

Towards a Compositional Typed Semantics for Universal Dependencies

Siva Reddy

School of Informatics
The University of Edinburgh

People



Mirella Lapata Mark Steedman



People



Mirella Lapata



Mark Steedman



Oscar Täckström



Dipanjan Das



Tom Kwiatkowski



Michael Collins



Slav Petrov

Syntax helps Semantics

kotini
monkey

aratipandu
banana

tinindi
eat

Syntax helps Semantics

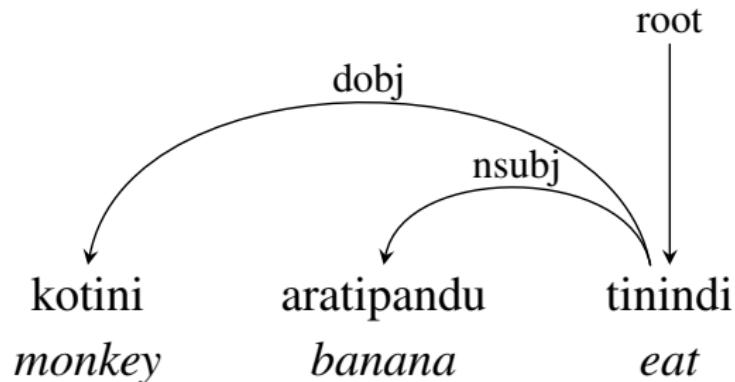
kotini
monkey

aratipandu
banana

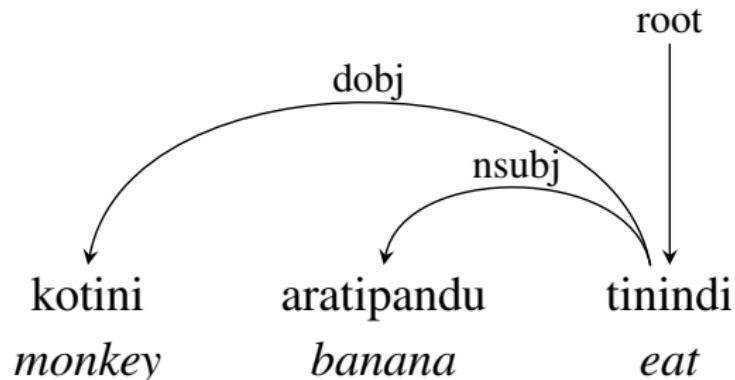
tinindi
eat



Syntax helps Semantics

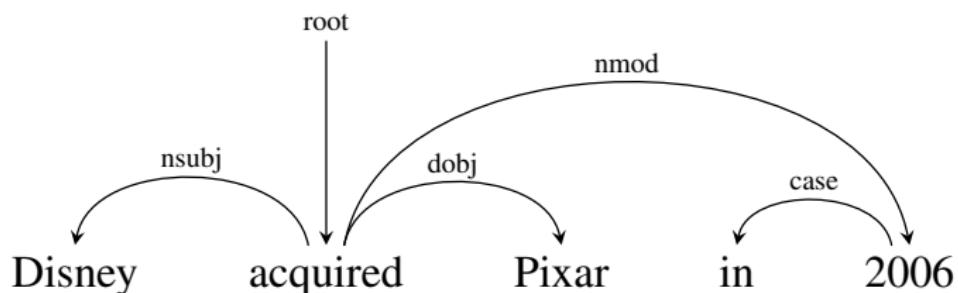


Syntax helps Semantics



Universal Dependencies

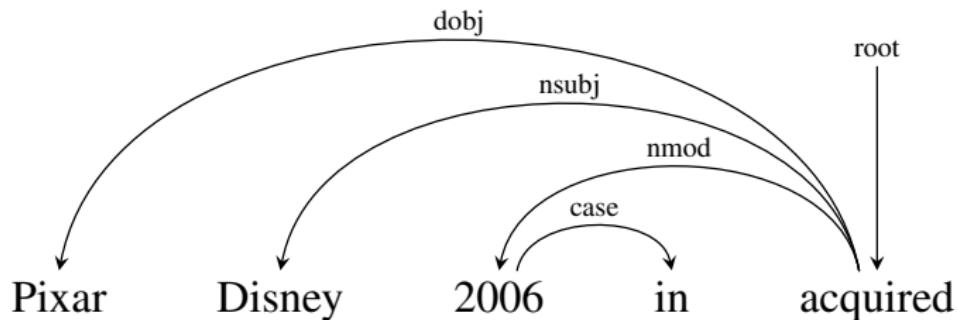
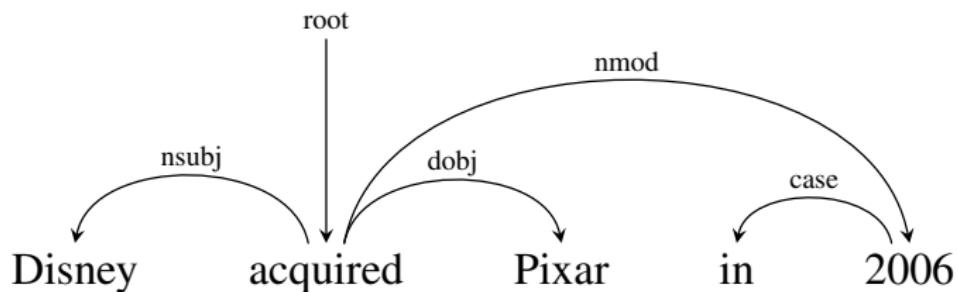
Homogeneous syntactic representation across languages



Pixar Disney 2006 in acquired

Universal Dependencies

Homogeneous syntactic representation across languages



Universal Dependencies

Dependency trees are human-friendly

Treebanks in 40 languages

40 dependency labels

Dependency Tree to Semantics

Semantics

- ▶ Symbolic representations such as logical forms
- ▶ Distributed representations such as vectors

Dependency Tree to Semantics

Principle of Compositionality: the semantics of a complex expression is determined by the semantics of its constituent expressions and the rules used to combine them

Complex expression is the dependency tree

Constituent expressions are subtree

Rules are the dependency labels

Dependency Tree to Semantics

Dependencies **lack** a formal mechanism to derive semantics from constituents

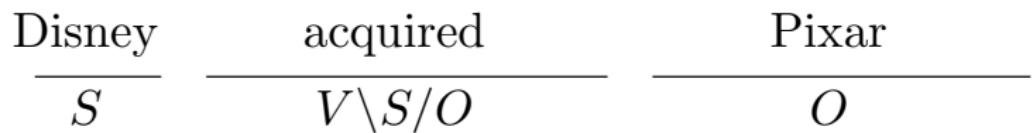
Syntax to semantics interface is important

Existing Syntax-Semantics interfaces

- ▶ **CCG** [Bos et al., 2004; Lewis & Steedman, 2013]
- ▶ **HPSG** [Copestake et al., 2001]
- ▶ **LFG** [Dalrymple et al., 1995]
- ▶ **TAG** [Joshi et al., 1995]

CCG is a popular choice

- ▶ Typing Mechanism
- ▶ Combinator Rules



CCG

$$\frac{\text{Disney}}{S} \quad \frac{\text{acquired}}{V \setminus S/O} \quad \frac{}{O}$$

Disney $\lambda y \lambda x \lambda e. \text{acquired}(e)$ Pixar

$\wedge \text{arg}_1(e, x)$

$\wedge \text{arg}_2(e, y)$

CCG

$$\begin{array}{ccc}
 \text{Disney} & \text{acquired} & \text{Pixar} \\
 \hline
 S & V \setminus S/O & O \\
 \text{Disney } \lambda y \lambda x \lambda e. \text{ acquired}(e) & & \text{Pixar} \\
 & \wedge \text{arg}_1(e, x) & \\
 & \wedge \text{arg}_2(e, y) & \\
 \hline
 & & \longrightarrow \\
 & V \setminus S & \\
 & \lambda x \lambda e. \text{ acquired}(e) & \\
 & \wedge \text{arg}_1(e, x) \wedge \text{arg}_2(e, \text{Pixar}) &
 \end{array}$$

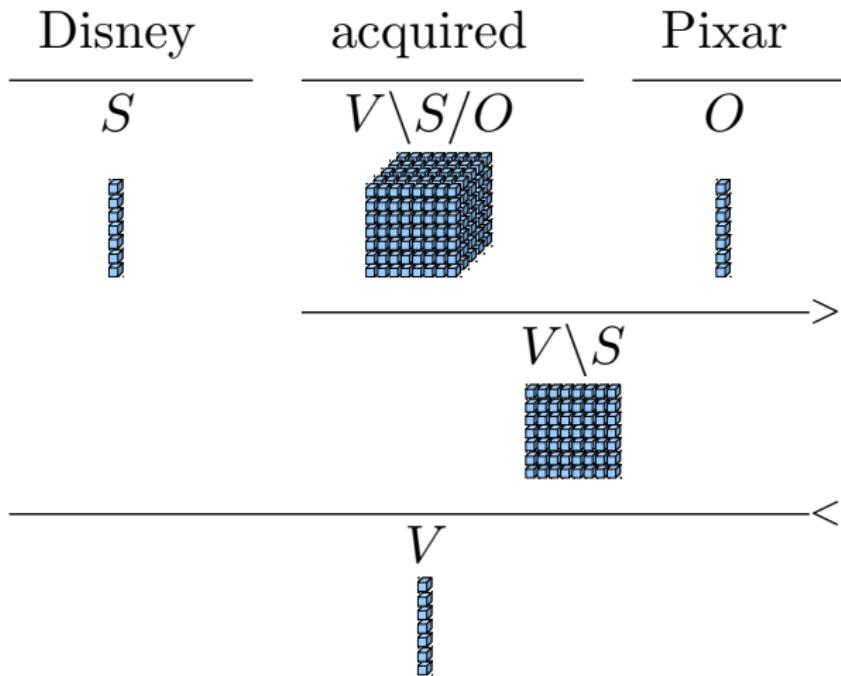
CCG

$$\frac{\text{Disney} \quad \text{acquired} \quad \text{Pixar}}{\frac{S \quad V \setminus S/O \quad O}{\text{Disney } \lambda y \lambda x \lambda e. \text{ acquired}(e) \wedge \arg_1(e, x) \wedge \arg_2(e, y) \quad \text{Pixar}}} \rightarrow$$

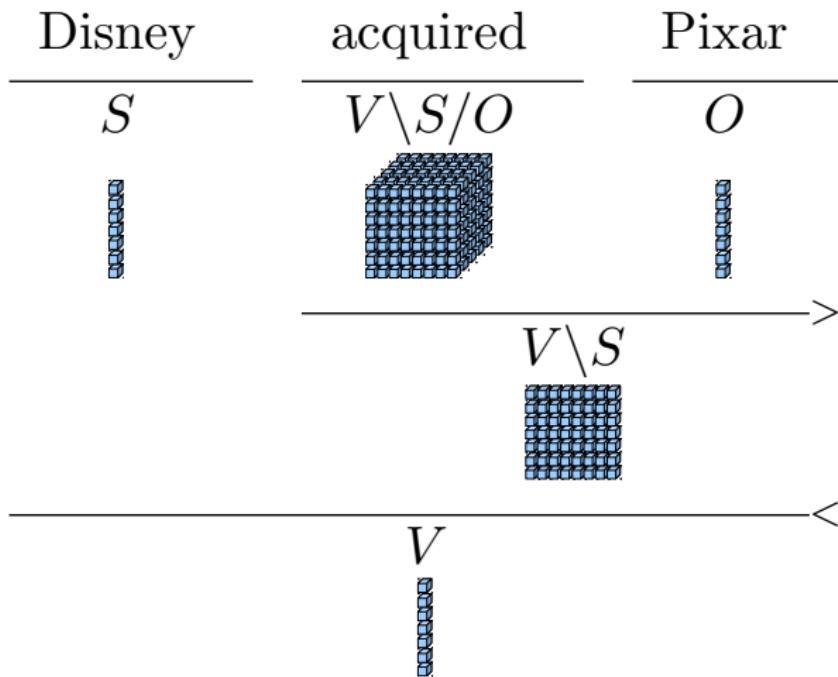
$$\frac{}{\lambda x \lambda e. \text{ acquired}(e) \wedge \arg_1(e, x) \wedge \arg_2(e, \text{Pixar})} \leftarrow$$

$$\frac{}{\lambda e. \text{ acquired}(e) \wedge \arg_1(e, \text{Disney}) \wedge \arg_2(e, \text{Pixar})}$$

CCG

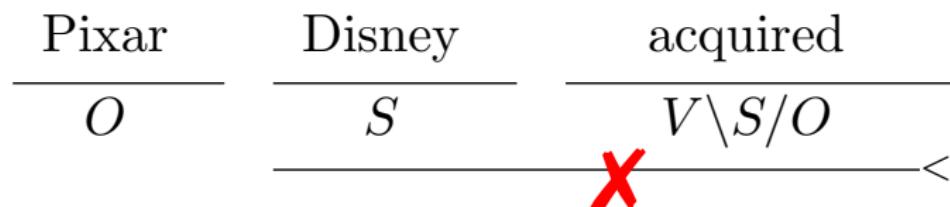
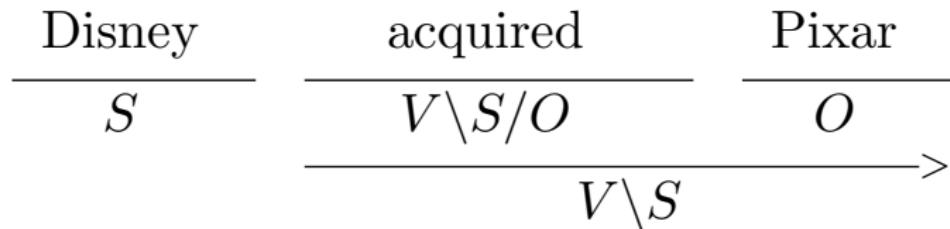


CCG



Typing and Combinator Rules allow
Synchronous Syntax-Semantics interface

CCG



- ▶ Language-specific types
- ▶ Heavy lexicalization, e.g., $(V \setminus S) / (V \setminus O) / O$

This Talk

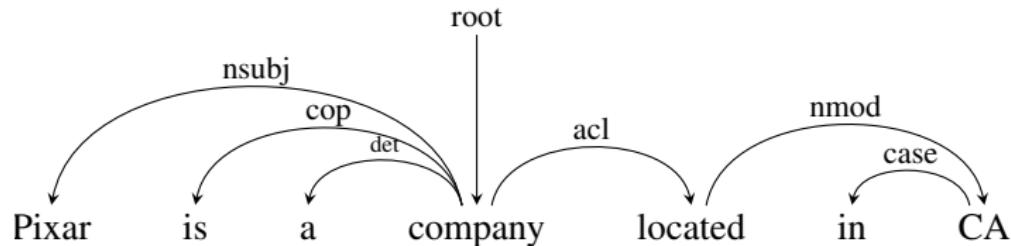
A Compositional Typed Semantic Interface for Dependencies

This Talk

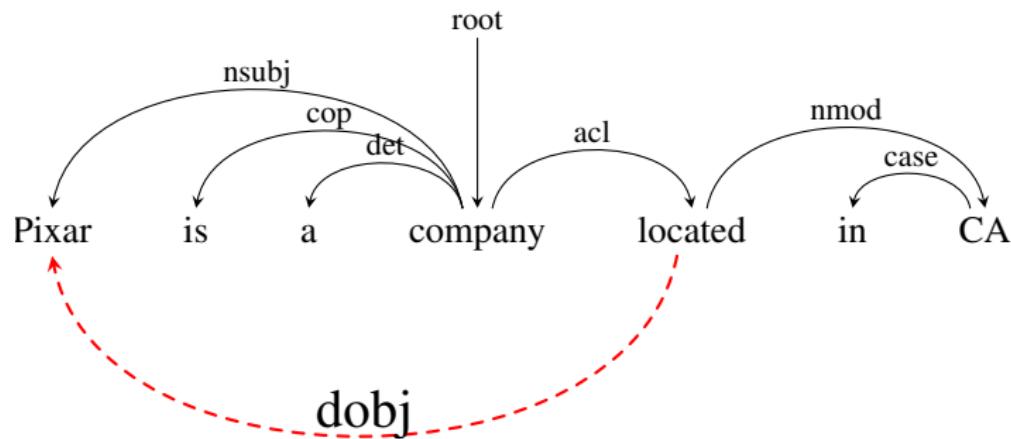
1. Dependencies to Logical Forms
2. Freebase Semantic Parsing using Logical Forms
3. Exploiting Text for Freebase Semantic Parsing

1. DepLambda: Dependencies to Logical Forms

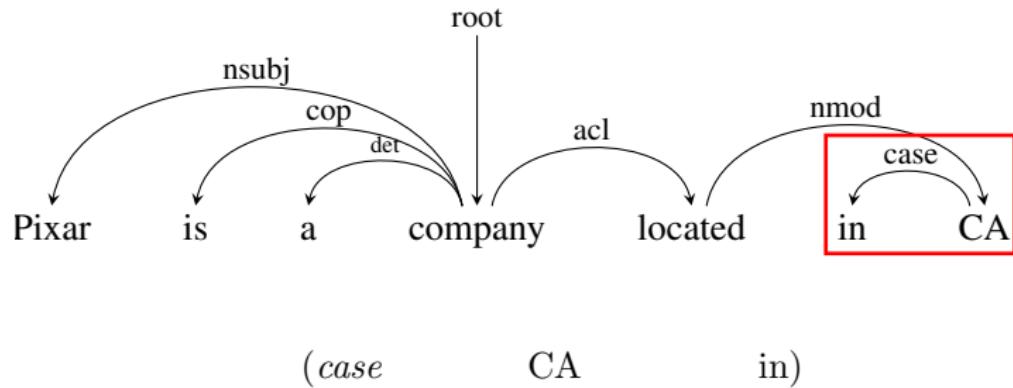
Dependencies to Logical Forms


$$\exists z. \text{company}(z) \wedge \text{located}(z_e) \wedge \text{arg}_2(z_e, \text{Pixar}) \wedge \text{arg}_{\text{in}}(z_e, \text{CA})$$

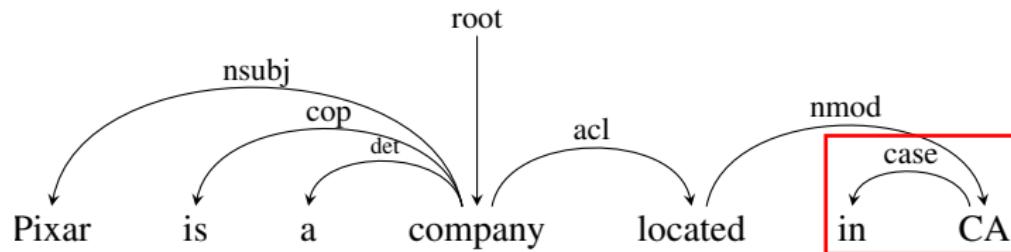
Dependencies to Logical Forms


$$\exists z. \text{company}(z, \text{Pixar}) \wedge \text{located}(z_e) \wedge \text{arg}_2(z_e, \text{Pixar}) \wedge \text{arg}_{\text{in}}(z_e, \text{CA})$$

Dependencies to Logical Forms

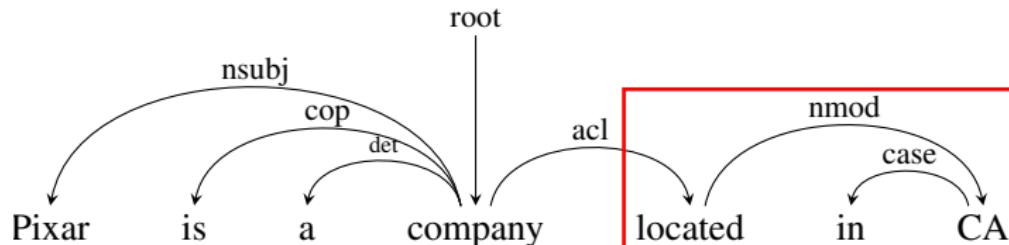


Dependencies to Logical Forms



$$\begin{array}{c} (\text{case} \quad \text{CA} \quad \text{in}) \\ \lambda f g x. f(x) \quad \lambda x. \text{CA}(x_a) \quad \lambda x. \text{empty}(x) \\ \hline \lambda x. \text{CA}(x_a) \end{array}$$

Dependencies to Logical Forms

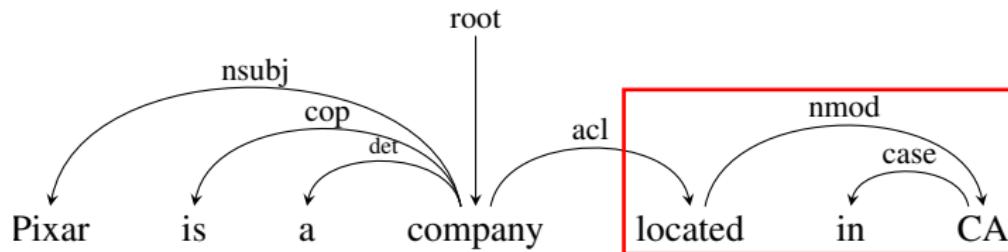


(*nmod*

located

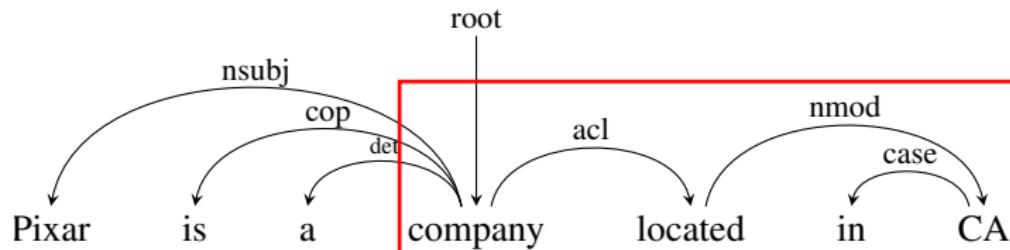
in CA)

Dependencies to Logical Forms



$$\frac{\begin{array}{c} (nmod \quad \text{located} \quad \text{in CA}) \\ \lambda f g z. \exists x. \quad \lambda x. \text{located}(x_e) \quad \hline \\ f(z) \wedge g(x) \wedge \\ \wedge \text{argin}(z_e, x_a) \end{array}}{\lambda z. \text{located}(z_e) \wedge \text{CA}(x_a) \wedge \text{argin}(z_e, x_a)}$$

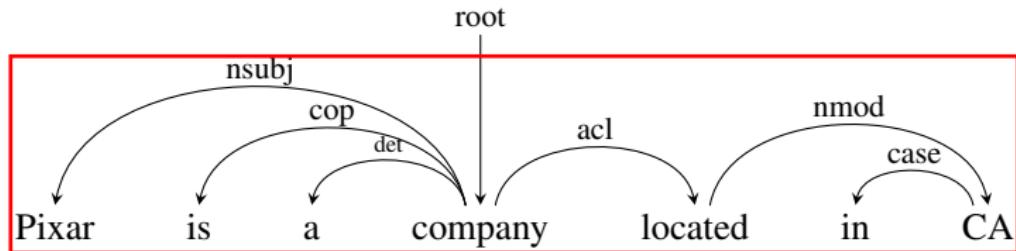
Dependencies to Logical Forms



$$\frac{\begin{array}{c} (acl \quad \text{company} \quad \text{located in CA}) \\ \lambda f g x. \exists z. \quad \lambda x. \text{compay}(x_a) \quad \hline \\ f(x) \wedge g(x) \wedge \\ \arg_2(z_e, x_a) \end{array}}{\lambda g z. \exists x. \text{located}(z_e) \wedge \text{CA}(x_a) \wedge \arg_{\text{in}}(z_e, x_a)}$$

$$\lambda x. \exists y z. \text{company}(x_a) \wedge \text{located}(z_e) \wedge \text{CA}(y_a) \wedge \arg_2(z_e, x_a) \wedge \arg_{\text{in}}(z_e, y_a)$$

Dependencies to Logical Forms


$$\exists z. \text{company}(\text{Pixar}) \wedge \text{located}(z_e) \wedge \text{arg}_2(z_e, \text{Pixar}) \wedge \text{arg}_{\text{in}}(z_e, \text{CA})$$

Quick Intro to Lambda Calculus

$$(\lambda x.M)N = M[x := N]$$

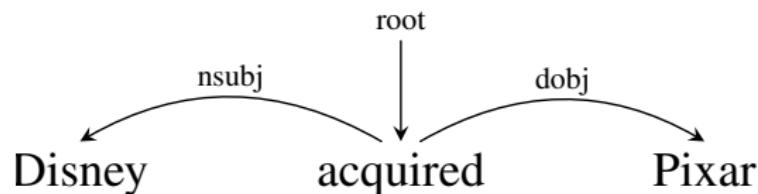
$$\begin{aligned} \text{sum}(2, 3) &= (\lambda x. \lambda y. (+\ x\ y))(2, 3) \\ &= (\lambda y. (+\ 2\ y))(3) \\ &= (+\ 2\ 3) \\ &= 5 \end{aligned}$$

$$\mathbf{TYPE}[\text{sum}] = \text{int} \rightarrow \text{int} \rightarrow \text{int}$$

$$\text{sum}(4, \text{sum}(2, 3)) = 9$$

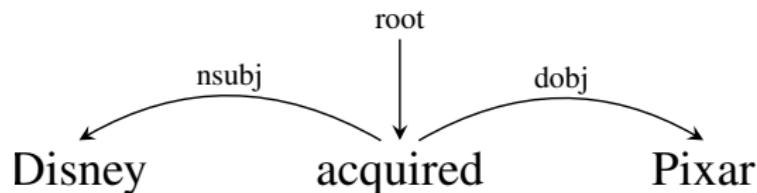
$$\text{sum}(2, \text{"three"}) = \text{TypeError}$$

Dependencies to Logical Forms


$$\lambda z. \exists xy. \text{acquired}(z_e) \wedge \text{Pixar}(y_a) \wedge \text{Disney}(x_a) \wedge \\ \text{arg}_1(z_e, x_a) \wedge \text{arg}_2(z_e, y_a)$$

Dependencies to Logical Forms

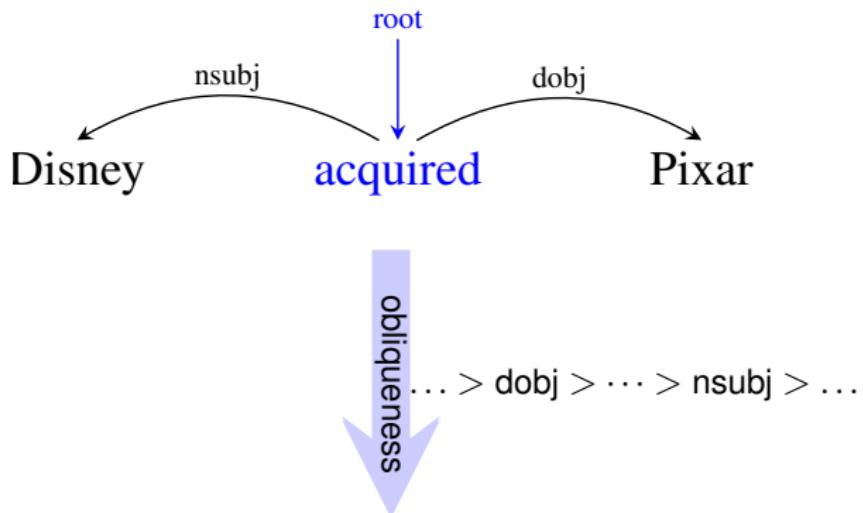
Our Approach



Let dependency labels drive the composition

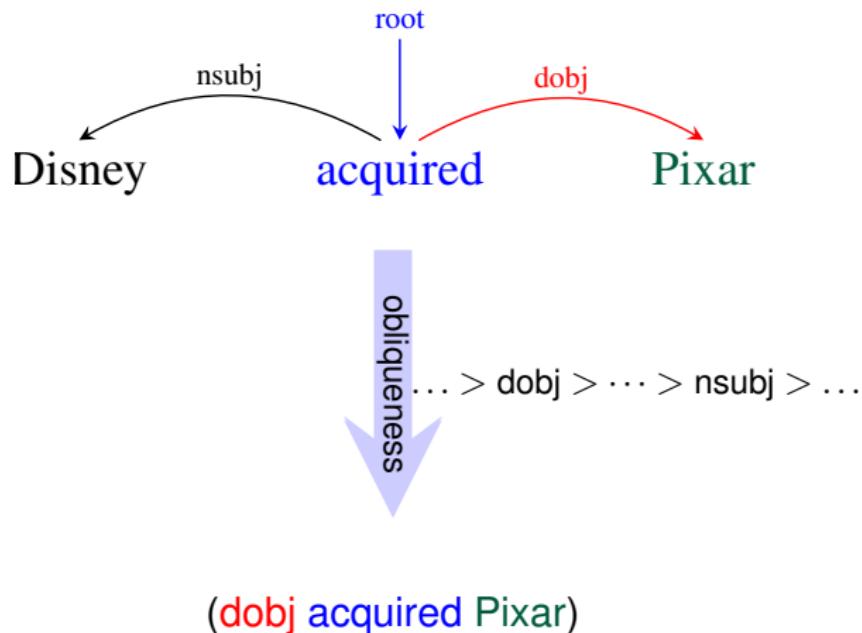
Dependencies to Logical Forms

Our Approach



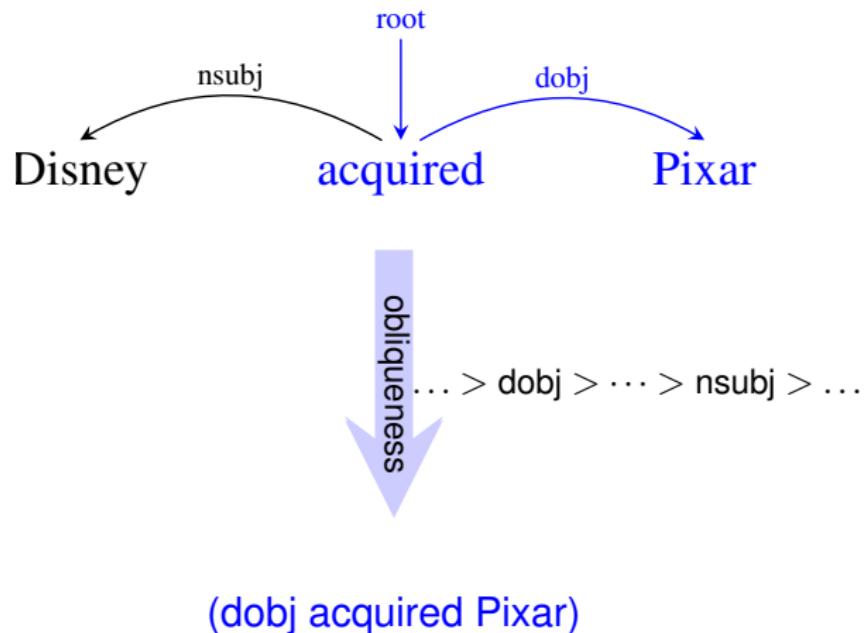
Dependencies to Logical Forms

Our Approach



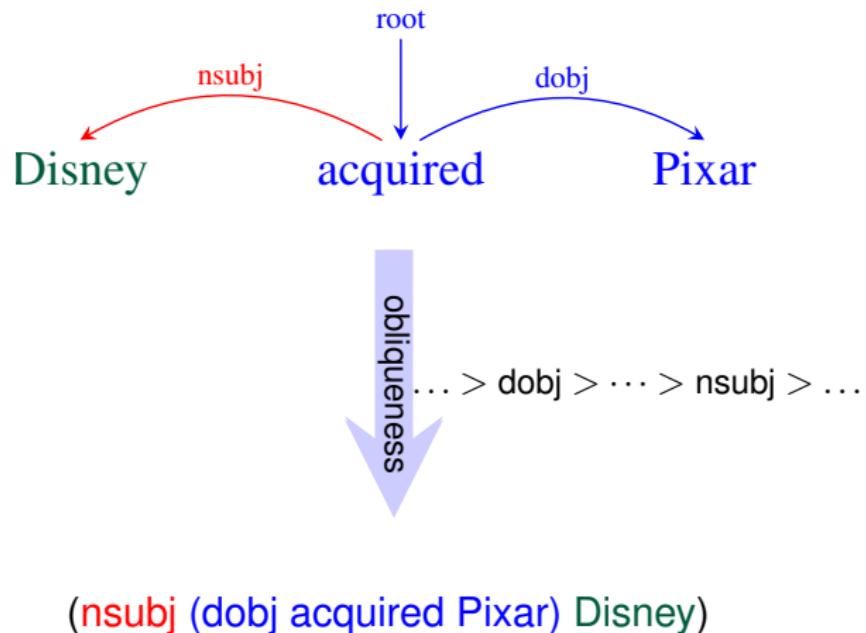
Dependencies to Logical Forms

Our Approach



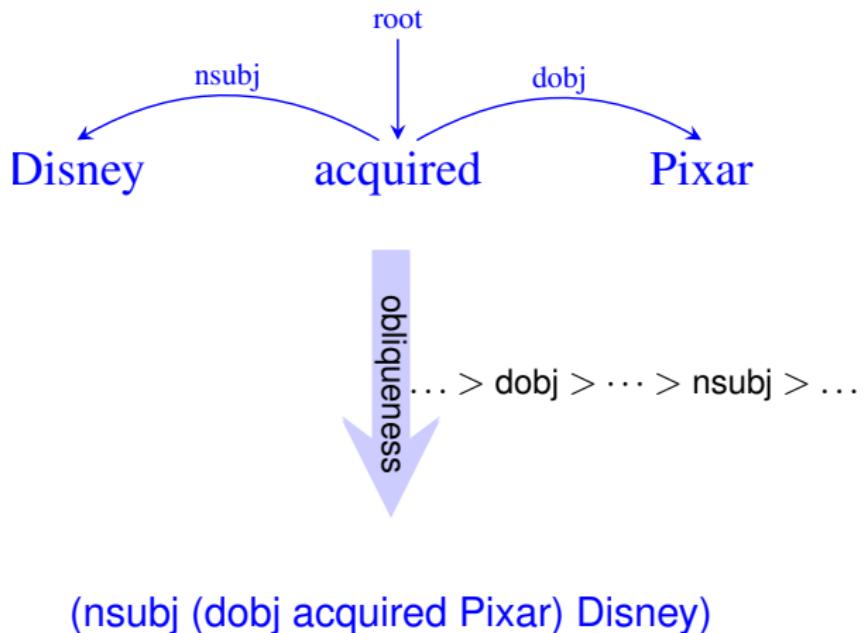
Dependencies to Logical Forms

Our Approach



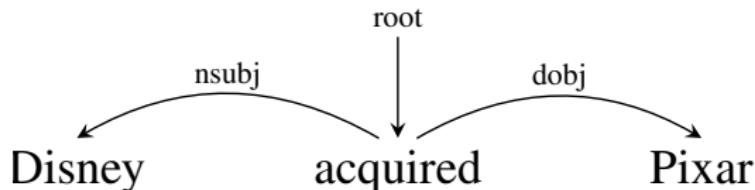
Dependencies to Logical Forms

Our Approach



Dependencies to Logical Forms

Our Approach

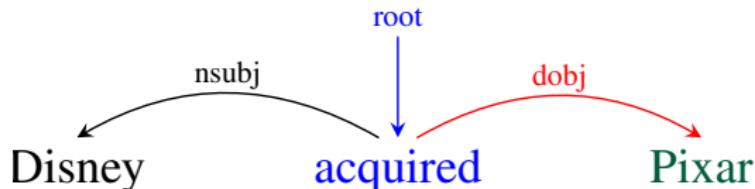


(nsubj (dobj acquired Pixar) Disney)

$$\lambda z. \exists xy. \text{acquired}(z_e) \wedge \text{Pixar}(y_a) \wedge \text{Disney}(x_a) \wedge \\ \text{arg}_1(z_e, x_a) \wedge \text{arg}_2(z_e, y_a)$$

Dependencies to Logical Forms

Lambda Calculus

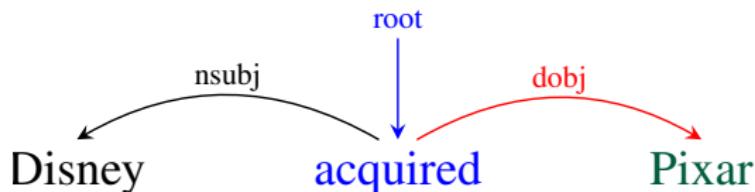


Lambda Calculus Basic Types

- ▶ Individuals: **Ind** (also denoted by $._a$)
- ▶ Events: **Event** (also denoted by $._e$)
- ▶ Truth values: **Bool**

Dependencies to Logical Forms

Lambda Calculus



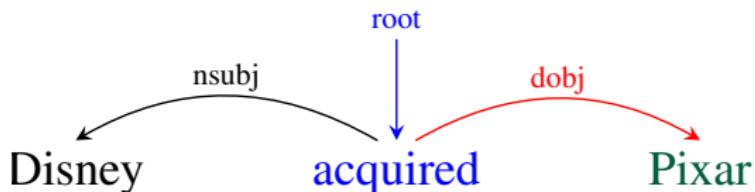
Lambda Expression for words

$$\text{acquired} \Rightarrow \lambda x. \text{acquired}(x_e)$$

$$\text{Pixar} \Rightarrow \lambda x. \text{Pixar}(x_a)$$

Dependencies to Logical Forms

Lambda Calculus

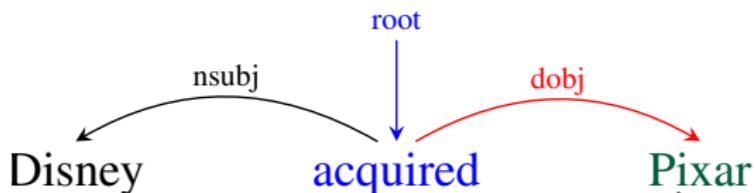


Lambda Expression for dependency labels

$$\text{dobj} \Rightarrow \lambda f\ g\ z . \exists x . f(z) \wedge g(x) \wedge \text{arg}_2(z_e, x_a)$$

Dependencies to Logical Forms

Lambda Calculus

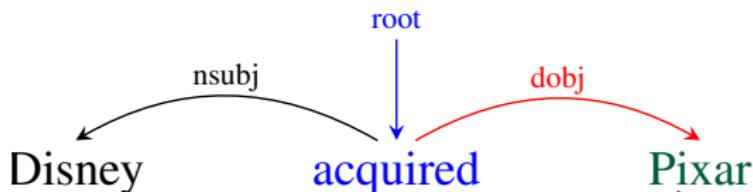


Lambda Expression for dependency labels

$$\text{dobj} \Rightarrow \lambda f\ g\ z . \exists x . f(z) \wedge g(x) \wedge \text{arg}_2(z_e, x_a)$$

Dependencies to Logical Forms

Lambda Calculus



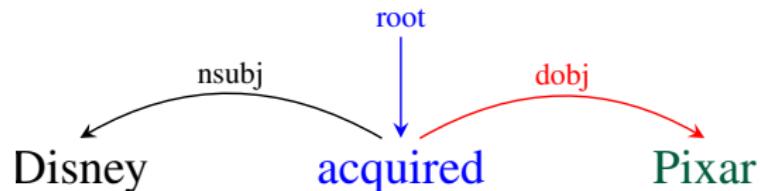
Lambda Expression for dependency labels

$$\text{dobj} \Rightarrow \lambda f\ g\ z . \exists x . f(z) \wedge g(x) \wedge \text{arg}_2(z_e, x_a)$$

This operation mirrors the tree structure

Dependencies to Logical Forms

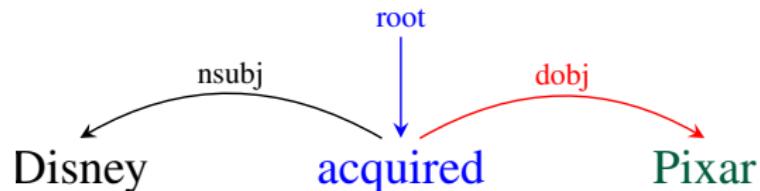
Composition



(**dobj** **acquired** **Pixar**)
 $\lambda f g z. \exists y. f(z) \wedge g(y) \wedge \arg_2(z_e, y_a)$
 $\lambda z. \text{acquired}(z_e)$
 $\lambda y. \text{Pixar}(y_a)$

Dependencies to Logical Forms

Composition

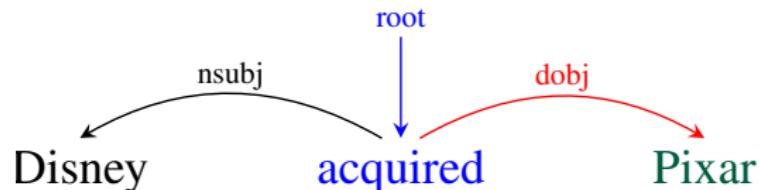


$$\begin{array}{ccc} (\text{dobj} & \text{acquired} & \text{Pixar}) \\ \lambda fgz. \exists y. & \lambda z. \text{acquired}(z_e) & \lambda y. \text{Pixar}(y_a) \\ f(z) \wedge g(y) \wedge \\ \text{arg}_2(z_e, y_a) \end{array}$$

$$\begin{array}{c} \lambda gz. \exists y. \text{acquired}(z_e) \wedge g(y) \\ \wedge \text{arg}_2(z_e, y_a) \end{array}$$

Dependencies to Logical Forms

Composition



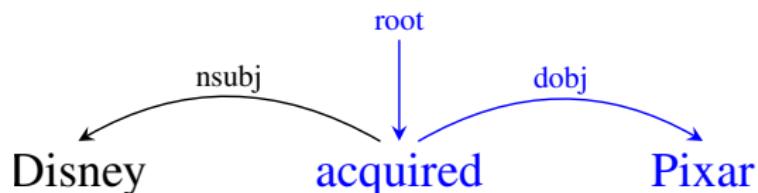
$$\begin{array}{ccc} (\mathbf{dobj}) & \mathbf{acquired} & \mathbf{Pixar} \\ \lambda f g z. \exists y. & \lambda z. \text{acquired}(z_e) & \lambda y. \text{Pixar}(y_a) \\ f(z) \wedge g(y) \wedge \\ \text{arg}_2(z_e, y_a) \end{array}$$

$$\begin{array}{c} \lambda g z. \exists y. \text{acquired}(z_e) \wedge g(y) \\ \wedge \text{arg}_2(z_e, y_a) \end{array}$$

$$\begin{array}{c} \lambda z. \exists y. \text{acquired}(z_e) \wedge \text{Pixar}(y_a) \\ \wedge \text{arg}_2(z_e, y_a) \end{array}$$

Dependencies to Logical Forms

Composition

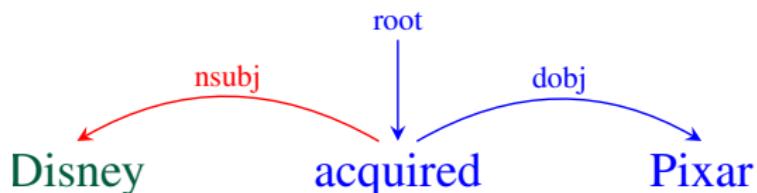


(**dobj** **acquired** **Pixar**)

$$\begin{aligned} \lambda z. \exists y. & \text{ acquired}(z_e) \wedge \text{Pixar}(y_a) \\ & \wedge \text{arg}_2(z_e, y_a) \end{aligned}$$

Dependencies to Logical Forms

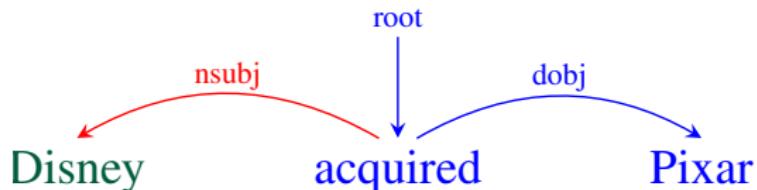
Composition



$$\begin{array}{cccc} \text{(nsubj} & \text{(dobj} & \text{acquired} & \text{Disney)} \\ \lambda f g z. \exists x. & \hline & \lambda z. \exists y. \text{acquired}(z_e) \wedge \text{Pixar}(y_a) \\ f(z) \wedge g(x) \wedge & & \wedge \text{arg}_2(z_e, y_a) \\ \text{arg}_1(z_e, x_a) & & & \lambda x. \text{Disney}(x_a) \end{array}$$

Dependencies to Logical Forms

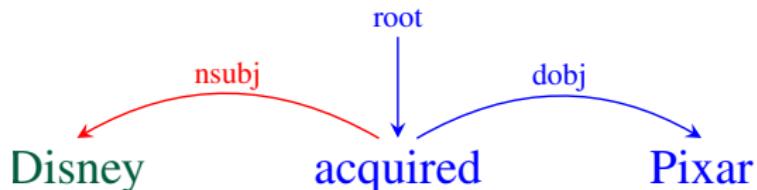
Composition



$$\begin{array}{cccc} \text{(nsubj} & \text{(dobj} & \text{acquired} & \text{Pixar)} \\ \lambda f g z. \exists x. & \hline & \lambda z. \exists y. \text{acquired}(z_e) \wedge \text{Pixar}(y_a) \\ f(z) \wedge g(x) \wedge & & \wedge \text{arg}_1(z_e, x_a) & \lambda x. \text{Disney}(x_a) \\ \text{arg}_1(z_e, x_a) & & \wedge \text{arg}_2(z_e, y_a) & \\ \hline \\ \lambda g z. \exists x y. \text{acquired}(z_e) \wedge \text{Pixar}(y_a) \wedge g(x) \wedge \\ & & \text{arg}_1(z_e, x_a) \wedge \text{arg}_2(z_e, y_a) & \end{array}$$

Dependencies to Logical Forms

Composition



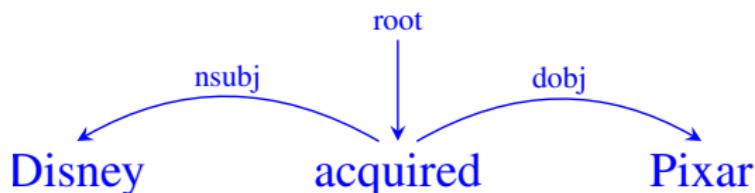
$$\begin{array}{cccc} \text{(nsubj} & \text{(dobj} & \text{acquired} & \text{Disney)} \\ \lambda f g z. \exists x. & \hline & \lambda z. \exists y. \text{acquired}(z_e) \wedge \text{Pixar}(y_a) \\ f(z) \wedge g(x) \wedge & & \wedge \text{arg}_1(z_e, x_a) & \lambda x. \text{Disney}(x_a) \\ \text{arg}_1(z_e, x_a) & & \text{arg}_2(z_e, y_a) & \end{array}$$

$$\begin{array}{c} \lambda g z. \exists x y. \text{acquired}(z_e) \wedge \text{Pixar}(y_a) \wedge g(x) \wedge \\ \text{arg}_1(z_e, x_a) \wedge \text{arg}_2(z_e, y_a) \end{array}$$

$$\begin{array}{c} \lambda z. \exists x y. \text{acquired}(z_e) \wedge \text{Pixar}(y_a) \wedge \text{Disney}(x_a) \wedge \\ \text{arg}_1(z_e, x_a) \wedge \text{arg}_2(z_e, y_a) \end{array}$$

Dependencies to Logical Forms

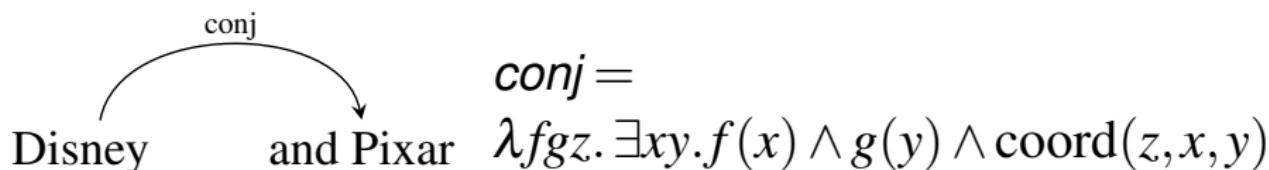
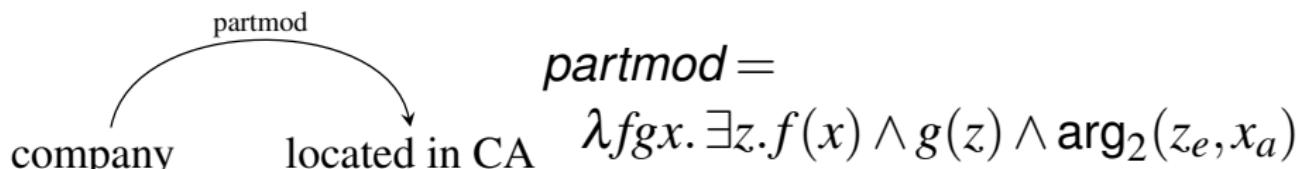
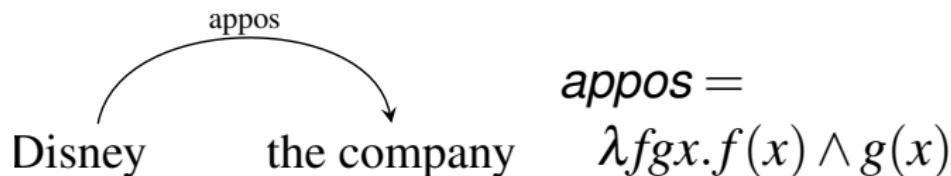
Composition



(**nsubj** (**dobj** **acquired** **Pixar**) **Disney**)

$$\lambda z. \exists xy. \text{acquired}(z_e) \wedge \text{Pixar}(y_a) \wedge \text{Disney}(x_a) \wedge \\ \text{arg}_1(z_e, x_a) \wedge \text{arg}_2(z_e, y_a)$$

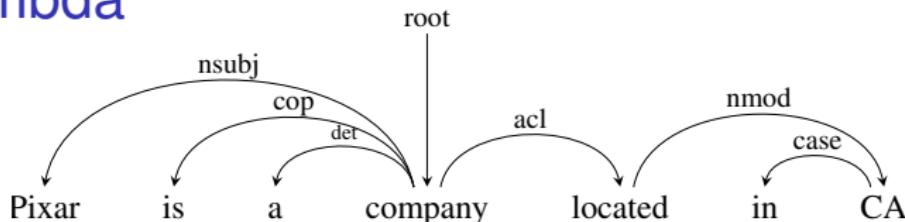
Dependencies to Logical Forms



Comparison with CCG

CCG	DepLambda
Lexicalized semantics $S \setminus NP/NP : \lambda y \lambda x \lambda e. \text{acquired}(e) \wedge \arg_1(e, x) \wedge \arg_2(e, y)$	Simple lexical semantics $\lambda x. \text{acquired}(x_e)$
Words drive composition	Dependencies drive composition
Language specific types	Mostly universal

DepLambda



... > **dobj** > ... > **nsubj** > ...

$$\dots \frac{(\text{acl } f(x) \wedge g(z)) \wedge \lambda x. \text{compay}(x_a) \quad (\text{nmod } f(z) \wedge g(x)) \wedge \lambda x. \text{located}(x_e) \quad (\text{case } \arg_2(z_e, x_a)) \wedge \lambda x. \text{CA}(x_a) \quad \lambda x. \text{empty}(x)}{\lambda x. \text{CA}(x_a)} \dots$$

lambda expression composition

$\exists z. \text{company}(\text{Pixar}) \wedge \text{located}(z_e) \wedge \arg_2(z_e, \text{Pixar}) \wedge \arg_{\text{in}}(z_e, \text{CA})$

DepLambda in a nutshell

Dependency tree is a series of compositions

Dependency label defines the composition function

Each function takes two typed-semantic sub-expressions

Returns typed-semantics of the larger expression

DepLambda: Current Status

Compositional Typed Semantics using Lambda Calculus

Two universal types

69 lambda calculus formulae

Output: General-purpose logical forms

DepLambda: Towards ...

Richer universal context-sensitive types

Richer composition functions, e.g., neural networks

Output: Any target semantic representation

- ▶ Semantic Parsing
- ▶ Machine Translation

This Talk

1. Dependencies to Logical Forms
2. Freebase Semantic Parsing using Logical Forms
3. Exploiting Text for Freebase Semantic Parsing

2. Freebase Semantic Parsing using Logical Forms

Reddy, Täckström, Kwiatkowski, Das, Collins, Steedman, Lapata (TACL 2016)
Reddy, Lapata, Steedman (TACL 2014)

Freebase Semantic Parsing

[Berant et al., 2013, Kwiatkowski et al., 2013]

Question

Who is the director of Titanic?

Answer

{James Cameron}



Titanic

1997 · Drama film/Romance · 3h 30m

7.7/10 · IMDb

88% · Rotten Tomatoes

James Cameron's "Titanic" is an epic, action-packed romance set against the ill-fated maiden voyage of the R.M.S. Titanic; the pride and joy of the White Star Line and, at the time, the larg... [More](#)

Initial release: November 18, 1997 ([London](#))

Director: James Cameron

Featured song: My Heart Will Go On

Cast



Leonardo
DiCaprio
Jack Dawson



Kate
Winslet
Rose DeWitt
Bukater



Billy
Zane
Caledon
Hockley



Gloria
Stuart
Mrs. Thayer
Rose DeWitt
Bukater



Kathy Bates
Cal's mother

Freebase Semantic Parsing

[Berant et al., 2013, Kwiatkowski et al., 2013]

Question

Who is the director of Titanic?

Logical Form

$$\lambda x. \exists e. \text{film director}(x) \wedge \text{Latent} \text{ film.directed_by}(e) \wedge \text{arg2}(y, x) \wedge \text{arg1}(e, \text{Titanic})$$

Answer

{James Cameron}



Titanic

1997 · Drama film/Romance · 3h 30m

7.7/10 · IMDb

88% · Rotten Tomatoes

James Cameron's "Titanic" is an epic, action-packed romance set against the ill-fated maiden voyage of the R.M.S. Titanic; the pride and joy of the White Star Line and, at the time, the larg... [More](#)

Initial release: November 18, 1997 ([London](#))

Director: James Cameron

Featured song: My Heart Will Go On

Cast



Leonardo
DiCaprio
Jack Dawson



Kate
Winslet
Rose DeWitt
Bukater



Billy
Zane
Caledon
Hockley



Gloria
Stuart
Mrs. Thayer



Kathy Bates
Cal Hockley

End-to-End Semantic Parsing

[Zelle & Mooney, 1996; Zettlemoyer & Collins, 2005; Kwiatkowski et al., 2010; Liang et al., 2011;
Artzi & Zettlemoyer, 2011; Krishnamurthy & Mitchell, 2012; Berant et al., 2013; Pasupat & Liang, 2015;
Yih et al., 2015]

Grammar learning

- ▶ who → $(S/(S/NP) : \lambda fx.f(EQ(x))$
- ▶ is → $(S\backslash NP)/NP : \lambda fgx.f(x) \wedge g(x)$
- ▶ director → $N : \lambda x. \text{film.director}(x)$
- ▶ of → $(NP\backslash NP)/NP :$
 $\lambda fgx. \exists ye. f(y) \wedge g(x) \wedge$
 $\text{film.directed_by}(e) \wedge \text{arg2}(e, x) \wedge$
 $\text{arg1}(e, y)$

Question

Who is the director of Titanic?

Logical Form

$$\lambda x. \exists e. \text{film.director}(x) \wedge \\ \text{film.directed_by}(e) \wedge \\ \text{arg2}(y, x) \wedge \text{arg1}(e, \text{Titanic})$$

Intermediate Semantic Parsing

[Kwiatkowski et al., 2013; Reddy et al., 2014; Choi et al., 2015; Artzi et al., 2015]

Language to
ungrounded logical form

Ungrounded logical form to
grounded logical form

Question

Who is the director of Titanic?

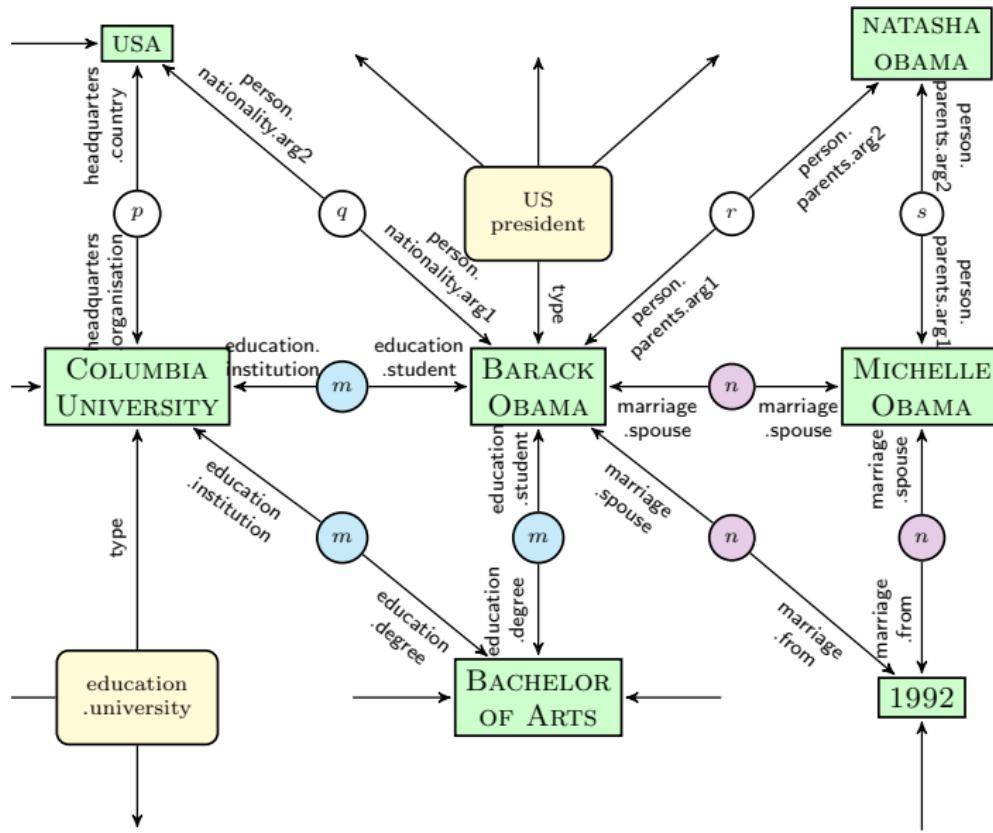
Ungrounded Logical Form

$$\lambda x. \text{TARGET}(x_a) \wedge \text{director}(x_a) \wedge \\ \text{director_event}(x_e) \wedge \\ \text{arg0}(x_e, x_a) \wedge \text{arg.of}(x_e, \text{Titanic})$$

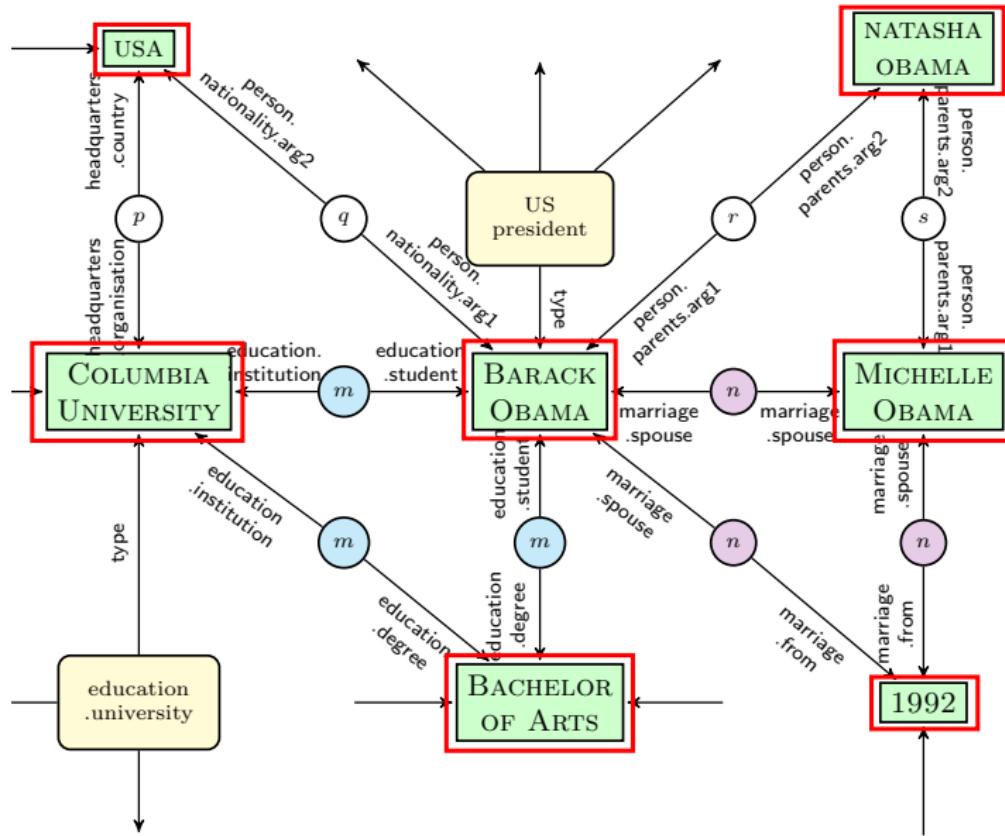
Logical Form

$$\lambda x. \exists e. \text{film.director}(x) \wedge \\ \text{film.directed_by}(e) \wedge \\ \text{arg2}(e, x) \wedge \text{arg1}(e, \text{Titanic})$$

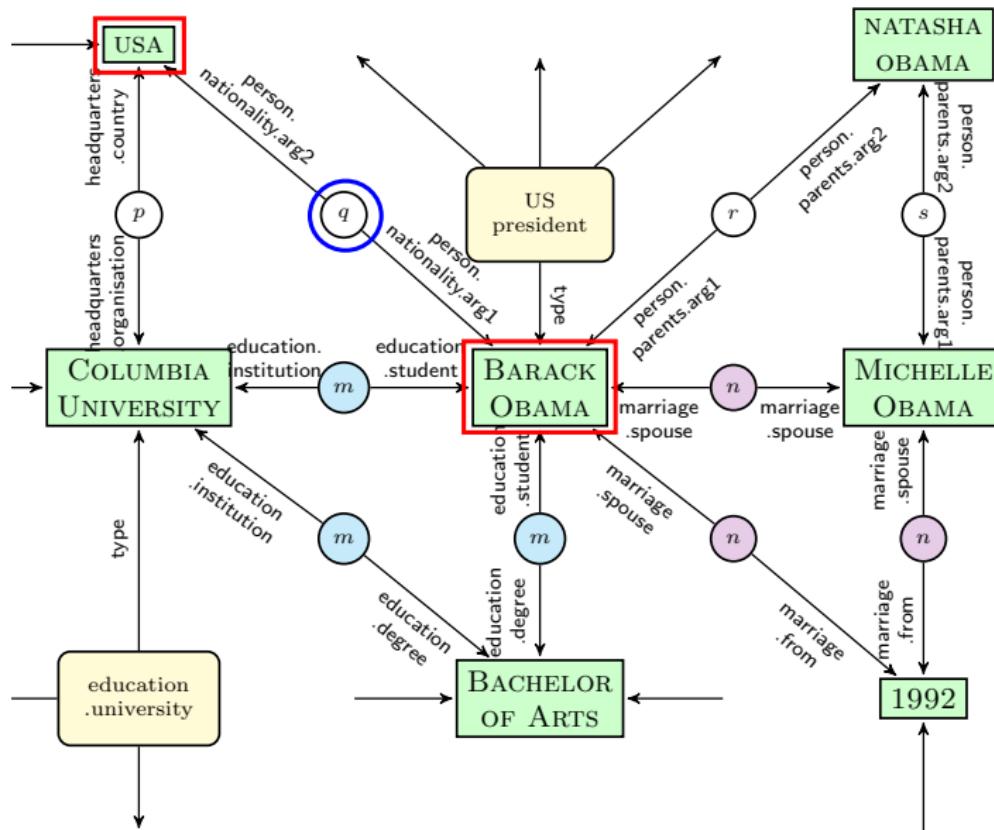
Freebase is a Graph



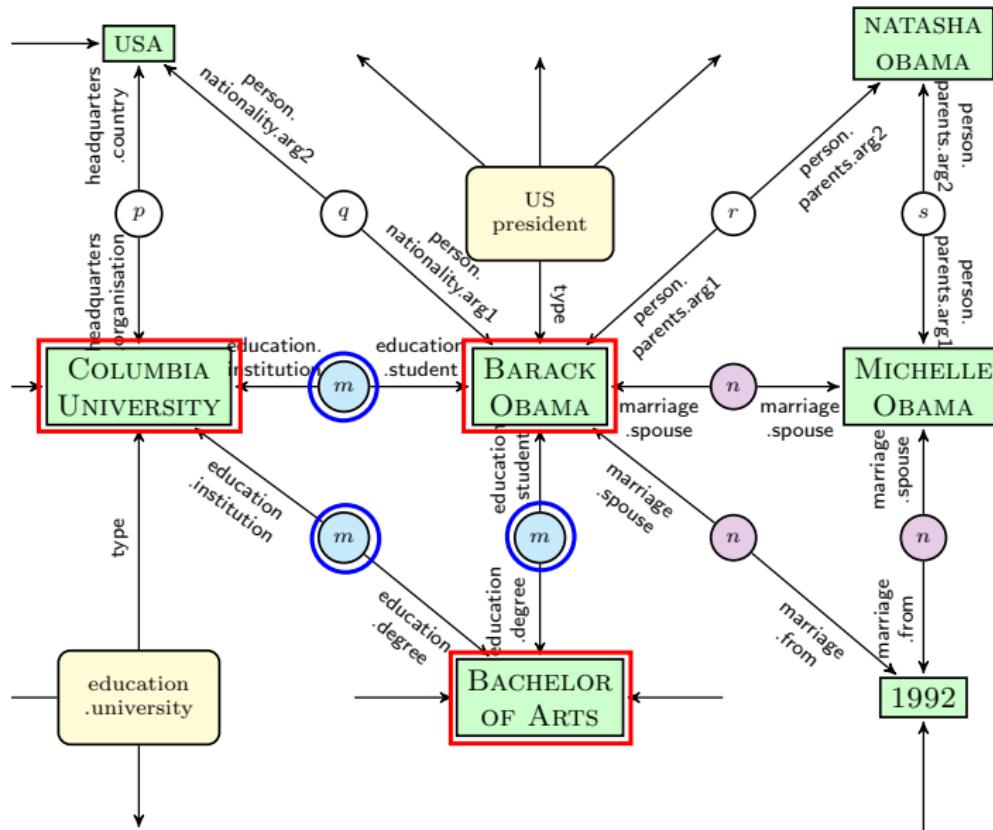
Freebase is a Graph



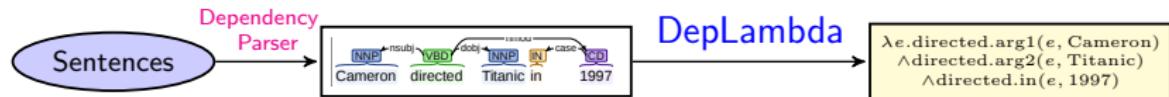
Freebase is a Graph



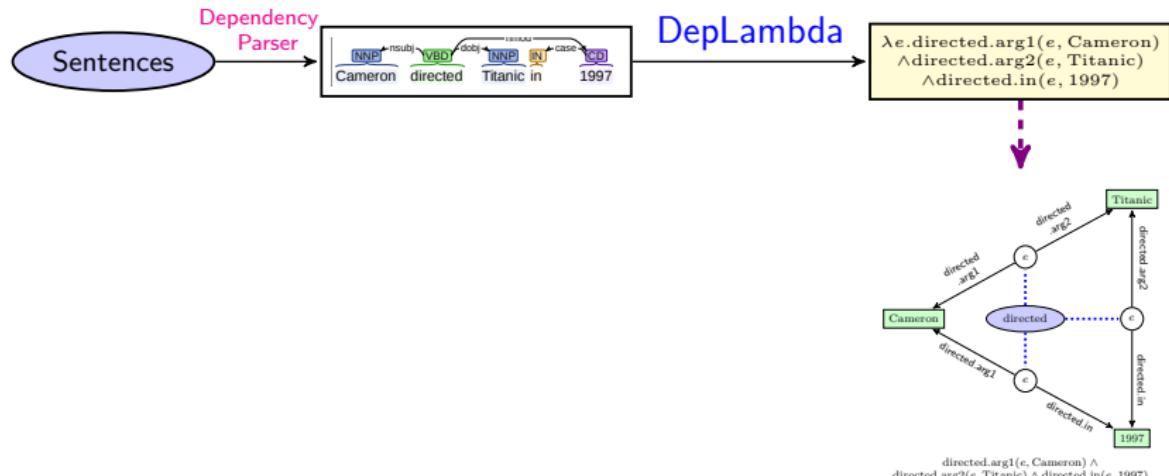
Freebase is a Graph



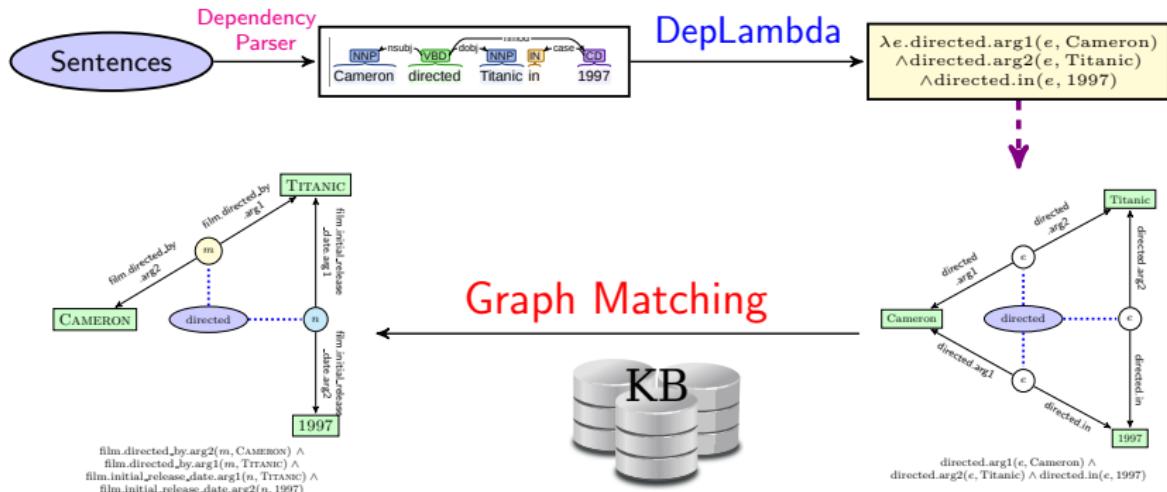
Semantic Parsing as Graph Matching



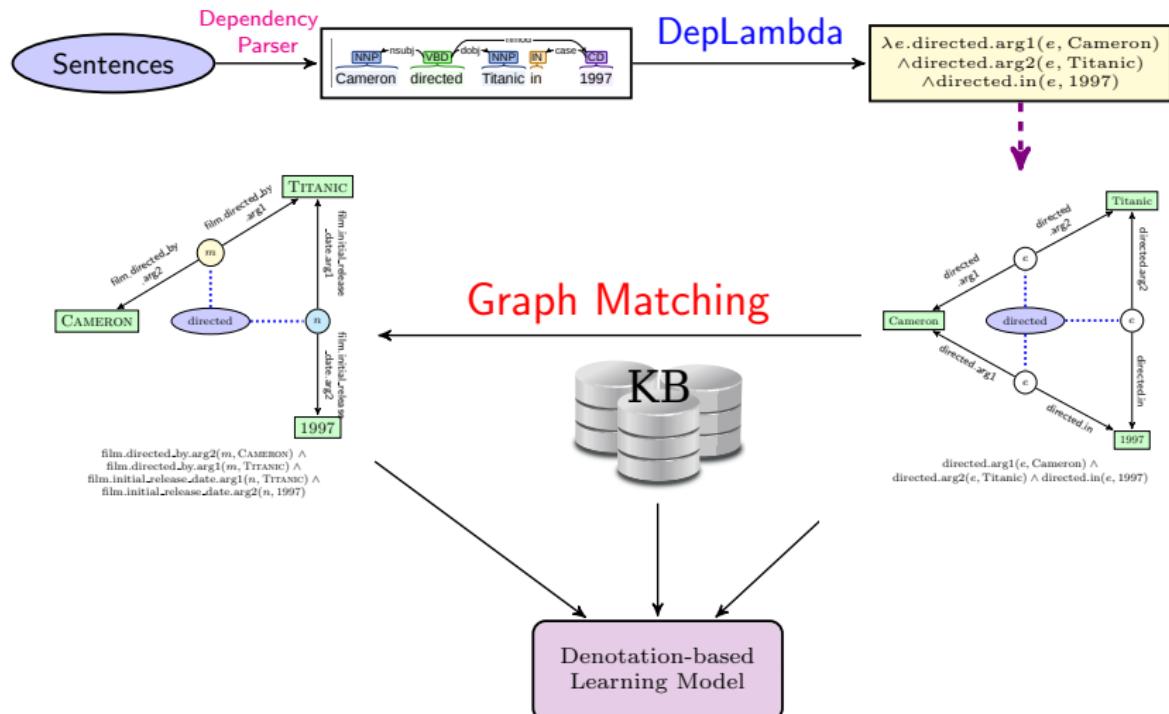
Semantic Parsing as Graph Matching



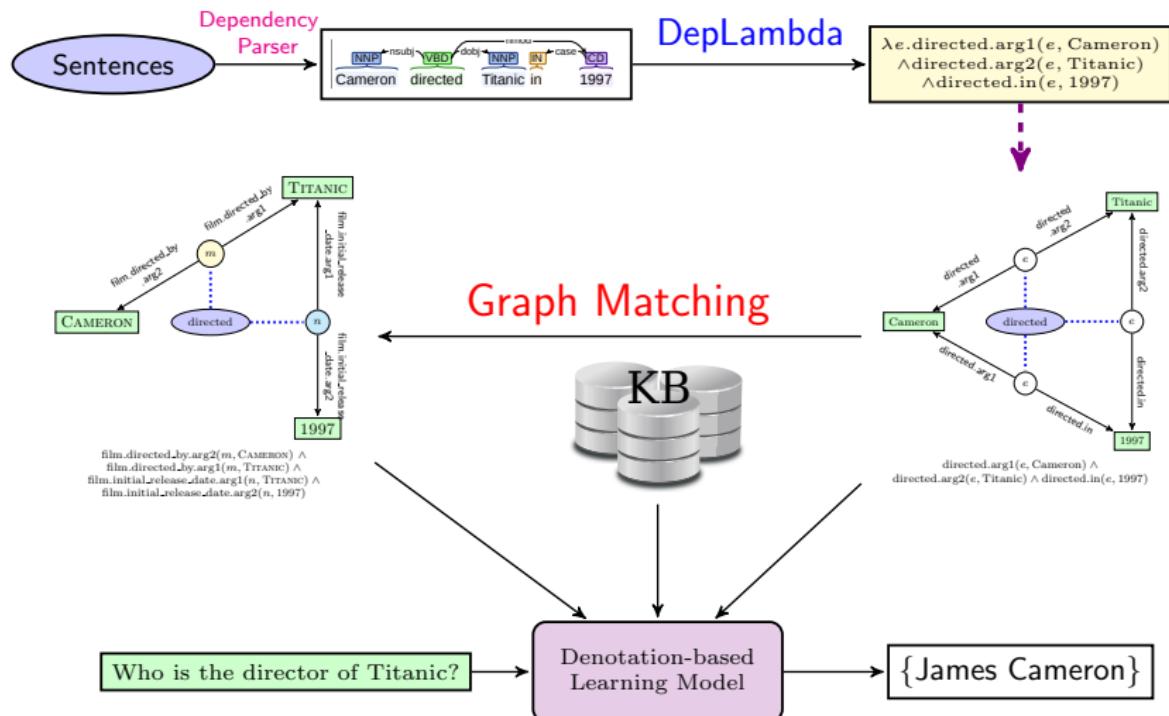
Semantic Parsing as Graph Matching



Semantic Parsing as Graph Matching



Semantic Parsing as Graph Matching



Logical Form to Ungrounded Graph

Cameron directed Titanic in 1997

$$\lambda e. \text{directed}.\text{arg1}(e, \text{Cameron}) \wedge \text{directed}.\text{arg2}(e, \text{Titanic}) \wedge \text{directed}.\text{in}(e, 1997)$$

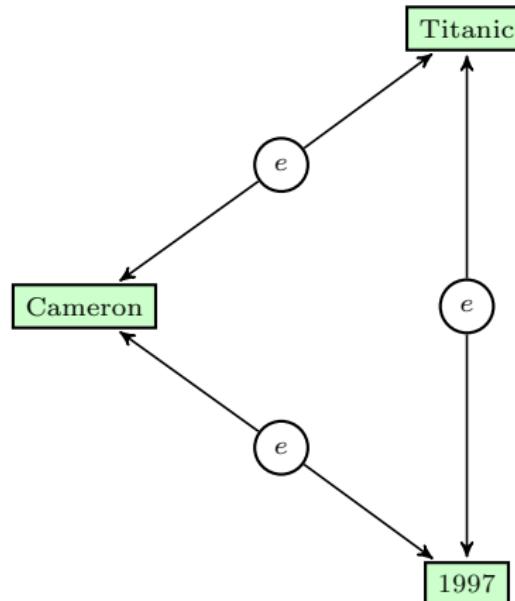
Titanic

Cameron

1997

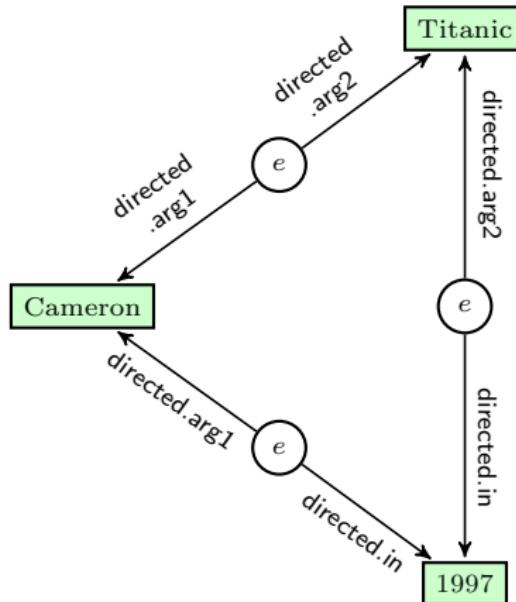
Logical Form to Ungrounded Graph

Cameron directed Titanic in 1997

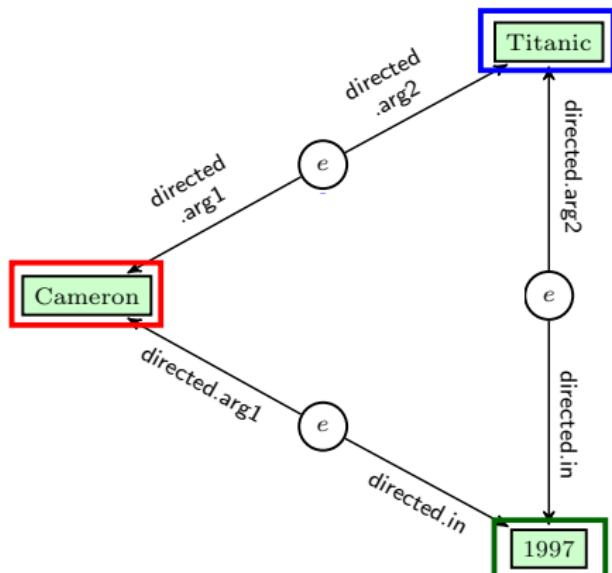
$$\lambda e. \text{directed}.\text{arg1}(e, \text{Cameron}) \wedge \text{directed}.\text{arg2}(e, \text{Titanic}) \wedge \text{directed}.\text{in}(e, 1997)$$


Logical Form to Ungrounded Graph

Cameron directed Titanic in 1997

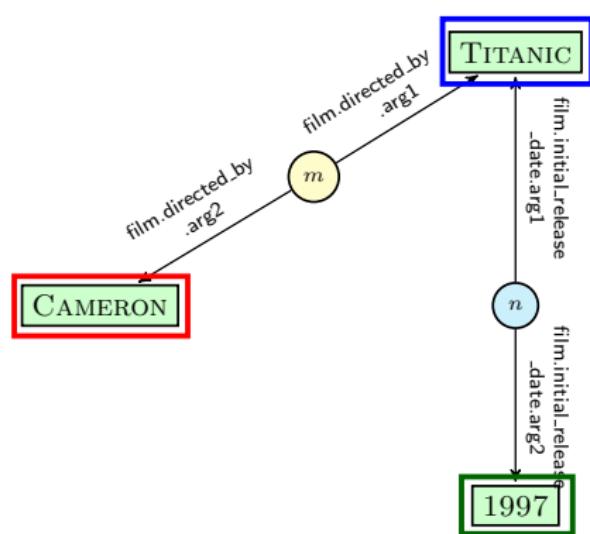
$$\lambda e. \text{directed.arg1}(e, \text{Cameron}) \wedge \text{directed.arg2}(e, \text{Titanic}) \wedge \\ \text{directed.in}(e, 1997)$$


Graph Matching



`directed.arg1(e, Cameron) ∧
directed.arg2(e, Titanic) ∧ directed.in(e, 1997)`

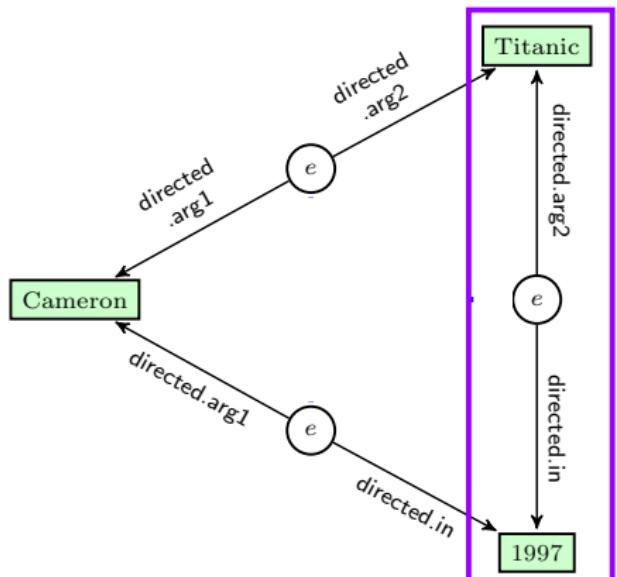
Ungrounded Graph



`film.directed_by.arg2(m, CAMERON) ∧
film.directed_by.arg1(m, TITANIC) ∧
film.initial_release_date.arg1(n, TITANIC) ∧
film.initial_release_date.arg2(n, 1997)`

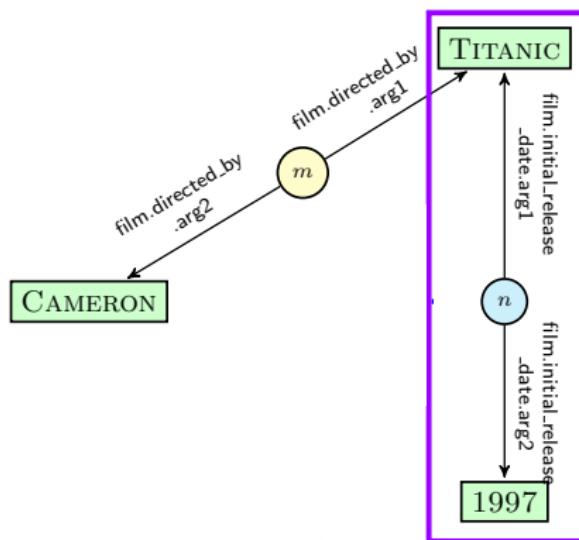
Grounded Graph

Graph Matching



$\text{directed.arg1}(e, \text{Cameron}) \wedge$
 $\text{directed.arg2}(e, \text{Titanic}) \wedge \text{directed.in}(e, 1997)$

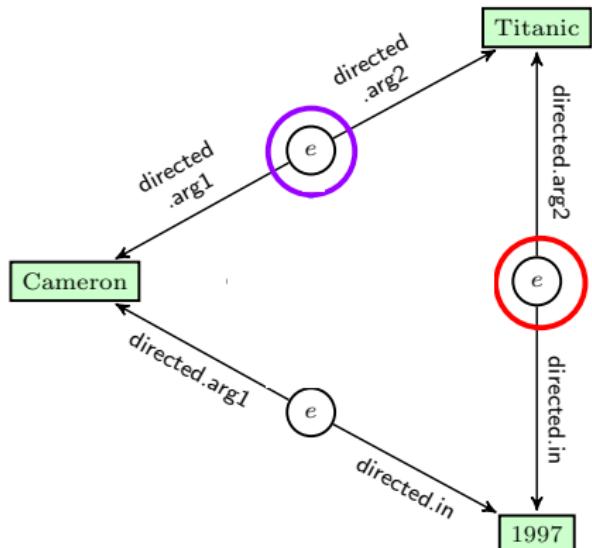
Ungrounded Graph



$\text{film.directed_by.arg2}(m, \text{CAMERON}) \wedge$
 $\text{film.directed_by.arg1}(m, \text{TITANIC}) \wedge$
 $\text{film.initial_release_date.arg1}(n, \text{TITANIC}) \wedge$
 $\text{film.initial_release_date.arg2}(n, 1997)$

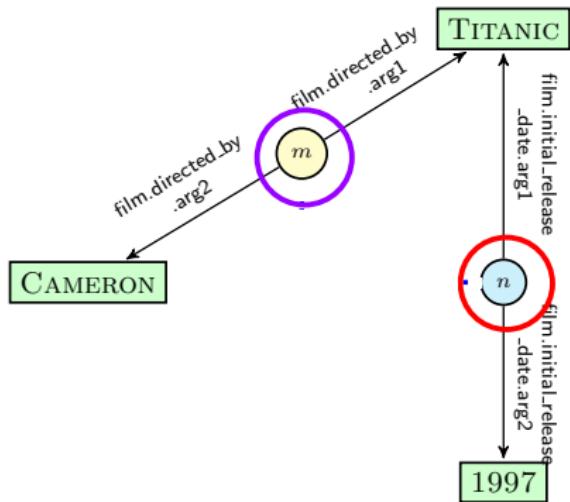
Grounded Graph

Graph Matching



$\text{directed.arg1}(e, \text{Cameron}) \wedge$
 $\text{directed.arg2}(e, \text{Titanic}) \wedge \text{directed.in}(e, 1997)$

Ungrounded Graph

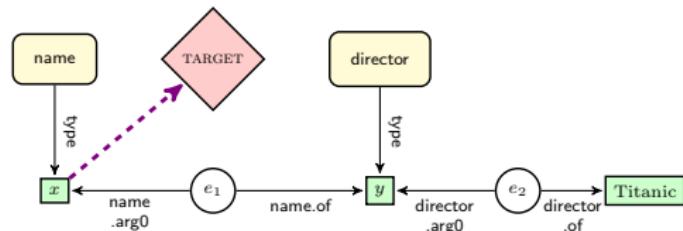


$\text{film.directed_by}(\text{arg1}(m, \text{CAMERON})) \wedge$
 $\text{film.directed_by}(\text{arg2}(m, \text{TITANIC})) \wedge$
 $\text{film.initial_release_date}(\text{arg1}(n, \text{TITANIC})) \wedge$
 $\text{film.initial_release_date}(\text{arg2}(n, 1997))$

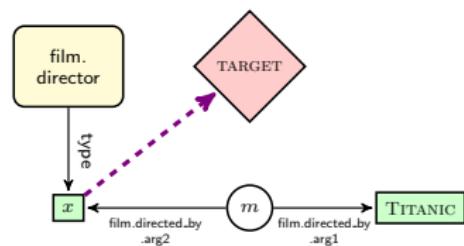
Grounded Graph

Graph Mismatch

What is the name of the director of *Titanic*?



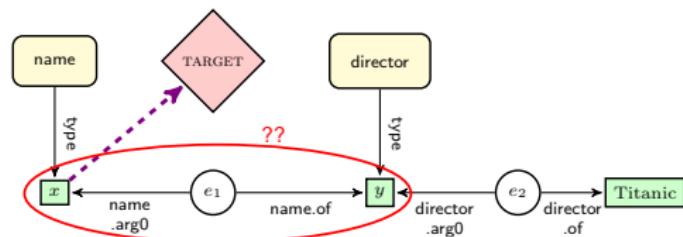
Ungrounded graph



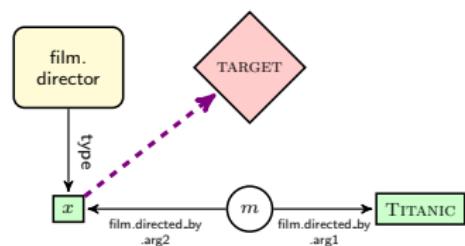
Freebase graph

Graph Mismatch

What is the name of the director of *Titanic*?



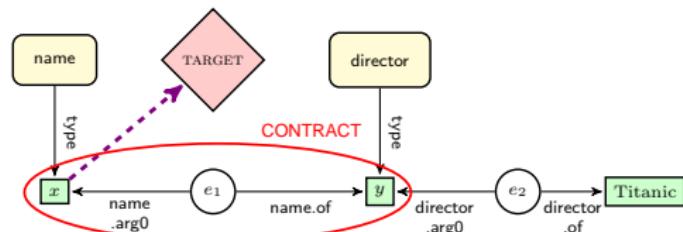
Ungrounded graph



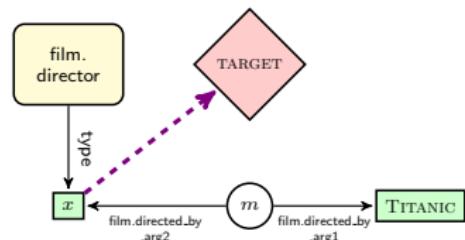
Freebase graph

Graph Mismatch

What is the name of the director of *Titanic*?



Ungrounded graph



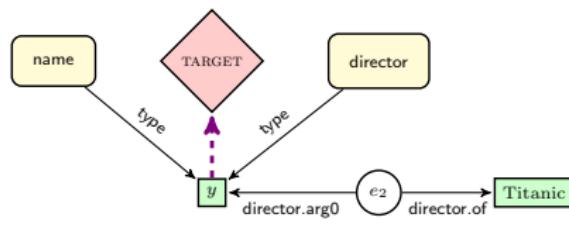
Freebase graph

- ▶ Paraphrasing is an alternative[†]

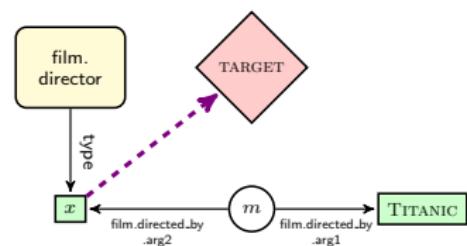
[†]Narayan, Reddy, Cohen (INLG 2016)

Graph Mismatch

What is the name of the director of *Titanic*?



Ungrounded graph



Freebase graph

Learning Model

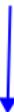
Structured Perceptron: Ranks grounded and ungrounded graph pairs

$$(\hat{g}, \hat{u}) = \arg \max_{g, u} \Phi(g, u, q, KB) \cdot \theta$$

↓
Feature Function

Learning Model

Structured Perceptron: Ranks grounded and ungrounded graph pairs

$$(\hat{g}, \hat{u}) = \arg \max_{g, u} \Phi(g, u, q, KB) \cdot \theta$$


Grounded Graph

Learning Model

Structured Perceptron: Ranks grounded and ungrounded graph pairs

$$(\hat{g}, \hat{u}) = \arg \max_{g, u} \Phi(g, u, q, KB) \cdot \theta$$


Ungrounded Graph

Learning Model

Structured Perceptron: Ranks grounded and ungrounded graph pairs

$$(\hat{g}, \hat{u}) = \arg \max_{g, u} \Phi(g, u, q, KB) \cdot \theta$$


Question

Learning Model

Structured Perceptron: Ranks grounded and ungrounded graph pairs

$$(\hat{g}, \hat{u}) = \arg \max_{g, u} \Phi(g, u, q, KB) \cdot \theta$$



Weight Vector

Learning Model

Structured Perceptron: Ranks grounded and ungrounded graph pairs

$$(\hat{g}, \hat{u}) = \arg \max_{g, u} \Phi(g, u, q, KB) \cdot \theta$$


Weight Vector

Training: Use gold graph to update weights

$$\theta \leftarrow \theta + \Phi(g^+, u^+, q, KB) - \Phi(\hat{g}, \hat{u}, q, KB)$$

Learning Model

- ★ We **do not** have access to gold graphs
- ★ Access only to the **answers** rather than the query
- ★ Instead use a **surrogate** gold graph

Learning Model

- ★ We **do not** have access to gold graphs
- ★ Access only to the **answers** rather than the query
- ★ Instead use a **surrogate** gold graph

Surrogate Gold Graph:

$$(g^+, u^+) = \underset{(g,u) \in O(q)}{\arg \max} \Phi(g, u, q, KB) \cdot \theta^t$$



Oracle Graphs

Baselines

SIMPLEGRAPH: All entities connected to a single event

Baselines

SIMPLEGRAPH: All entities connected to a single event

CCGGGRAPH: CCG logical forms

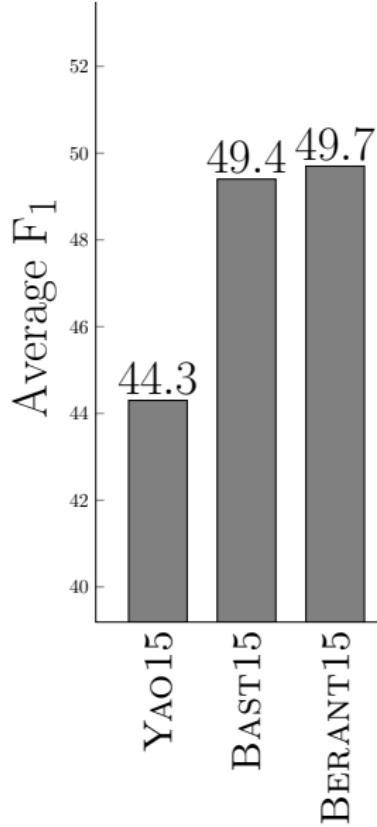
Baselines

SIMPLEGRAPH: All entities connected to a single event

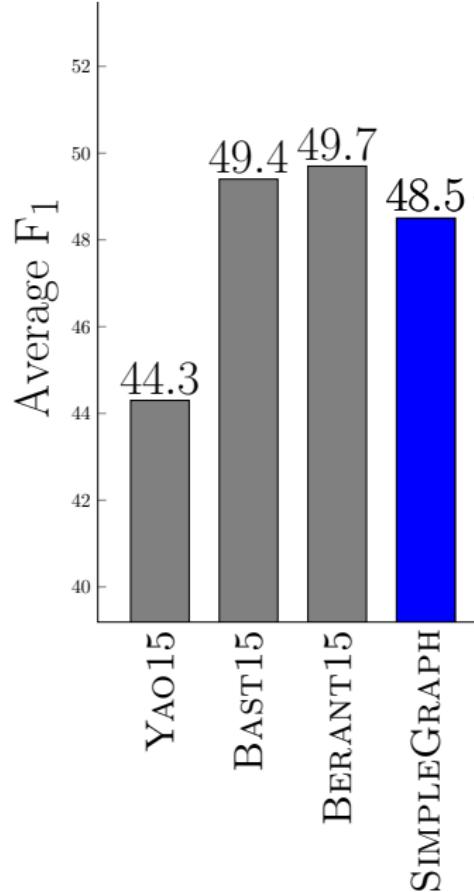
CCGGGRAPH: CCG logical forms

DEPTREE: Directly transduce a dependency tree to target graph

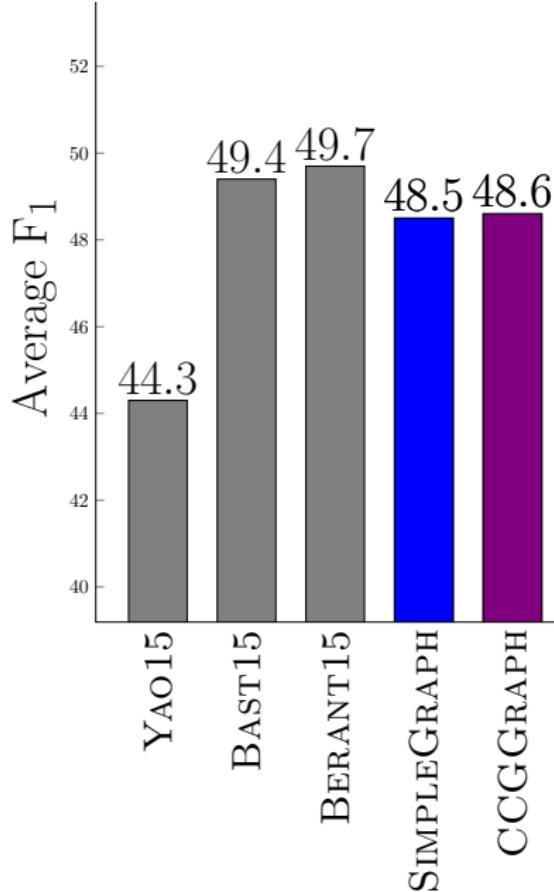
Results on WebQuestions



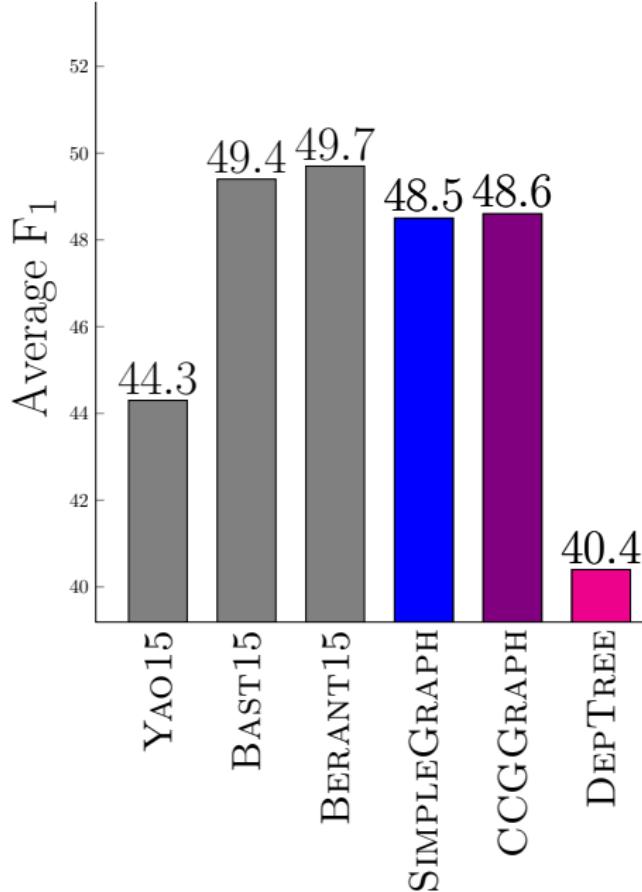
Results on WebQuestions



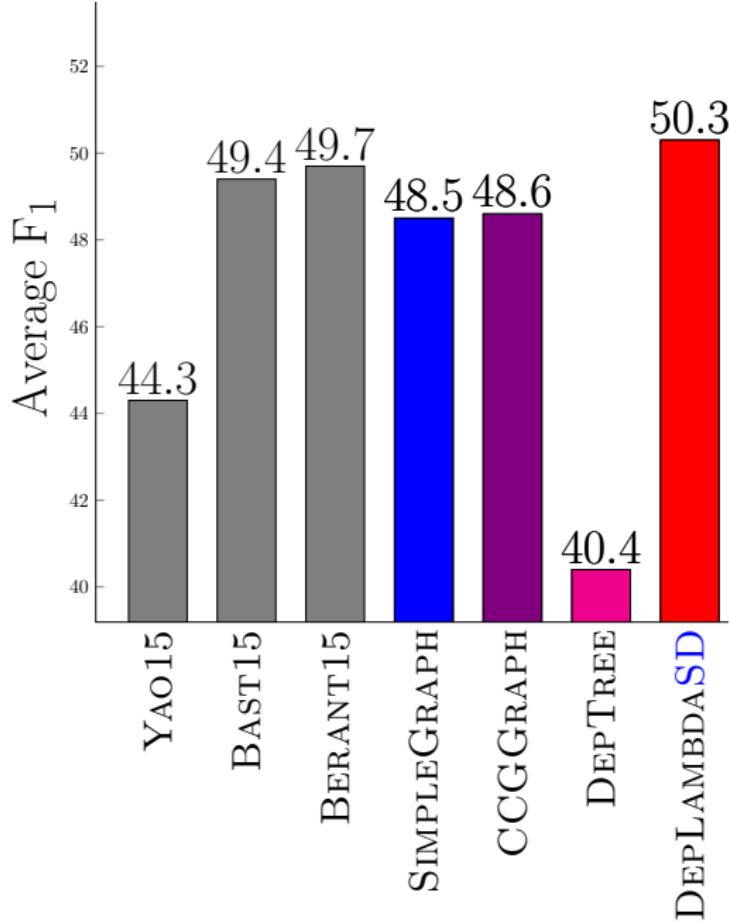
Results on WebQuestions



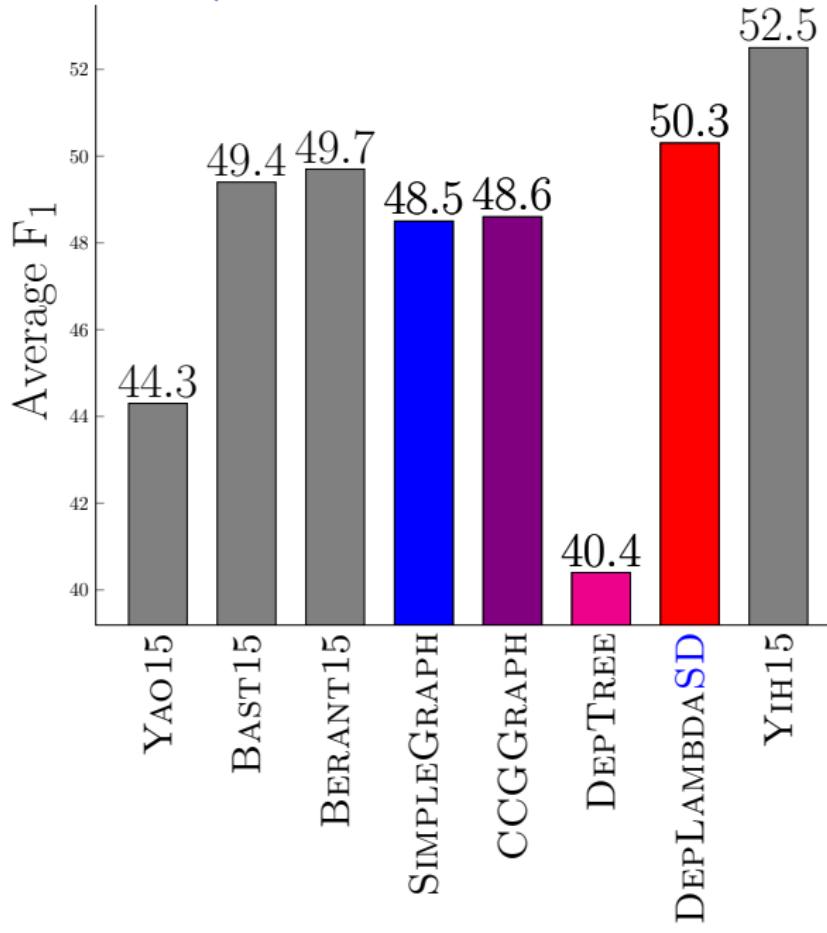
Results on WebQuestions



Results on WebQuestions



Results on WebQuestions



Results on Multilingual WebQuestions

Works for English, German,
Spanish. Graph: TODO

Error Analysis

CCGGGRAPH fails to produce logical forms for 4.5%.

- ▶ Sensitive to grammatical errors
- ▶ e.g., *what nestle owns?*

DepLambda fails only for 0.9%

- ▶ e.g., *what to do washington december*

This Talk

1. Dependencies to Logical Forms
2. Freebase Semantic Parsing using Logical Forms
3. Exploiting Text for Freebase Semantic Parsing

3. Exploiting Text for Freebase Semantic Parsing

Bisk, Reddy, Blitzer, Hockenmaier (EMNLP 2016)

Xu, Reddy, Feng, Huang, Zhao (ACL 2016)

Reddy, Lapata, Steedman (TACL 2014)

Semantic Parsing without QA pairs

Question

Who is the director of Titanic?

Logical Form

$\lambda x. \text{film.directed_by}(\text{Titanic}, x)$

Answer

{James Cameron}



Titanic

1997 · Drama film/Romance · 3h 30m

7.7/10 · [IMDb](#)

88% · [Rotten Tomatoes](#)

James Cameron's "Titanic" is an epic, action-packed romance set against the ill-fated maiden voyage of the R.M.S. Titanic; the pride and joy of the White Star Line and, at the time, the larg... [More](#)

Initial release: November 18, 1997 ([London](#))

Director: James Cameron

Featured song: [My Heart Will Go On](#)

Cast



Leonardo
DiCaprio
Jack Dawson



Kate
Winslet
Rose DeWitt
Bukater



Billy Zane
Caledon
Hockley



Gloria
Stuart
Mrs. Thelma
Brown



Kathy Bates
Molly Brown

Semantic Parsing without QA pairs

Question

Who is the director of Titanic?

Logical Form

Latent

$$\lambda x. \text{film.director}(x)(\text{Titanic}, x)$$

Answer

{James Cameron}



Titanic

1997 · Drama film/Romance · 3h 30m

7.7/10 · [IMDb](#)

88% · [Rotten Tomatoes](#)

James Cameron's "Titanic" is an epic, action-packed romance set against the ill-fated maiden voyage of the R.M.S. Titanic; the pride and joy of the White Star Line and, at the time, the larg... [More](#)

Initial release: November 18, 1997 ([London](#))

Director: James Cameron

Featured song: [My Heart Will Go On](#)

Cast



Leonardo
DiCaprio
Jack Dawson



Kate
Winslet
Rose DeWitt
Bukater



Billy Zane
Caledon
Hockley



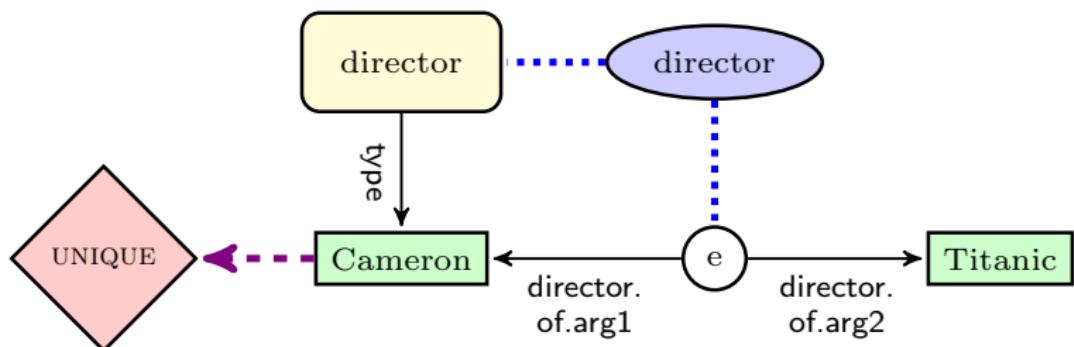
Gloria
Stuart
Mrs. Thelma
Brown



Kathy Bates
Molly Brown

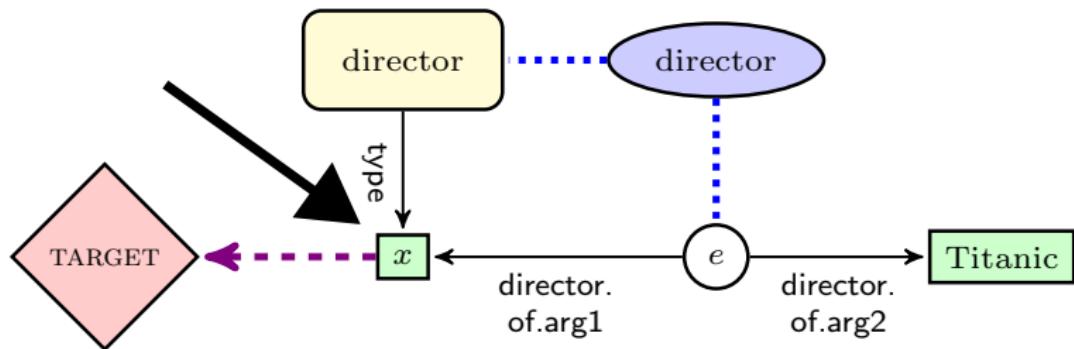
Learning from Text

Cameron is the director of Titanic.


$$\text{UNIQUE}(\text{Cameron}) \wedge \text{director}(\text{Cameron}) \wedge \\ \text{director.of.arg1}(e, \text{Cameron}) \wedge \text{director.of.arg2}(e, \text{Titanic})$$

Learning from Text

Cameron is the director of Titanic.

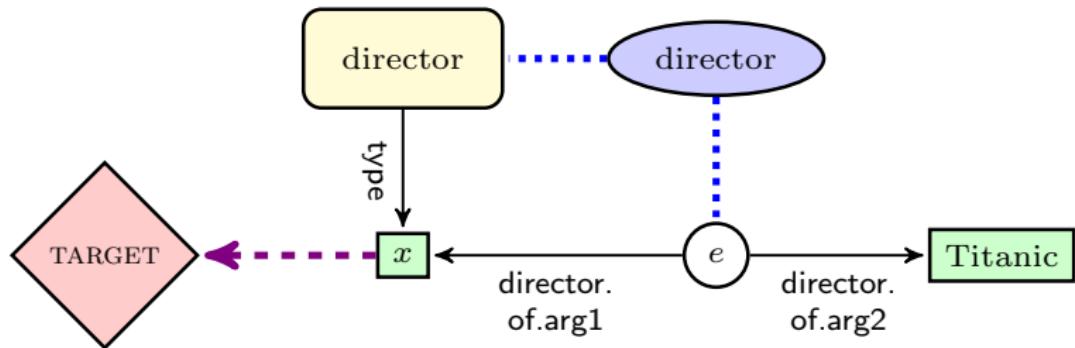


$\text{TARGET}(x) \wedge \text{director}(x) \wedge \text{director.of.arg1}(e, x) \wedge \text{director.of.arg2}(e, \text{Titanic})$

Select one of the entities as target and replace it with x (here *Cameron*).

Learning from Text

Who is the director of Titanic?

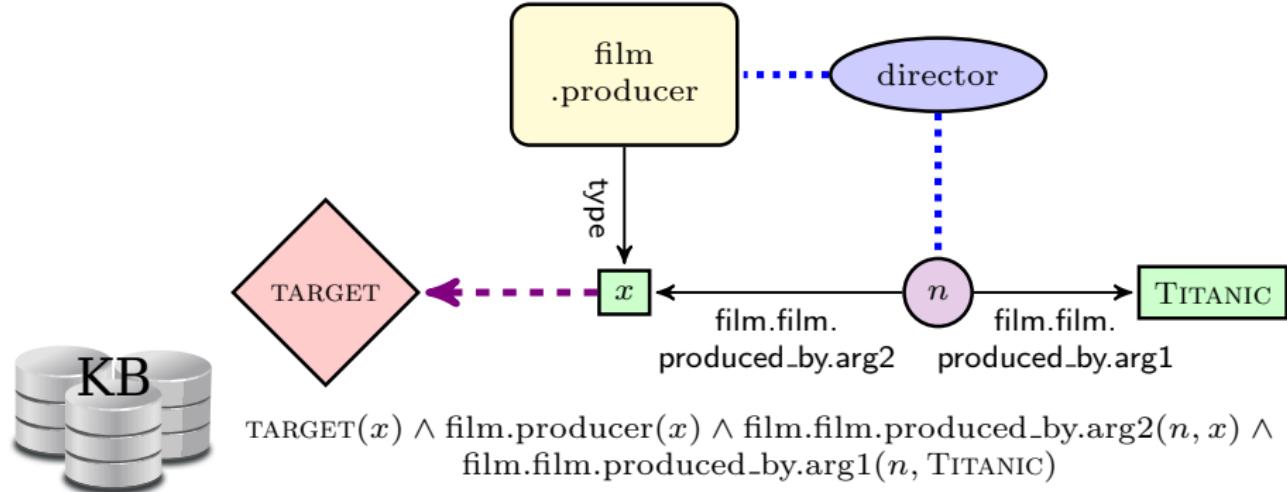


$\text{TARGET}(x) \wedge \text{director}(x) \wedge \text{director.of.arg1}(e, x) \wedge \text{director.of.arg2}(e, \text{Titanic})$

Build all knowledge base subgraphs with x uninstantiated.

Learning from Text

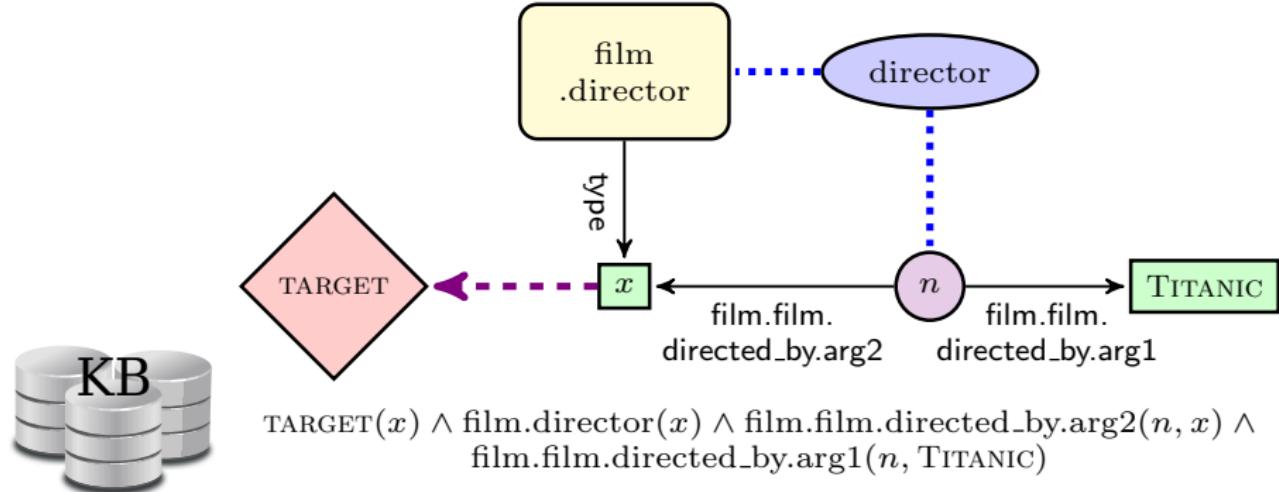
Who is the director of Titanic?



Build all knowledge base subgraphs with x uninstantiated.

Learning from Text

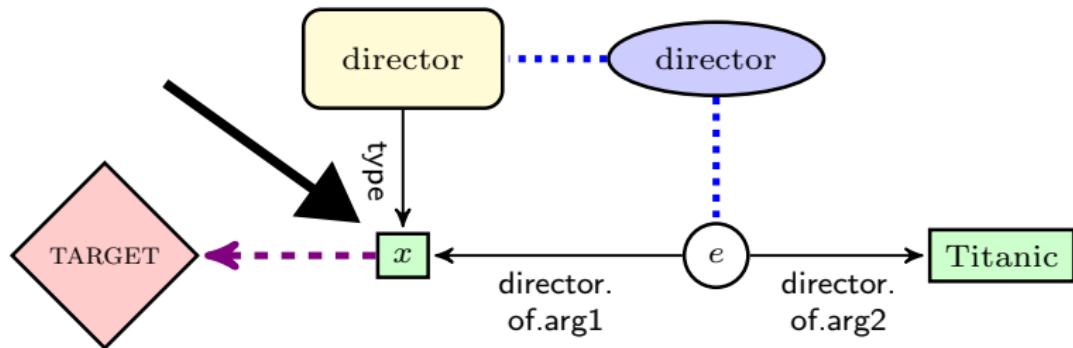
Who is the director of Titanic?



Build all knowledge base subgraphs with x uninstantiated.

Learning from Text

Cameron is the director of Titanic.



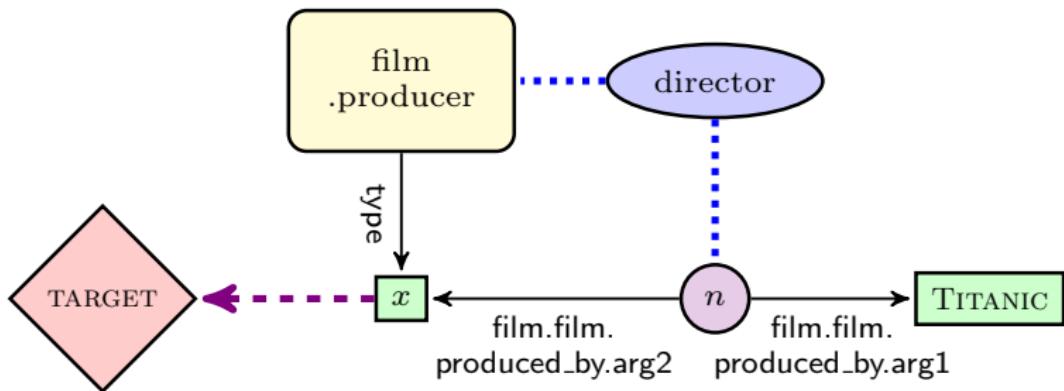
$\text{TARGET}(x) \wedge \text{director}(x) \wedge \text{director.of.arg1}(e, x) \wedge \text{director.of.arg2}(e, \text{Titanic})$

$$[[x]]_{NL} = \{\text{CAMERON}\}$$

Use denotations in NL and Freebase to select a surrogate gold graph.

Learning from Text

Who is the director of Titanic?



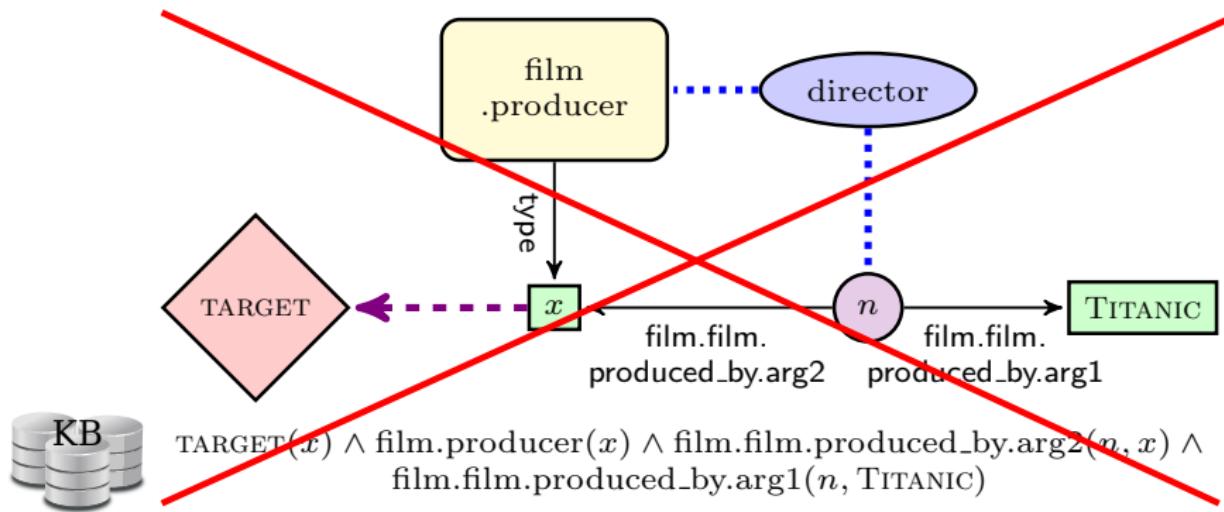
$\text{TARGET}(x) \wedge \text{film.producer}(x) \wedge \text{film.film.produced_by.arg2}(n, x) \wedge \text{film.film.produced_by.arg1}(n, \text{TITANIC})$

$$[[x]]_{KB} = \{\text{CAMERON}, \text{JON LANDAU}\}$$

Use denotations in NL and Freebase to select a surrogate gold graph.

Learning from Text

Who is the director of Titanic?

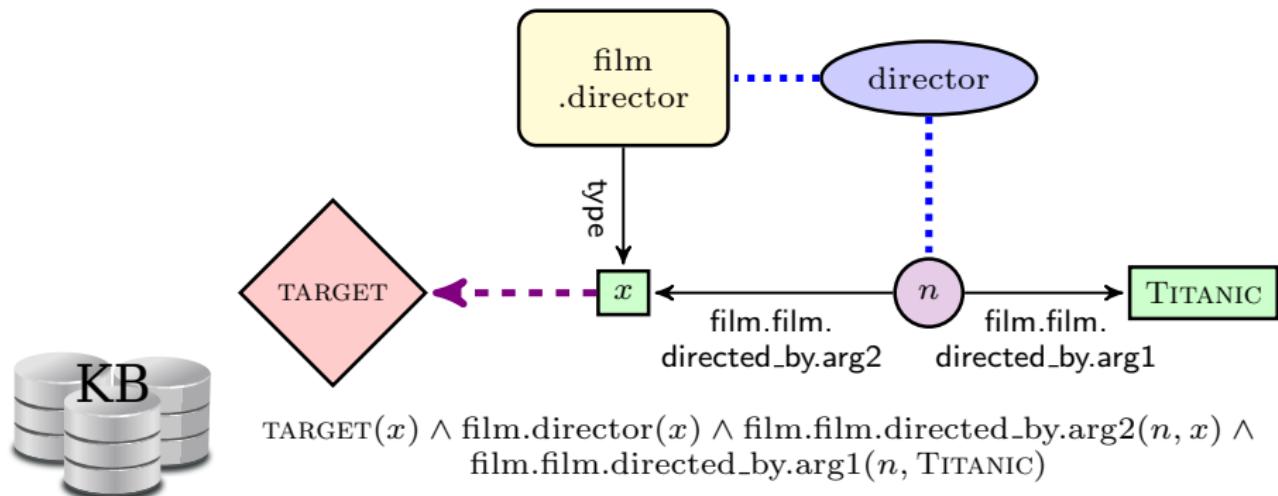


$$[\![x]\!]_{KB} = \{\text{CAMERON}, \text{JONLANDAU}\} \neq [\![x]\!]_{NL}$$

Use denotations in NL and Freebase to select a surrogate gold graph.

Learning from Text

Who is the director of Titanic?

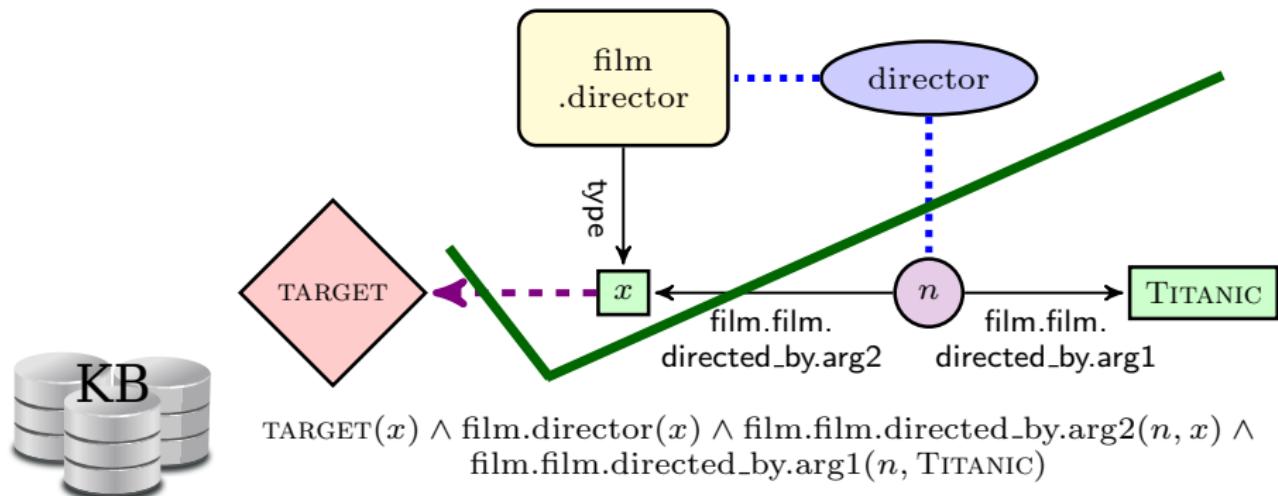


$$[[x]]_{KB} = \{\text{CAMERON}\}$$

Use denotations in NL and Freebase to select a surrogate gold graph.

Learning from Text

Who is the director of Titanic?



$$[x]_{KB} = \{\text{CAMERON}\} = [x]_{NL}$$

Use denotations in NL and Freebase to select a surrogate gold graph.

Learning from Text: Summary

Learning proceeds by creating **question-like graphs** by replacing named entities in logical forms mined from web text with a variable.

Then try to find the subgraph of the knowledge graph with the the most **similar denotation**.

Experiments

Target Domains: Business, Film, People

- ▶ Largest domains of Freebase
- ▶ 120m triples, 411 relations and 210 types.

Training data: 99K sentences from ClueWeb09

Experiments

Test data:

- ▶ Free917 [Cai and Yates, 2013]
 - ▶ Syntactically well-formed queries
 - ▶ 124 queries for our target domains
- ▶ WebQuestions [Berant et al., 2013]
 - ▶ Google search queries starting with *wh* question words.
 - ▶ 570 queries for our target domains

Results on Free917

System	Precision	Recall	F1
MWG	52.6	49.1	50.8
KCAZ13	72.6	66.1	69.2
This Work	81.9	76.6	79.2 +10.0

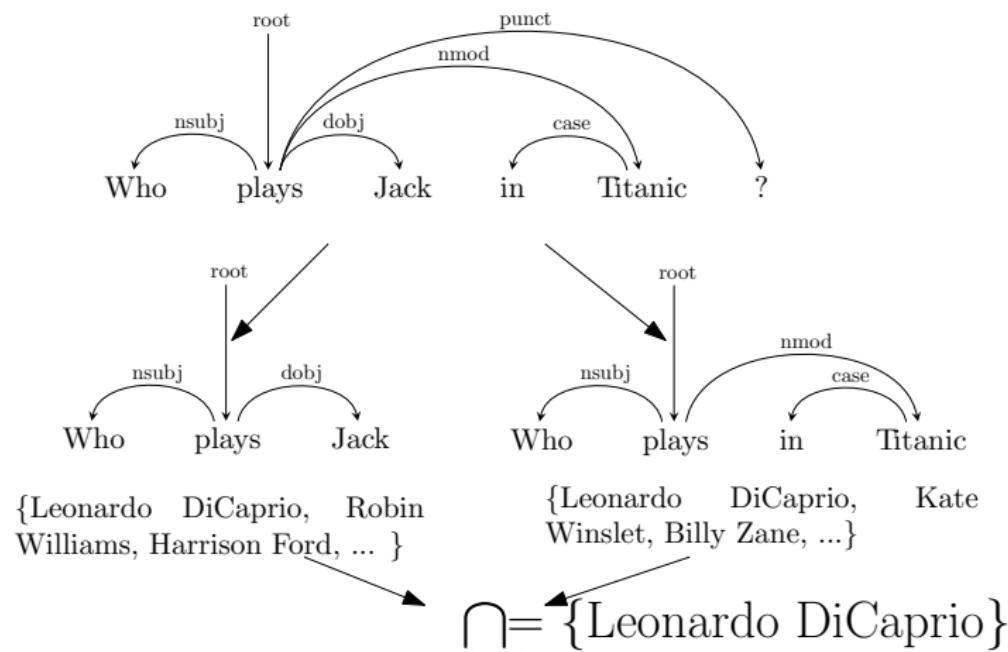
- ▶ **MWG:** Greedy Maximum Weighted Graph
- ▶ **KCAZ13:** Kwiatkowski et al. 2013
 - ▶ Manually annotated QA pairs (612 pairs)
 - ▶ Wiktionary

Results on WebQuestions

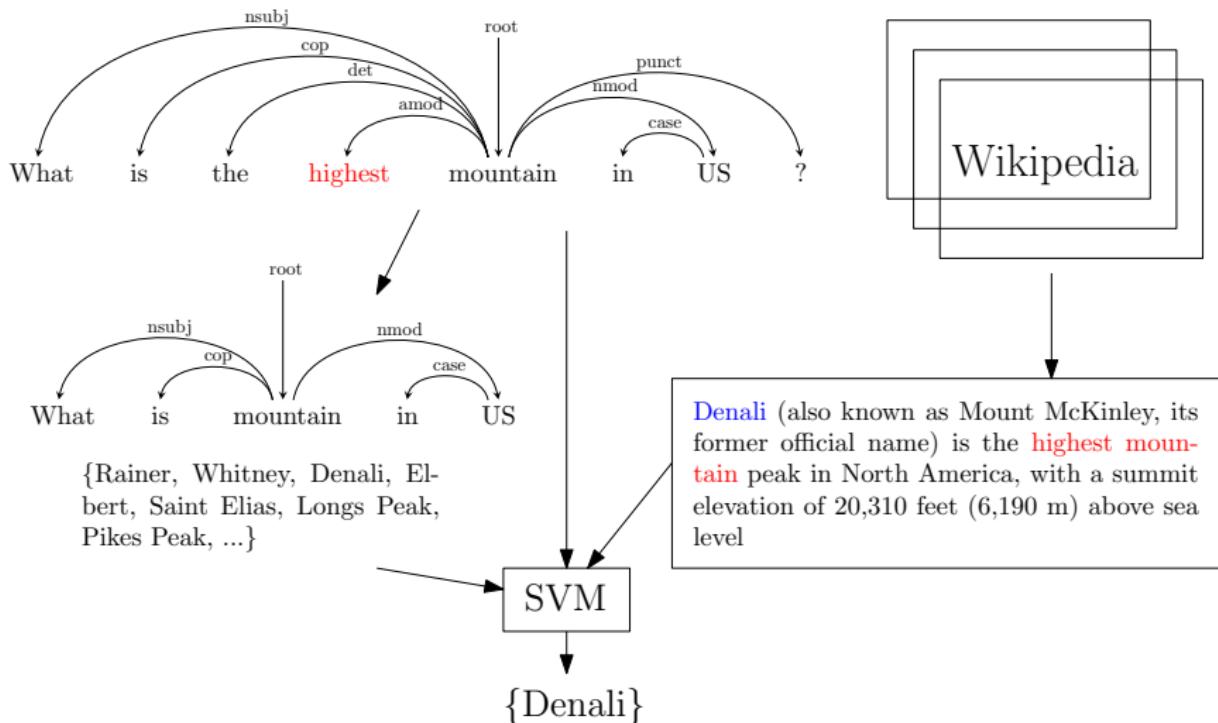
System	Precision	Recall	F1	<i>AvgF1</i> (BL14)
MWG	39.4	34.0	36.5	41.6
BL14	—	—	—	37.5
This Work	41.9	37.0	39.3	47.7 +10.2

- ▶ **MWG:** Greedy Maximum Weighted Graph
- ▶ **BL14:** Berant and Liang, 2014
 - ▶ ClueWeb09
 - ▶ Manually annotated QA pairs (1115 pairs)
 - ▶ Paraphrases

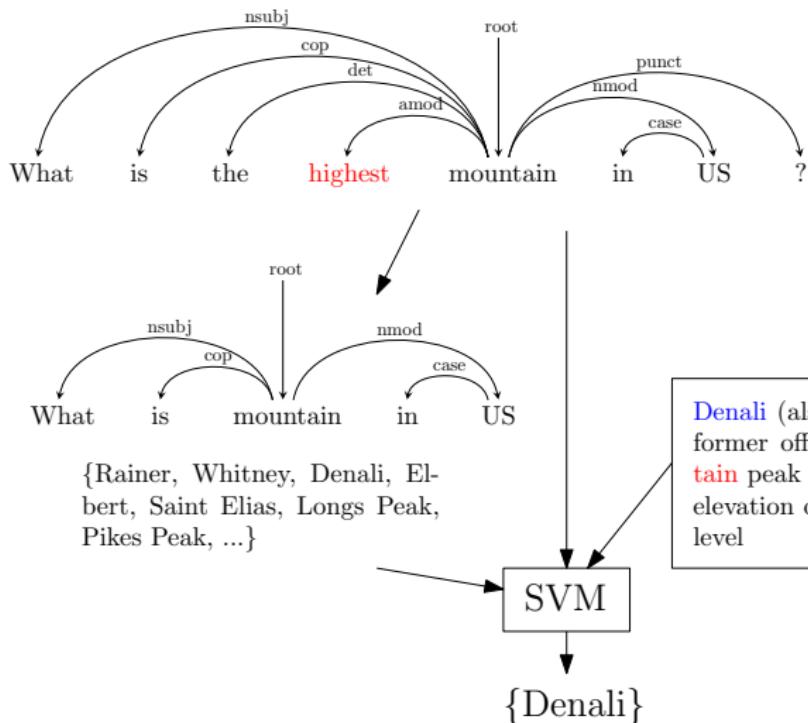
Semantic Parsing with Natural Logic and IE



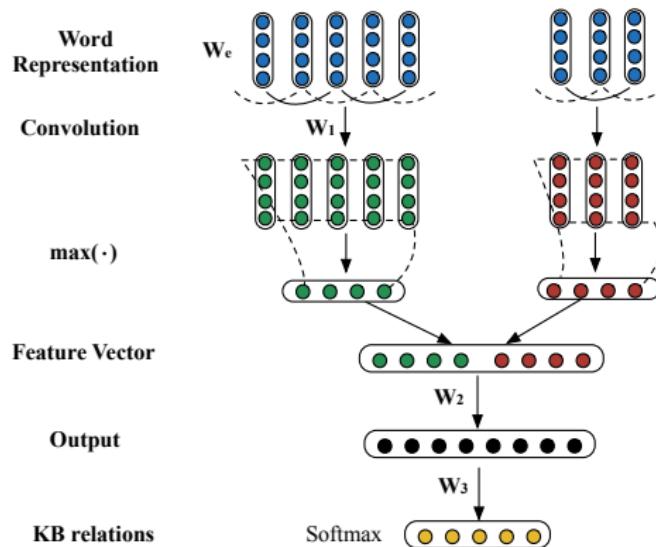
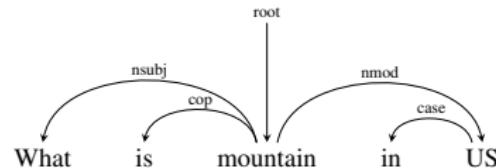
Natural-Logic styled Semantic Parsing



Natural-Logic styled Semantic Parsing



Natural-Logic styled Semantic Parsing



Summary

- ▶ Lambda Calculus for converting Dependencies to Logical Forms
- ▶ Semantic parsing as Graph Transduction
- ▶ Semantic Parsing with Natural Logic and IE

Summary

- ▶ Lambda Calculus for converting Dependencies to Logical Forms
- ▶ Semantic parsing as Graph Transduction
- ▶ Semantic Parsing with Natural Logic and IE

Demo at

<https://sivareddy.in/deplambda.html>

Syntax in humans?

*Studies on Peruvian Indian bilinguals
indicate Quechua word order influences
the local varieties of Spanish
[Odlin, 1989]*



Syntax in humans?

Studies on Peruvian Indian bilinguals indicate Quechua word order influences the local varieties of Spanish [Odlin, 1989]



Arabs show strong preference for SVO in Dutch, whereas Turks for SOV [Jansen et al., 1981; Appel, 1984]

Learning Model

Structured Perceptron: Ranks a pair of grounded and ungrounded graph

$$(\hat{g}, \hat{u}) = \arg \max_{g,u} \Phi(g, u, q, KB) \cdot \theta$$

Learning Model

Structured Perceptron: Ranks a pair of grounded and ungrounded graph

$$(\hat{g}, \hat{u}) = \arg \max_{g,u} \Phi(g, u, q, KB) \cdot \theta$$

Features: Φ is defined over sentence, grounded and ungrounded graph

Learning Model

Structured Perceptron: Ranks a pair of grounded and ungrounded graph

$$(\hat{g}, \hat{u}) = \arg \max_{g, u} \Phi(g, u, q, KB) \cdot \theta$$

Features: Φ is defined over sentence, grounded and ungrounded graph

Training: Use a surrogate graph (dynamic oracle) to update weights

$$\theta \leftarrow \theta + \Phi(g^+, u^+, q, KB) - \Phi(\hat{g}, \hat{u}, q, KB)$$

Learning Model

Oracle Graphs: Search for all the grounded graphs reachable via the ungrounded graph.

Pick all the graphs with minimal F_1 -loss against the gold answer.

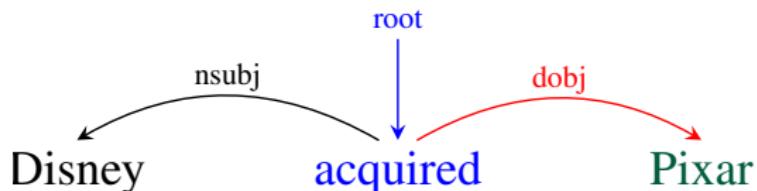
Surrogate Gold Graph:

$$(u^+, g^+) = \arg \max_{(u, g) \in O_{KB, A}(q)} \theta^t \cdot \Phi(u, g, q, KB),$$

Beam Search: Limit the predictions to 100 graph pairs, and choose the best prediction for update.

Dependencies to Logical Forms

Single Type System

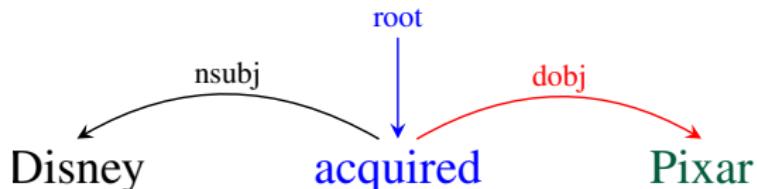


All constituents are of the same lambda expression type

$\text{TYPE}[\text{acquired}] = \text{TYPE}[\text{Pixar}] = \text{TYPE}[(\text{dobj} \text{ acquired } \text{Pixar})]$

Dependencies to Logical Forms

Single Type System

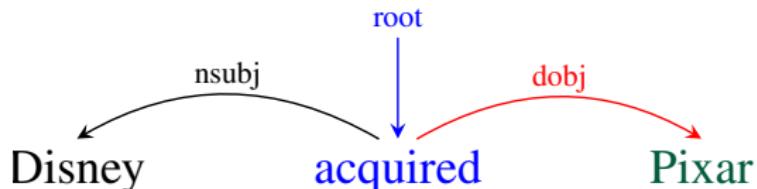


All **words** have a *lambda expression* of type η

- ▶ $\text{TYPE}[\text{acquired}] = \eta$
- ▶ $\text{TYPE}[\text{Pixar}] = \eta$

Dependencies to Logical Forms

Single Type System

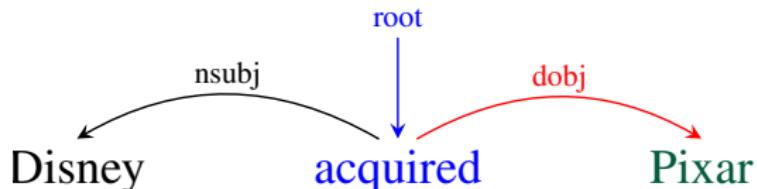


All **constituents** have a *lambda expression* of type η

- ▶ $\text{TYPE}[\text{acquired}] = \eta$
- ▶ $\text{TYPE}[\text{Pixar}] = \eta$
- ▶ $\text{TYPE}[(\text{dobj acquired Pixar})] = \eta$

Dependencies to Logical Forms

Single Type System

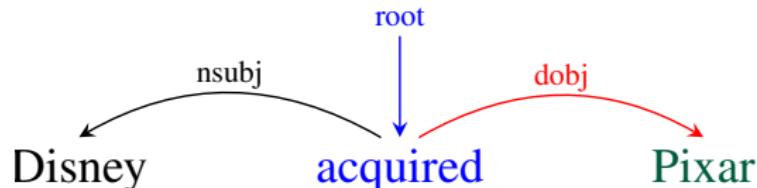


All **constituents** have a *lambda expression* of type η

- ▶ $\text{TYPE}[\text{acquired}] = \eta$
 - ▶ $\text{TYPE}[\text{Pixar}] = \eta$
 - ▶ $\text{TYPE}[(\text{dobj acquired Pixar})] = \eta$
- $\implies \text{TYPE}[\text{dobj}] = \eta \rightarrow \eta \rightarrow \eta$

Dependencies to Logical Forms

Lambda Calculus for Single Type System



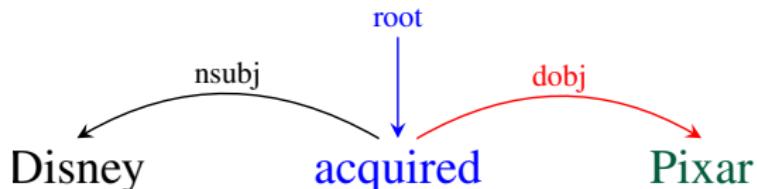
Lambda Expression for words

$$\text{acquired} \Rightarrow \lambda x_e. \text{acquired}(x_e)$$

$$\text{Pixar} \Rightarrow \lambda x_a. \text{Pixar}(x_a)$$

Dependencies to Logical Forms

Lambda Calculus for Single Type System



Lambda Expression for words

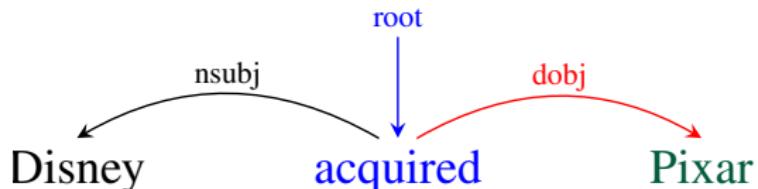
$$\text{acquired} \Rightarrow \lambda x_e. \text{acquired}(x_e) \Rightarrow \text{TYPE} = \text{Event} \rightarrow \text{Bool}$$

$$\text{Pixar} \Rightarrow \lambda x_a. \text{Pixar}(x_a) \Rightarrow \text{TYPE} = \text{Ind} \rightarrow \text{Bool}$$

Here $\text{TYPE}[\text{acquired}] \neq \text{TYPE}[\text{Pixar}]$ \times

Dependencies to Logical Forms

Lambda Calculus for Single Type System



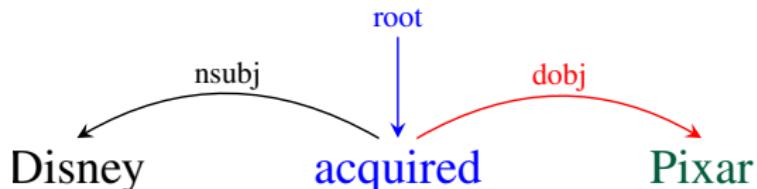
Lambda Expression for words

$$\text{acquired} \Rightarrow \lambda \mathbf{x_a} x_e. \text{acquired}(x_e)$$

$$\text{Pixar} \Rightarrow \lambda x_a \mathbf{x_e}. \text{Pixar}(x_a)$$

Dependencies to Logical Forms

Lambda Calculus for Single Type System



Lambda Expression for words

$\text{acquired} \Rightarrow \lambda \mathbf{x_a} x_e. \text{acquired}(x_e)$ $\Rightarrow \mathbf{TYPE} = \mathbf{Ind} \times \mathbf{Event} \rightarrow \mathbf{Bool}$

$\text{Pixar} \Rightarrow \lambda x_a \mathbf{x_e}. \text{Pixar}(x_a)$ $\Rightarrow \mathbf{TYPE} = \mathbf{Ind} \times \mathbf{Event} \rightarrow \mathbf{Bool}$

Here $\eta = \mathbf{TYPE}[\text{acquired}] = \mathbf{TYPE}[\text{Pixar}] \checkmark$

Lambda calculus for “*Single Type*” System

More dependency labels

- ▶ (appos Disney the_company)

$$appos = \lambda f g x. f(x) \wedge g(x)$$

This function unifies two nodes

Lambda calculus for “*Single Type*” System

More dependency labels

- ▶ (appos Disney the_company)

$$appos = \lambda f g x. f(x) \wedge g(x)$$

This function unifies two nodes

- ▶ (partmod a_company acquired_by_Disney)

$$partmod = \lambda f g x. \exists z. f(x) \wedge g(z) \wedge \text{arg}_1(z_e, x_a)$$

This function reverses the dependency arc direction, but still returns the head

Lambda calculus for “*Single Type*” System

More dependency labels

- ▶ (conj Disney_and Pixar)

$$conj = \lambda fgz. \exists xy. f(x) \wedge g(y) \wedge \mathbf{coord}(z, x, y)$$

This function creates a struct with two variables

Lambda calculus for “*Single Type*” System

More dependency labels

- ▶ (conj Disney_and Pixar)

$$\text{conj} = \lambda f g z. \exists x y. f(x) \wedge g(y) \wedge \mathbf{coord}(z, x, y)$$

This function creates a struct with two variables

- ▶ (rcmod Disney which_acquired_Pixar)

(rcmod Disney
 (wh-dobj (**BIND** f (nsubj (dobj acquired f) Pixar))
 which))

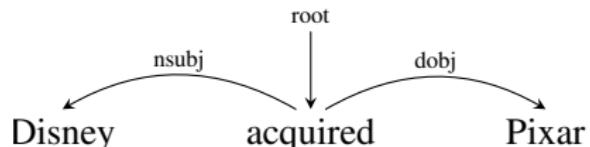
CCG to Logical Forms

Steedman, 2000, 2012; Bos et al., 2004; Lewis & Steedman, 2013; Reddy et al., 2014

$$\frac{\begin{array}{ccc} \text{Disney} & \text{acquired} & \text{Pixar} \\ \hline NP & S \setminus NP/NP & NP \end{array}}{\frac{\text{Disney } \lambda y \lambda x \lambda e. \text{ acquired}(e) \wedge \arg_1(e, x) \wedge \arg_2(e, y)}{S \setminus NP}} \rightarrow$$
$$\frac{\lambda x \lambda e. \text{ acquired}(e) \wedge \arg_1(e, x) \wedge \arg_2(e, \text{ Pixar})}{S} \leftarrow$$
$$\lambda e. \text{ acquired}(e) \wedge \arg_1(e, \text{ Disney}) \wedge \arg_2(e, \text{ Pixar})$$

Dependencies to Logical Forms

Challenges



The obvious idea

if *proper noun* **then**

 assign $\lambda x.\text{word}(x)$

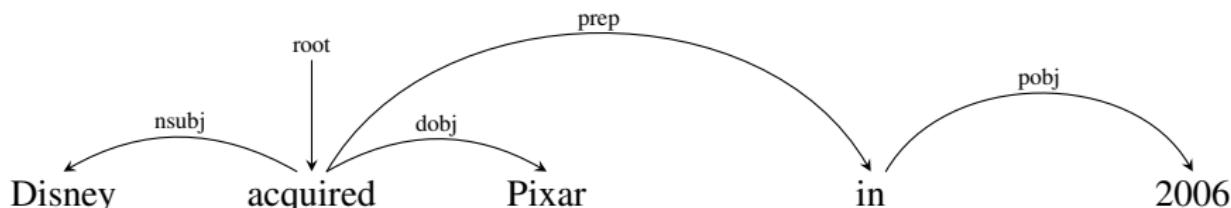
else if *verb with subject and object* **then**

 assign $\lambda fge.\exists xy.f(x) \wedge g(y) \wedge \text{word}(e) \wedge \text{arg}_1(e, x) \wedge \text{arg}_2(e, y)$

end if

Dependencies to Logical Forms

Challenges



The obvious idea

if proper noun **then**

 assign $\lambda x.\text{word}(x)$

else if verb with subject and object **then**

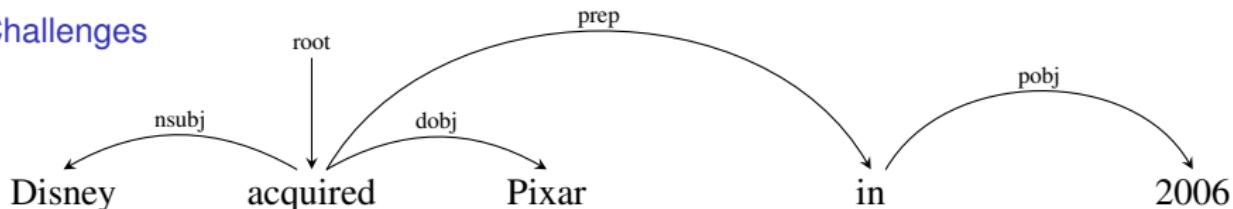
 assign $\lambda fge.\exists xy.f(x) \wedge g(y) \wedge \text{word}(e) \wedge \text{arg}_1(e, x) \wedge \text{arg}_2(e, y)$

end if

But, what about?

Dependencies to Logical Forms

Challenges



Problems

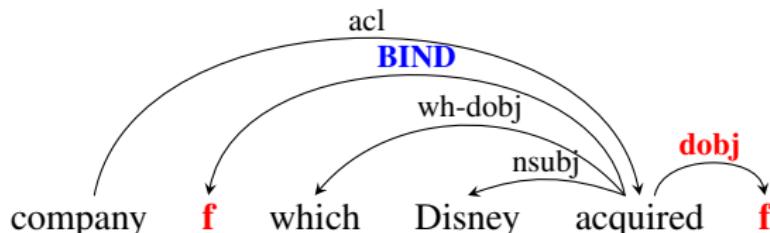
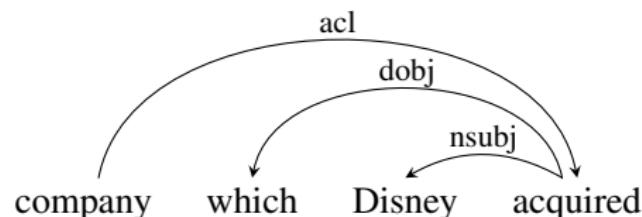
1. Rules \propto dependency label permutations
2. Complex lexical semantics
3. Highly sensitive to parse errors
4. Prone to type collisions

Relative Clause in CCG

company	which	Disney	acquired
N $\lambda x. company(x)$	$(N \setminus N) / (S_{dcl} / NP_x)$ $\lambda p \lambda q \lambda x. q(x) \wedge \exists e [p(x, e)]$	N $\lambda x \lambda y \lambda e. acquire(e) \wedge A0(y, e) \wedge A1(x, e)$	$(S_{dcl} \setminus NP) / NP$ $\lambda x \lambda e. acquire(e) \wedge A0(disney, e) \wedge A1(x, e)$
		NP <i>disney</i>	
		$S_X / (S_X \setminus NP)$ $\lambda p \lambda e. p(disney, e)$	
			S_{dcl} / NP $\lambda x \lambda e. acquire(e) \wedge A0(disney, e) \wedge A1(x, e)$
		N\N $\lambda p \lambda x. p(x) \wedge \exists e [acquire(e) \wedge A0(disney, e) \wedge A1(x, e)]$	
		N $\lambda x. company(x) \wedge \exists e [acquire(e) \wedge A0(disney, e) \wedge A1(x, e)]$	

Relative Clause in DepLambda

following Carpenter (1998)



Comparison with CCG

Handling of control verbs is painful.

Sentence:

John persuaded Jim to acquire Pixar.

Binarized Tree:

(nsubj (xcomp (dobj persuaded Jim) to_acquire_Pixar) John)

Comparison with CCG

Handling of control verbs is painful.

Sentence:

John persuaded Jim to acquire Pixar.

Binarized Tree:

(nsubj (xcomp (dobj persuaded Jim) to_acquire_Pixar) John)

Elegant handling in CCG

persuaded: $((S[dcl] \setminus NP) / (S[to] \setminus NP_x)) / NP_x$

Conjunctions

Sentence:

Eminem signed to Interscope and discovered 50 Cent.

Binarized tree:

(nsubj (conj-vp (cc s_to_I and) d_50) Eminem)

Conjunctions

Sentence:

Eminem signed to Interscope and discovered 50 Cent.

Binarized tree:

(nsubj (conj-vp (cc s_to_I and) d_50) Eminem)

Substitution:

conj-vp $\Rightarrow \lambda f g x. \exists y z. f(y) \wedge g(z) \wedge \text{coord}(x, y, z)$

Logical Expression:

$$\begin{aligned} \lambda w. \exists x y z. & \text{Eminem}(x_a) \wedge \text{coord}(w, y, z) \\ & \wedge \text{arg}_1(w_e, x_a) \wedge \text{s_to_I}(y) \wedge \text{d_50}(z) \end{aligned}$$

Conjunctions

Sentence:

Eminem signed to Interscope and discovered 50 Cent.

Binarized tree:

(nsubj (conj-vp (cc s_to_I and) d_50) Eminem)

Substitution:

conj-vp $\Rightarrow \lambda f g x. \exists y z. f(y) \wedge g(z) \wedge \text{coord}(x, y, z)$

Logical Expression:

$$\begin{aligned} \lambda w. \exists x y z. & \text{Eminem}(x_a) \wedge \text{coord}(w, y, z) \\ & \wedge \text{arg}_1(w_e, x_a) \wedge \text{s_to_I}(y) \wedge \text{d_50}(z) \end{aligned}$$

Post processing:

$$\begin{aligned} \lambda e. \exists x y z. & \text{Eminem}(x_a) \wedge \text{arg}_1(y_e, x_a) \\ & \wedge \text{arg}_1(z_e, x_a) \wedge \text{s_to_I}(y) \wedge \text{d_50}(z) \end{aligned}$$

Relative Clause

following Moortgat (1988); Pereira (1990); Carpenter (1998)

Sentence:

Apple which Jobs founded

Binarized tree:

(rcmod Apple
 (wh-dobj (**BIND** f (nsubj (dobj founded f) Jobs))
 which))

Relative Clause

following Moortgat (1988); Pereira (1990); Carpenter (1998)

Sentence:

Apple which Jobs founded

Binarized tree:

(rcmod Apple
 (wh-dobj (**BIND** f (nsubj (dobj founded f) Jobs))
 which))

Substitution:

$$\begin{aligned} \text{wh-dobj} &\Rightarrow \lambda f g z. f(z) \\ \text{rcmod} &\Rightarrow \lambda f g z. f(z) \wedge g(z) \end{aligned}$$

Logical Expression:

$$\begin{aligned} \lambda u. \exists xy. \text{founded}(x_e) \wedge \text{Jobs}(y_a) \\ \wedge \text{arg}_1(x_e, y_a) \wedge \text{arg}_2(x_e, u_a) \wedge \text{Apple}(u_a) \end{aligned}$$

Expressivity

How isomorphic are the representations compared to Knowledge Graph?

Average Oracle F_1 Table.

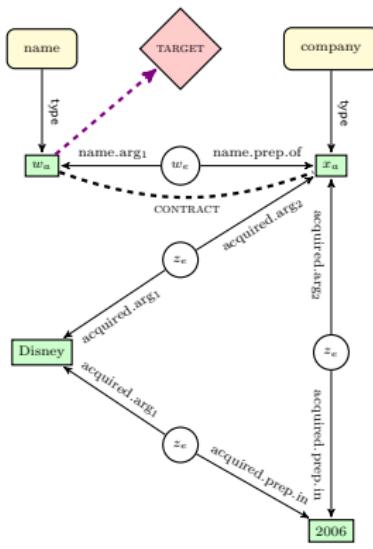
Search Space

How many ways to reach an answer?

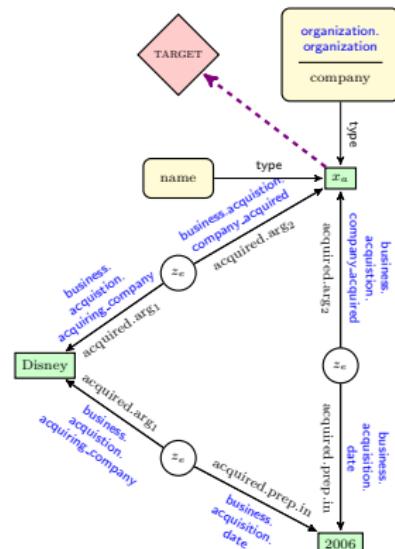
Average Oracle F_1 Table.

Graph Transformation: CONTRACT operation

What is the **name** of the company which Disney acquired in 2006?



Ungrounded graph



Grounded graph

Graph Mismatch: EXPAND operation

What to do Washington DC December?

Before EXPAND

- ▶ $\lambda z. \exists xyw. \text{TARGET}(x_a) \wedge \text{do}(z_e) \wedge \arg_1(z_e, x_a) \wedge \text{Washington_DC}(y_a) \wedge \text{December}(w_a)$

After EXPAND

- ▶ $\lambda z. \exists xyw. \text{TARGET}(x_a) \wedge \text{do}(z_e) \wedge \arg_1(z_e, x_a) \wedge \text{Washington_DC}(y_a) \wedge \text{dep}(z_e, y_a) \wedge \text{December}(w_a) \wedge \text{dep}(z_e, w_a)$