Scalable Factoid QA: Big Knowledge Bases and Complex Questions

Luke Zettlemoyer

joint work with
Yoav Artzi, Eunsol Choi, Oren Etzioni, Anthony Fader, and Tom Kwiakowski
Question Answering (QA)

Semantic Parsing

How many states have a higher point than the highest point of the state with the largest capital city in the US?

Zelle & Mooney, ’96,
Zettlemoyer & Collins, ’05,
Liang et al., ’11,
and many many more

Answer Retrieval

Who is Tom Cruise married to?

Tom Cruise is married to Katie Holmes, also an actress.

Voorhees & Tice, ‘00
Ravichandran & Hovy '02
and many many more
Semantic Parsing

Q: How many people live in Seattle?

MR: $\lambda x.\text{population}(\text{seattle}, x)$

620,778
Two Big Challenges

• Part 1: How do we understand complex questions against large, varied KBs?  
  [Kwiatkowski et al, 2013]

• Part II: How do we get enough facts to answer any question?  
  [Fader et al, 2014]
Open Domain QA

Q: Who managed Liverpool F.C. from 2004 to June 2010?
A: Rafael Benitez

Q: What architectural style is the Brooklyn Bridge?
A: Gothic Revival architecture

Q: What are the symptoms of prostate cancer?
A: {Hematuria, Nocturia, Dysuria, ... }
Who managed Liverpool F.C. from 2004 to June 2010?

Rafael Benitez

What are the symptoms of prostate cancer?

Hematuria, Nocturia, Dysuria, Urinary frequency, Hematocrit

What architectural style is the Brooklyn Bridge?

Gothic Revival architecture

How many people ride the monorail in Seattle daily?

40 million

Open Domain QA

Freebase is a community authored knowledge base with:

- 40 Million Entities
- 2 Billion Facts
- 20,000 Relations
- 10,000 Types
- 100 Domains
Query is Domain Dependent

How many people live in Seattle?
Query is Domain Dependent

How many people live in Seattle?

\[ \lambda x. \text{eq}(x, \text{count}(\lambda y. \text{person}(y) \land \text{home}(y, \text{seattle}))) \]

<table>
<thead>
<tr>
<th>Person</th>
<th>Home</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eunsol</td>
<td>Seattle</td>
</tr>
<tr>
<td>Luke</td>
<td>Seattle</td>
</tr>
<tr>
<td>Jane</td>
<td>Boston</td>
</tr>
</tbody>
</table>
Query is Domain Dependent

How many people live in Seattle?

\[ \lambda x. \text{eq}(x, \text{count}(\lambda y. \text{person}(y) \land \text{home}(y, \text{seattle}))) \]

<table>
<thead>
<tr>
<th>City</th>
<th>Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seattle</td>
<td>620778</td>
</tr>
<tr>
<td>Boston</td>
<td>636479</td>
</tr>
</tbody>
</table>

Person | Home
-------|-----
Eunsol | Seattle
Luke   | Seattle
Jane   | Boston
How many people live in Seattle?

<table>
<thead>
<tr>
<th>City</th>
<th>Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seattle</td>
<td>620778</td>
</tr>
<tr>
<td>Boston</td>
<td>636479</td>
</tr>
</tbody>
</table>

- Requires different syntax for different domains
  - Grammars do not generalize well
  - Grammars are hard to learn
Query is Domain Dependent

How many people live in Seattle?
\[ \lambda x. \text{population(seattle, } x) \]

How many people have won the Nobel peace prize?

\[ \lambda x. \text{eq(x, count(} \lambda y. \text{person(y)} \land \text{won(y, nobel_peace_prize)))} \]

<table>
<thead>
<tr>
<th>Person</th>
<th>Award</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nelson M.</td>
<td>Nobel P.P.</td>
</tr>
<tr>
<td>Mother T.</td>
<td>Nobel P.P.</td>
</tr>
<tr>
<td>Leymah G.</td>
<td>Nobel P.P.</td>
</tr>
</tbody>
</table>
Query is Domain Dependent

How many people live in Seattle?

$$\lambda x. \text{population} \left( \text{seattle}, x \right)$$

How many people have won the Nobel peace prize?

$$\lambda x. \text{eq} \left( x, \text{count} \left( \lambda y. \text{person} \left( y \right) \land \text{won} \left( y, \text{nobel \_ peace \_ prize} \right) \right) \right)$$

<table>
<thead>
<tr>
<th>Person</th>
<th>Award</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nelson M.</td>
<td>Nobel P.P.</td>
</tr>
<tr>
<td>Mother T.</td>
<td>Nobel P.P.</td>
</tr>
<tr>
<td>Leymah G.</td>
<td>Nobel P.P.</td>
</tr>
</tbody>
</table>
How many people live in Seattle?
2 Stage Semantic Parsing

1. Domain independent, linguistically motivated parse.

\[ \lambda x. \text{eq}(x, \text{count}(\lambda y. \text{people}(y) \land \exists \text{ev}. \text{live}(y, \text{ev}) \land \text{in(seattle, ev)))) \]
2 Stage Semantic Parsing

2. Domain specific ontology match.

How many people live in Seattle?

$$\lambda x. eq(x, \text{count}(\lambda y. \text{person}(y) \wedge \text{in}(\text{seattle}, ev)))$$
2 Stage Semantic Parsing

2. Domain specific ontology match.

How many people live in Seattle?

\[ \lambda x. \text{population}(\text{seattle}, x) \]

\[ \lambda x. \text{eq}(x, \text{count}(\lambda y. \text{people}(y) \land \exists e. \text{live}(y, e) \land \text{in}(\text{seattle}, e))) \]

\[ \lambda x. \text{eq}(x, \text{count}(\lambda y. \text{person}(y) \land \text{home}(y, \text{seattle}))) \]

<table>
<thead>
<tr>
<th>City</th>
<th>Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seattle</td>
<td>620778</td>
</tr>
<tr>
<td>Boston</td>
<td>636479</td>
</tr>
</tbody>
</table>

Person | Home
------|------
Eunsol | Seattle
Luke   | Seattle
Jane   | Boston
2 Stage Semantic Parsing

- How many people live in Seattle?

- All domains use same syntax that generalizes well

- Ontology match can be guided by the structure of the underspecified logical form

<table>
<thead>
<tr>
<th>City</th>
<th>Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seattle</td>
<td>620778</td>
</tr>
<tr>
<td>Boston</td>
<td>636479</td>
</tr>
</tbody>
</table>

- Eunsol: Seattle
- Luke: Seattle
- Jane: Boston
2 Stage Semantic Parsing

Domain Independent Parse

<table>
<thead>
<tr>
<th>Structure Match</th>
<th>Ontology Match</th>
<th>Seagrove</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S/(S\backslash NP)/N )</td>
<td>( \lambda x.eq(x, \text{count} (\lambda y.\exists ev.\text{people}(y) \land \text{live}(y, ev) \land \text{in}(ev, \text{seattle}))) )</td>
<td>( \lambda x.\text{population}(\text{seattle}, x) )</td>
</tr>
<tr>
<td>( \text{People} )</td>
<td>( \lambda x.\text{people}(x) )</td>
<td></td>
</tr>
<tr>
<td>( \text{Live in} )</td>
<td>( \lambda x.\text{live}(x, ev) )</td>
<td></td>
</tr>
<tr>
<td>( \text{In} )</td>
<td>( \lambda x.\text{in}(x, ev) )</td>
<td></td>
</tr>
</tbody>
</table>

\( \lambda x.\text{eq}(x, \text{count}(\lambda y.\exists ev.\text{people}(y) \land \text{live}(y, ev) \land \text{in}(ev, \text{seattle}))) \)
# Domain Independent Parsing

## 49 domain independent lexical items:

<table>
<thead>
<tr>
<th>Word</th>
<th>Syntax</th>
<th>Underspecified semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>How many</td>
<td>$S/(S\backslash NP)/N$</td>
<td>$\lambda f \lambda g \lambda x . \text{eq}(x, \text{count}(\lambda y . f(y) \land g(y)))$</td>
</tr>
<tr>
<td>What</td>
<td>$S/(S\backslash NP)/N$</td>
<td>$\lambda f \lambda g \lambda x . f(x) \land g(x)$</td>
</tr>
<tr>
<td>most</td>
<td>$NP/N$</td>
<td>$\lambda f . \text{max_count}(\lambda y . f(y))$</td>
</tr>
</tbody>
</table>

etc.

## 56 underspecified lexical categories:

<table>
<thead>
<tr>
<th>Part-of-Speech</th>
<th>Syntax</th>
<th>Underspecified semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>proper noun</td>
<td>$NP$</td>
<td>$C$</td>
</tr>
<tr>
<td>noun</td>
<td>$N$</td>
<td>$\lambda x . P(x)$</td>
</tr>
<tr>
<td>noun</td>
<td>$N/N$</td>
<td>$\lambda f \lambda x . f(x) \land P(x)$</td>
</tr>
<tr>
<td>verb</td>
<td>$S\backslash NP$</td>
<td>$\lambda x \lambda e v . P(x, e v)$</td>
</tr>
<tr>
<td>preposition</td>
<td>$N\backslash NP$</td>
<td>$\lambda f \lambda x \lambda y . P(y, x) \land f(y)$</td>
</tr>
<tr>
<td>preposition</td>
<td>$S\backslash S\backslash NP$</td>
<td>$\lambda f \lambda x \exists e v . P(e v, x) \land f(e v)$</td>
</tr>
</tbody>
</table>

etc.
2 Step Semantic Parsing

Domain Independent Parse

 Ontology Match

\[ \lambda x.\text{eq}(x, \text{count}(\lambda y.\exists \text{ev}.\text{people}(y) \land \text{live}(y, \text{ev}) \land \text{in}(\text{ev, seattle}))) \]

Structure Match

\[ \lambda x.\text{how\_many\_people\_live\_in}(\text{seattle}, x) \]

\[ \lambda x.\text{population}(\text{seattle}, x) \]
Structural Match

Collapse and expand subexpressions in underspecified logical form with operators that:

1. Collapse simple typed sub-expression
2. Collapse complex typed sub-expression
3. Expand predicate

New example

How many people ride the monorail in Seattle daily?

\[ \lambda x.\text{eq}(x, \text{count}(\lambda y.\text{people}(y) \land \exists e.\text{ride}(y, \text{z.monorail}(z) \land \text{in}(z, \text{seattle}), e) \land \text{daily}(e))) \]

\[ \equiv \lambda x.\text{transit-system/daily riders(seattle monorail, x)} \]
Structural Match

1. Find subexpression with type allowed in KB

\[ \lambda x.\text{eq}(x, \text{count}(\lambda y.\text{people}(y)) \land \exists e. \text{ride}(y, iz.\text{monorail}(z) \land \text{in}(z, \text{seattle}), e) \land \text{daily}(e)) \]

2. Replace with new underspecified constant
1. Find subexpression with type allowed in KB

\[ \lambda x. \text{eq}(x, \text{count}(\lambda y. \text{people}(y)) \land \exists e. \text{ride}(y, iz. \text{monorail}(z) \land \text{in}(z, \text{seattle})e) \land \text{daily}(e)) \]

2. Replace with new underspecified constant

\[ \lambda x. \text{eq}(x, \text{count}(\lambda y. \text{people}(y)) \land \exists e. \text{ride}(y, \text{monorail}_{-\text{in}} \text{seattle}, e) \land \text{daily}(e)) \]
1. Find subexpression with type allowed in KB

\[ \lambda x. \text{eq}(x, \text{count}(\lambda y. \text{people}(y) \land \exists e. \text{ride}(y, iz. \text{monorail}(z) \land \text{in}(z, \text{seattle}), e) \land \text{daily}(e))) \]

2. Replace with new underspecified constant

\[ \lambda x. \text{eq}(x, \text{how_many_people_ride_daily_the_monorail_in_seattle}) \]
Constant Match

Replace constants with constants from KB

\( \lambda x. \text{how\_many\_people\_ride\_daily(} \text{the\_monorail\_in\_seattle,} \ x) \)

Assume constants have English string labels!
Constant Match

Replace constants with constants from KB

\( \lambda x. \text{how\_many\_people\_ride\_daily}(\text{the\_monorail\_in\_seattle}, x) \)

\( \Rightarrow \lambda x. \text{transit\_system\_daily\_riders}(\text{seattle\_monorail}, x) \)
Constant Match

Replace constants with constants from KB

\[ \lambda x.\text{how\_many\_people\_ride\_daily}(\text{the\_monorail\_in\_seattle}, x) \]

\[\equiv\]

\[ \lambda x.\text{how\_many\_people\_ride\_daily}(\text{seattle\_monorail}, x) \]
Constant Match

Replace constants with constants from KB

\[ \lambda x.\textit{how\_many\_people\_ride\_daily}(\textit{the\_monorail\_in\_seattle}, x) \]

\[ \approx \]

\[ \lambda x.\textit{transit\_system/daily\_riders}(\textit{seattle\_monorail}, x) \]
2 Stage Semantic Parsing

Domain Independent Parse

<table>
<thead>
<tr>
<th>How many</th>
<th>people</th>
<th>live</th>
<th>in</th>
<th>Seattle</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S/(S/NP)/N$</td>
<td>$\lambda x.\text{people}(x)$</td>
<td>$\lambda x \lambda e v .\text{live}(x, e v)$</td>
<td>$S\backslash S/NP$</td>
<td>$\lambda x \lambda f \exists e v .\text{in}(e v, x)$</td>
</tr>
<tr>
<td>$\lambda f \lambda g \lambda x .\text{eq}(x, \text{count}(\lambda y .g(y) \land f(y)))$</td>
<td></td>
<td></td>
<td>$\land f(e v)$</td>
<td></td>
</tr>
</tbody>
</table>

Ontology Match

$\lambda x .\text{eq}(x, \text{count}(\lambda y .\exists e v .\text{people}(y) \land \text{live}(y, e v) \land \text{in}(e v, \text{seattle})))$

Structure Match

$\lambda x .\text{how\_many\_people\_live\_in}(\text{seattle}, x)$

$\lambda x .\text{population}(\text{seattle}, x)$
Learning

Input
Q/A pairs \( \{ (x_i, a_i) : i = 1, \ldots, n \} \)
Knowledge Base, Wiktionary, Underspecified Lexicon

Algorithm
For \( i = 1, \ldots, n \)

\[
\begin{align*}
C &\leftarrow \text{Max scoring correct parses } \alpha^c_i \\
W &\leftarrow \text{Margin violating incorrect parses } \alpha^i_i
\end{align*}
\]

\[
\theta = \theta + \frac{1}{|C|} \sum_{c \in C} \phi(c) - \frac{1}{|W|} \sum_{w \in W} \phi(w)
\]
Experiments

Works well on Freebase benchmark

Also works on older, more compositional benchmark
Example Parses

How many operating systems is Adobe Flash compatible with?

\[ \lambda x. \text{eq}(x, \text{count}(\lambda y. \text{software_compatibility} \ . \text{operating_system}(\text{adobe_flash}, y))) \]

Who is the CEO of Save-A-Lot?

\[ \lambda x. \text{person}(x) \land \exists y. \text{organization}(y, \text{savealot}) \land \text{board_member.leader_of}(x, y) \land \text{leadership.role}(y, \text{ceo}) \]
Example Errors

How many children does Jerry Seinfeld have?

Target:
\[ \lambda x. \text{eq}(x, \text{count}(\lambda y. \text{person} \cdot \text{children}(\text{jerry} \_ \text{seinfeld}, y))) \]

Prediction:
\[ \lambda x. \text{eq}(x, \text{count}(\lambda y. \text{person} \cdot \text{children}(y, \text{jerry} \_ \text{seinfeld}))) \]
Example Errors

What programming languages were used for AOL instant messenger?

Target:
\[ \lambda x.\text{languages	extunderscore used}(\text{aol	extunderscore instant	extunderscore messenger},x) \]

Prediction:
\[ \lambda x.\text{languages	extunderscore used}(\text{aol	extunderscore instant	extunderscore messenger},x) \wedge \text{programming	extunderscore language}(x) \]
Two Big Challenges

• Part 1: How do we understand complex questions against large, varied KBs?
  - Use underspecified semantic parser
  - Learn to match meaning to target domain

• Part II: How do we get enough facts to answer any question?
Freebase

Open Information Extraction (ReVerb)

Multiple Knowledge Sources
Billions of Facts
Millions of Relationships

ProBase

NELL: Never-Ending Language Learning
Knowledge Base

Open IE

ProBase

Don’t require:
Normalization
Canonicalization
Ontologization

(banana, source of, potassium)
(mushroom, is-a, pizza topping)
(quinoa, compatible with dietary restrictions, gluten-free diet)
Freebase Triples

**Required**

- arg1: McDonald’s
- rel: Advertising characters
- arg2: Mac Tonight
- namespace: Freebase

- arg1 id: /m/0jg57
- arg2 id: /m/01qq3s

**Optional**

- arg1: McDonald’s
- rel: Revenue
- arg2: Currency: United States Dollar
  Amount: 23,000,000,000
  Valid Date: 12/31/2008

Threw away non-binary relations
Bill Clinton ate at McDonald's, went jogging, and confessed a fondness for trashy spy novels.

Lin et al. 2012
McDonald’s characters such as Grimace or Mac Tonight...

**Required**

<table>
<thead>
<tr>
<th>arg1: Grimace</th>
<th>rel: is a</th>
<th>arg2: McDonald’s characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace: Probase</td>
<td></td>
<td></td>
</tr>
<tr>
<td>arg1 frequency: 17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>arg2 frequency: 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>popularity: 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>arg1 vagueness: 0.67</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

...
Sammy Sosa, who played for the Cubs in...
Cubs player Sammy Sosa came from the...
Last night Sammy Sosa hit a run for the Cubs...

| arg1: | sammy sosa |
| rel: | plays for the team |
| arg2: | cubs |
| namespace: | nell |
| confidence: | 0.92 |
# Knowledge Base Statistics

<table>
<thead>
<tr>
<th>Source</th>
<th>Type</th>
<th># Triples</th>
<th># Relation Phrases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freebase</td>
<td>Curated</td>
<td>300M</td>
<td>18K</td>
</tr>
<tr>
<td>Open IE</td>
<td>Extracted</td>
<td>500M</td>
<td>6M</td>
</tr>
<tr>
<td>Probase</td>
<td>Extracted</td>
<td>200M</td>
<td>1</td>
</tr>
<tr>
<td>NELL</td>
<td>Extracted</td>
<td>2M</td>
<td>300</td>
</tr>
</tbody>
</table>
WikiAnswers Paraphrase Corpus

What cures a hangover?
How do you lose a hangover?
The best way to recover from a hangover?
What takes away hangover symptoms?
What is a benefit of social networks?
What are social networking sites used for?
Why do people use social networks?
Why are social networks used a lot?
Who wrote Winnie the Pooh?
Who is the author of Winnie the Pooh?
Who is Pooh's creator?
Name of the author of Winnie the Pooh?
How can I tell if I have strep throat?
What are signs of strep throat?
Am I sick with strep throat?
What are strep throat symptoms?

20 million user-created clusters
First used for Open QA in Paralex
(Fader, Zettlemoyer, and Etzioni, 2012)
How can I tell if I have the flu?

What are signs of the flu?

- (flu, symptoms, ?)
- the chills

What are signs of strep throat?

Am I sick with strep throat?

- (cough, cold) (fever, strep)

Paraphrase

Parse

Rewrite

Execute

score(... → the chills) = 1.2
score(... → virus) = -0.3
Query Language

What fruits are a source of vitamin C?

?x: (?x, is a, fruit) (?x, source of, vitamin c)

<table>
<thead>
<tr>
<th>arg1</th>
<th>rel</th>
<th>arg2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lychee</td>
<td>is a</td>
<td>fruit</td>
</tr>
<tr>
<td>star-fruit</td>
<td>is a</td>
<td>tropical fruit</td>
</tr>
<tr>
<td>pepper</td>
<td>is a</td>
<td>fresh fruit</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>arg1</th>
<th>rel</th>
<th>arg2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lychees</td>
<td>good source of</td>
<td>vitamin C</td>
</tr>
<tr>
<td>starfruits</td>
<td>source of</td>
<td>vitamin C</td>
</tr>
<tr>
<td>peppers</td>
<td>provides a source of</td>
<td>vitamins C and A</td>
</tr>
</tbody>
</table>
## Mining Paraphrase Templates

<table>
<thead>
<tr>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$\text{count}(t_1, t_2)$</th>
<th>$\text{count}(t_1)$</th>
<th>$\text{count}(t_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>How can I tell if I have _</td>
<td>What are signs of _</td>
<td>8</td>
<td>21</td>
<td>45</td>
</tr>
<tr>
<td>How can I tell if I have _</td>
<td>Am I sick with _</td>
<td>11</td>
<td>21</td>
<td>52</td>
</tr>
<tr>
<td>How can I tell if I have strep _</td>
<td>What are signs of strep _</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

5 million $\{t_1, t_2\}$ pairs with $\text{count}(t_1, t_2) \geq 5$
<table>
<thead>
<tr>
<th>Template 1</th>
<th>Template 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>How does _ affect your body?</td>
<td>What body system does _ affect?</td>
</tr>
<tr>
<td>What is the latin name for _?</td>
<td>What is _’s scientific name?</td>
</tr>
<tr>
<td>Why do we use _?</td>
<td>What did _ replace?</td>
</tr>
<tr>
<td>What to use instead of _?</td>
<td>What is a substitute for _?</td>
</tr>
<tr>
<td>Was _ ever married?</td>
<td>Who has _ been married to?</td>
</tr>
</tbody>
</table>
Mining Query Rewrite Operators

<table>
<thead>
<tr>
<th>arg1</th>
<th>arg2</th>
<th>sign of</th>
<th>symptom$^1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>cough</td>
<td>cold</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>jealousy</td>
<td>love</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>diziness</td>
<td>Meniere's</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>chills</td>
<td>flu</td>
<td></td>
<td>✔️</td>
</tr>
</tbody>
</table>

74 million $\{r_1, r_2\}$ pairs with $\geq 10$ shared arguments

DIRT (Lin and Pantel, 2001)
Sempre (Berant et al., 2013)
<table>
<thead>
<tr>
<th>Source Relation</th>
<th>Target Relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(?x, children, ?y)</td>
<td>(?y, was born to, ?x)</td>
</tr>
<tr>
<td>(?x, birthdate, ?y)</td>
<td>(?x, date of birth, ?y)</td>
</tr>
<tr>
<td>(?x, headquartered in, ?y)</td>
<td>(?y, is based in, ?x)</td>
</tr>
<tr>
<td>(?x, invented, ?y)</td>
<td>(?y, was invented by, ?x)</td>
</tr>
<tr>
<td>(?x, is the language of, ?y)</td>
<td>(?y, languages spoken, ?x)</td>
</tr>
</tbody>
</table>
Learning and Inference

Latent-Variable Structured Perceptron: Liang et al., 2006; Sun et al., 2009

How can I tell if I have the flu?

Derivations not available in training data

\[ w = w + f(... \rightarrow \text{the chills}) - f(... \rightarrow \text{virus}) \]
Question Sets

WebQuestions  Berant et al., 2013
where was nicki minaj born?

TREC  Vorhees and Tice, 2000
What other countries do curds live in ?

WikiAnswers  Held-out from corpus
Who is a retired gay nfl player?
Experiments

WebQuestions

TREC

WikiAnswers

Precision

Recall

0%
20%
40%

0%
15%
30%

0%
4%
8%

WebQuestions

TREC

WikiAnswers

Precision

Recall

0%
20%
40%

0%
15%
30%

0%
4%
8%
Effect of Knowledge Sources

WebQuestions

- All KBs
- No Open IE
- No Freebase
- No NELL

F1: 0.35

TREC

- All KBs
- No Open IE
- No Freebase
- No NELL

F1: 0.29

WikiAnswers

- All KBs
- No Open IE
- No Freebase
- No NELL

F1: 0.08
Cherry-Picked Examples

Input | Who did Michael J Fox marry?
 Parse | ?x: (Michael J Fox, marry, ?x)
 Rewrite | ?x: (Michael J Fox, has wife, ?x)
 Execute | **Tracy Pollan:**
 | (Michael J. Fox, has wife, Tracy Pollan)

Input | What are brake pads made of?
 Parse | What material are brake pads made of?
 Execute | ?x: (?x, is-a, material) (brake pads, made of, ?x)
 | **copper:**
 | (copper, is-a, material)
 | (The brake pads, were made of, copper)
<table>
<thead>
<tr>
<th>Input</th>
<th>What animal represents California?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parse</td>
<td>What are California’s symbols?</td>
</tr>
<tr>
<td>Execute</td>
<td>?x: (california, symbols, ?x)</td>
</tr>
</tbody>
</table>

**CWT:**
(California Water Service, Trading symbol, CWT)

<table>
<thead>
<tr>
<th>Input</th>
<th>What actor first portrayed James Bond?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parse</td>
<td>?x: (?x, is-a, actor first)</td>
</tr>
<tr>
<td></td>
<td>(?x, portrayed, James Bond)</td>
</tr>
</tbody>
</table>

**Daniel Craig:**
(Daniel Craig, is-a, first class actor)
(Danny Craig, portrays, James Bond)
An Embarrassingly Funny Example

What kind of sound does a unicorn make?

(a really cute unicorn, sounds like, fun)
Two Big Challenges

• Part 1: How do we understand complex questions against large, varied KBs?
  - Use underspecified semantic parser
  - Learn to match meaning to target domain

• Part II: How do we get enough facts to answer any question?
  - Use large, unstructured tuple store
  - Learn from paraphrases and KB facts
A Few Open Questions

• How many facts do we need, and can we ever enumerate them all?
• Can we jointly learn to extract facts and answer questions?
• Can we apply similar techniques to understand non-question sentences?