Based Weak Supervision for Information Extraction of Overlapping Relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*.

Boris Katz. 1997. Annotating the World Wide Web using Natural Language. In *RIAO*, pages 136–159.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics*.

Cody Kwok, Oren Etzioni, and Daniel S. Weld. 2001. Scaling Question Answering to the Web. *ACM Trans. Inf. Syst.*, 19(3):242–262.

Percy Liang, Michael Jordan, and Dan Klein. 2011. Learning Dependency-Based Compositional Semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*.

Yuval Marton, Chris Callison-Burch, and Philip Resnik. 2009. Improved Statistical Machine Translation Using Monolingually-Derived Paraphrases. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*.

Ryan McDonald, Keith Hall, and Gideon Mann. 2010. Distributed training strategies for the structured perceptron. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant Supervision for Relation Extraction Without Labeled Data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL*.

Franz Josef Och and Hermann Ney. 2000. Improved Statistical Alignment Models. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*.

Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates. 2004. Modern Natural Language Interfaces to Databases: Composing Statistical Parsing with Semantic Tractability. In *Proceedings of the Twentieth International Conference on Computational Linguistics*.

Chris Quirk, Chris Brockett, and William Dolan. 2004. Monolingual Machine Translation for Paraphrase Generation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling Relations and Their Mentions without Labeled Text. In *Proceedings of the 2010 European conference on Machine learning and Knowledge Discovery in Databases*.

Lappoon R. Tang and Raymond J. Mooney. 2001. Using Multiple Clause Constructors in Inductive Logic Programming for Semantic Parsing.

Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. 2012. Template-Based Question Answering over RDF Data. In *Proceedings of the 21st World Wide Web Conference 2012*.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, Maya Ramanath, Volker Tresp, and Gerhard Weikum. 2012. Natural Language Questions for the Web of Data. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2012. Unsupervised Relation Discovery with Sense Disambiguation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*.

John M. Zelle and Raymond J. Mooney. 1996. Learning to Parse Database Queries Using Inductive Logic Programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars. In *Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence*.