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

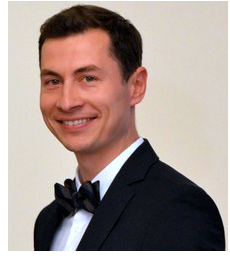
Mark Steedman

Oscar Täckström

Dipanjan Das

Tom Kwiatkowski

Michael Collins



Slav Petrov

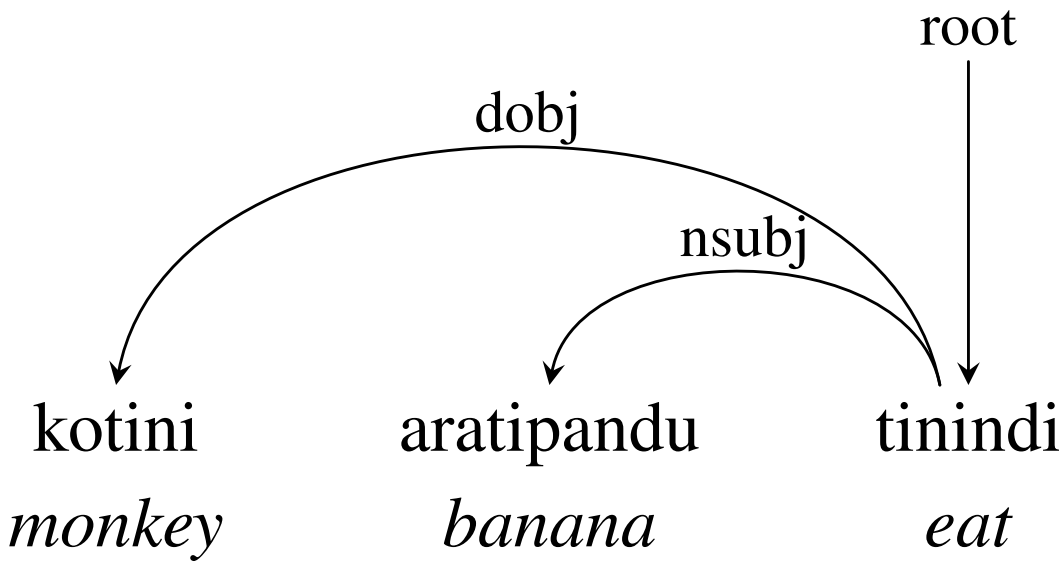
Dependency Trees help Semantics

kotini	aratipandu	tinindi
monkey	*banana*	*eat*

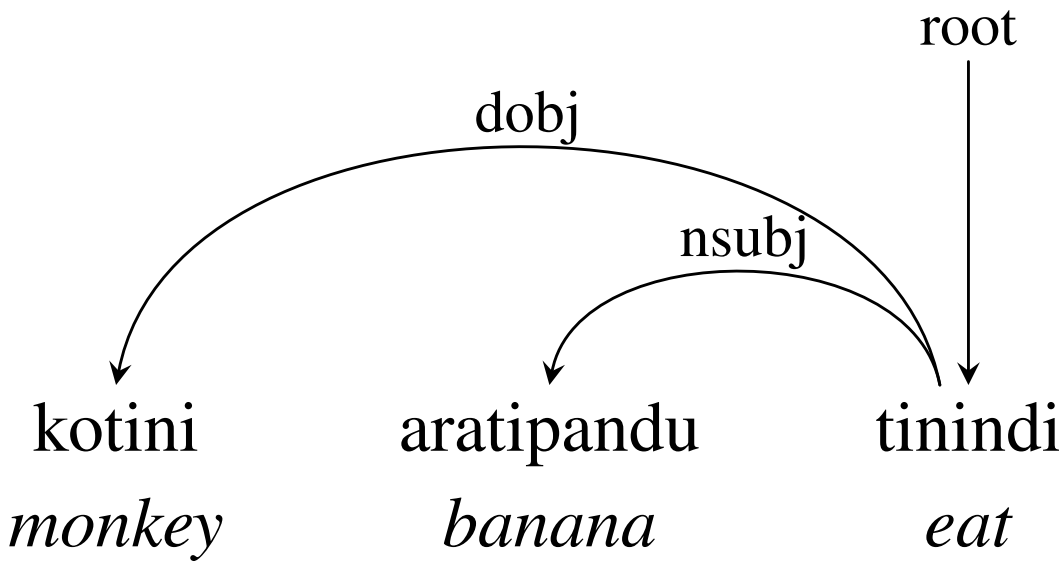
Dependency Trees help Semantics

kotini	aratipandu	tinindi
monkey	*banana*	*eat*

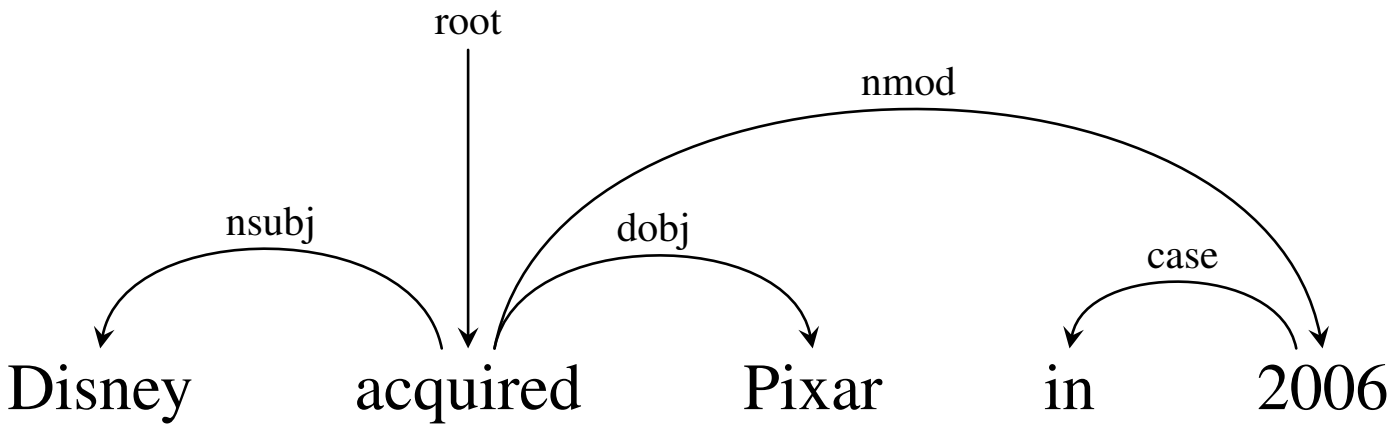
Dependency Trees help Semantics



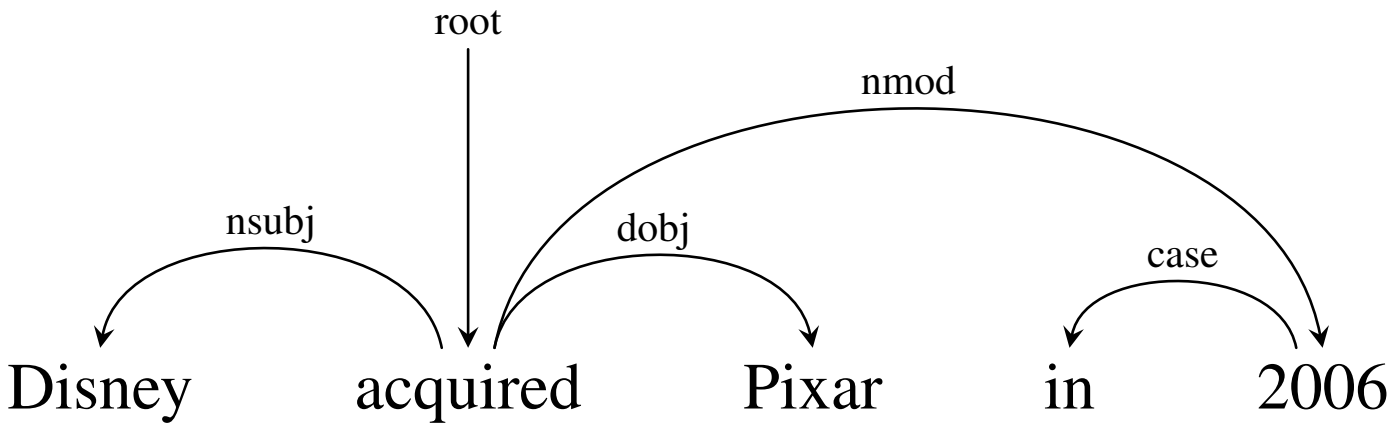
Dependency Trees help Semantics



Universal Dependencies

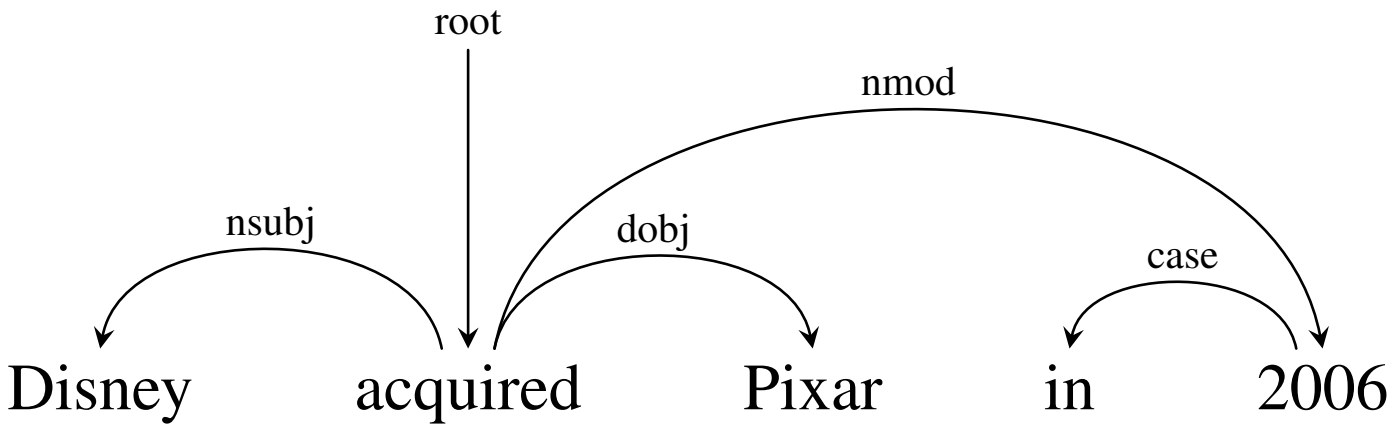


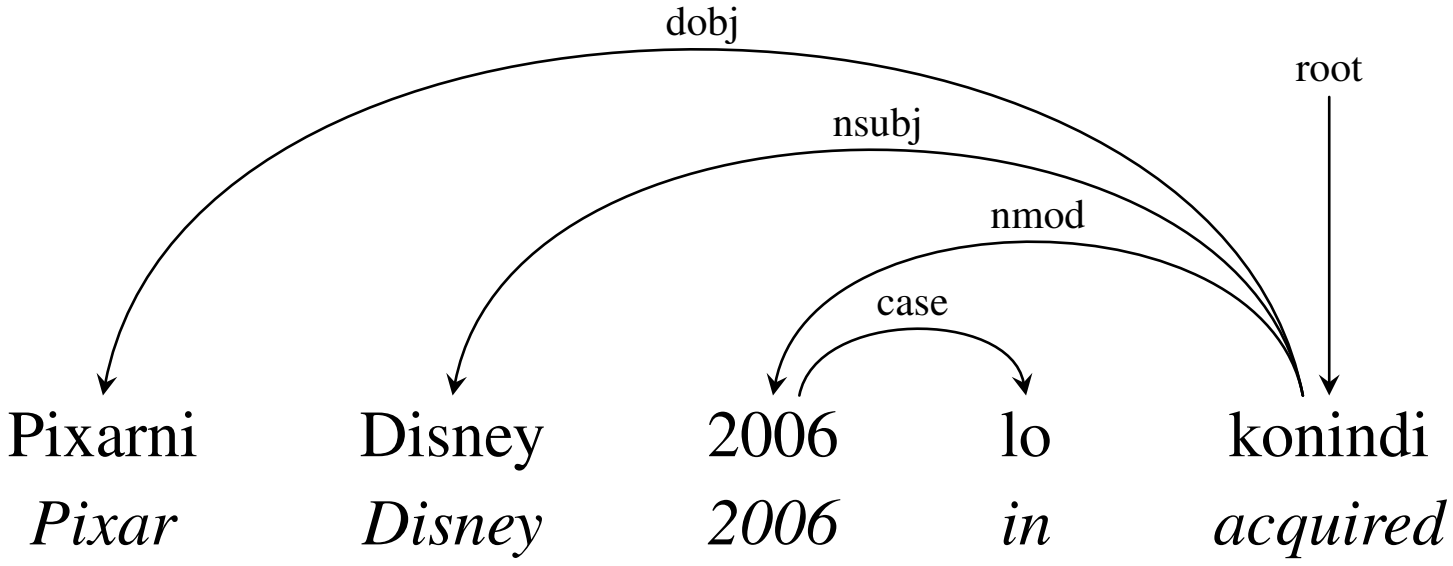
Universal Dependencies



Pixarni	Disney	2006	lo	konindi
Pixar	*Disney*	*2006*	*in*	*acquired*

Universal Dependencies





Universal Dependencies

Homogeneous syntactic representation across languages

Treebanks in 40 languages

40 dependency labels

Dependency Tree to Semantics

Dependencies lack a formal theory of semantics

This Talk

A Compositional Typed Semantic Interface for Dependencies

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

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

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 subtrees

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 subtrees

Rules are the dependency labels

Existing Syntax-Semantics interfaces

CCG [Steedman, 2000; Bos et al., 2004]

HPSG [Copestake et al., 2001]

LFG [Dalrymple et al., 1995]

TAG [Joshi et al., 1995]

CCG

Disney	acquired	Pixar
NP	$S\backslash NP/NP$	NP

CCG

Disney	acquired	Pixar
NP	$S\backslash NP/NP$	NP
Disney	$\lambda y \lambda x \lambda e.\ \text{acquired}(e) \wedge \text{arg}_1(e, x) \wedge \text{arg}_2(e, y)$	Pixar

CCG

Lambda Calculus

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

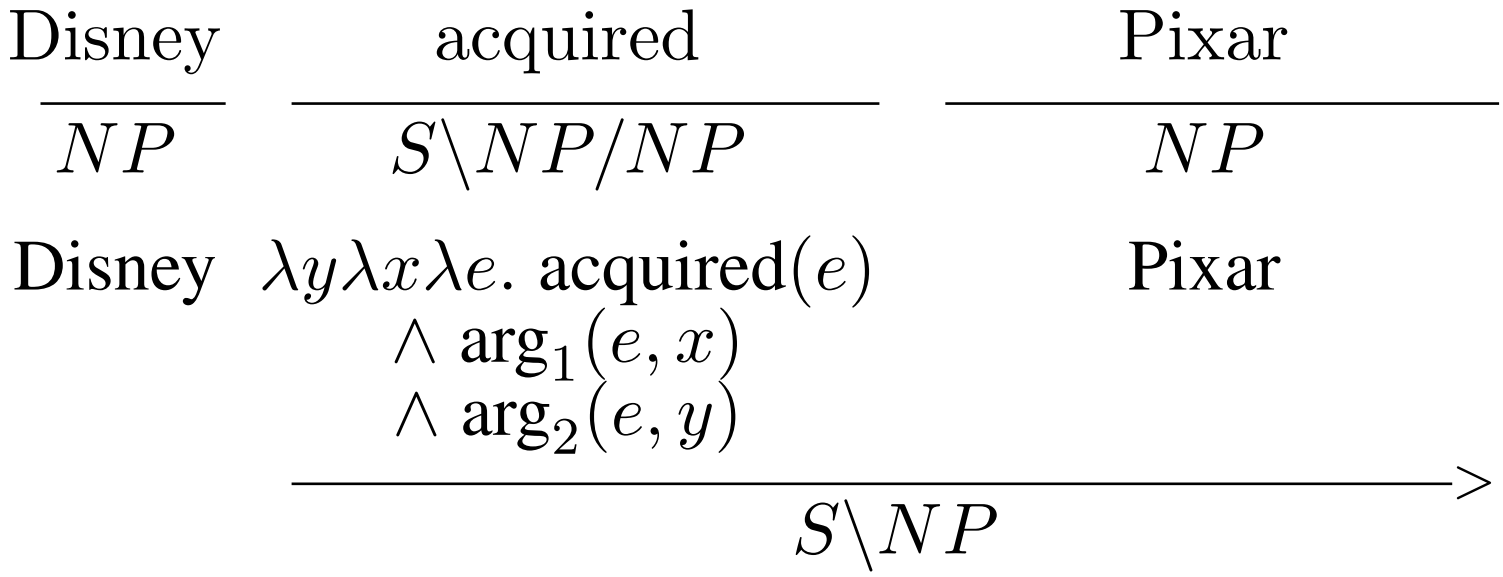
$$\begin{aligned} sum(2,3) &= (\lambda x \lambda y.(+\ x\ \ y))(2)(3) \\ &= (\lambda y.(+\ 2\ \ y))(3) \\ &= (+\ 2\ \ 3) \\ &= 5 \end{aligned}$$

$$\begin{aligned} \textbf{TYPE}[sum] &= int \rightarrow int \rightarrow int \\ sum(4, sum(2,3)) &= 9 \end{aligned}$$

CCG

Disney	acquired	Pixar
NP	$S\backslash NP/NP$	NP
Disney	$\lambda y \lambda x \lambda e.\ \text{acquired}(e) \wedge \text{arg}_1(e, x) \wedge \text{arg}_2(e, y)$	Pixar

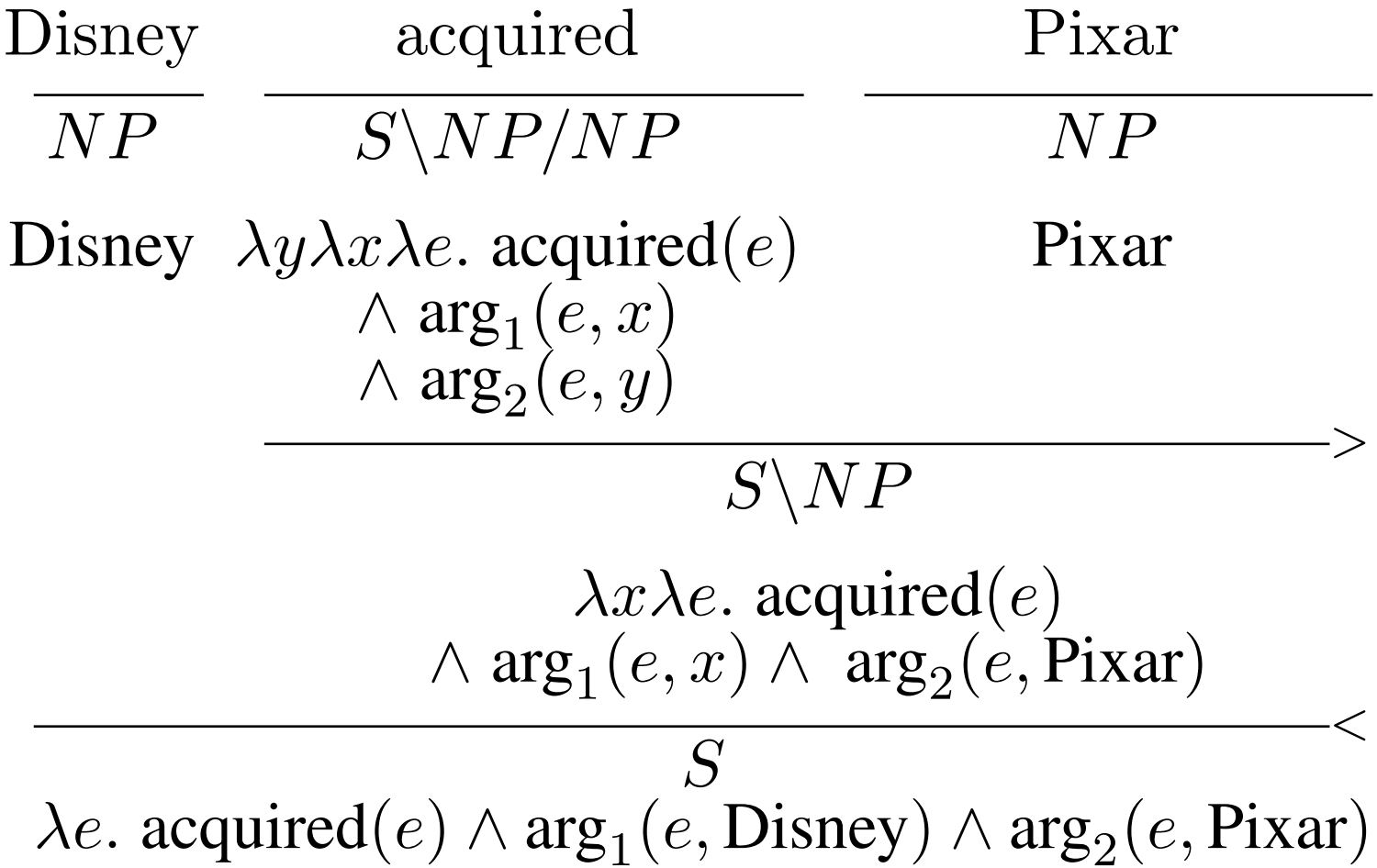
CCG


Disney
acquired
Pixar
NP
S\NP/NP
NP
Disney
λyλxλe. acquired(e)
∧ arg1(e, x)
∧ arg2(e, y)
Pixar
>
S\NP

CCG

Disney	acquired	Pixar
NP	$S\backslash NP/NP$	NP
Disney	$\lambda y \lambda x \lambda e.\ \text{acquired}(e) \wedge \text{arg}_1(e, x) \wedge \text{arg}_2(e, y)$	Pixar
	$S\backslash NP$ (>)	
	$\lambda x \lambda e.\ \text{acquired}(e) \wedge \text{arg}_1(e, x) \wedge \text{arg}_2(e, \text{Pixar})$	

CCG


Disney
acquired
Pixar
NP
S\NP/NP
NP
Disney
λyλxλe. acquired(e)
∧ arg1(e, x)
∧ arg2(e, y)
Pixar
>
S\NP
λxλe. acquired(e)
∧ arg1(e, x) ∧ arg2(e, Pixar)
<
S
λe. acquired(e) ∧ arg1(e, Disney) ∧ arg2(e, Pixar)

CCG

$$\frac{\text{Disney}}{NP} \qquad \frac{\text{acquired}}{S\backslash NP/NP} \qquad \frac{\text{Pixar}}{NP}$$

$$\text{Disney} \qquad \lambda y \lambda x \lambda e.\ \text{acquired}(e) \wedge \text{arg}_1(e, x) \wedge \text{arg}_2(e, y) \qquad \text{Pixar}$$

$$\frac{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}{S\backslash NP}\ >$$

$$\lambda x \lambda e.\ \text{acquired}(e) \wedge \text{arg}_1(e, x) \wedge\ \text{arg}_2(e, \text{Pixar})$$

$$\frac{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}{S}\ <$$

$$\lambda e.\ \text{acquired}(e) \wedge \text{arg}_1(e, \text{Disney}) \wedge \text{arg}_2(e, \text{Pixar})$$

Typing and Combinator Rules allow
Synchronous Syntax-Semantics interface

Why from dependencies?

Easy to annotate

Treebanks in many languages

Very accurate parsers

[Andor et al., 2016, Dyer et al., 2015, Chen & Manning, 2014]

Friendly to read

Outline

DepLambda: Dependencies to Logical Forms

Freebase Semantic Parsing using DepLambda

Results on English, German, and Spanish

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

DepLambda:
Dependencies to Logical Forms

Why Logical Forms?

Semantic Parsing [Zelle & Mooney, 1996; Zettlemoyer & Collins, 2005]

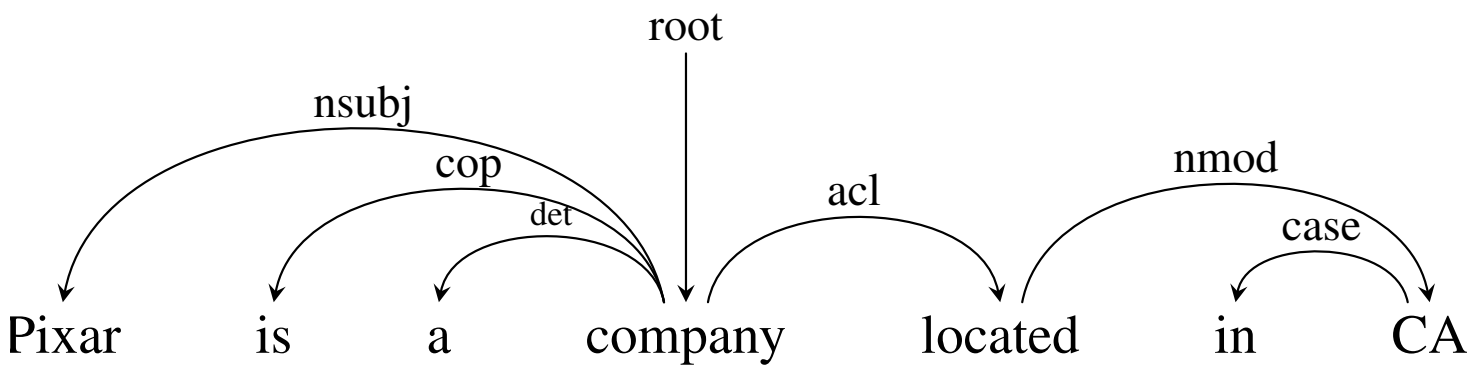
Simplification [Narayan & Gardent, 2014]

Paraphrasing [Pavlick et al., 2015]

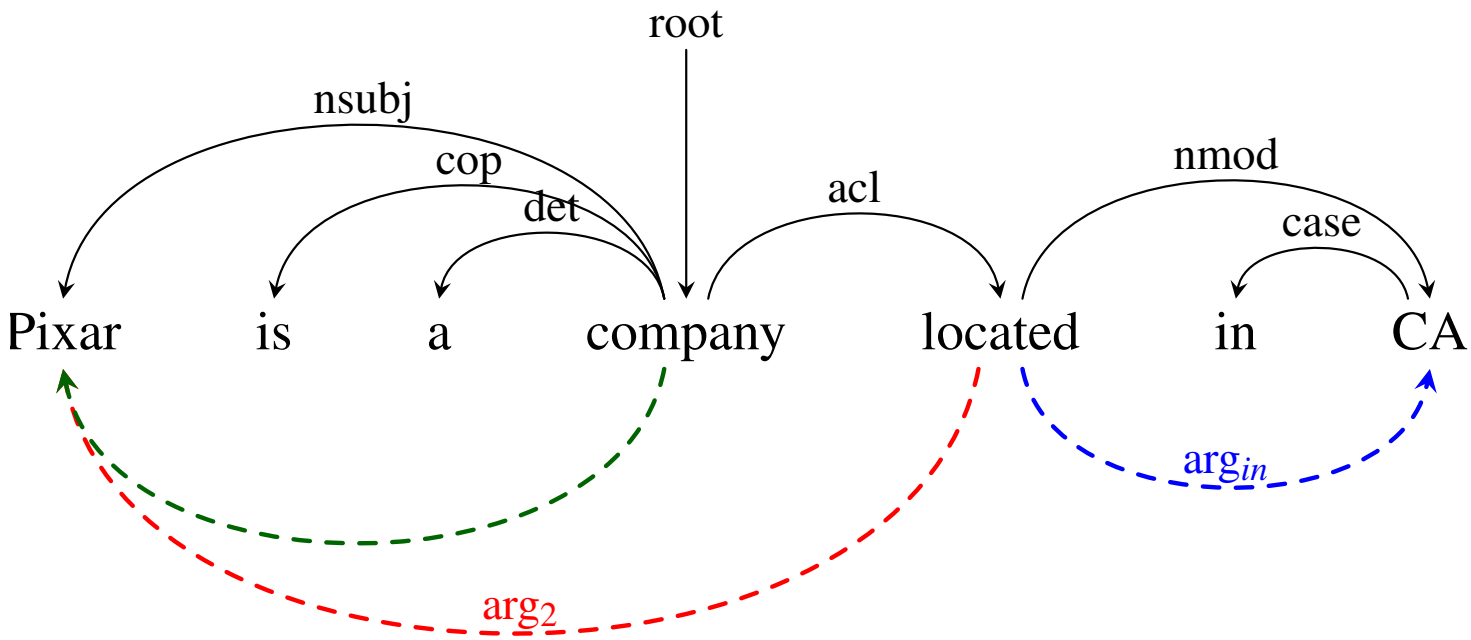
Information Extraction [Rocktäschel et al., 2015]

Summarization [Liu et al., 2015]

Dependencies to Logical Forms

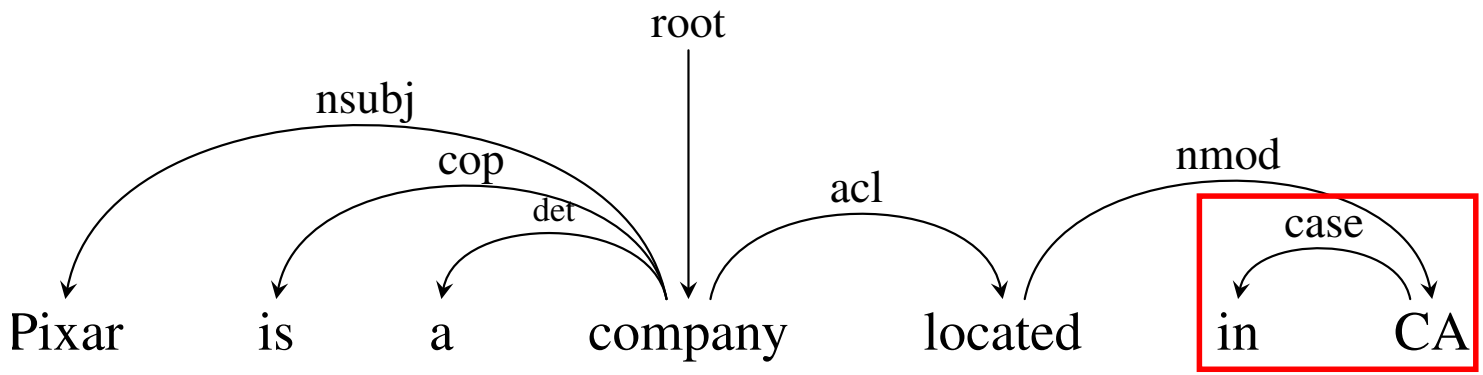


Dependencies to Logical Forms



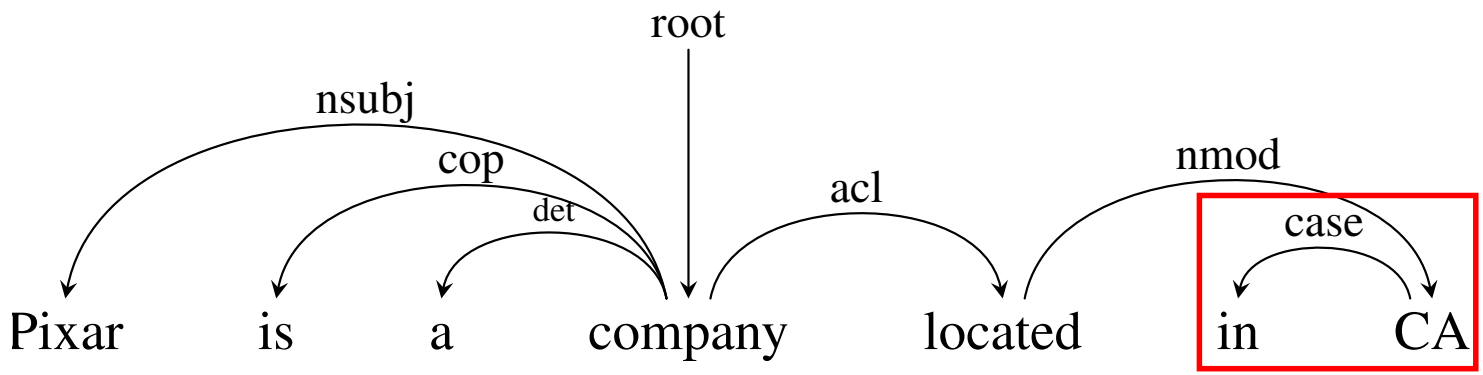
$$\lambda x.\exists yz.\,\text{located}(z_e) \wedge \text{Pixar}(x_a) \wedge \text{CA}(y_a) \wedge$$
$$\text{company}(x_a) \wedge \text{arg}_2(z_e, x_a) \wedge \text{arg}_{\text{in}}(z_e, y_a)$$

Dependencies to Logical Forms



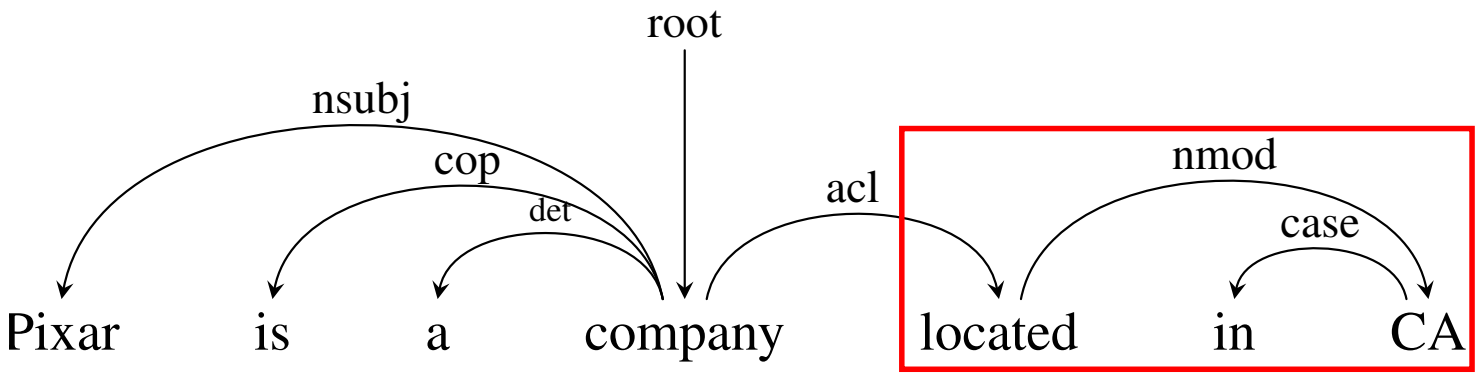
$(case \quad \text{CA} \quad \text{in})$

Dependencies to Logical Forms



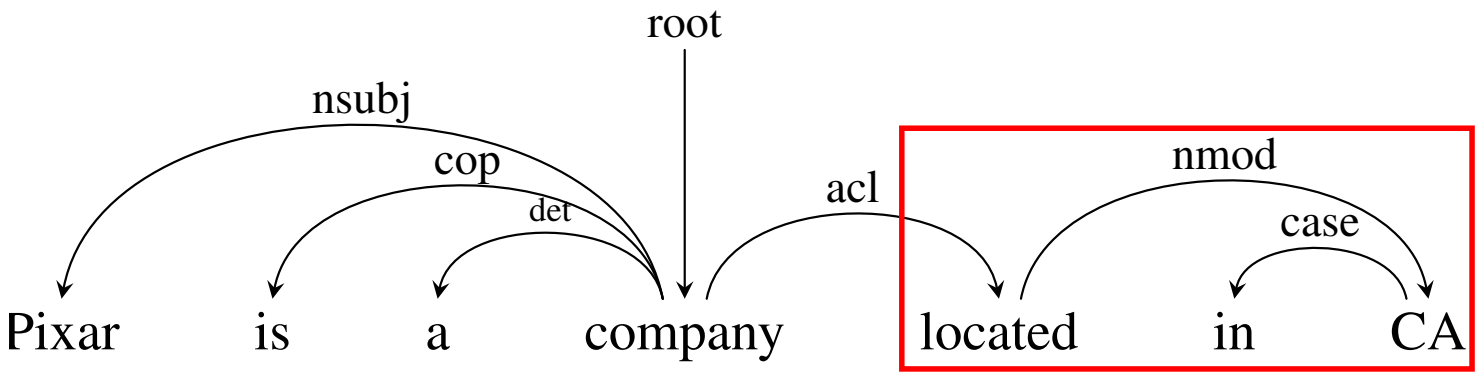
$$\frac{(case \quad \text{CA} \quad \text{in})}{\lambda x.\ \text{CA}(x_a)}$$

Dependencies to Logical Forms



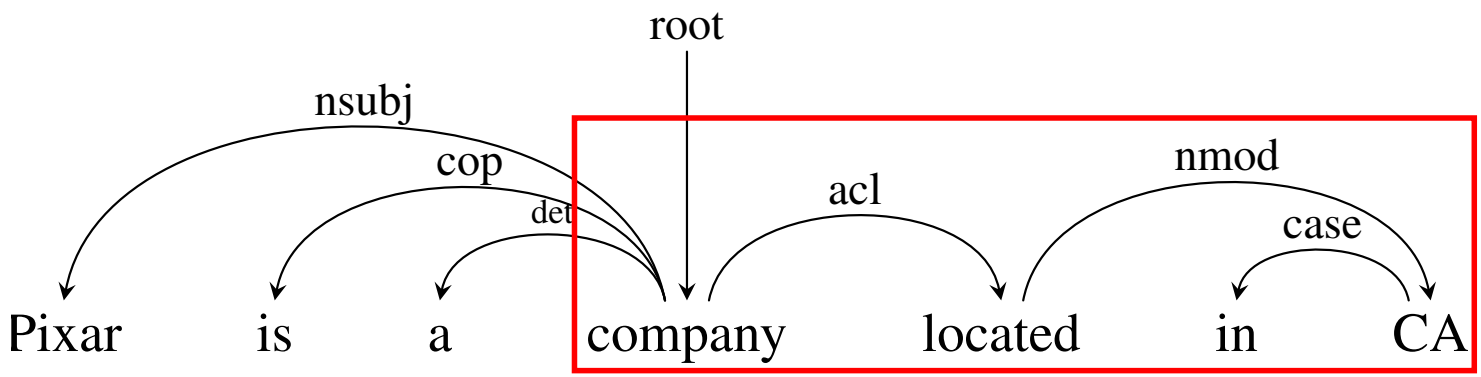
$(nmod \quad \text{located} \quad \text{in CA})$

Dependencies to Logical Forms



$$\frac{(nmod \qquad \text{located} \qquad \text{in CA})}{\lambda z.\text{located}(z_e) \wedge \text{CA}(x_a) \wedge \text{arg}_{\text{in}}(z_e, x_a)}$$

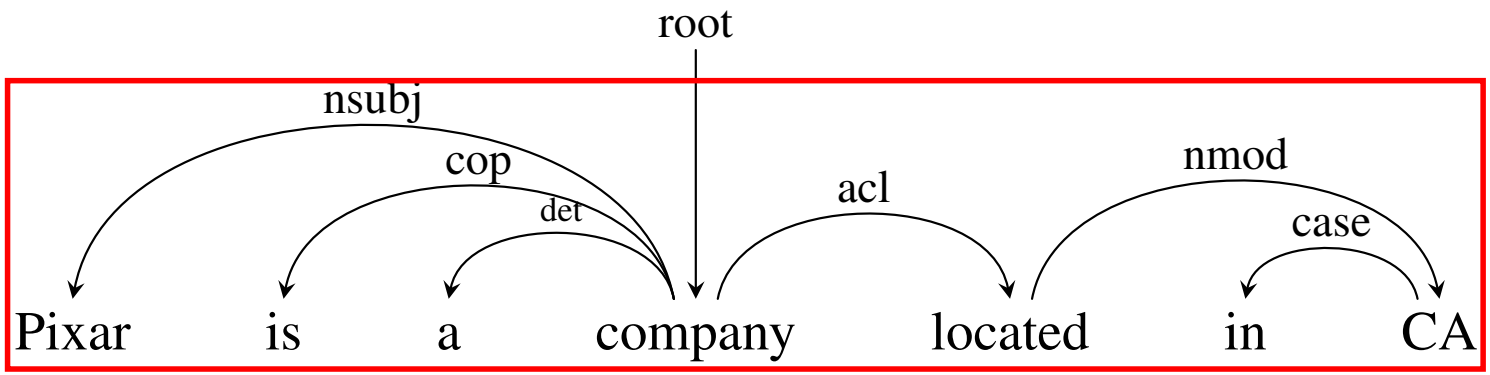
Dependencies to Logical Forms



$(acl \qquad \text{company} \qquad \text{located in CA})$

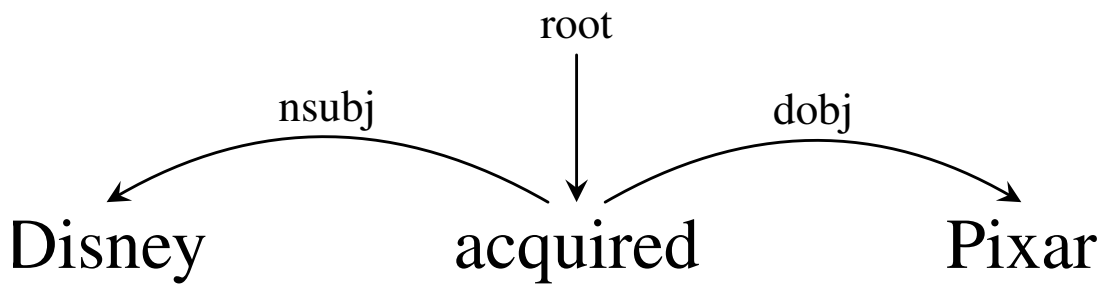
$$\lambda x.\exists yz.\text{company}(x_a) \wedge \text{located}(z_e) \wedge \text{CA}(y_a) \wedge \text{arg}_2(z_e, x_a) \wedge \text{arg}_{\text{in}}(z_e, y_a)$$

Dependencies to Logical Forms



$$\lambda x.\exists yz.\,\text{located}(z_e) \wedge \text{Pixar}(x_a) \wedge \text{CA}(y_a) \wedge$$
$$\text{company}(x_a) \wedge \text{arg}_2(z_e, x_a) \wedge \text{arg}_{\text{in}}(z_e, y_a)$$

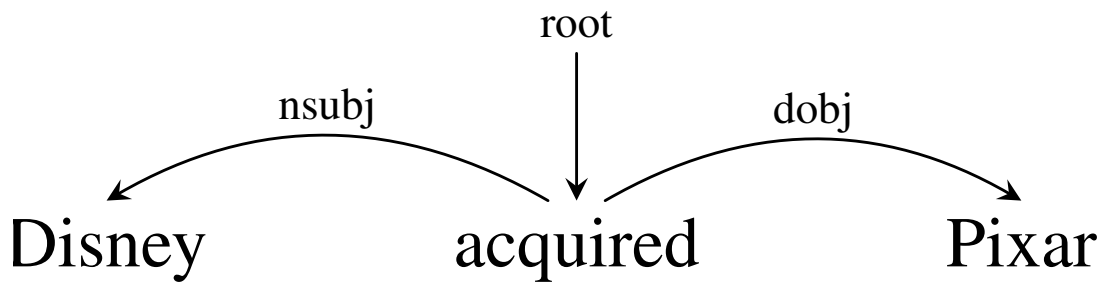
Dependencies to Logical Forms



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

Dependencies to Logical Forms

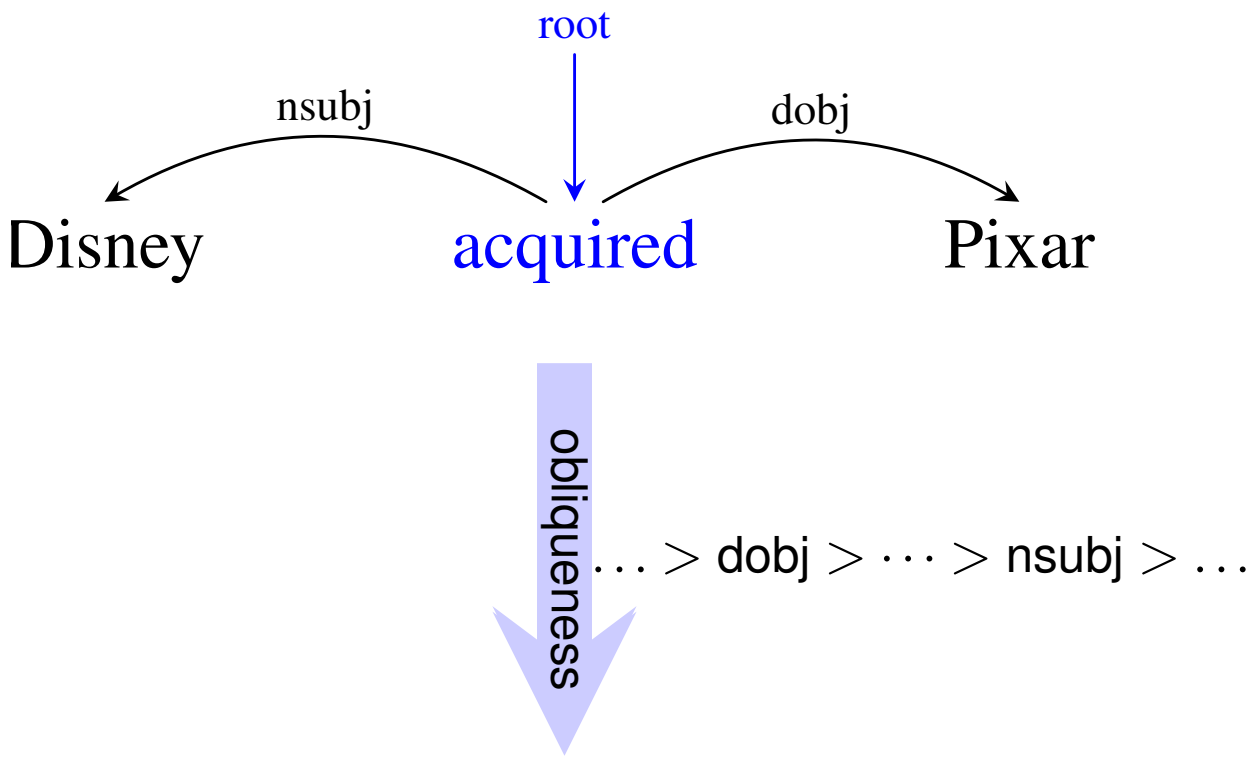
Our Approach



Let dependency labels drive the composition

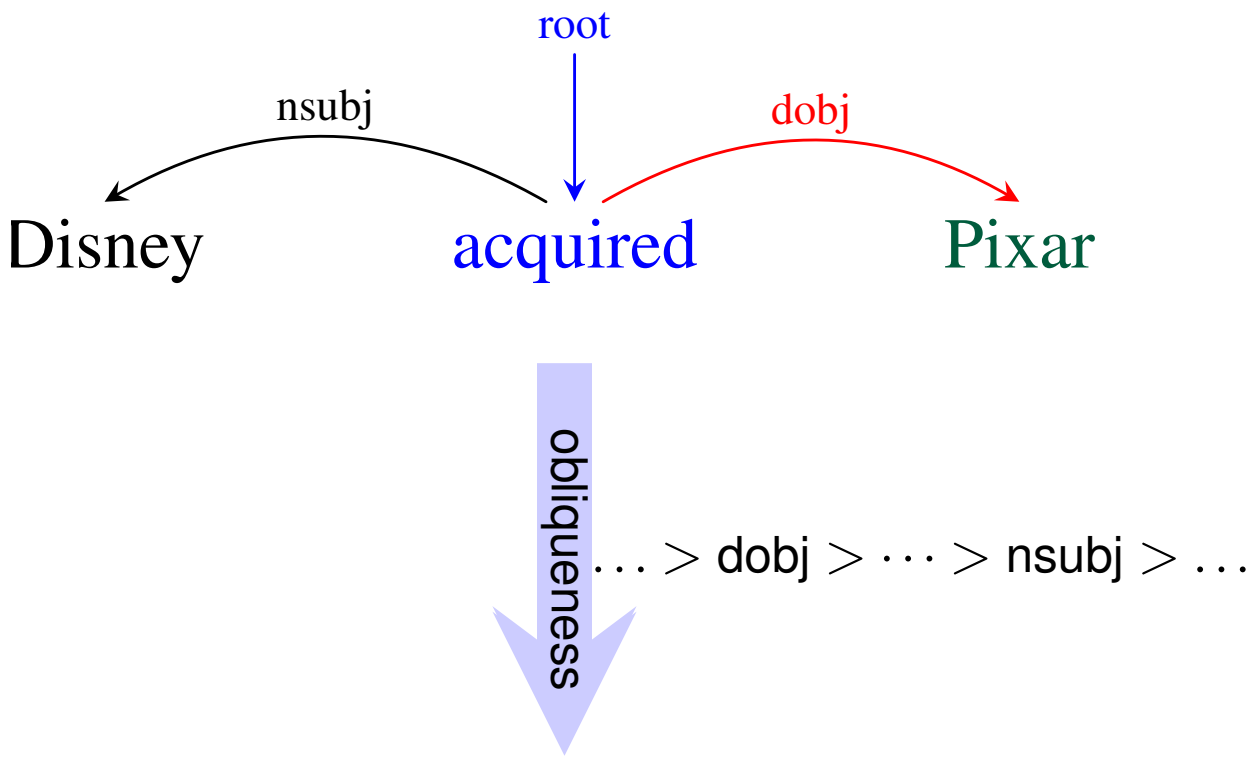
Dependencies to Logical Forms

Our Approach



Dependencies to Logical Forms

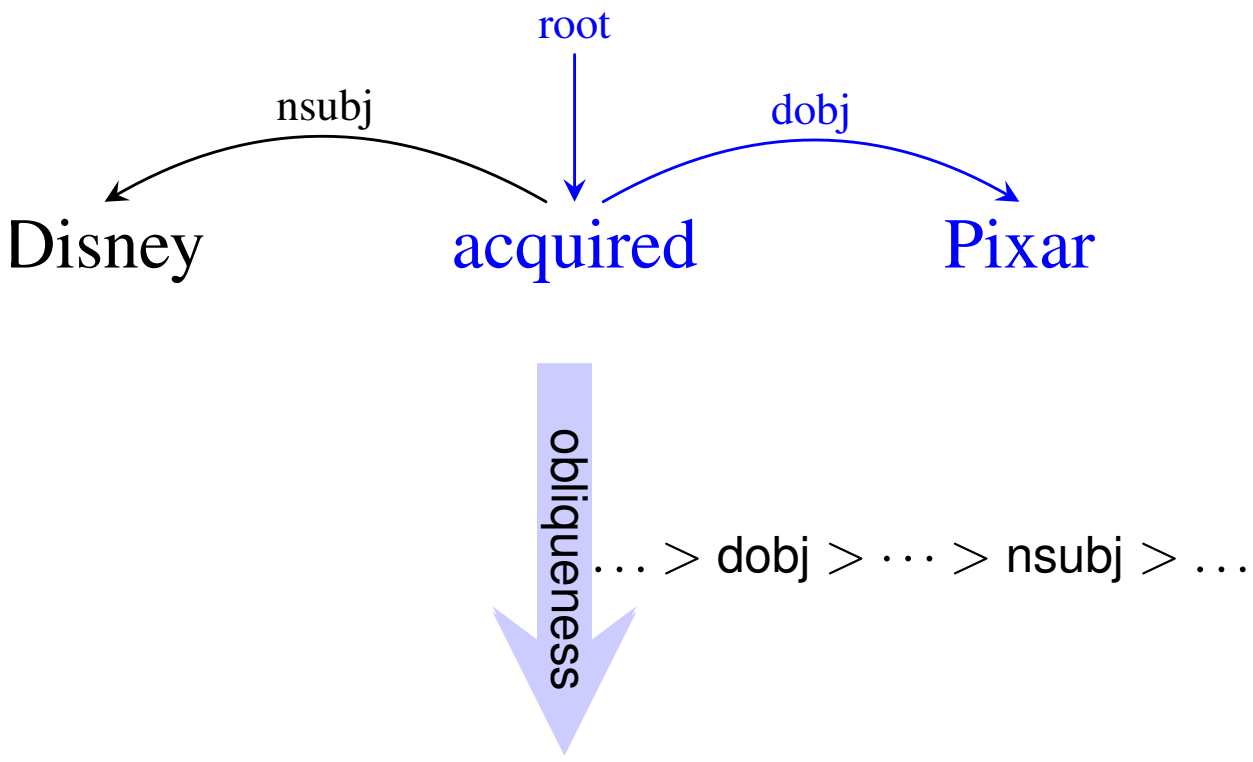
Our Approach



(dobj acquired Pixar)

Dependencies to Logical Forms

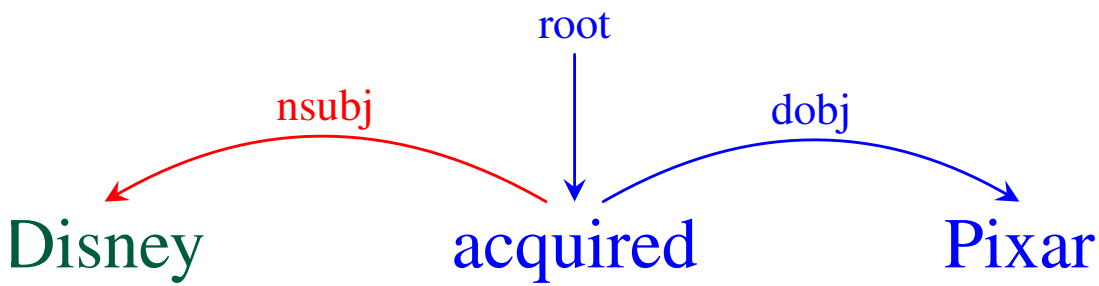
Our Approach

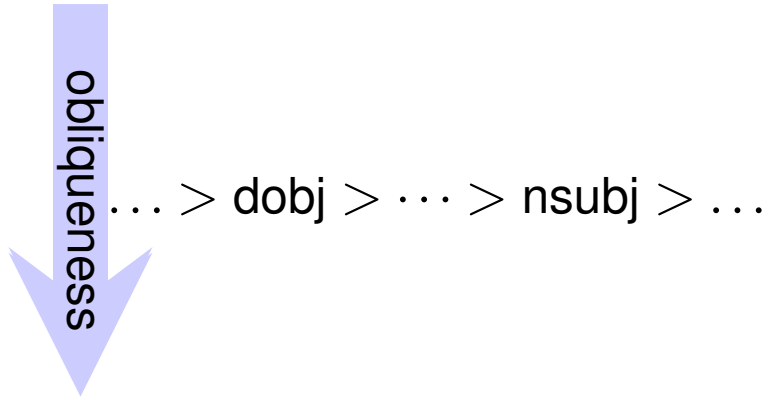


(dobj acquired Pixar)

Dependencies to Logical Forms

Our Approach

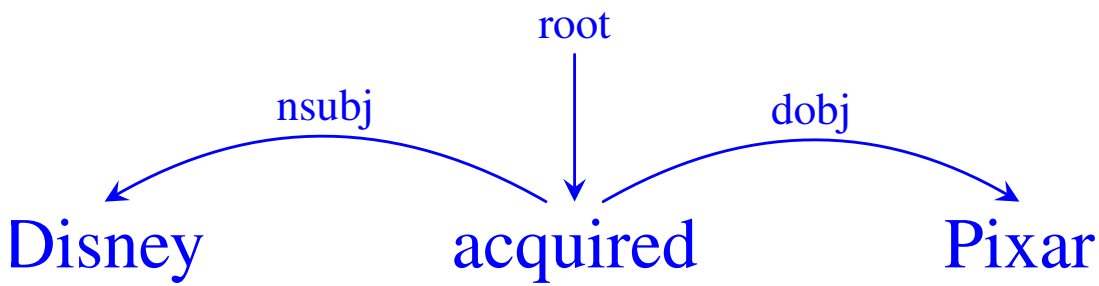


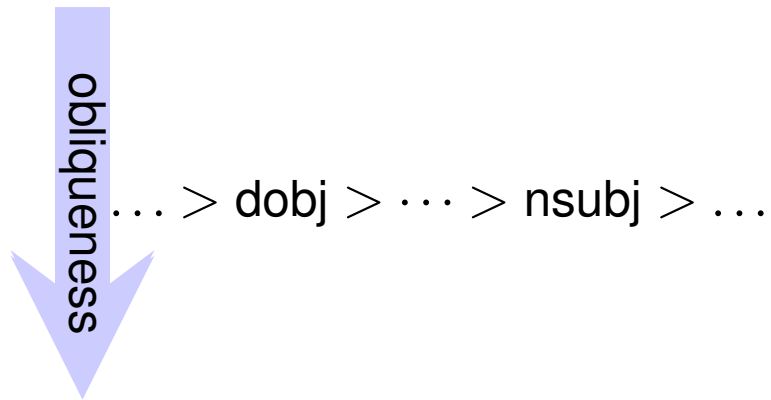


(nsubj (dobj acquired Pixar) Disney)

Dependencies to Logical Forms

Our Approach

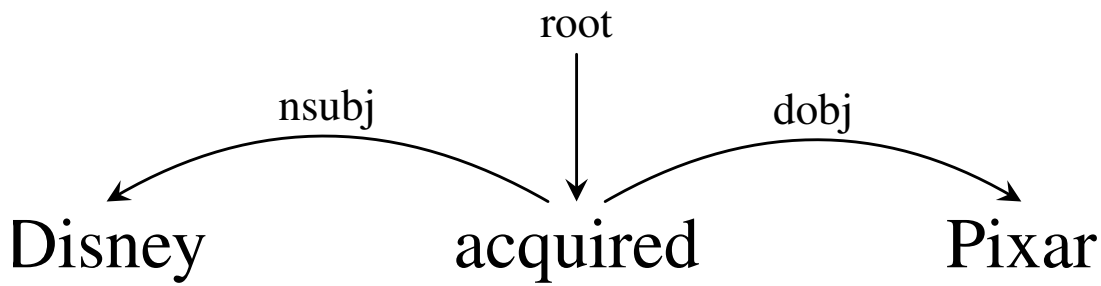




(nsubj (dobj acquired Pixar) Disney)

Dependencies to Logical Forms

Our Approach

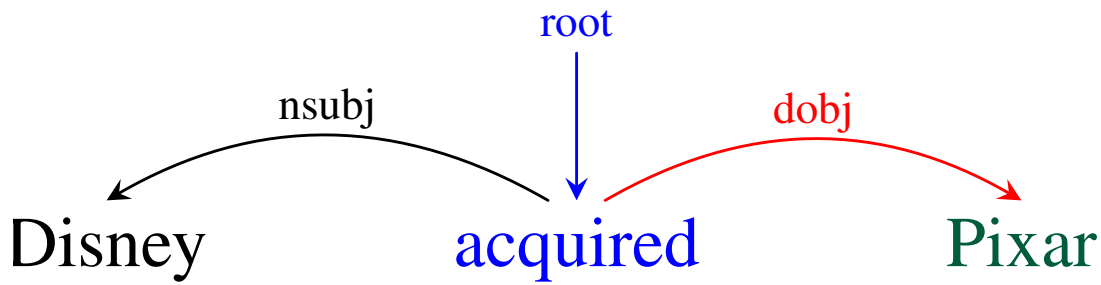


(nsubj (dobj acquired Pixar) Disney)

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

Dependencies to Logical Forms

Lambda Calculus

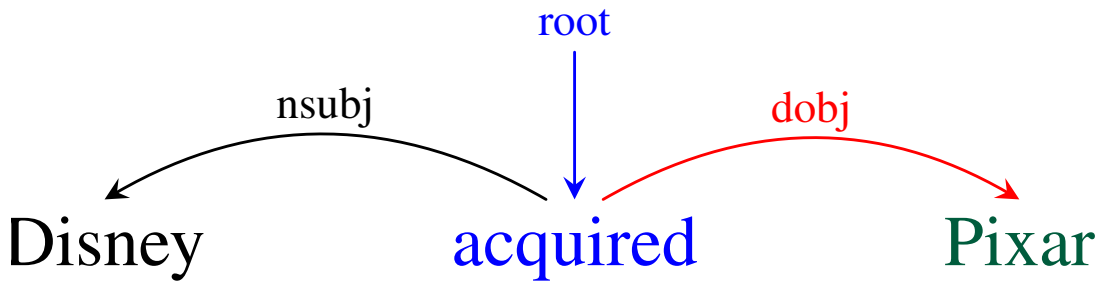


Lambda Calculus Basic Types

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

Dependencies to Logical Forms

Lambda Calculus



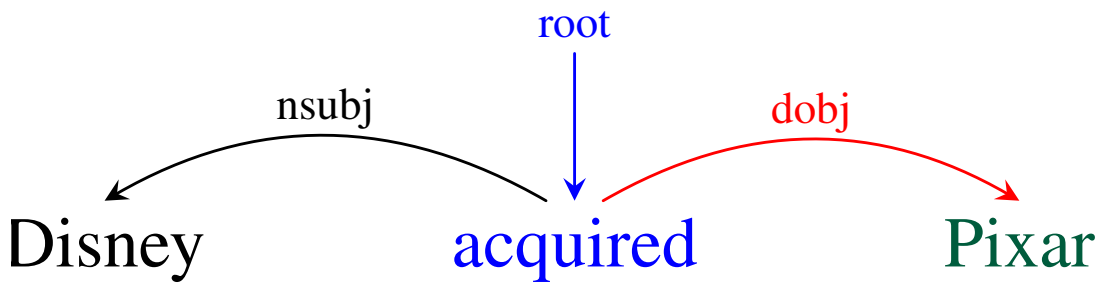
Lambda Expression for words

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

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

Dependencies to Logical Forms

Lambda Calculus

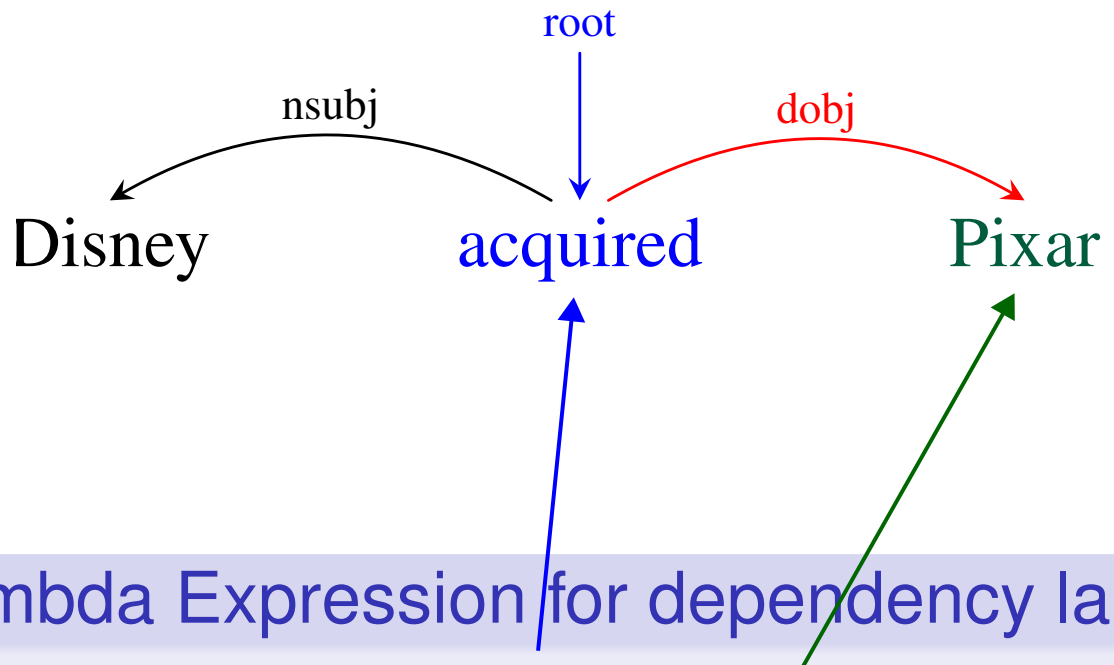


Lambda Expression for dependency labels

$$\text{dobj} \Rightarrow \lambda \mathbf{f}\ \lambda \mathbf{g}\ \lambda \mathbf{z}\ .\ \exists \mathbf{x}\ .\ \mathbf{f}(\mathbf{z}) \quad \wedge \quad \mathbf{g}(\mathbf{x}) \quad \wedge \quad \mathbf{arg_2}(\mathbf{z_e}, \mathbf{x_a})$$

Dependencies to Logical Forms

Lambda Calculus

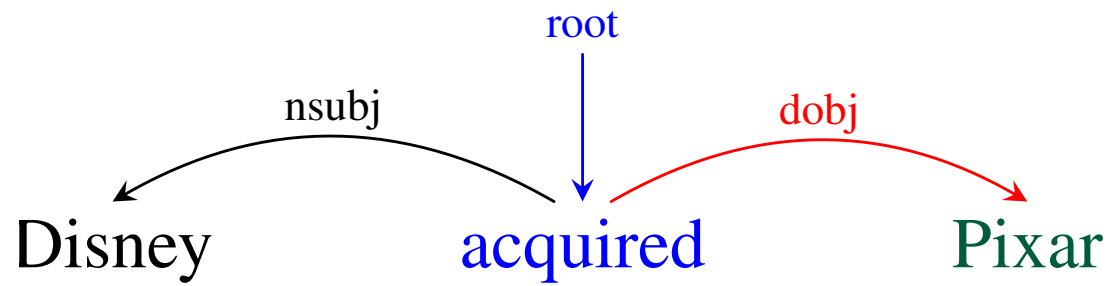


Lambda Expression for dependency labels

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

Dependencies to Logical Forms

Composition



$$(\textbf{dobj} \quad \textbf{acquired} \quad \textbf{Pixar})$$

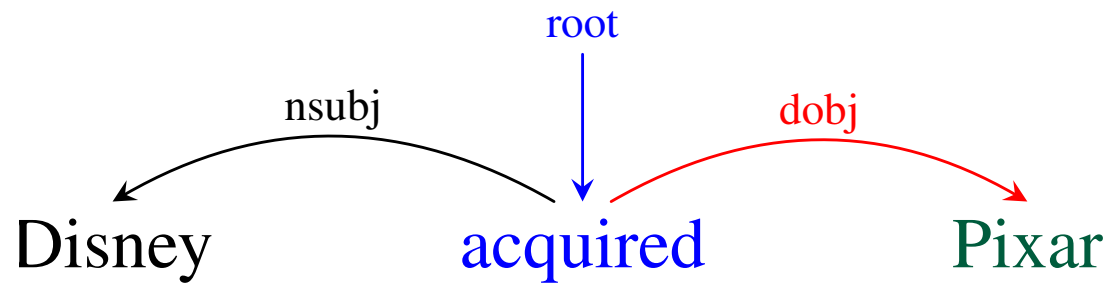
$$\lambda f \lambda g \lambda z. \exists y. \quad \lambda z.\text{acquired}(z_e) \quad \lambda y.\text{Pixar}(y_a)$$

$$f(z) \wedge g(y) \wedge$$

$$\arg_2(z_e, y_a)$$

Dependencies to Logical Forms

Composition



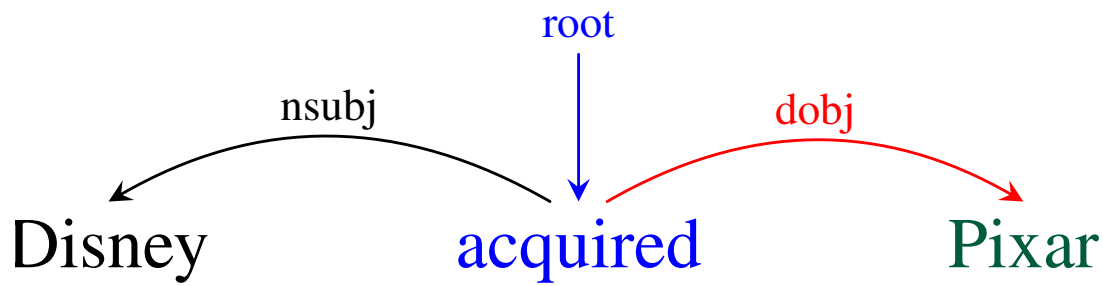
$$(\textbf{dobj} \quad \textbf{acquired} \quad \textbf{Pixar})$$

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

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

Dependencies to Logical Forms

Composition



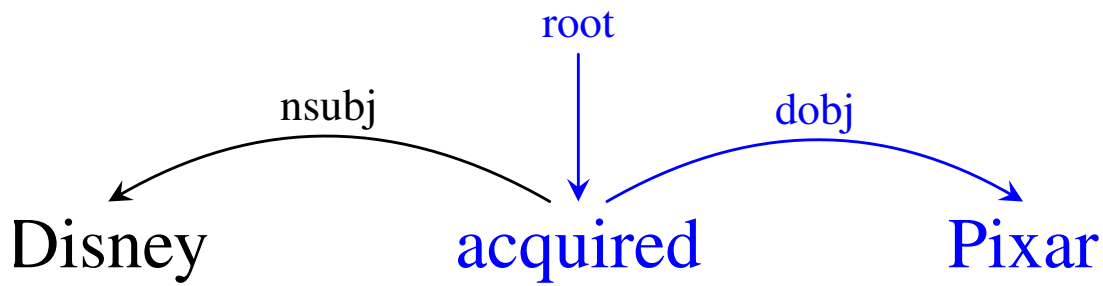
(**dobj**	**acquired**	**Pixar**)
$\lambda f \lambda g \lambda z. \exists y.$ $f(z) \wedge g(y) \wedge$ $\arg_2(z_e, y_a)$	$\lambda z.\text{acquired}(z_e)$	$\lambda y.\text{Pixar}(y_a)$

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

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

Dependencies to Logical Forms

Composition

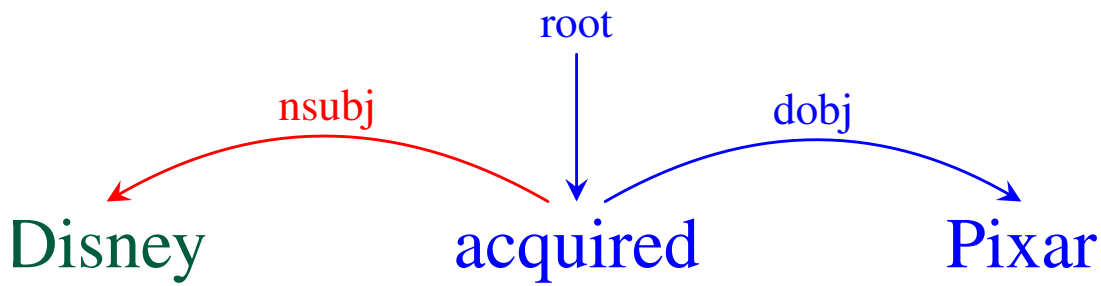


$$(\textbf{dobj} \quad \textbf{acquired} \quad \textbf{Pixar})$$

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

Dependencies to Logical Forms

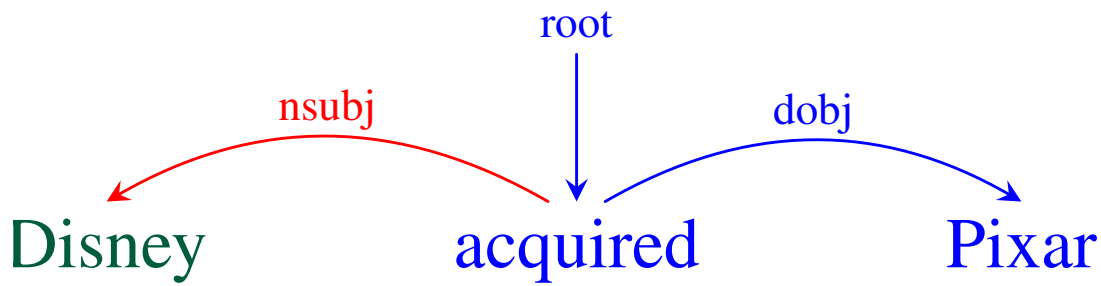
Composition



$$\frac{\begin{array}{c}(\textbf{nsubj} \\ \lambda f \lambda g \lambda z.\ \exists x. \\ f(z) \wedge g(x) \wedge \\ \arg_1(z_e, x_a)\end{array} \quad \frac{(\textbf{dobj} \quad \textbf{acquired} \quad \textbf{Pixar})}{\begin{array}{c}\lambda z.\ \exists y.\ \text{acquired}(z_e) \wedge \text{Pixar}(y_a) \\ \wedge\ \arg_2(z_e, y_a)\end{array}} \quad \begin{array}{c}\textbf{Disney}) \\ \lambda x.\text{Disney}(x_a)\end{array}}{}$$

Dependencies to Logical Forms

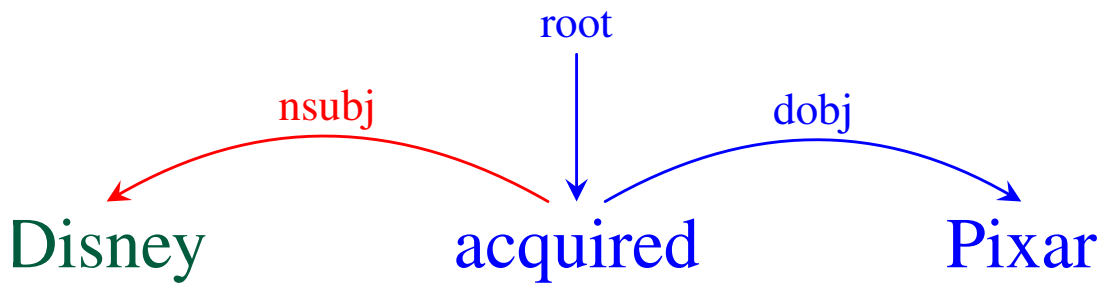
Composition



$$\frac{\dfrac{(\textbf{nsubj} \quad \lambda f \lambda g \lambda z.\ \exists x.\ f(z) \wedge g(x) \wedge \arg_1(z_e, x_a) \quad \dfrac{(\textbf{dobj} \quad \textbf{acquired} \quad \textbf{Pixar})}{\lambda z.\ \exists y.\ \text{acquired}(z_e) \wedge \text{Pixar}(y_a) \wedge \arg_2(z_e, y_a)}}{\lambda g \lambda z. \exists x y. \text{acquired}(z_e) \wedge \text{Pixar}(y_a) \wedge \text{g}(x) \wedge \arg_1(z_e, x_a) \wedge \arg_2(z_e, y_a)} \quad \textbf{Disney})}{\lambda x. \text{Disney}(x_a)}$$

Dependencies to Logical Forms

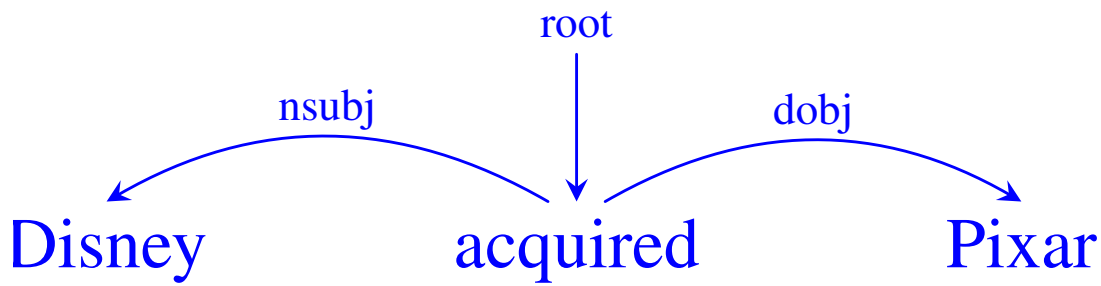
Composition



$$\frac{\dfrac{(\textbf{nsubj} \quad \lambda f \lambda g \lambda z.\ \exists x.\ f(z) \wedge g(x) \wedge \arg_1(z_e, x_a)) \qquad \dfrac{(\textbf{dobj} \quad \textbf{acquired} \quad \textbf{Pixar}) \qquad \textbf{Disney})}{\lambda z.\ \exists y.\ \text{acquired}(z_e) \wedge \text{Pixar}(y_a) \wedge \arg_2(z_e, y_a)} \quad \lambda x.\text{Disney}(x_a)}{\lambda g \lambda z. \exists x y. \text{acquired}(z_e) \wedge \text{Pixar}(y_a) \wedge \text{g}(x) \wedge \arg_1(z_e, x_a) \wedge \arg_2(z_e, y_a)}}{\lambda z. \exists x y. \text{acquired}(z_e) \wedge \text{Pixar}(y_a) \wedge \text{Disney}(x_a) \wedge \arg_1(z_e, x_a) \wedge \arg_2(z_e, y_a)}$$

Dependencies to Logical Forms

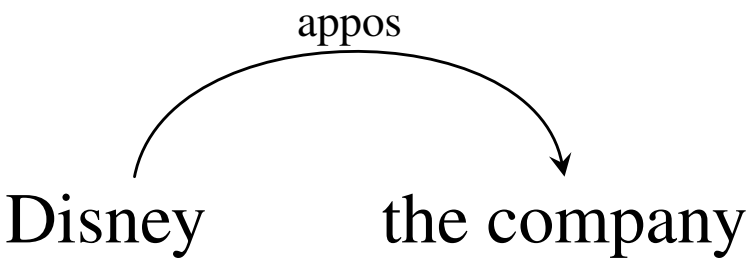
Composition



$$(\textbf{nsubj} \quad (\textbf{dobj} \quad \textbf{acquired} \quad \textbf{Pixar}) \quad \textbf{Disney})$$

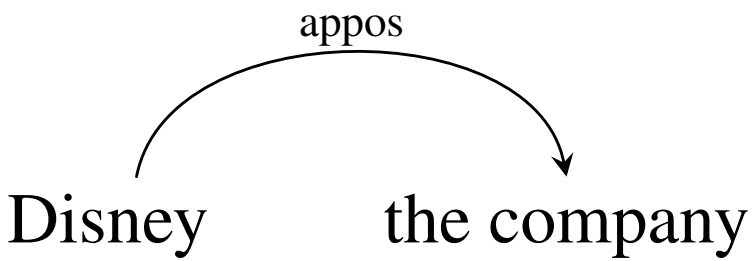
$$\lambda z.\exists xy.\text{acquired}(z_e) \wedge \text{Pixar}(y_a) \wedge \text{Disney}(x_a) \wedge \text{arg}_1(z_e, x_a) \wedge \text{arg}_2(z_e, y_a)$$

Dependencies to Logical Forms

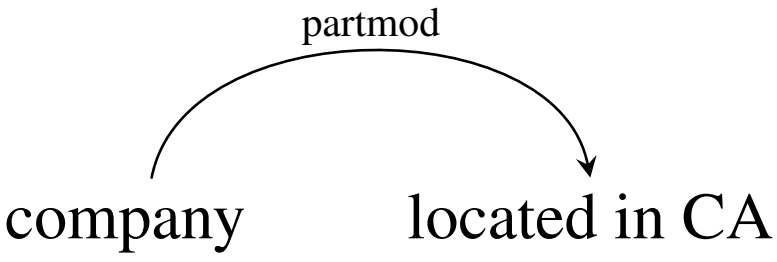


$$\textit{appos} = \lambda f \lambda g \lambda x . f(x) \wedge g(x)$$

Dependencies to Logical Forms

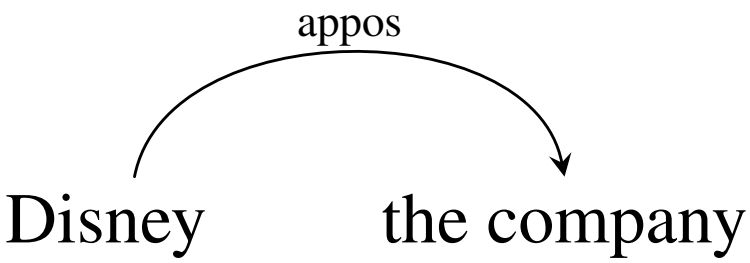


$$\textit{appos} = \lambda f \lambda g \lambda x . f(x) \wedge g(x)$$

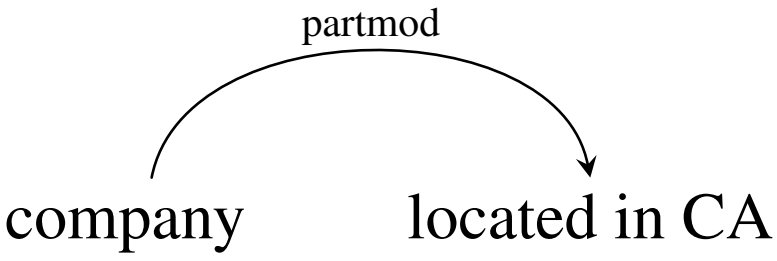


$$\textit{partmod} = \lambda f \lambda g \lambda x . \exists z . f(x) \wedge g(z) \wedge \text{arg}_2(z_e, x_a)$$

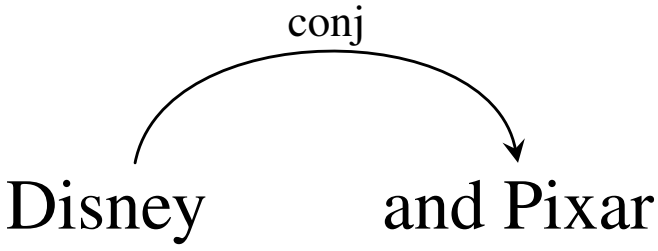
Dependencies to Logical Forms



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

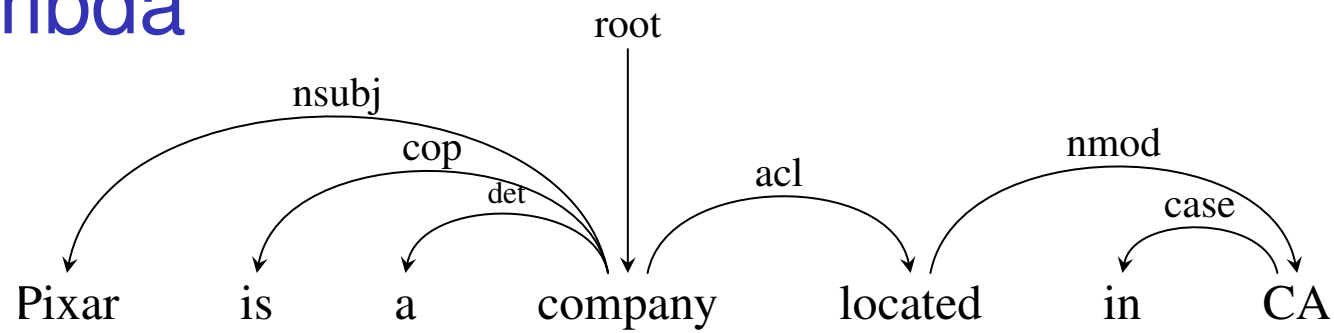


$$\textit{partmod} = \lambda f \lambda g \lambda x . \exists z . f(x) \wedge g(z) \wedge \text{arg}_2(z_e, x_a)$$



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

DepLambda



$\ldots > \text{dobj} > \cdots > \text{nsubj} > \ldots$

... $(acl\ \lambda fgx.\exists z.\ f(x) \wedge g(z) \wedge \text{arg}_2(z_e, x_a))$ company $\lambda x.\text{compay}(x_a)$ $(nmod\ \lambda fgz.\exists x.\ f(z) \wedge g(x)\ arg_{in}(z_e, x_a))$ located $\lambda x.\text{located}(x_e)$ $(case\ \lambda fgx.f(x))$ CA $\lambda x.\text{CA}(x_a)$ in) $\lambda x.\text{empty}(x)$))...

$\lambda x.\ \text{CA}(x_a)$

lambda expression composition

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

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

Freebase Semantic Parsing using DepLambda

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
Rose DeWitt Bukater


Kathy Bates
Molly Brown

Freebase Semantic Parsing

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

Question

Who is the director of Titanic?

Grounded Logical Form

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

Latent

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)

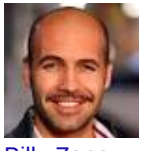
Director: James Cameron

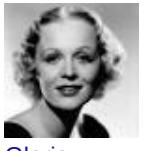
Featured song: My Heart Will Go On

Cast


Leonardo DiCaprio
Jack Dawson


Kate Winslet
Rose DeWitt Bukater


Billy Zane
Caledon Hockley


Gloria Stuart
Rose DeWitt Bukater


Kathy Bates
Molly Brown

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 problem

Question

Who is the director of Titanic?

Grounded Logical Form

$\lambda x.\exists e.$ film.director$(x) \wedge$

film.directed_by$(e) \wedge$

arg1$(e,$Titanic$) \wedge$ arg2(e,x)

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 problem

- director $\rightarrow N : \lambda x.\,\text{film.director}(x)$
- of $\rightarrow (NP\backslash NP)/NP$:
  $\lambda f \lambda g \lambda x.\exists y \exists e.f(y) \wedge g(x) \wedge$
  $\text{film.directed\_by}(e) \wedge$
  $\text{arg1}(e,y) \wedge \text{arg2}(e,x)$

Question

Who is the director of Titanic?

Grounded Logical Form

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

Intermediate Semantic Parsing

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

Question

Who is the director of Titanic?

Language to ungrounded logical form

Ungrounded Logical Form

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

Ungrounded logical form to grounded logical form

Grounded 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 Semantic Parsing: Task Setting

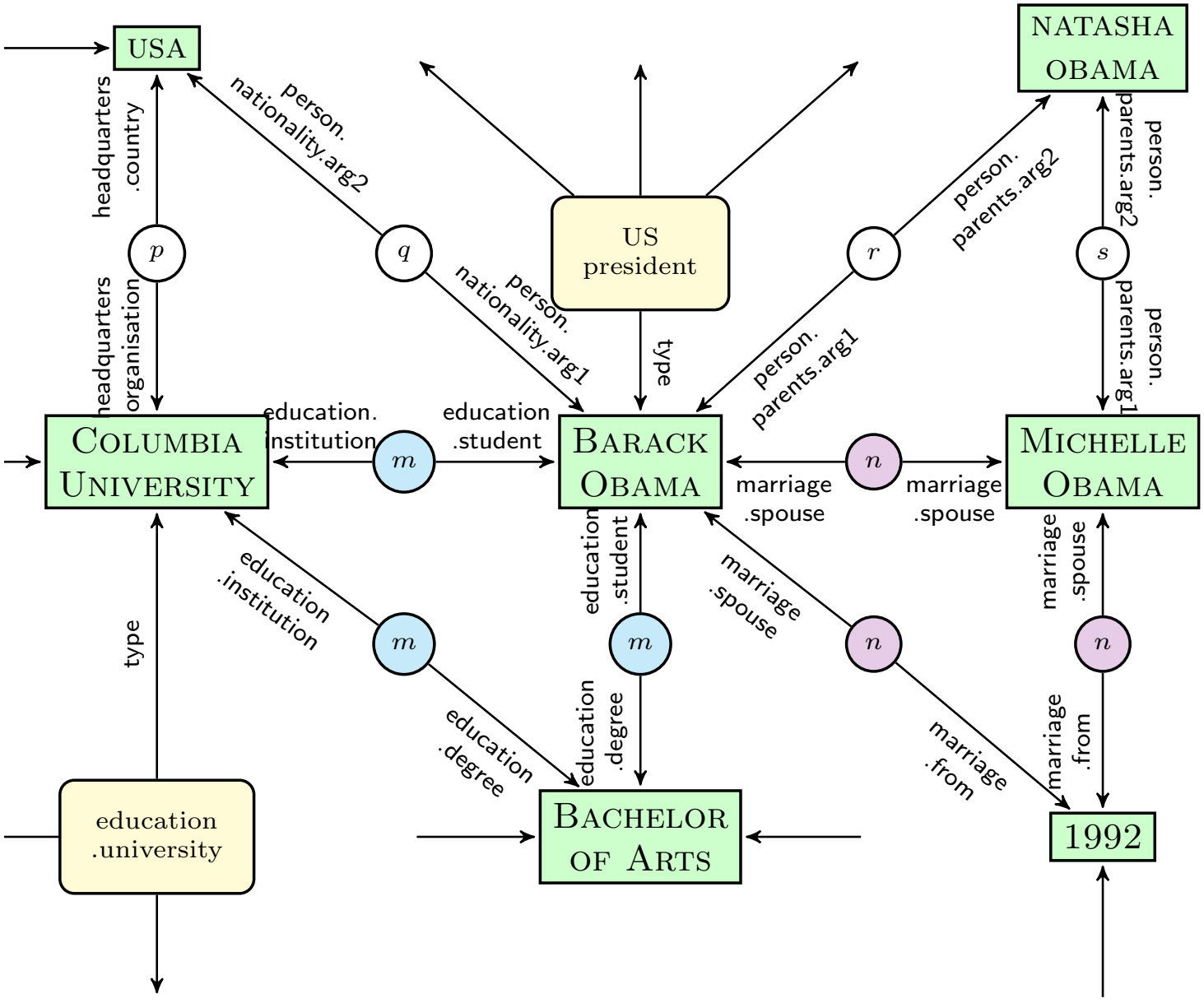
Training Data: Question and Answer Pairs

Evaluation: Question Answering on Freebase

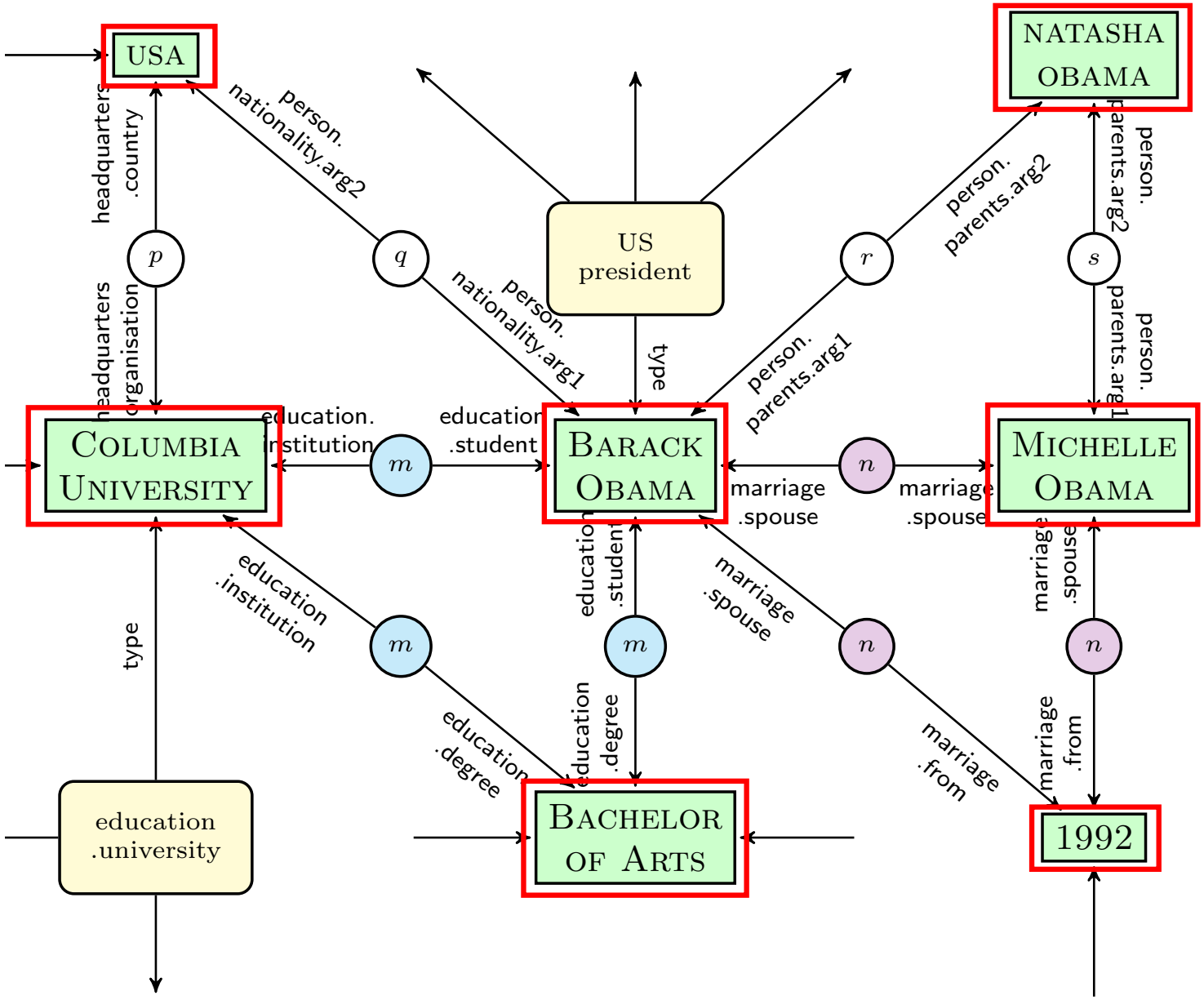
Resources: Dependency Parser, DepLambda

Hypothesis: DepLambda logical forms are useful

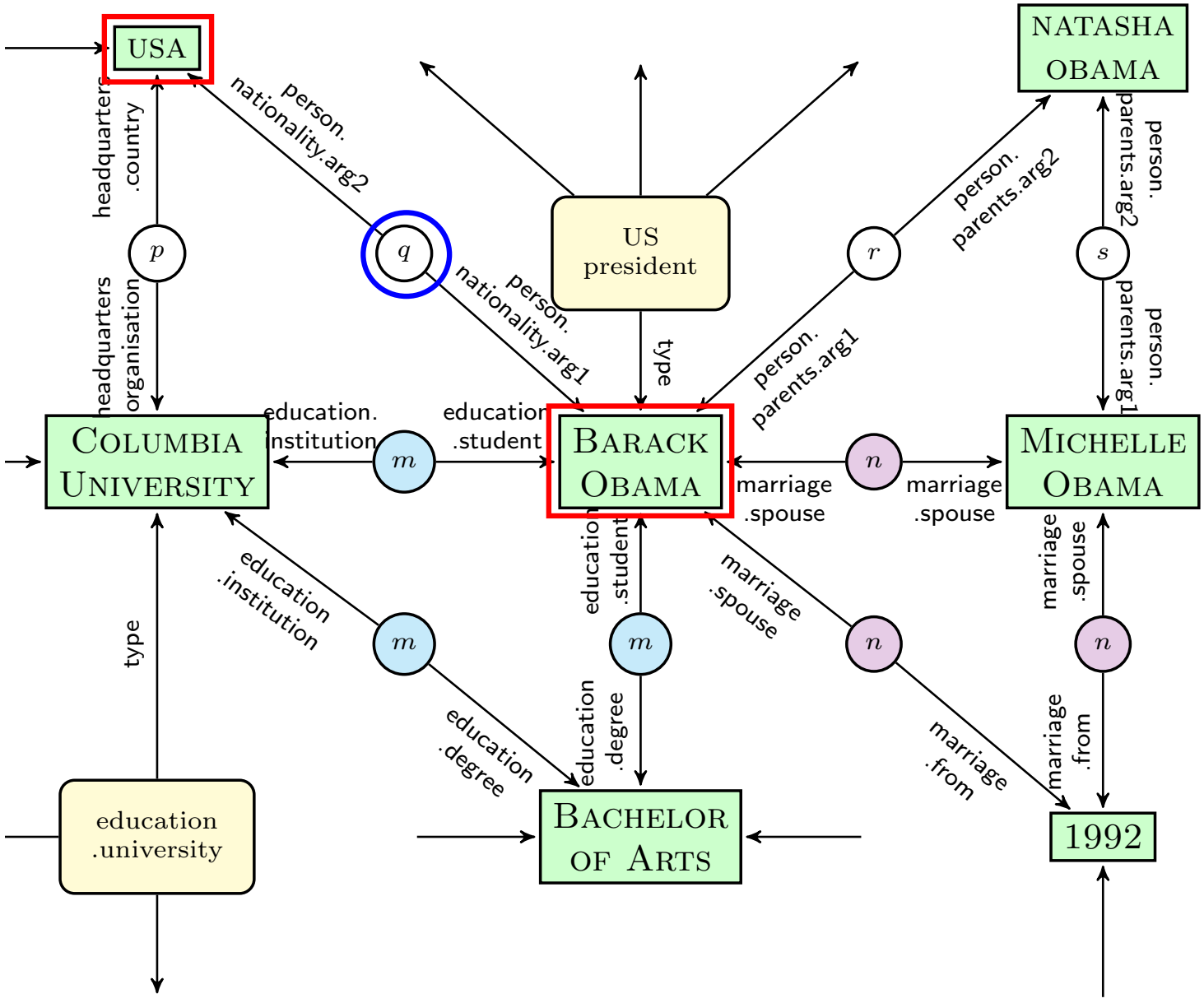
Freebase is a Graph



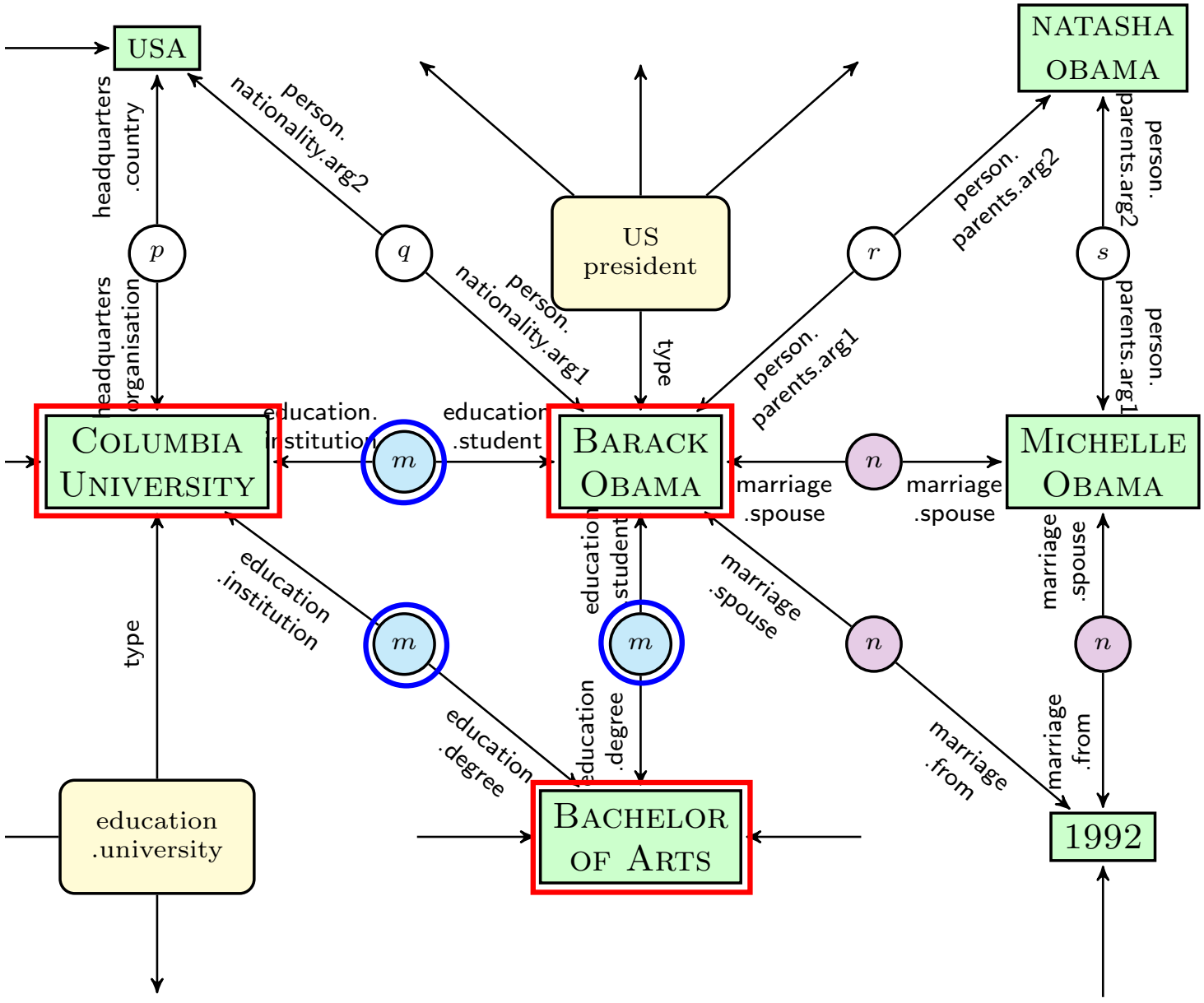
Freebase is a Graph



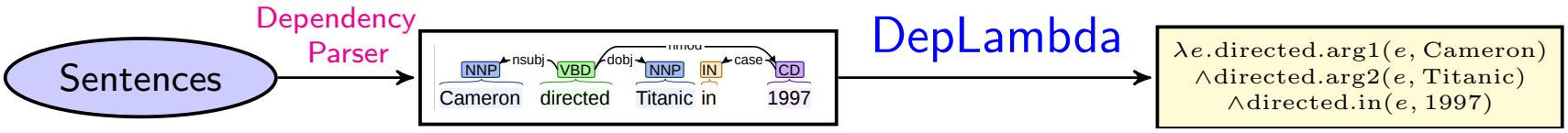
Freebase is a Graph



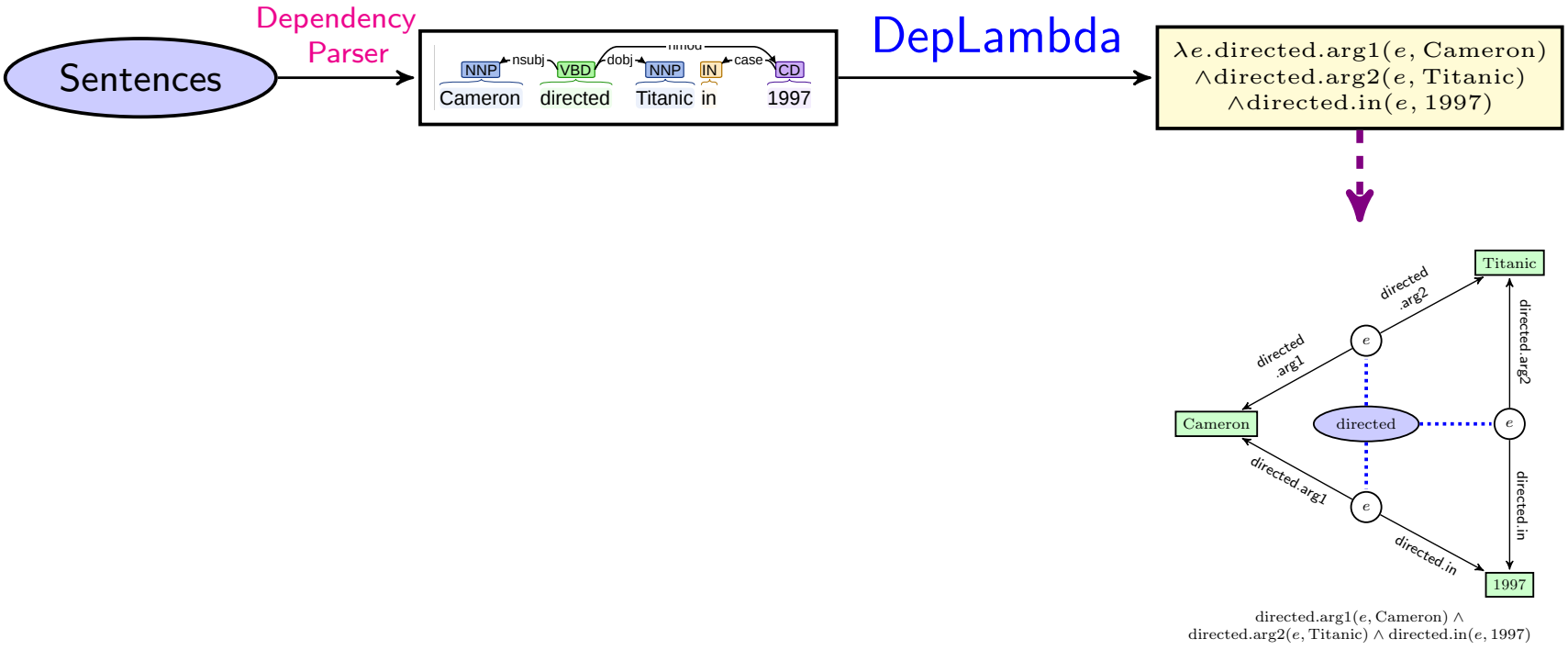
Freebase is a Graph



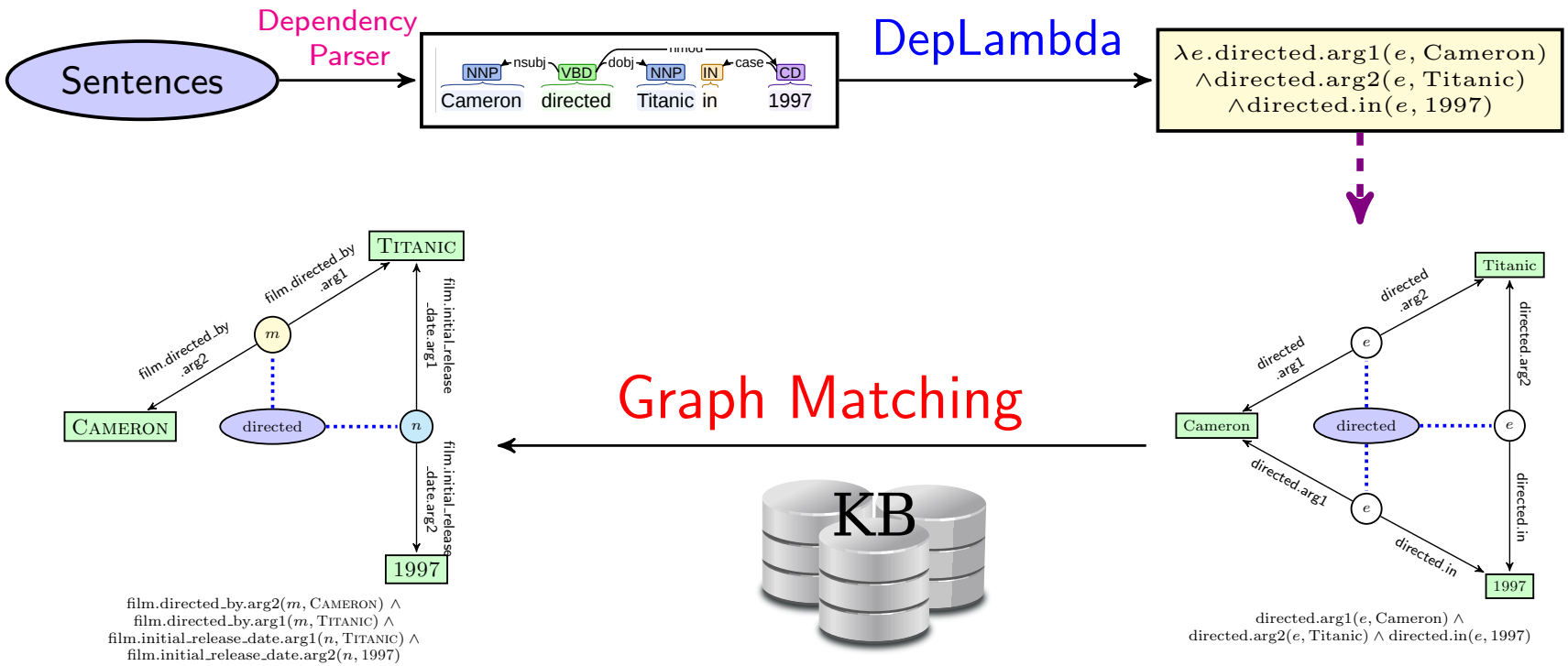
Our Approach



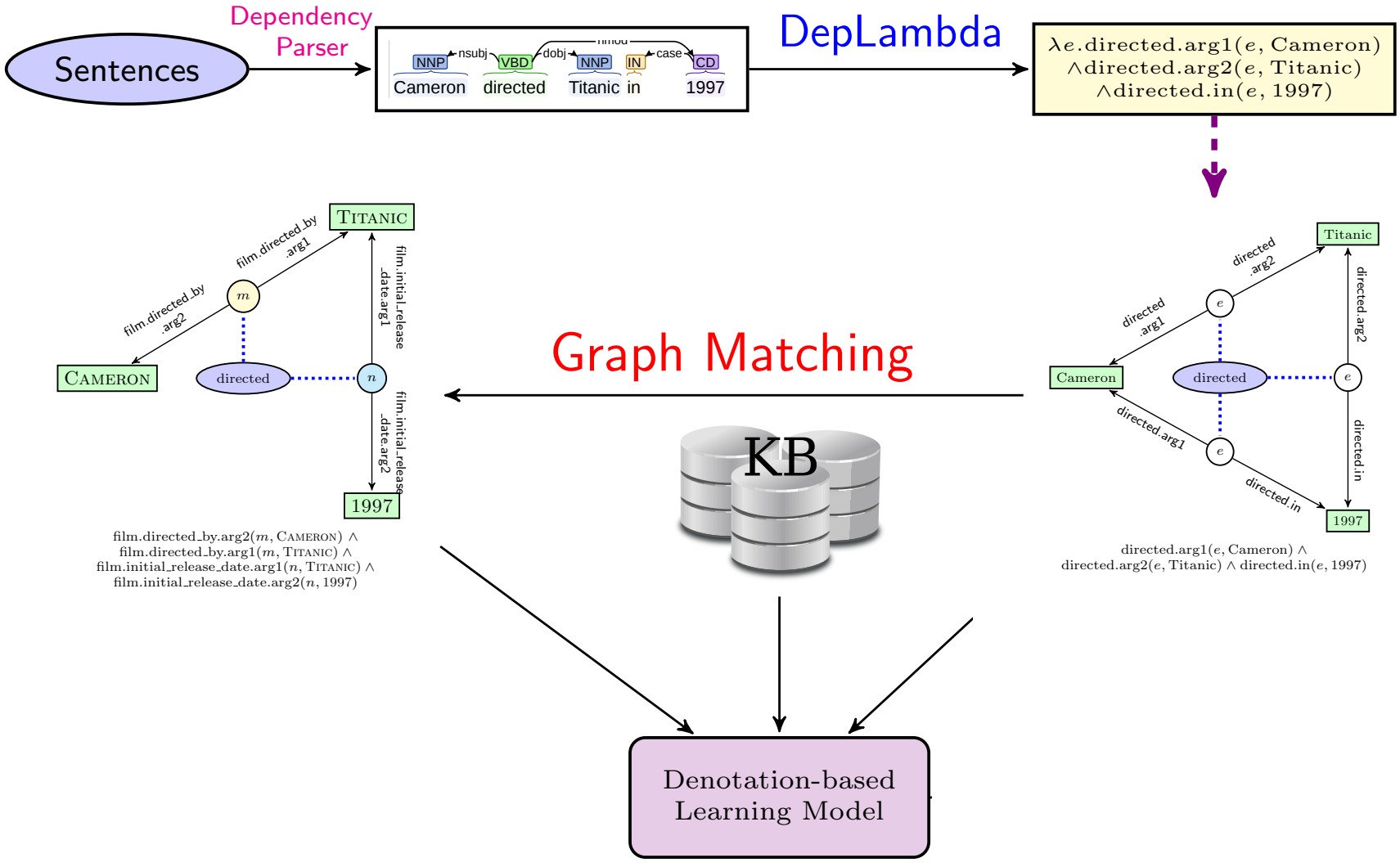
Our Approach



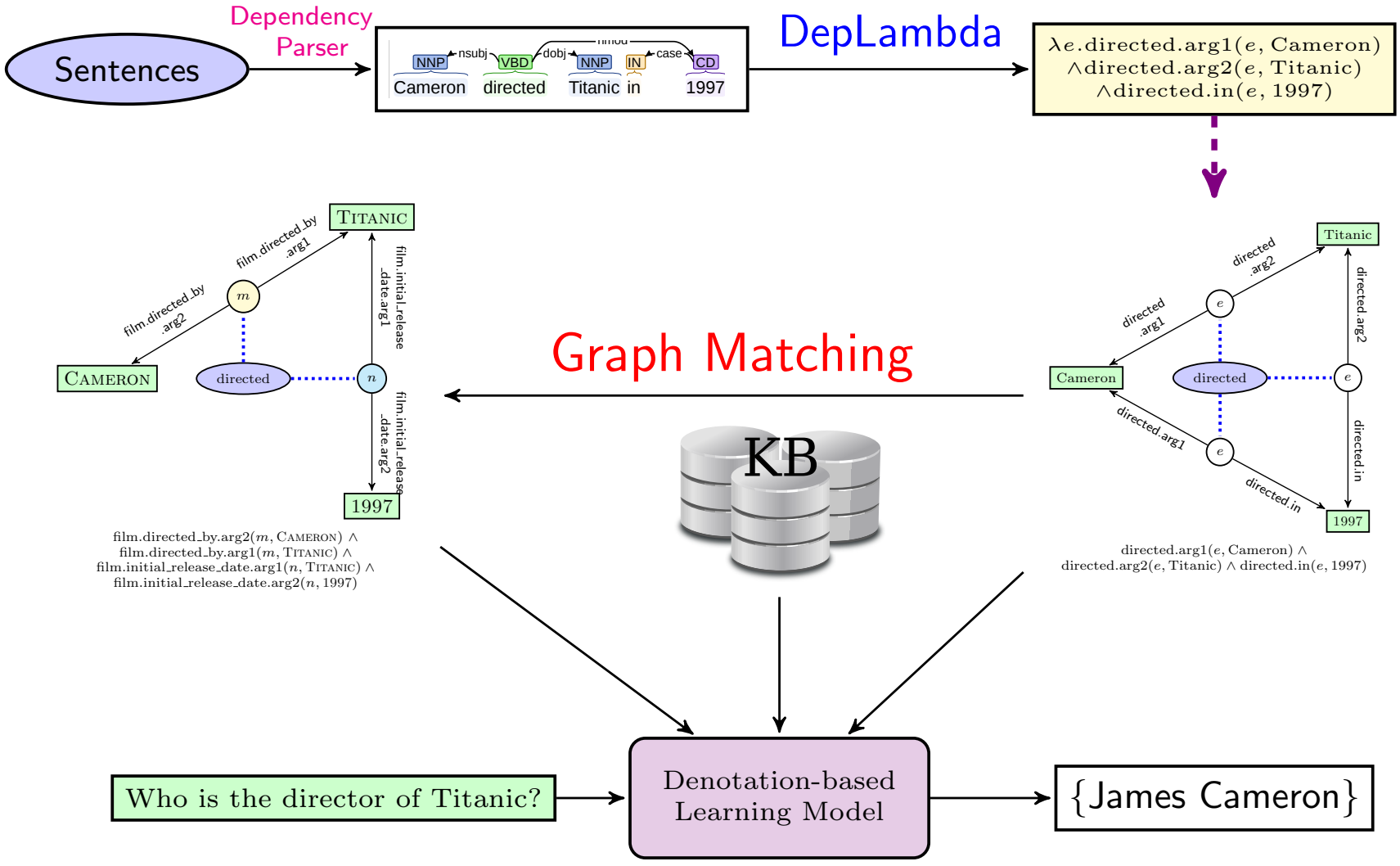
Our Approach



Our Approach



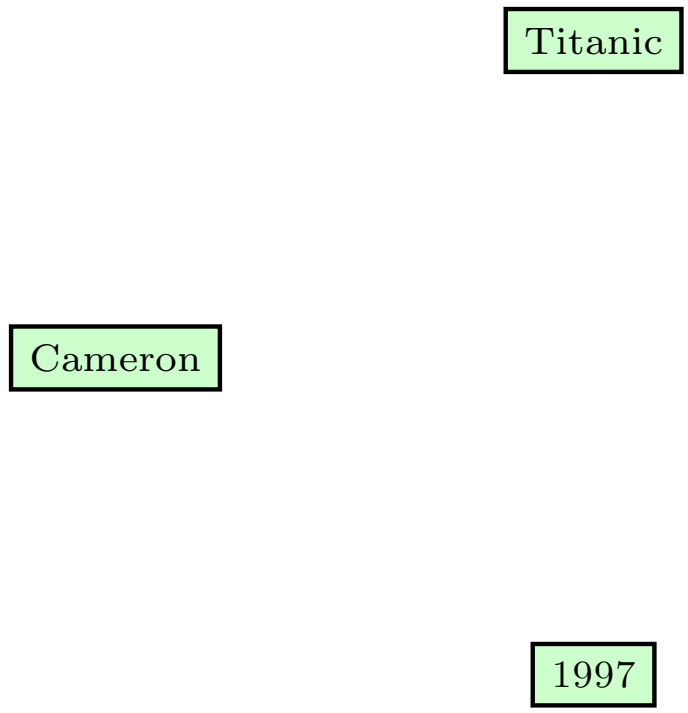
Our Approach



Logical Form to Ungrounded Graph

Cameron directed Titanic in 1997

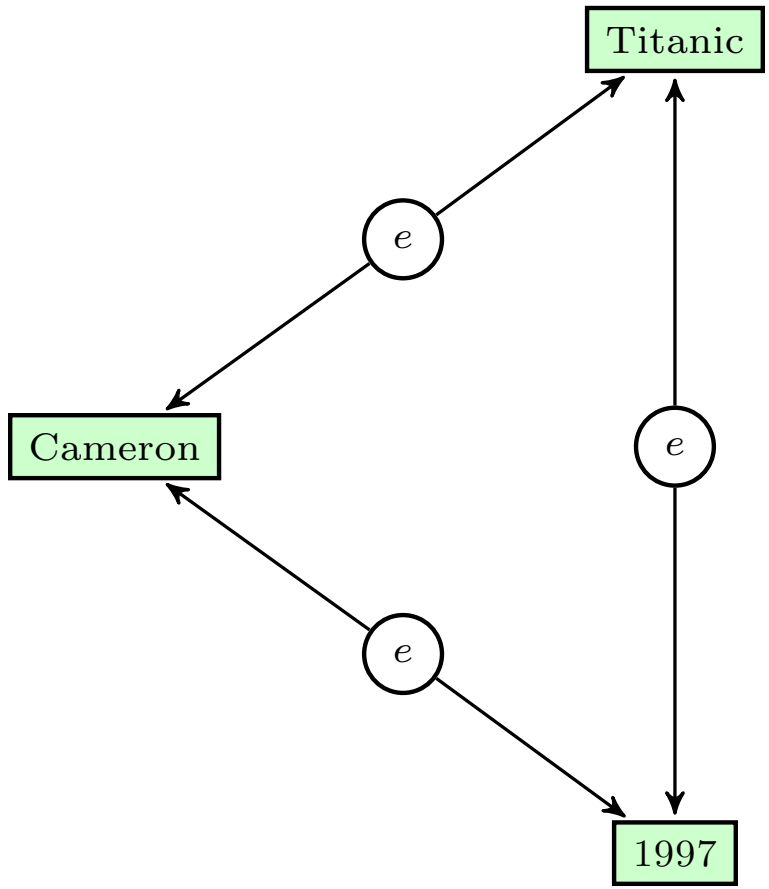
$$\lambda e.\text{directed.arg1}(e, \text{Cameron}) \wedge \text{directed.arg2}(e, \text{Titanic}) \wedge \text{directed.in}(e, 1997)$$



Logical Form to Ungrounded Graph

Cameron directed Titanic in 1997

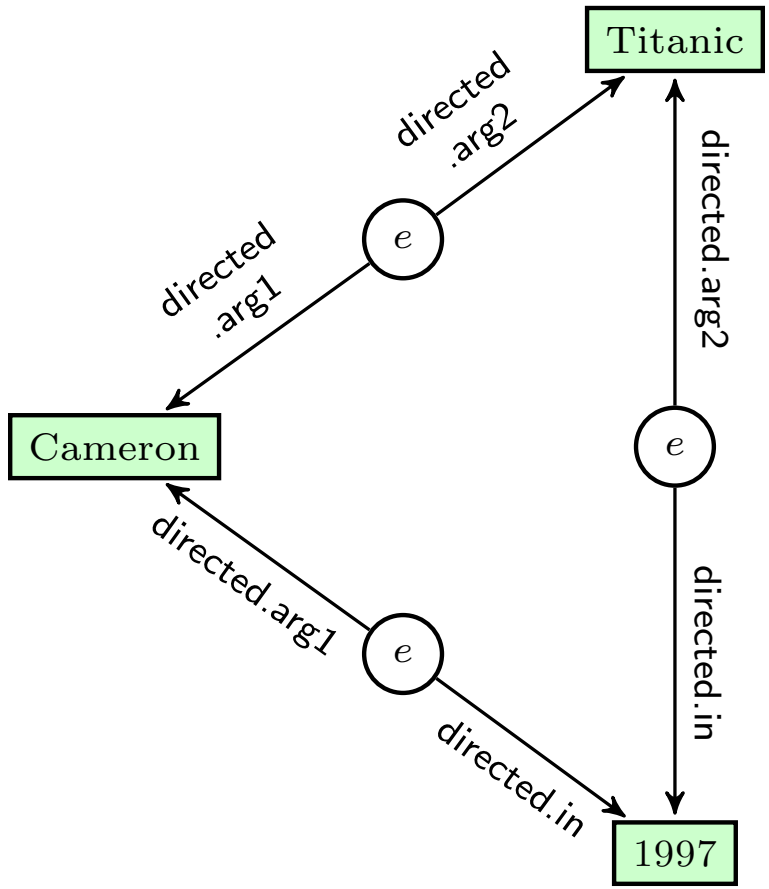
$$\lambda e.\text{directed.arg1}(\mathbf{e}, \texttt{Cameron}) \wedge \text{directed.arg2}(\mathbf{e}, \texttt{Titanic}) \wedge \text{directed.in}(\mathbf{e}, 1997)$$



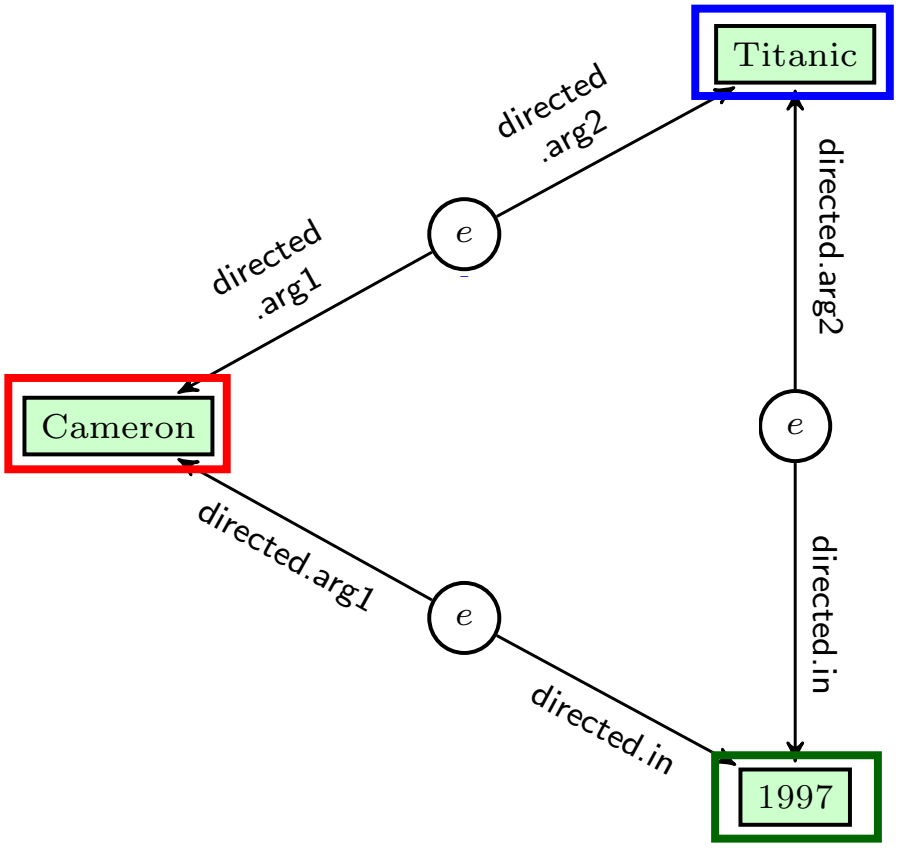
Logical Form to Ungrounded Graph

Cameron directed Titanic in 1997

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

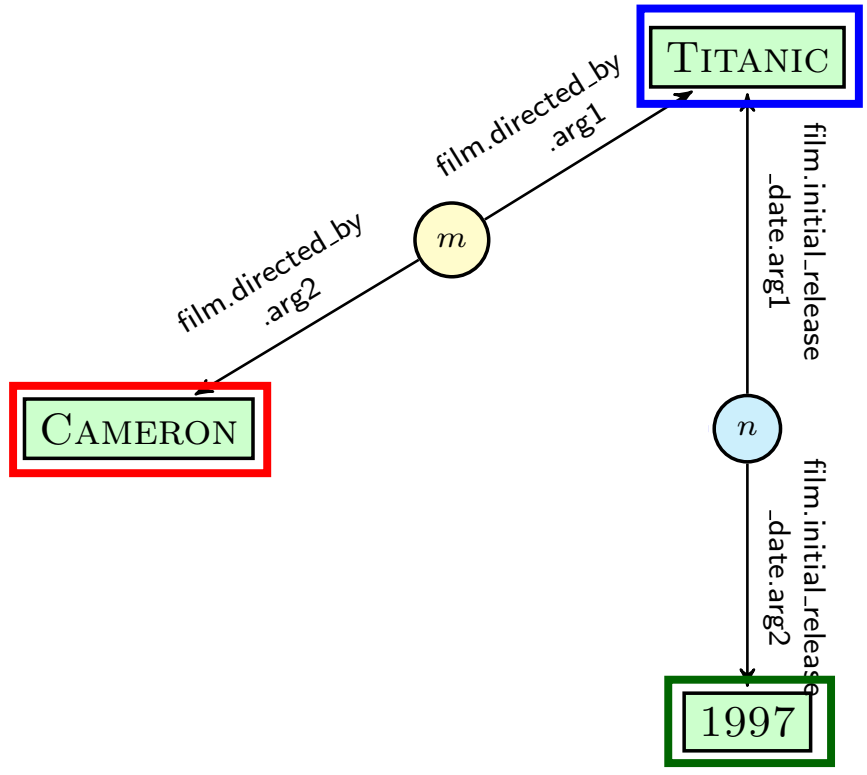


Graph Matching



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

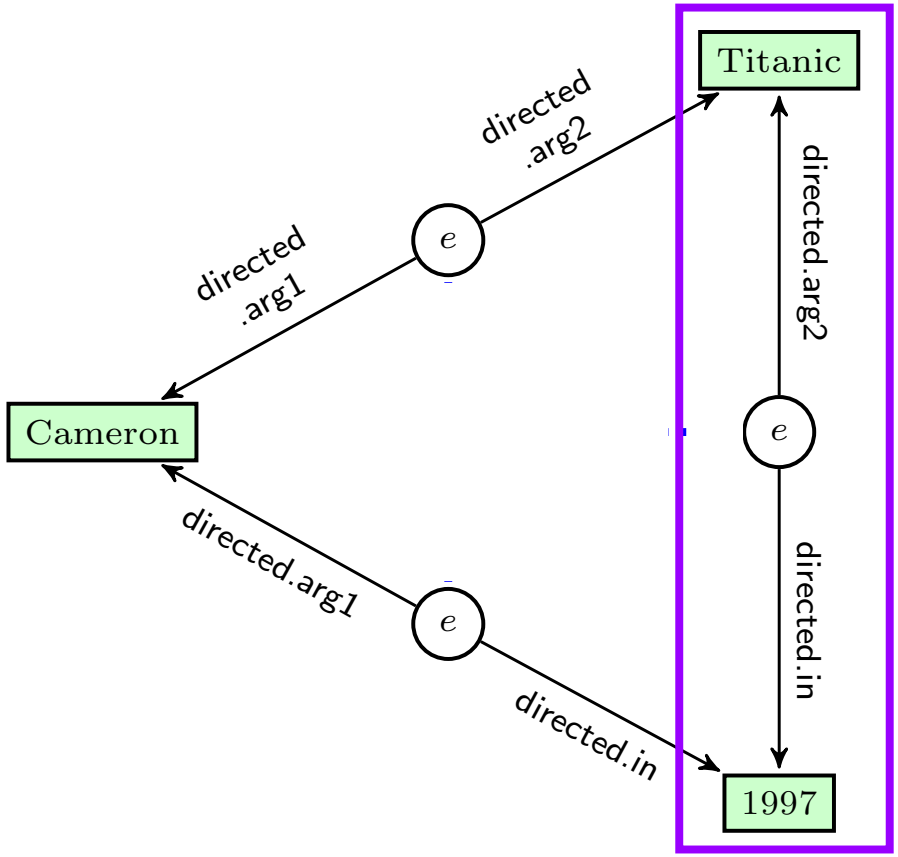
Ungrounded Graph



$\text{film.directed\_by.arg2}(m, \textsc{Cameron}) \wedge$
$\text{film.directed\_by.arg1}(m, \textsc{Titanic}) \wedge$
$\text{film.initial\_release\_date.arg1}(n, \textsc{Titanic}) \wedge$
$\text{film.initial\_release\_date.arg2}(n, 1997)$

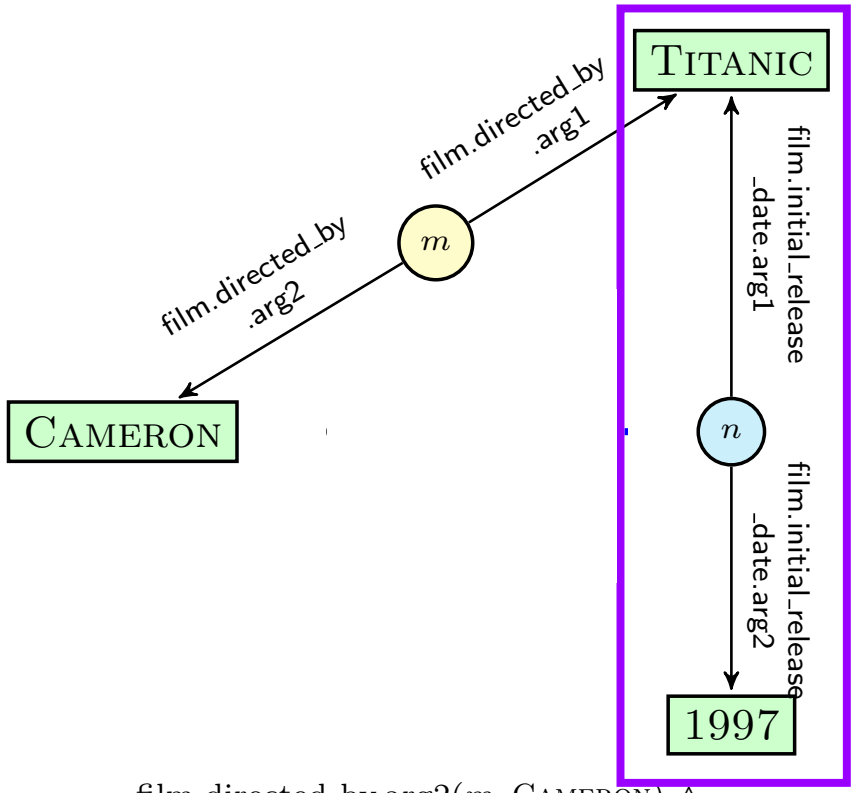
Grounded Graph

Graph Matching



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

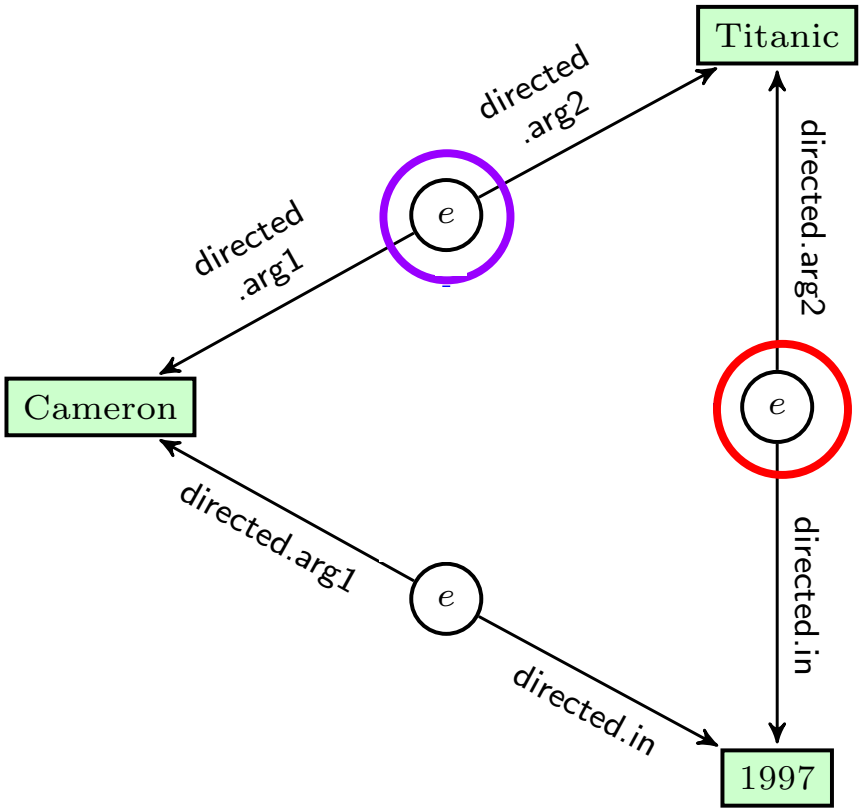
Ungrounded Graph



$\text{film.directed\_by.arg2}(m, \textsc{Cameron}) \wedge$
$\text{film.directed\_by.arg1}(m, \textsc{Titanic}) \wedge$
$\text{film.initial\_release\_date.arg1}(n, \textsc{Titanic}) \wedge$
$\text{film.initial\_release\_date.arg2}(n, 1997)$

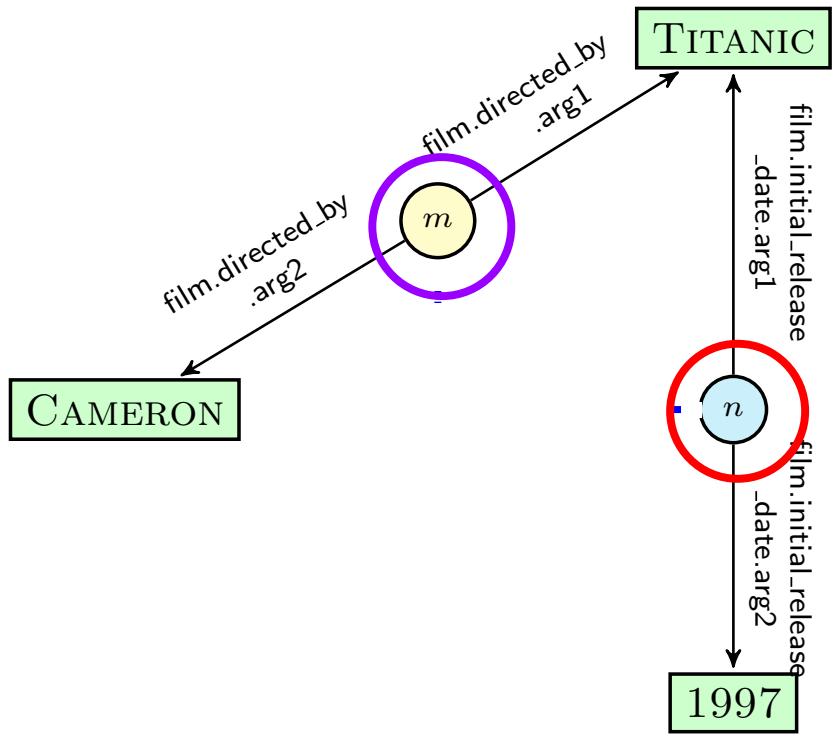
Grounded Graph

Graph Matching



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

Ungrounded Graph

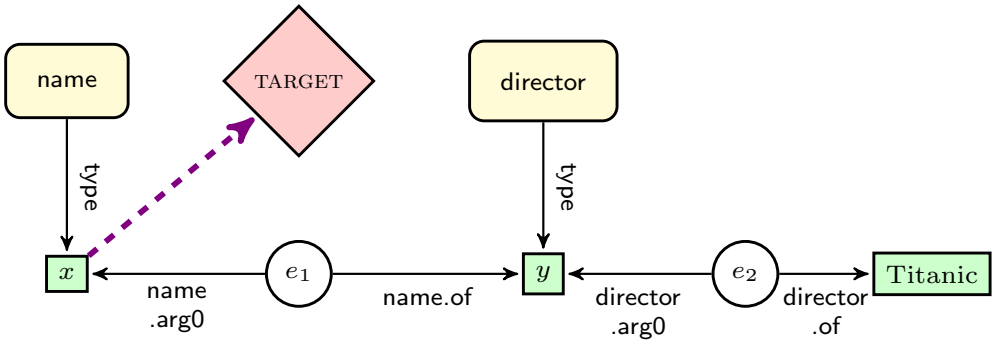


$\text{film.directed\_by.arg2}(m, \textsc{Cameron}) \wedge$
$\text{film.directed\_by.arg1}(m, \textsc{Titanic}) \wedge$
$\text{film.initial\_release\_date.arg1}(n, \textsc{Titanic}) \wedge$
$\text{film.initial\_release\_date.arg2}(n, 1997)$

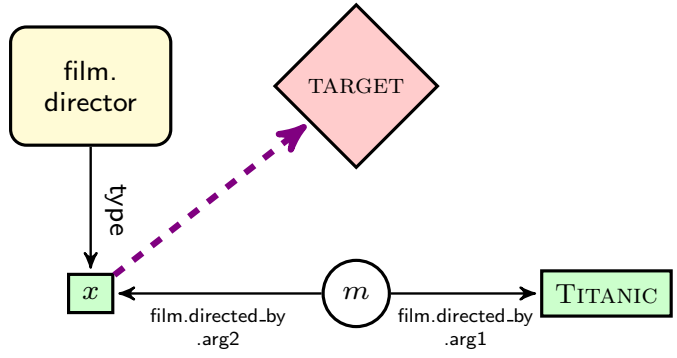
Grounded Graph

Graph Mismatch

What is the name of the director of Titanic?



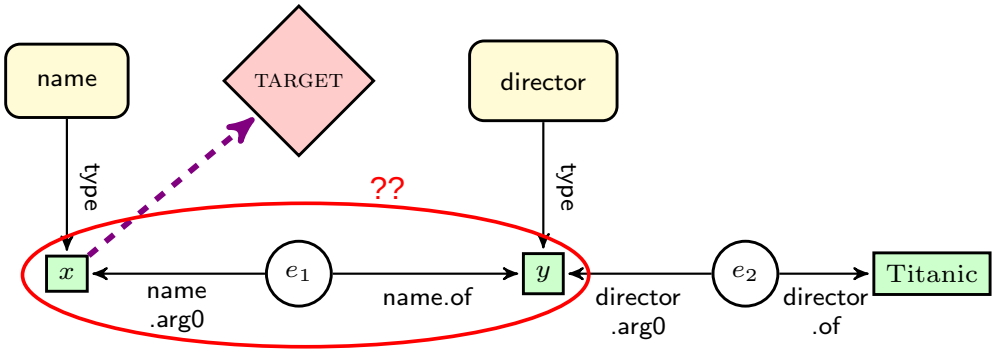
Ungrounded graph



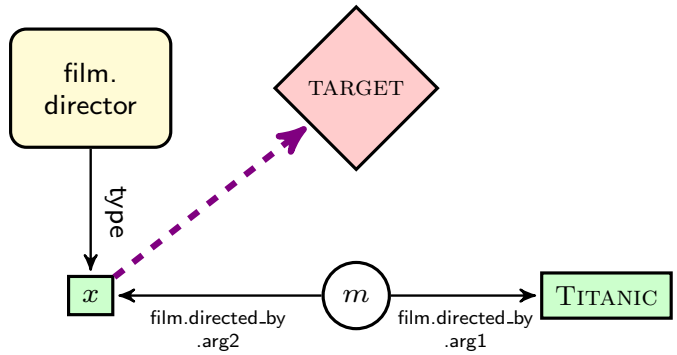
Grounded graph

Graph Mismatch

What is the name of the director of Titanic?



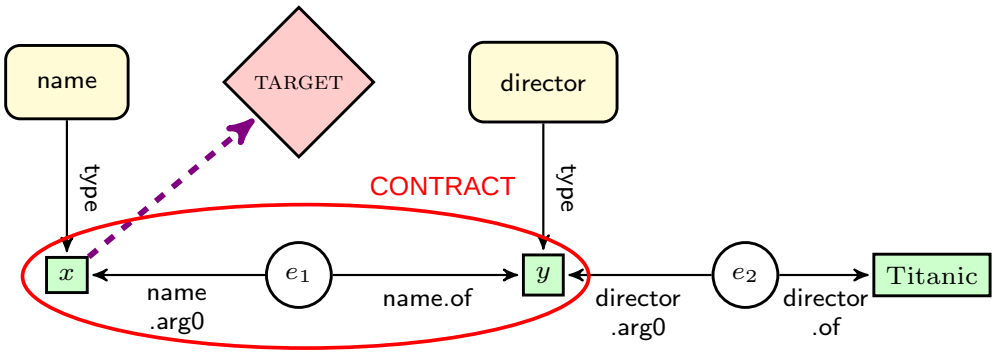
Ungrounded graph



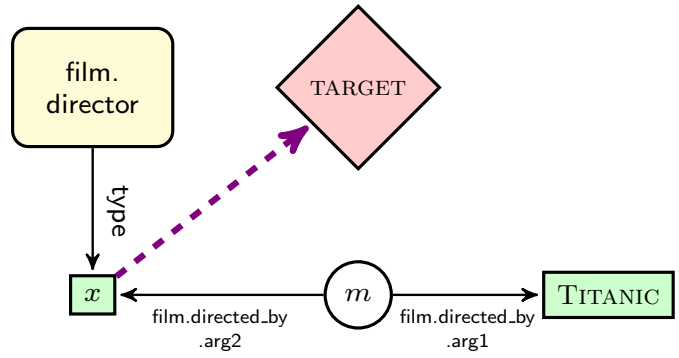
Grounded graph

Graph Mismatch

What is the name of the director of Titanic?



Ungrounded graph



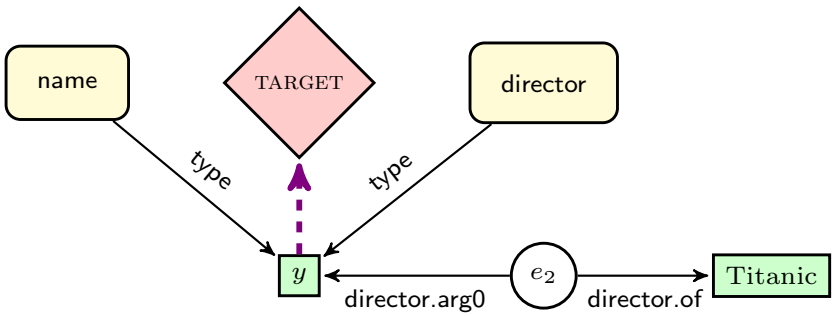
Grounded graph

- Paraphrasing is an alternative†

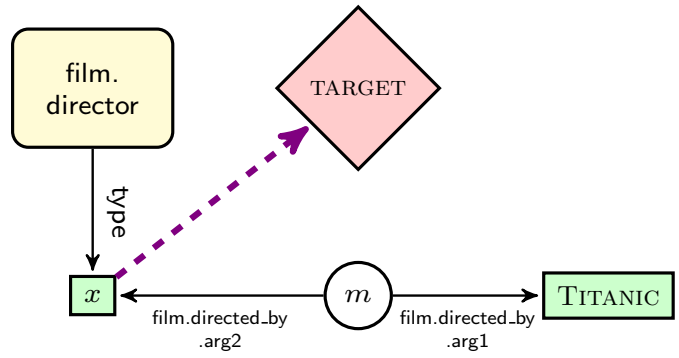
†Narayan, Reddy, Cohen (INLG 2016)

Graph Mismatch

What is the name of the director of Titanic?



Ungrounded graph



Grounded graph

Learning Model

Structured Perceptron: Ranks grounded and ungrounded graph pairs

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

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
- ★ Solution: use a surrogate gold graph

Learning Model

- ★ We do not have access to gold graphs
- ★ Access only to the answers rather than the query
- ★ Solution: 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

Experimental Setup: UD

69 lambda calculus formulae

WebQuestions in English, German, and Spanish

BiLSTM Parser [Kipperwiser and Goldberg (2016)]

- English: 81.73
- German: 79.13
- Spanish: 85.76

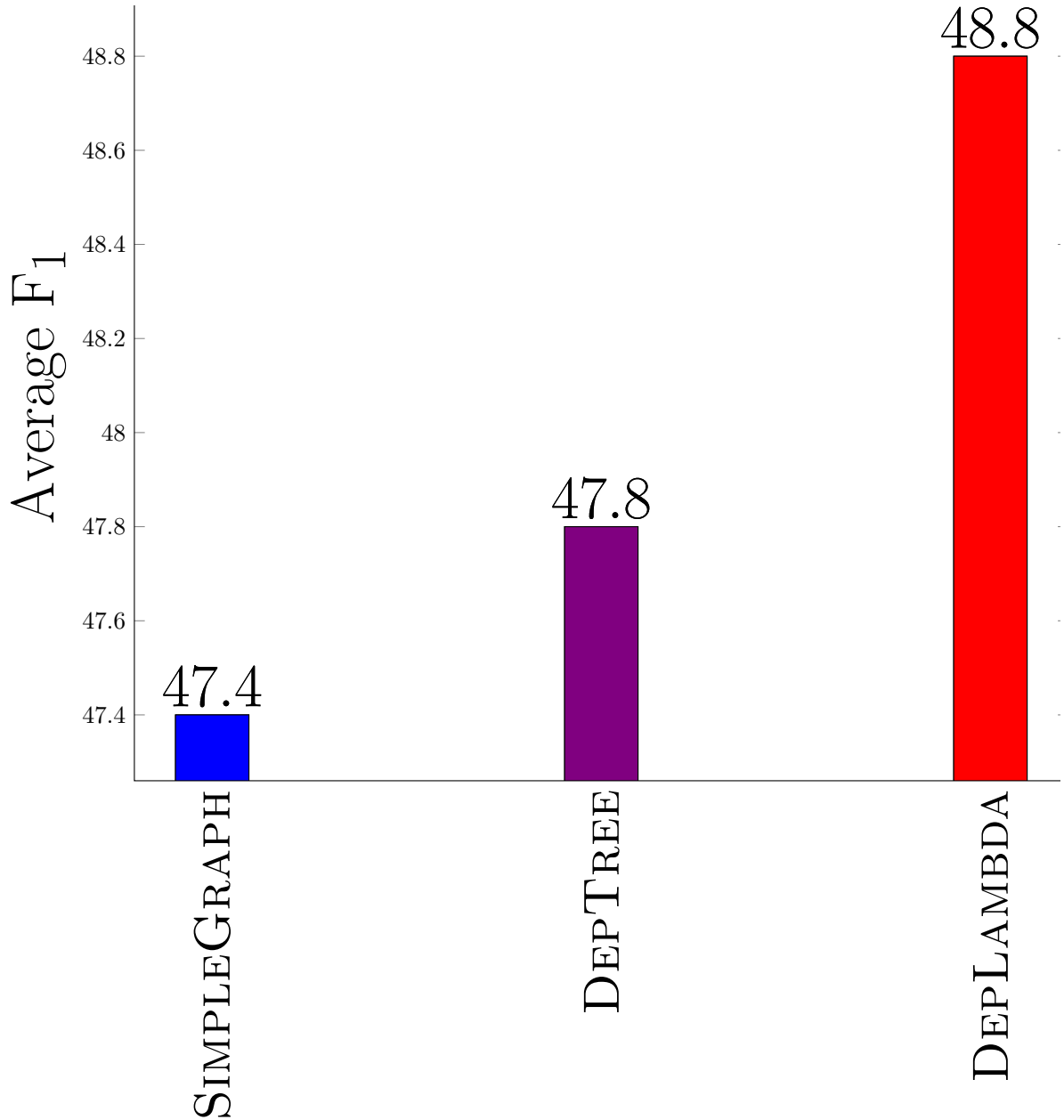
Baselines

SIMPLEGRAPH : All entities connected to a single event
bag of words

DEPTREE: Transduce a dependency tree to target graph

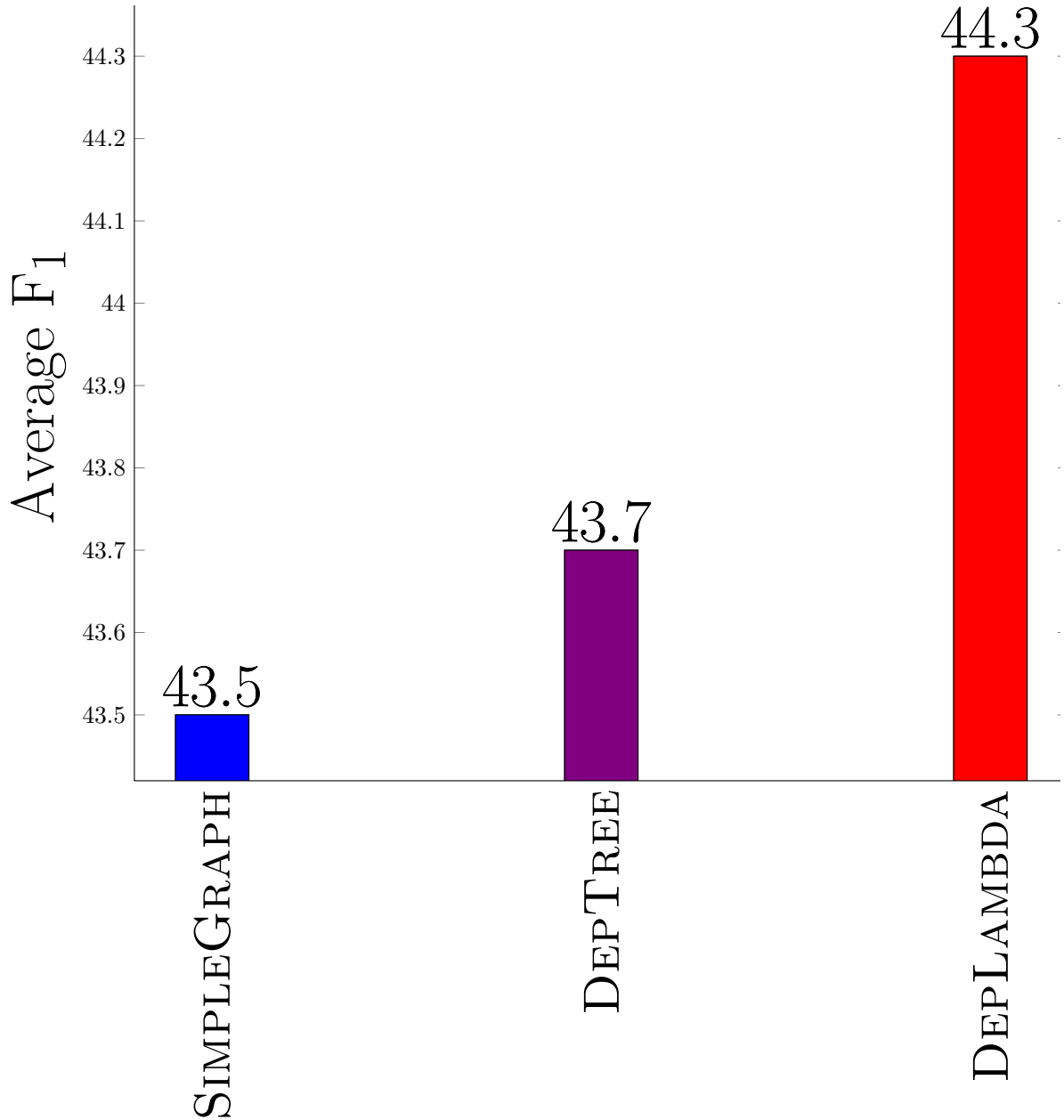
Results on Multilingual WebQuestions

English



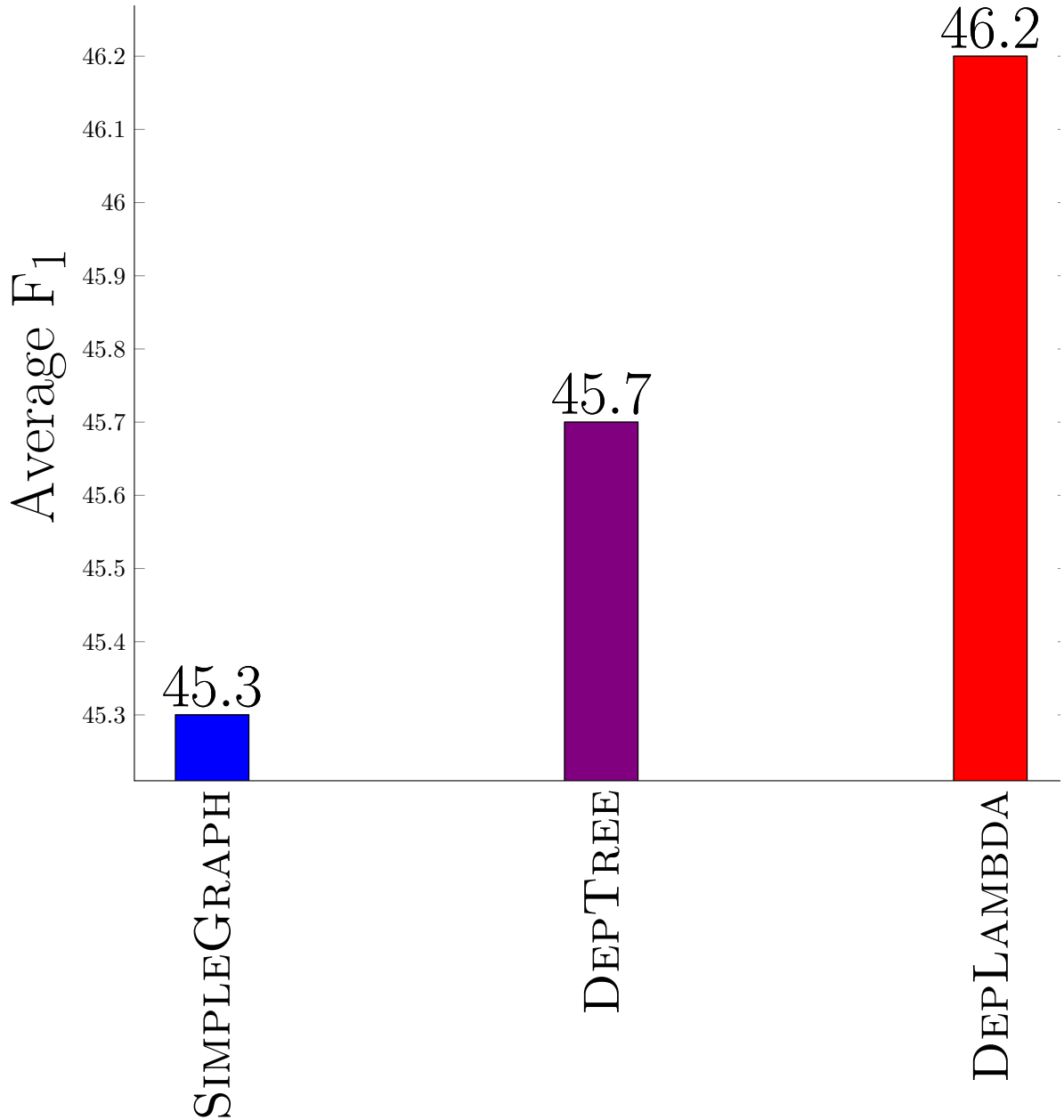
Results on Multilingual WebQuestions

German



Results on Multilingual WebQuestions

Spanish



Experimental Setup: Stanford Dependencies

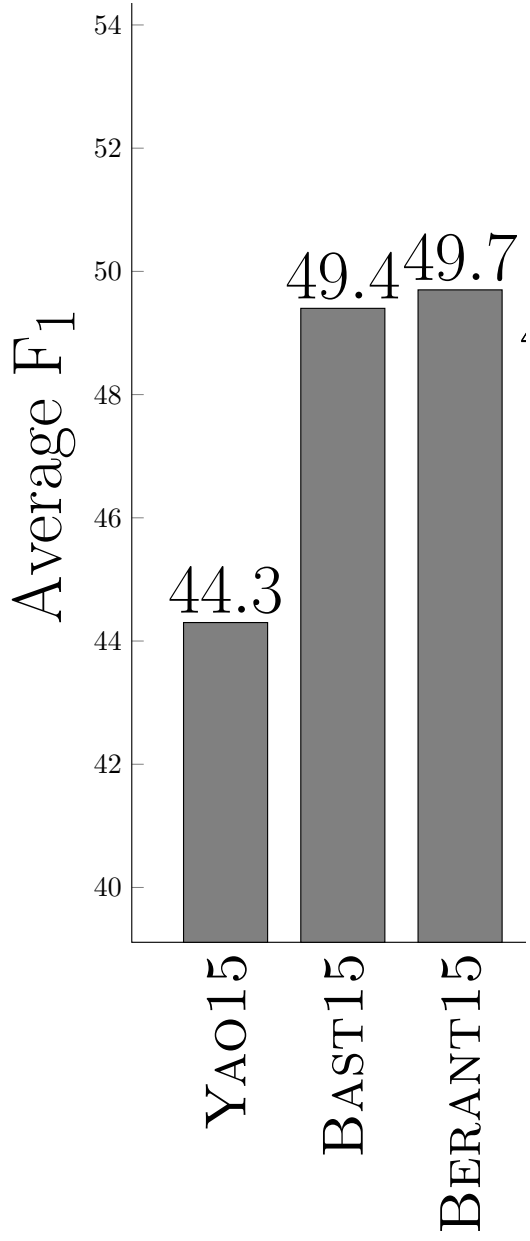
78 lambda calculus formulae

Hypergraph parser [Zhang & McDonald (2014)]

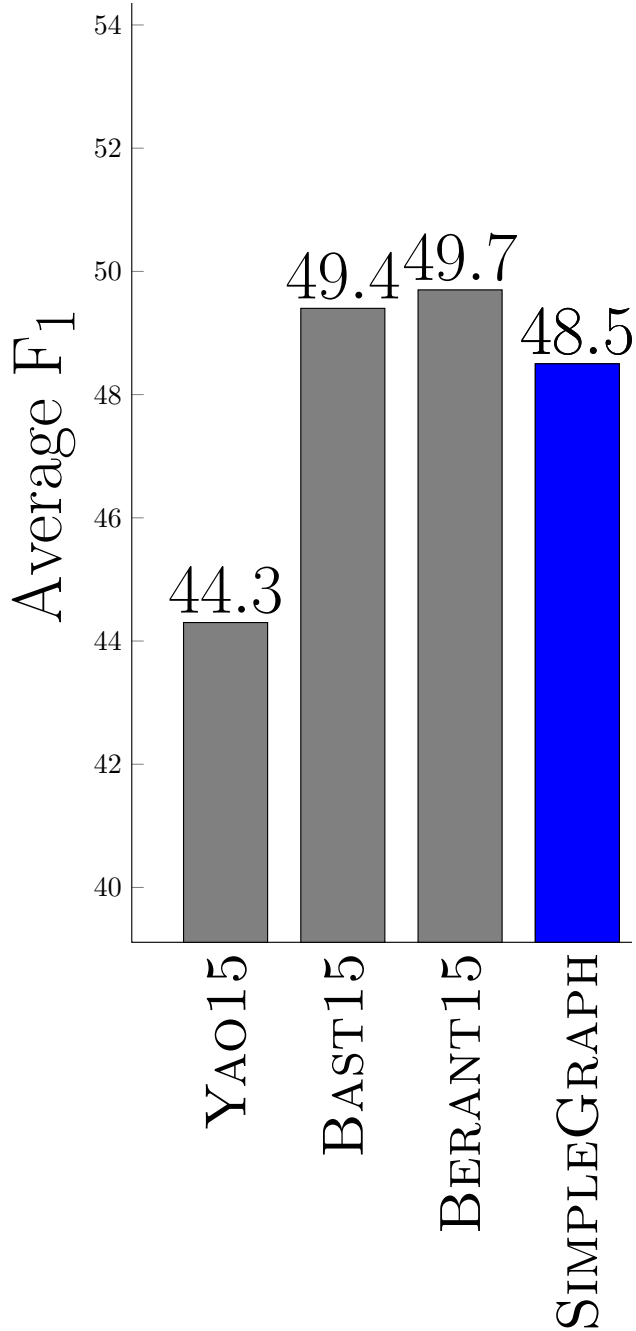
- English: 90.64

Additional Baseline: CCGGRAPH – CCG logical forms

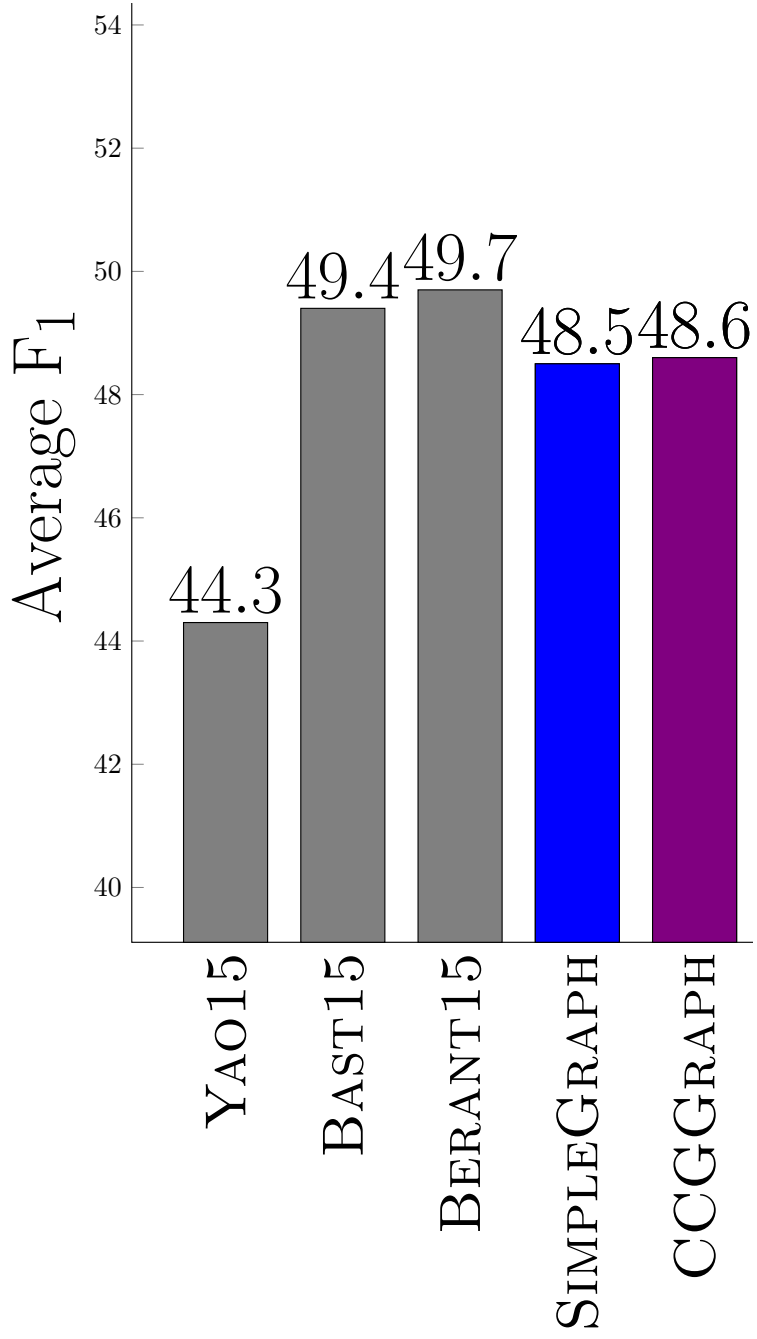
Results on WebQuestions



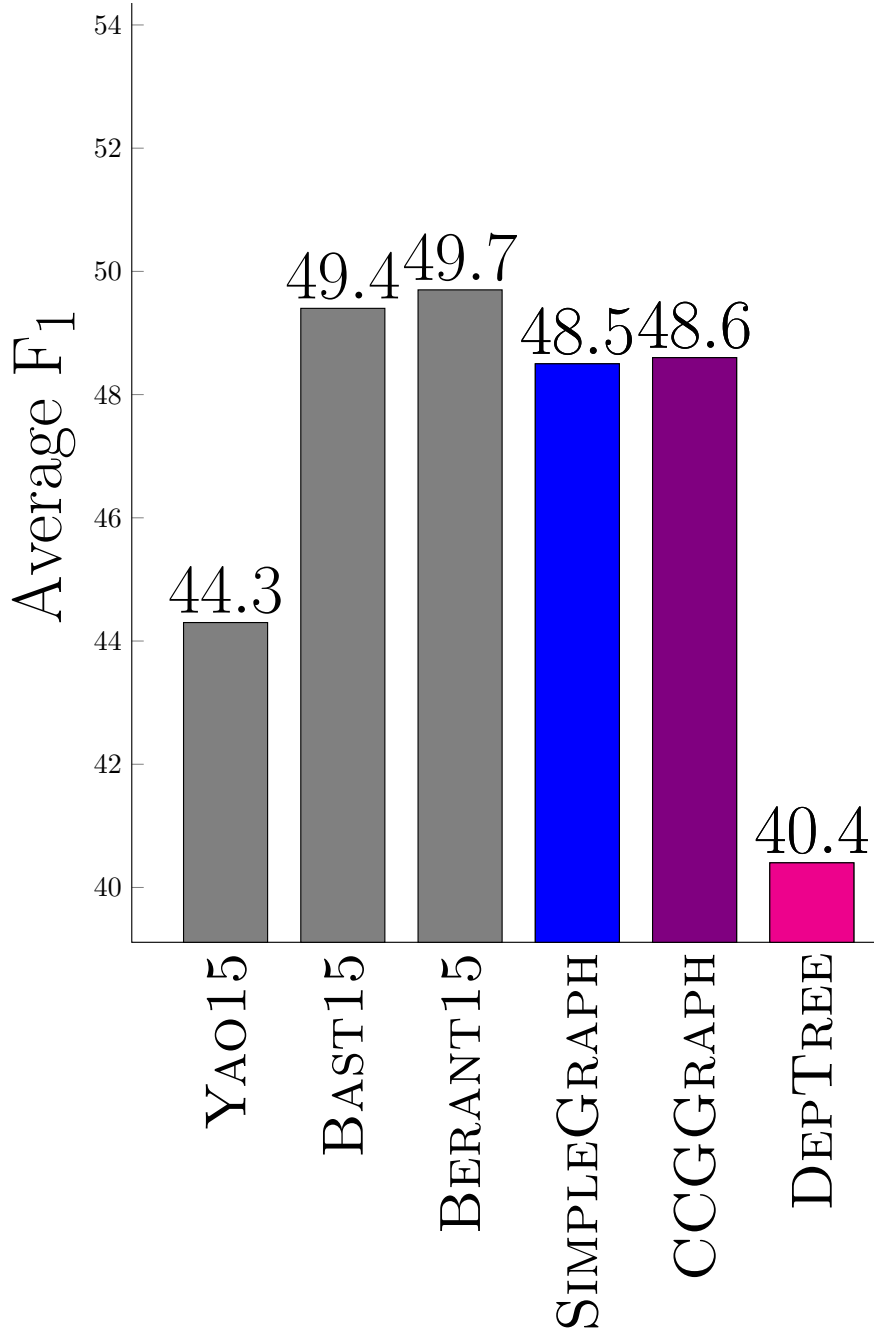
Results on WebQuestions



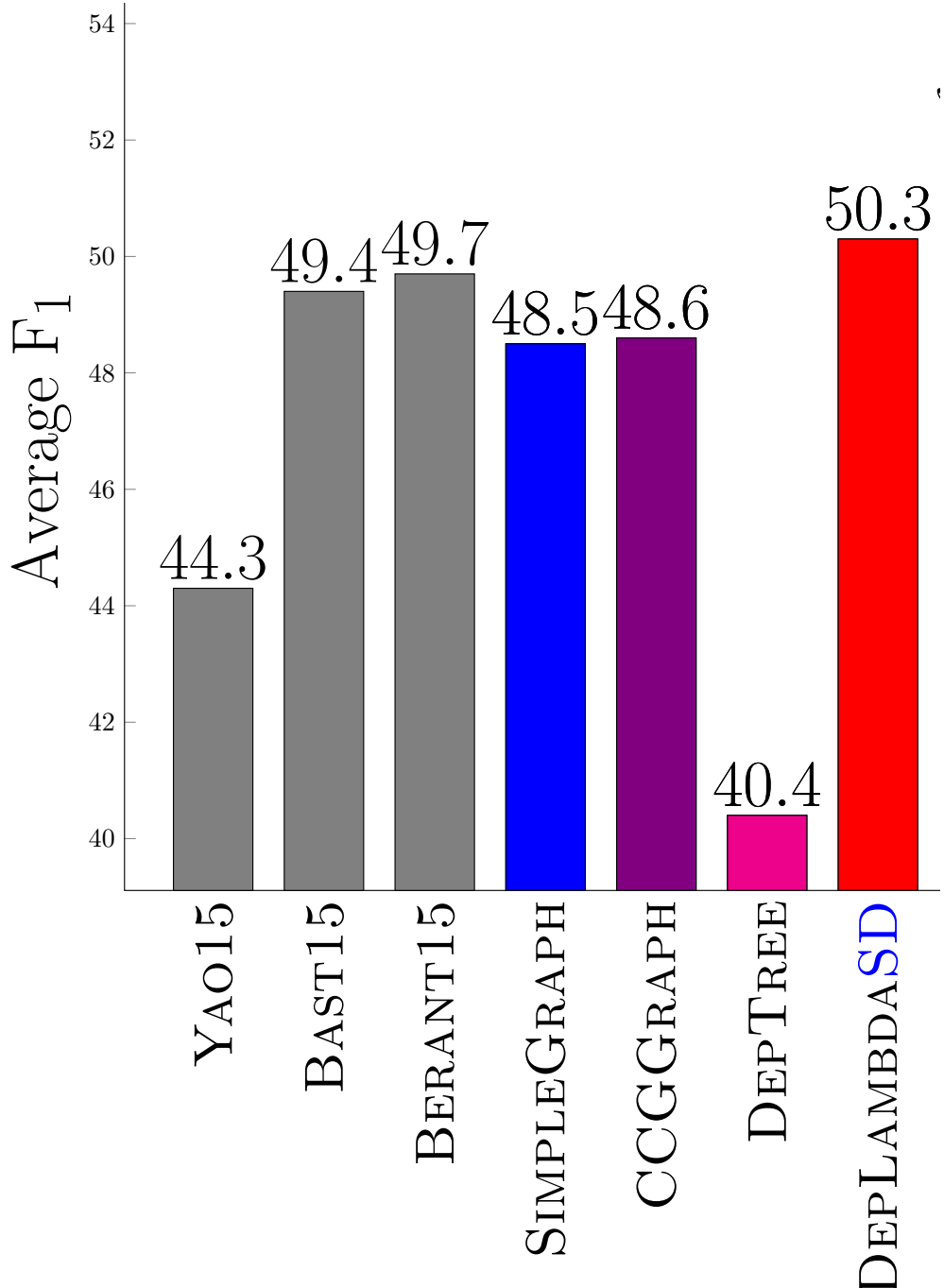
Results on WebQuestions



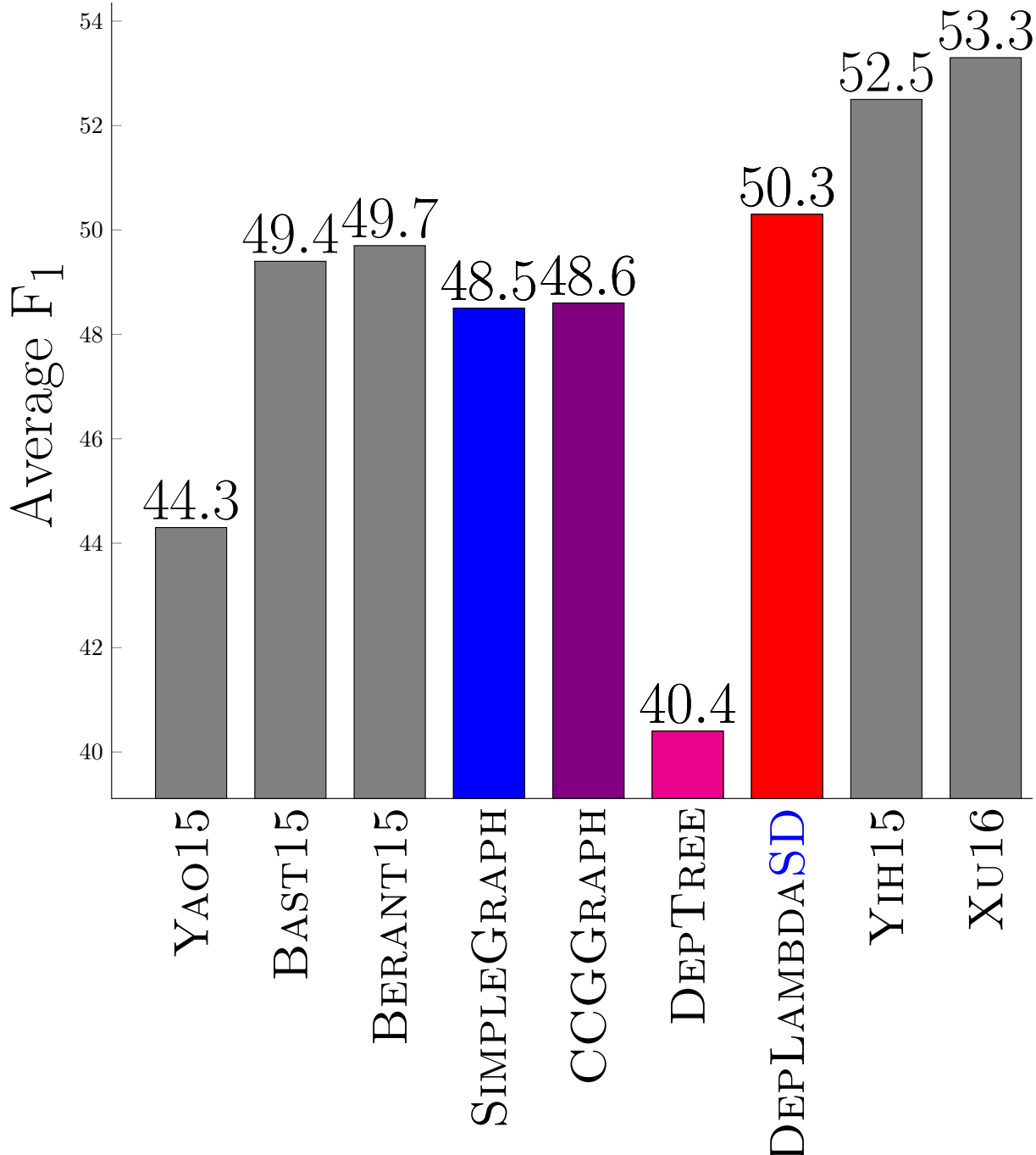
Results on WebQuestions



Results on WebQuestions



Results on WebQuestions



Error Analysis

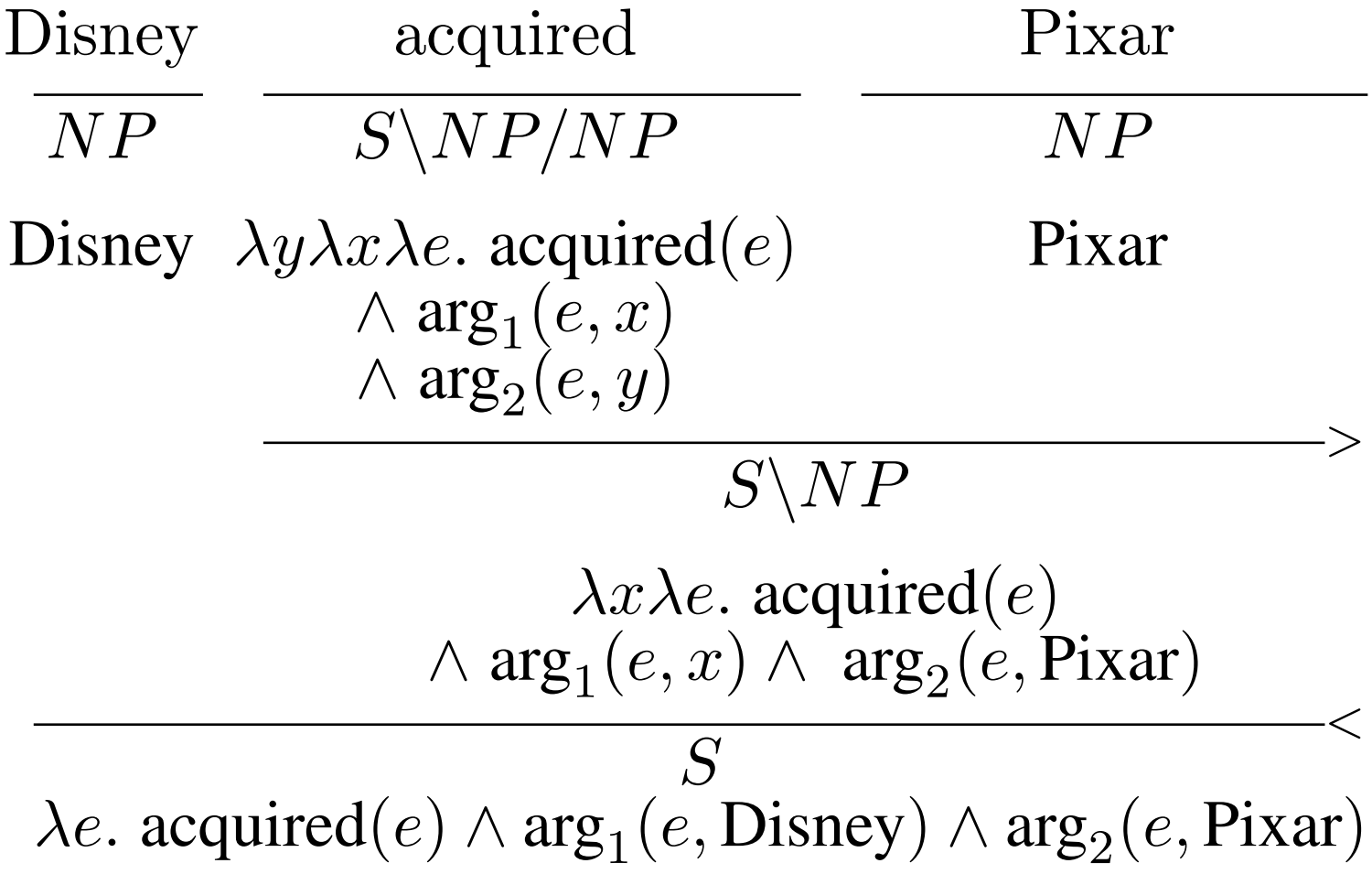
CCGGRAPH fails to produce logical forms for 4.5%.

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

DepLambda fails only for 0.9%

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

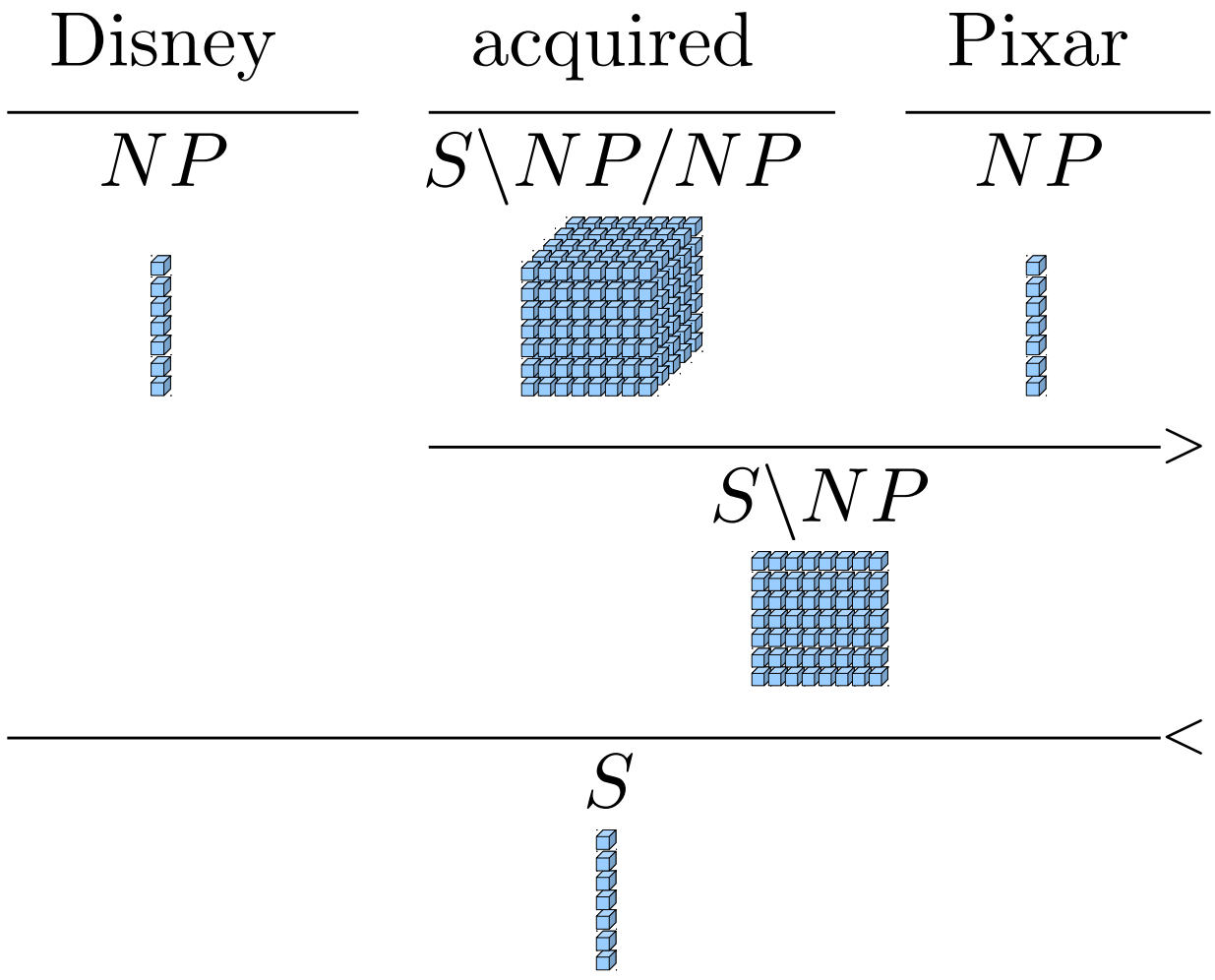
Comparison with CCG



Comparison with CCG

CCG	DepLambda
Lexicalized semantics	Simple lexical semantics
S\NP/NP : $\lambda y \lambda x \lambda e.\text{acquired}(e) \wedge \text{arg}_1(e,x) \wedge \text{arg}_2(e,y)$	$\lambda x.\text{acquired}(x_e)$
Words drive composition	Dependencies drive composition
Language specific types	Mostly universal

Comparison with CCG



DepLambda: Present

Compositional Typed Semantics using Lambda Calculus

DepLambda: Towards ...

Compositional Typed Semantics using Lambda Calculus

- Richer composition functions e.g. neural networks
- Pipelined vs. Synchronous Syntax-Semantics

DepLambda: Present

Compositional Typed Semantics using Lambda Calculus

- Richer composition functions e.g. neural networks
- Pipelined vs. Synchronous Syntax-Semantics

Two universal types

DepLambda: Towards ...

Compositional Typed Semantics using Lambda Calculus


Richer composition functions e.g. neural networks

Pipelined vs. Synchronous Syntax-Semantics

Two universal types


Richer universal types

DepLambda: Present

Compositional Typed Semantics using Lambda Calculus


Richer composition functions e.g. neural networks

Pipelined vs. Synchronous Syntax-Semantics

Two universal types


Richer universal types

Output: General-purpose logical forms

DepLambda: Towards ...

Compositional Typed Semantics using Lambda Calculus


Richer composition functions e.g. neural networks

Pipelined vs. Synchronous Syntax-Semantics

Two universal types


Richer universal types

Output: General-purpose logical forms


Any target representation

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

- Xu, Reddy, Feng, Huang, Zhao (ACL 2016)
- Bisk, Reddy, Blitzer, Hockenmaier (EMNLP 2016)
- Reddy, Lapata, Steedman (TACL 2014)

Summary

Compositional Typed Semantic interface for Dependencies

Dependencies to Logical Forms

Freebase Semantic Parsing using Logical Forms

Demo at

https://sivareddy.in/deplambda.html

Thank You!

Semantic Parsing without QA pairs

Question

Who is the director of Titanic?

Logical Form

$\lambda x.\ \text{film.directed\_by}(\textit{Titanic}, x)$

Latent

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)

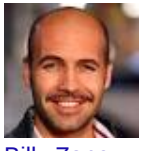
Director: James Cameron

Featured song: My Heart Will Go On

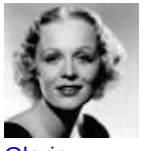
Cast

Leonardo DiCaprio
Jack Dawson

Kate Winslet
Rose DeWitt Bukater



Billy Zane
Caledon Hockley



Gloria Stuart
Rose DeWitt Bukater

Kathy Bates
Molly Brown

Semantic Parsing without QA pairs

Question

Who is the director of Titanic?

Logical Form

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

Latent

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
Rose DeWitt Bukater

Kathy Bates
Molly Brown

Semantic Parsing without QA pairs

NL Leonardo DiCaprio starred as Jack in Titanic which was directed by James Cameron.

Parser

KB $\text{cast}(\text{TITANIC}, \text{DICAPRIO}, \text{JACK}) \wedge \text{director}(\text{TITANIC}, \text{CAMERON})$

Execute

TRUE

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
Rose DeWitt Bukater


Kathy Bates
Molly Brown

Task Setup

Training Data: Web Corpus

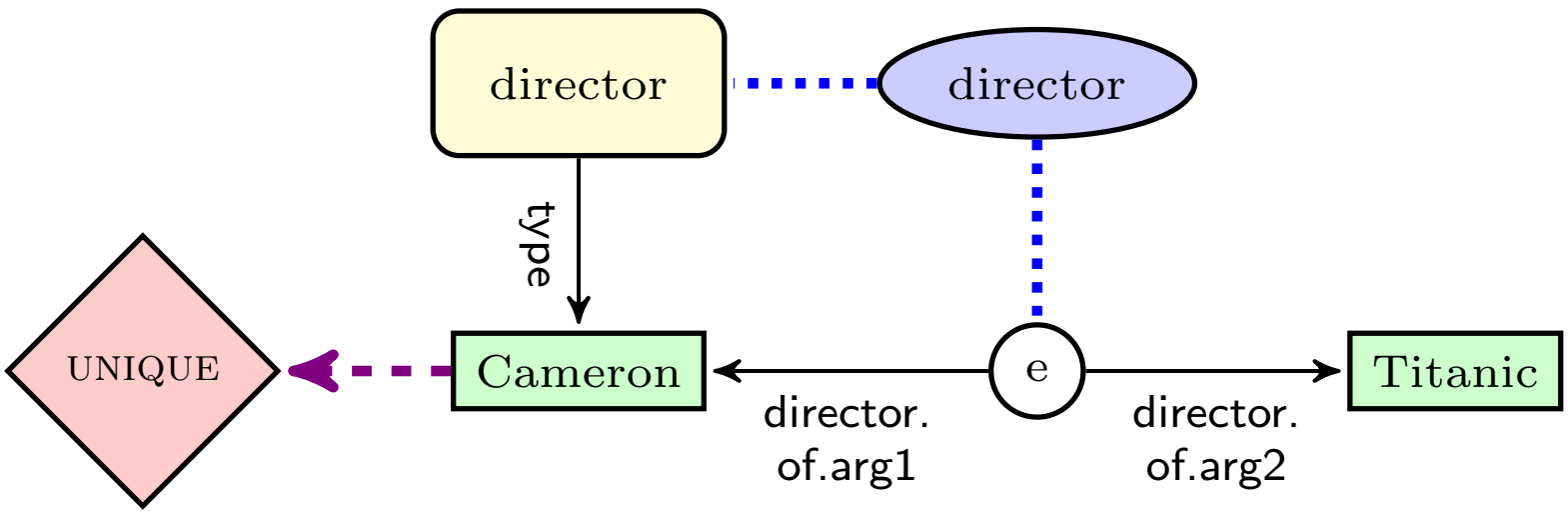
Evaluation: Question Answering on Freebase

Resources: Ungrounded Logical Forms

Hypothesis: We do not need manual question answer pairs

Learning from Text

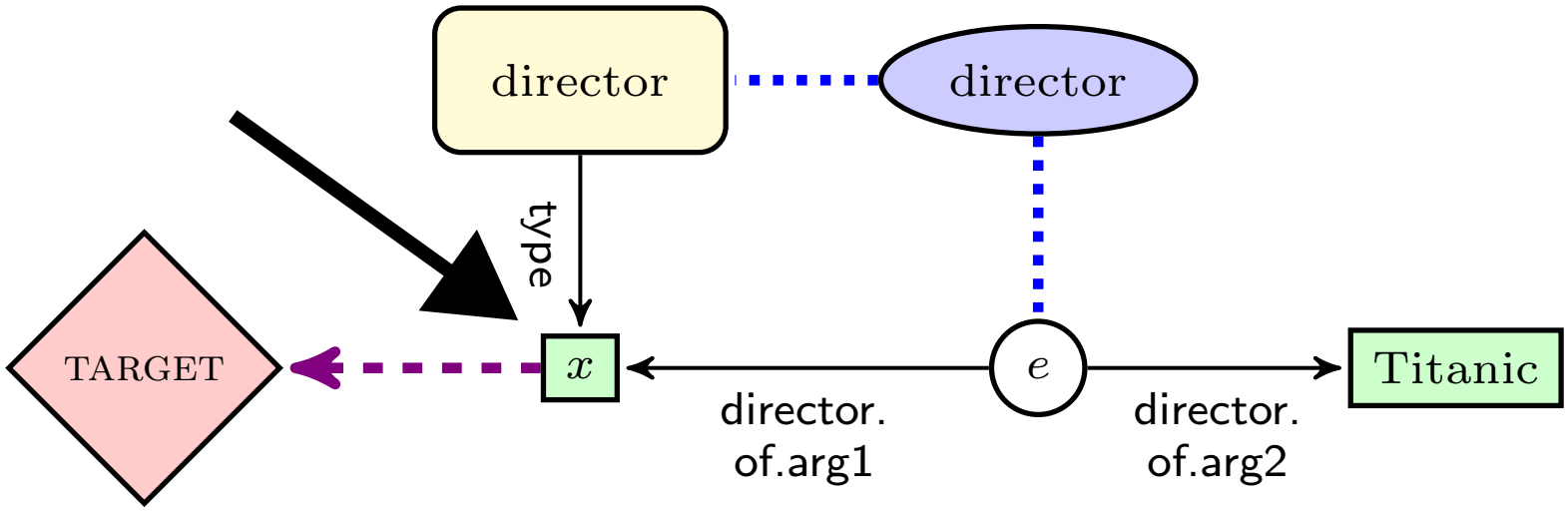
Cameron is the director of Titanic.



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

Learning from Text

Cameron is the director of Titanic.

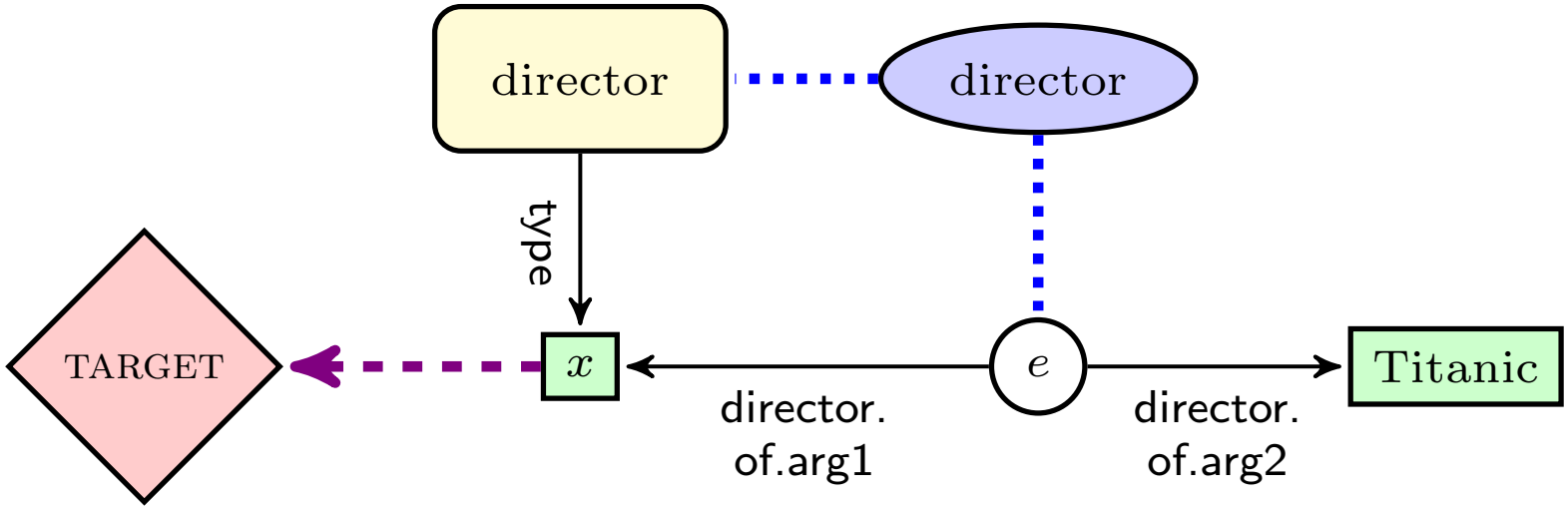


$$\text{TARGET}(x) \wedge \text{director}(x) \wedge \text{director.of.arg1}(e, x) \wedge \text{director.of.arg2}(e, \text{Titanic})$$

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

Learning from Text

Who is the director of Titanic?

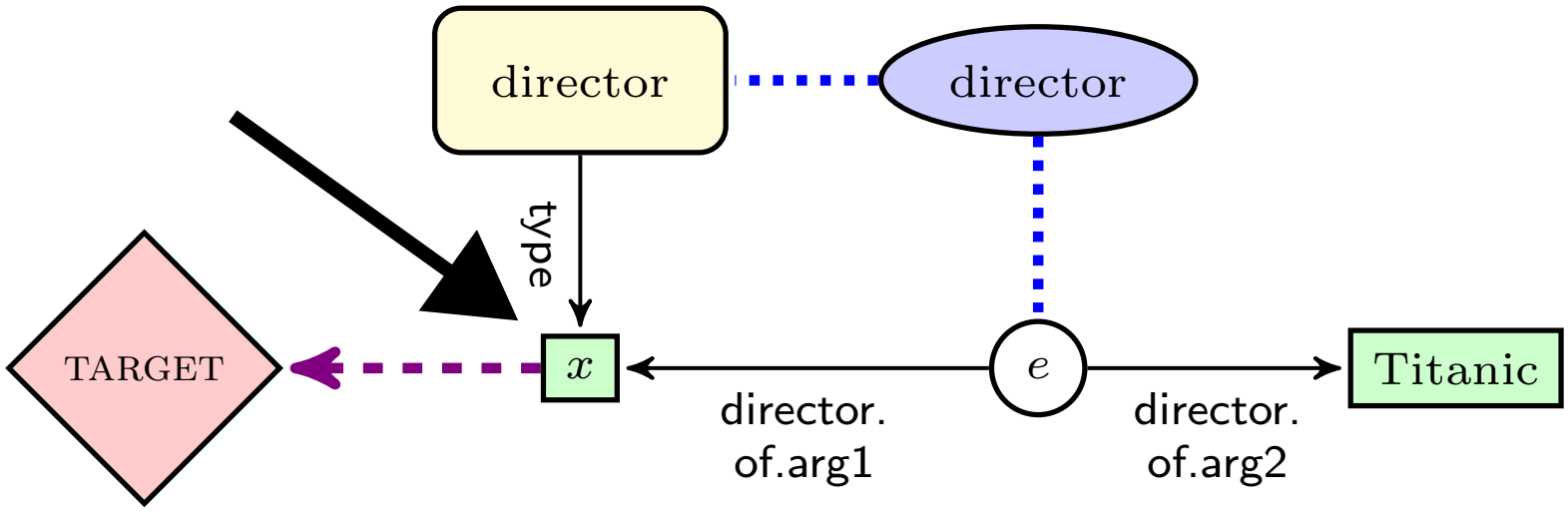


$$\text{TARGET}(x) \wedge \text{director}(x) \wedge \text{director.of.arg1}(e, x) \wedge \text{director.of.arg2}(e, \text{Titanic})$$

Build all knowledge base subgraphs with x uninstantiated.

Learning from Text

Cameron is the director of Titanic.



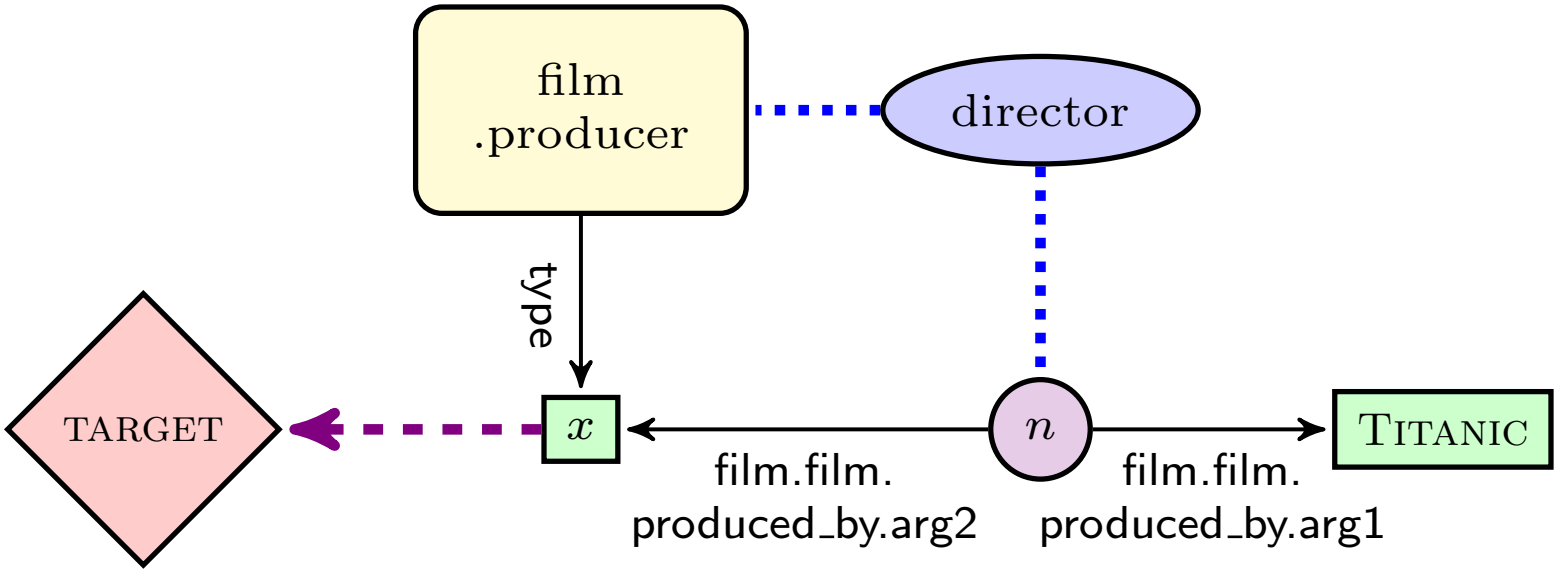
$$\text{TARGET}(x) \wedge \text{director}(x) \wedge \text{director.of.arg1}(e, x) \wedge \text{director.of.arg2}(e, \text{Titanic})$$

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

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

Learning from Text

Who is the director of Titanic?



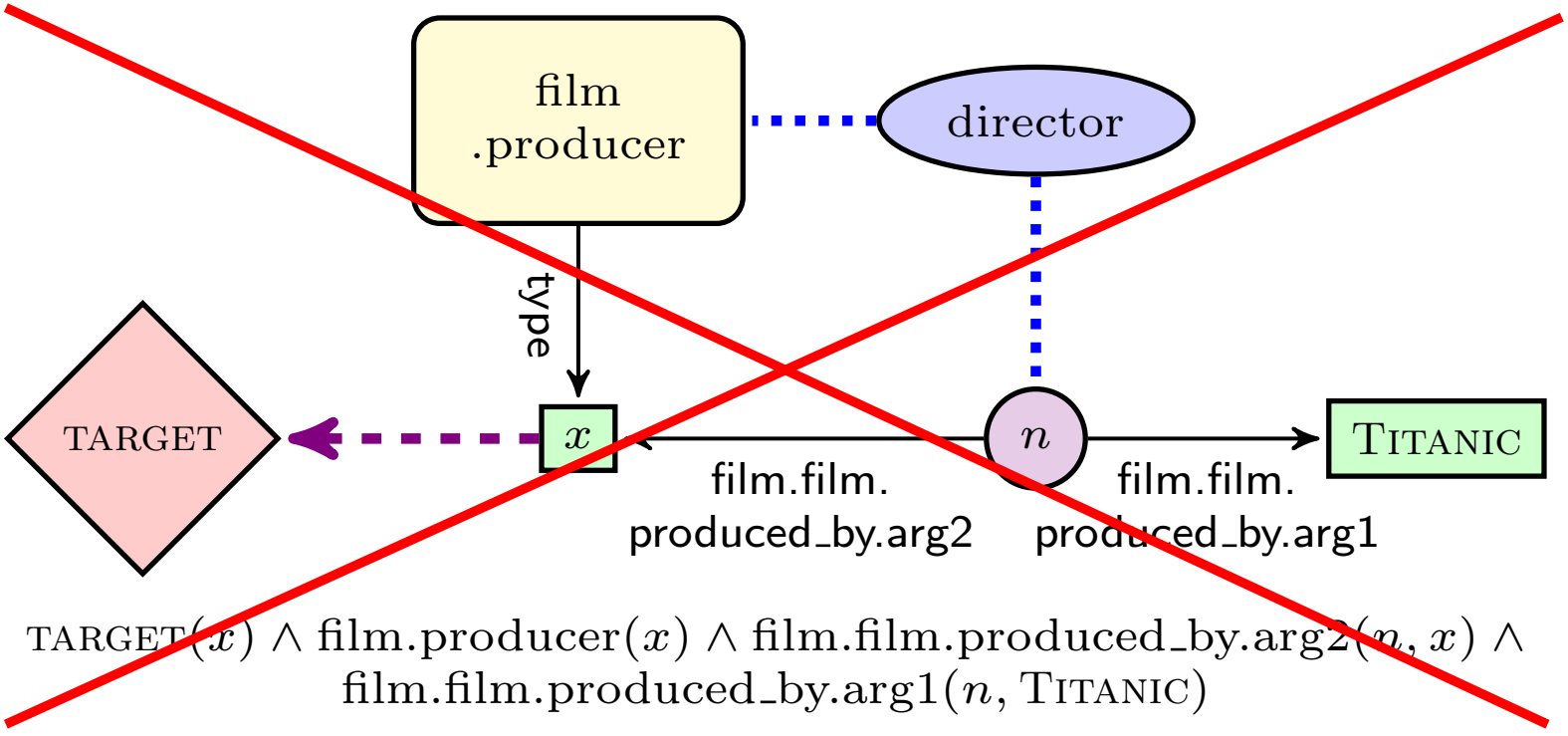
$$\textsc{target}(x) \wedge \text{film.producer}(x) \wedge \text{film.film.produced\_by.arg2}(n, x) \wedge \text{film.film.produced\_by.arg1}(n, \textsc{Titanic})$$

$$[\![x]\!]_{KB} = \{\textsc{Cameron}, \textsc{JonLandau}\}$$

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

Learning from Text

Who is the director of Titanic?



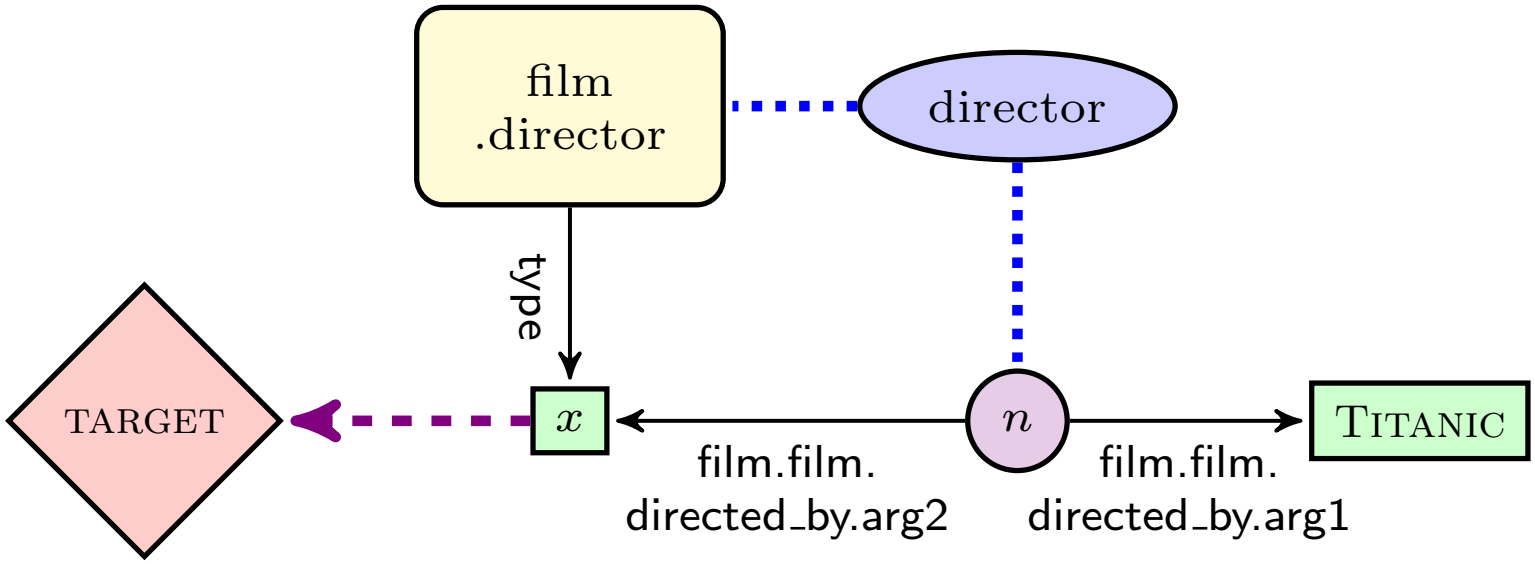
$\text{TARGET}(x) \wedge \text{film.producer}(x) \wedge \text{film.film.produced\_by.arg2}(n, x) \wedge \text{film.film.produced\_by.arg1}(n, \text{TITANIC})$

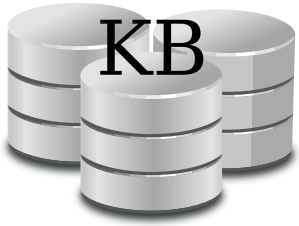
$[\![x]\!]_{KB} = \{\text{CAMERON}, \text{JONLANDAU}\} \neq [\![x]\!]_{NL}$

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

Learning from Text

Who is the director of Titanic?





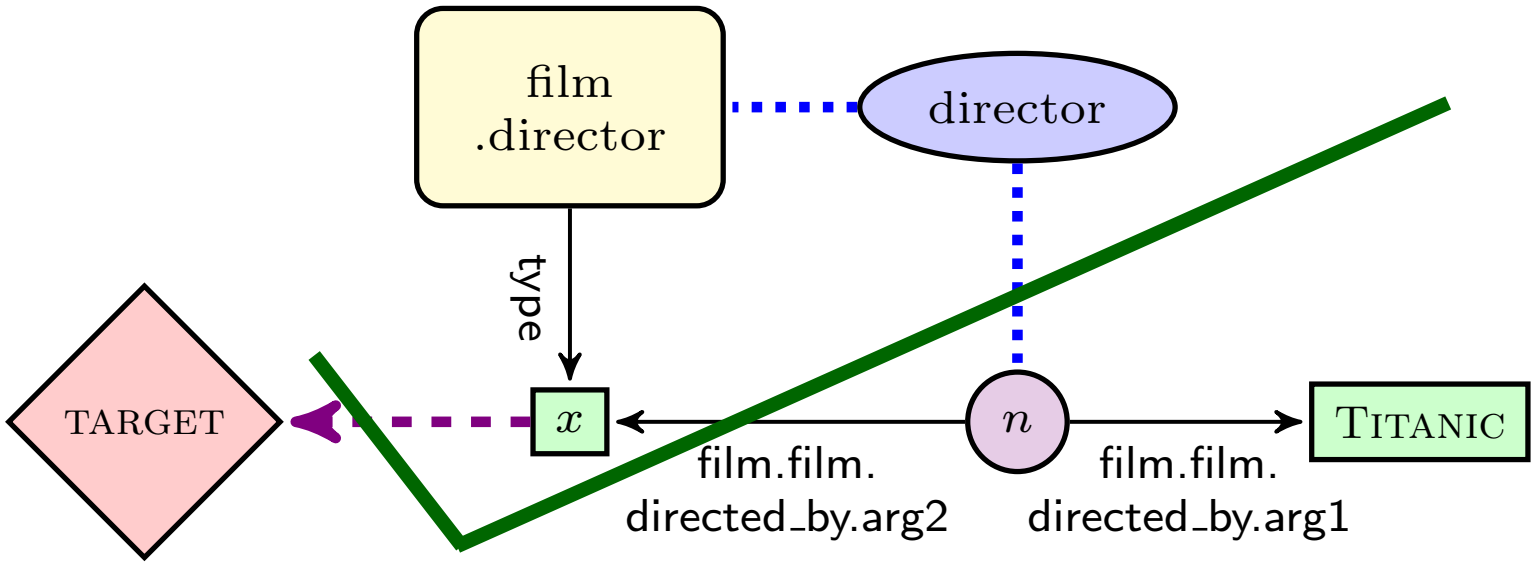
$$\text{TARGET}(x) \wedge \text{film.director}(x) \wedge \text{film.film.directed\_by.arg2}(n, x) \wedge \text{film.film.directed\_by.arg1}(n, \text{TITANIC})$$

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

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

Learning from Text

Who is the director of Titanic?



KB

$$\text{TARGET}(x) \wedge \text{film.director}(x) \wedge \text{film.film.directed\_by.arg2}(n, x) \wedge \text{film.film.directed\_by.arg1}(n, \text{TITANIC})$$

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

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.

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	$AvgF1$ (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
  - Manually annotated QA pairs (1115 pairs)
  - Paraphrases and Web Corpus

This Talk

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

QA without logical forms

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

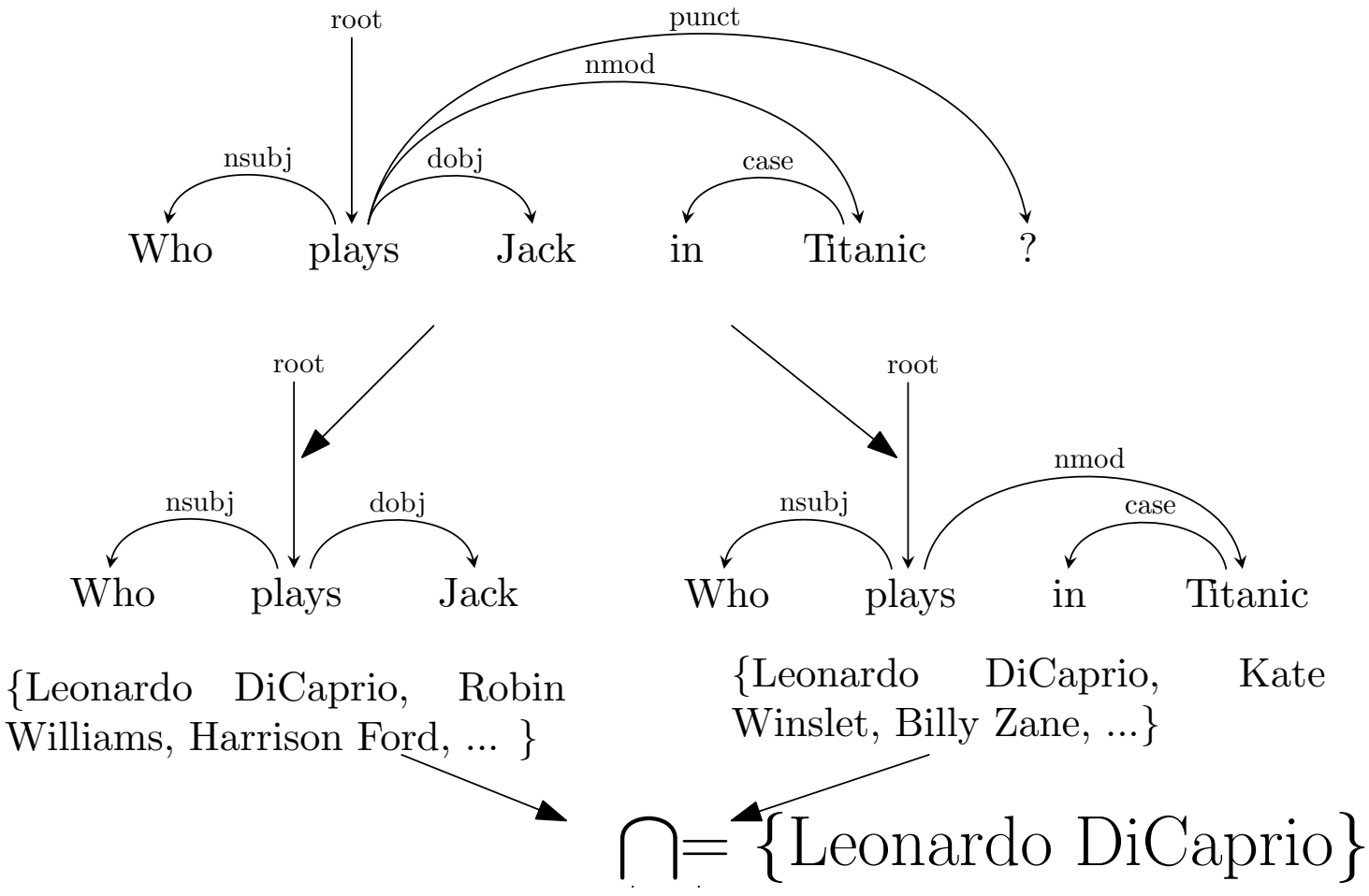
Task Setting

Training Data: Question and Answer Pairs, Web Corpus

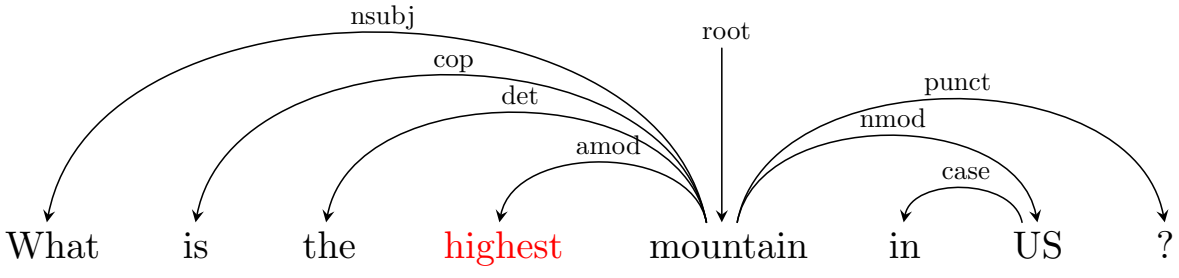
Evaluation: Question Answering on Freebase

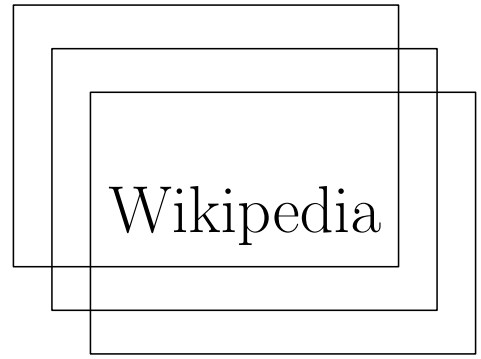
Resources: Dependency Parser

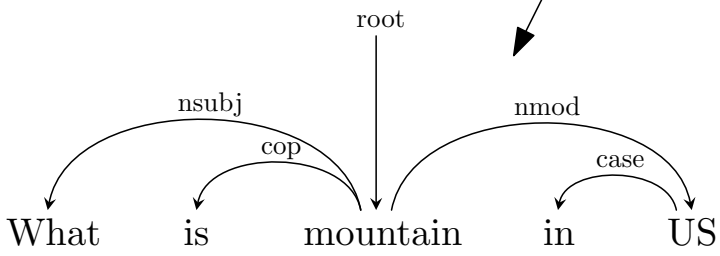
QA with Relation Extraction and Text Evidence



QA with Relation Extraction and Text Evidence







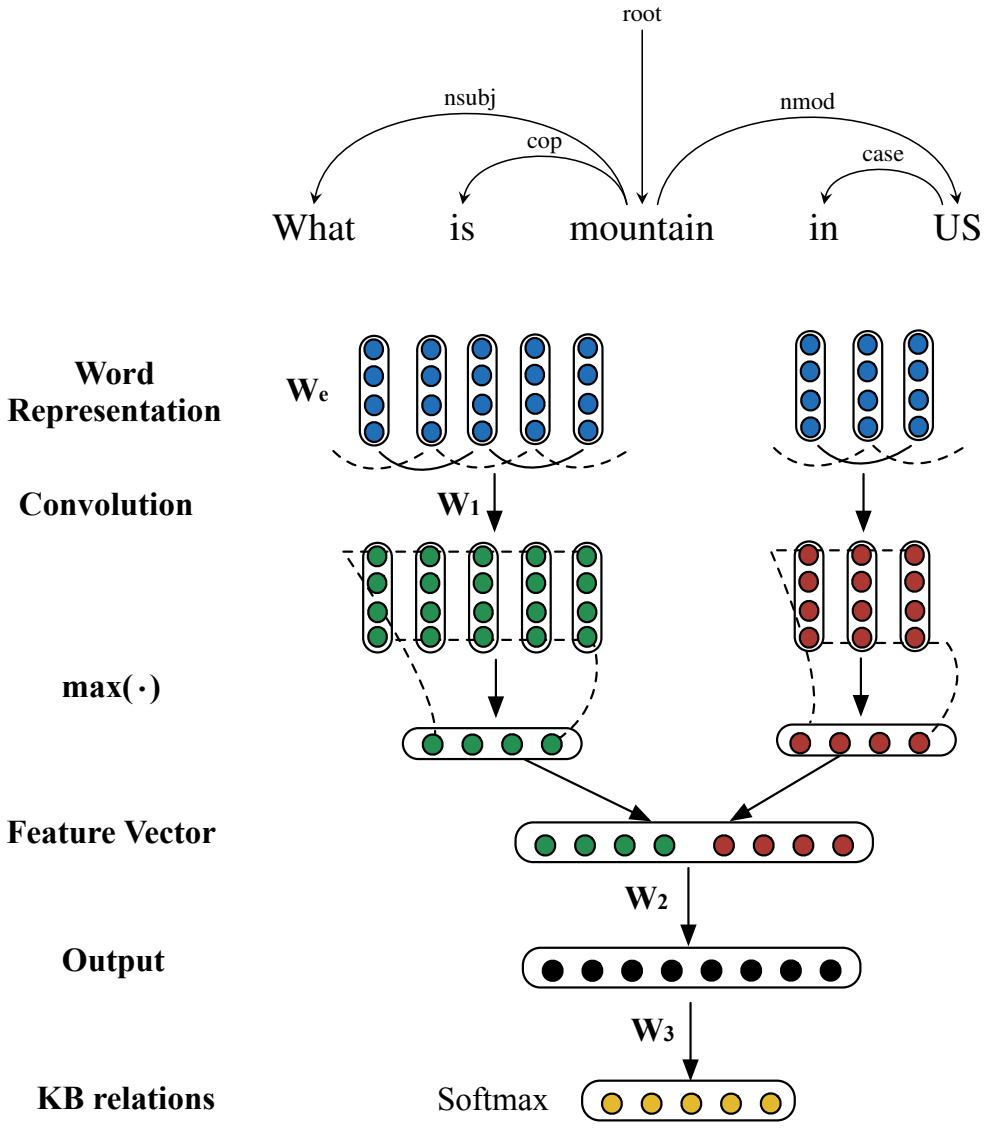
{Rainer, Whitney, Denali, Elbert, Saint Elias, Longs Peak, Pikes Peak, ...}

Denali (also known as Mount McKinley, its former official name) is the highest mountain peak in North America, with a summit elevation of 20,310 feet (6,190 m) above sea level

SVM

{Denali}

QA with Relation Extraction and Text Evidence



Syntax in humans?

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

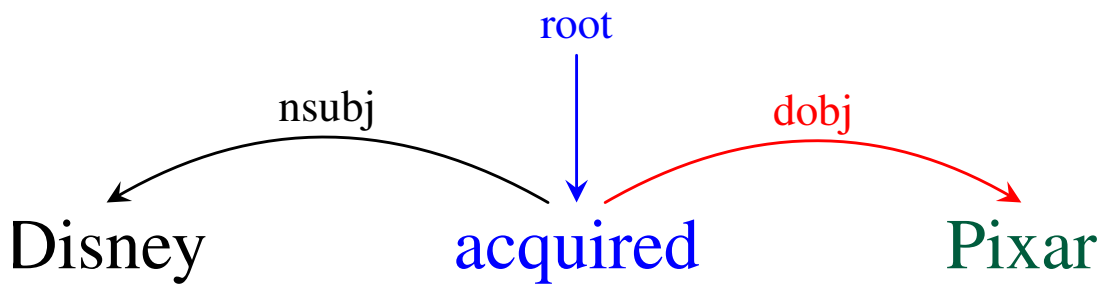
Syntax in humans?

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]

Dependencies to Logical Forms

Single Type System

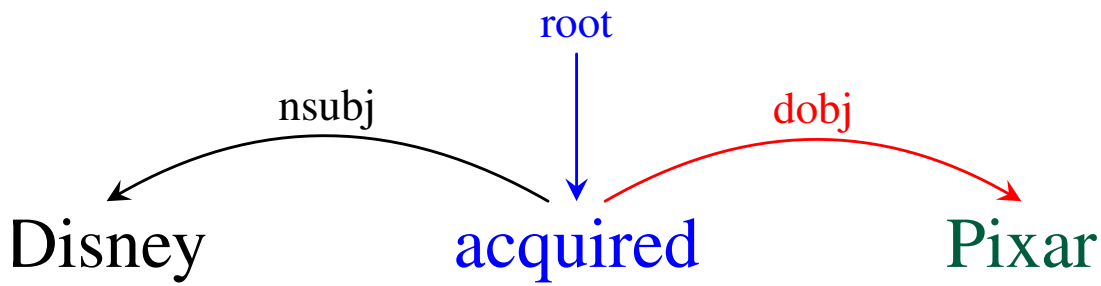


All constituents are of the same lambda expression type

TYPE[acquired] = **TYPE**[Pixar] = **TYPE**[(dobj acquired Pixar)]

Dependencies to Logical Forms

Single Type System

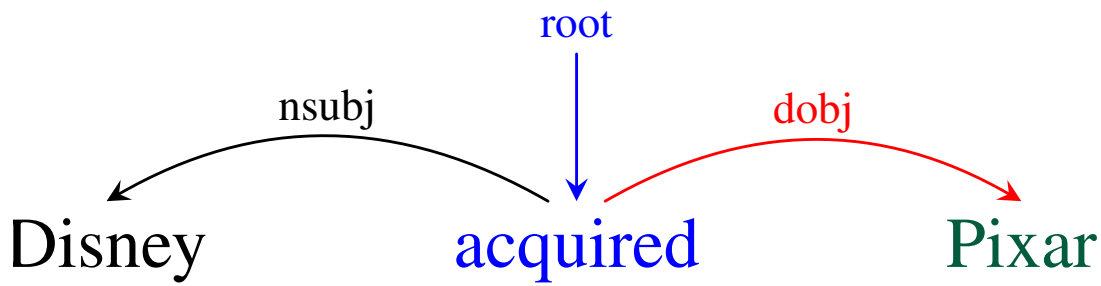


All words have a *lambda expression* of type η

- **TYPE**[acquired] = η
- **TYPE**[Pixar] = η

Dependencies to Logical Forms

Single Type System

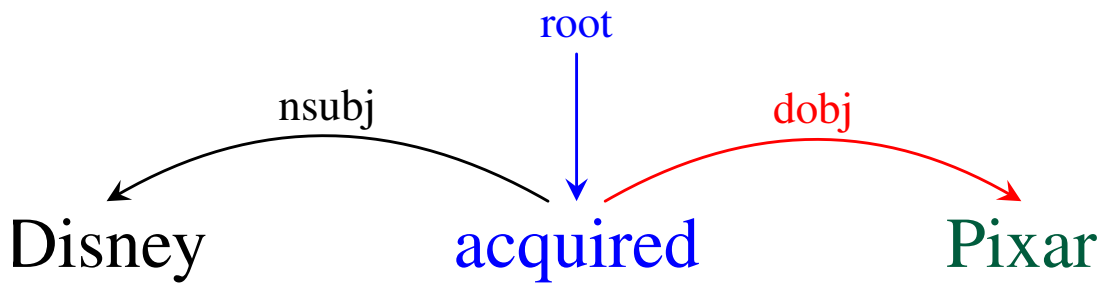


All constituents have a *lambda expression* of type η

- **TYPE**[acquired] = η
- **TYPE**[Pixar] = η
- **TYPE**[(dobj acquired Pixar)] = η

Dependencies to Logical Forms

Single Type System



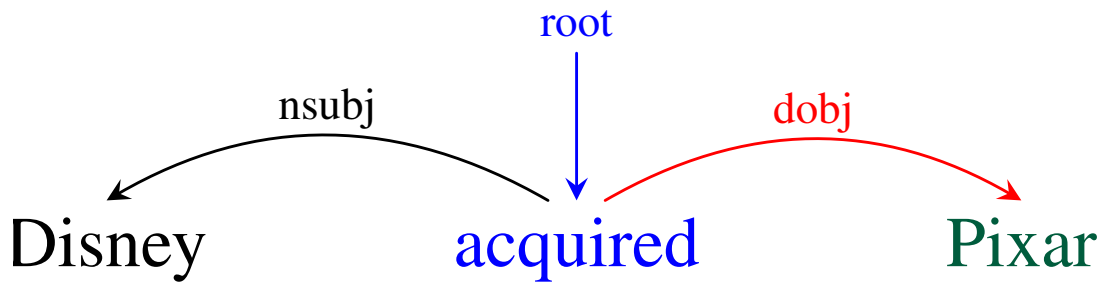
All constituents have a *lambda expression* of type η

- **TYPE**[acquired] = η
- **TYPE**[Pixar] = η
- **TYPE**[(dobj acquired Pixar)] = η

$\Longrightarrow$ **TYPE**[dobj] = $\eta \rightarrow \eta \rightarrow \eta$

Dependencies to Logical Forms

Lambda Calculus for Single Type System



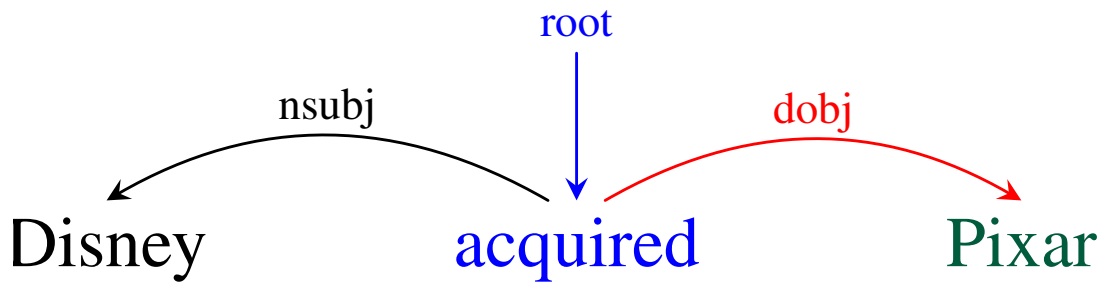
Lambda Expression for words

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

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

Dependencies to Logical Forms

Lambda Calculus for Single Type System



Lambda Expression for words

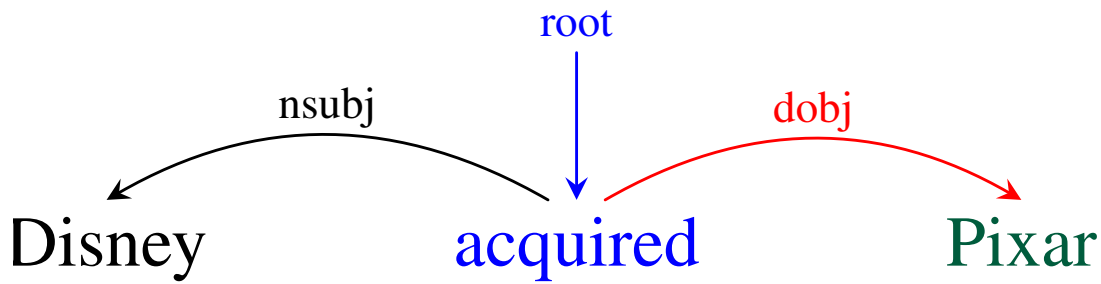
$$\text{acquired} \Rightarrow \lambda x_e.\,\text{acquired}(x_e) \quad \Rightarrow_{\textbf{TYPE}} = \textbf{Event} \rightarrow \textbf{Bool}$$

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

Here **TYPE**[acquired] $\neq$ **TYPE**[Pixar] ✗

Dependencies to Logical Forms

Lambda Calculus

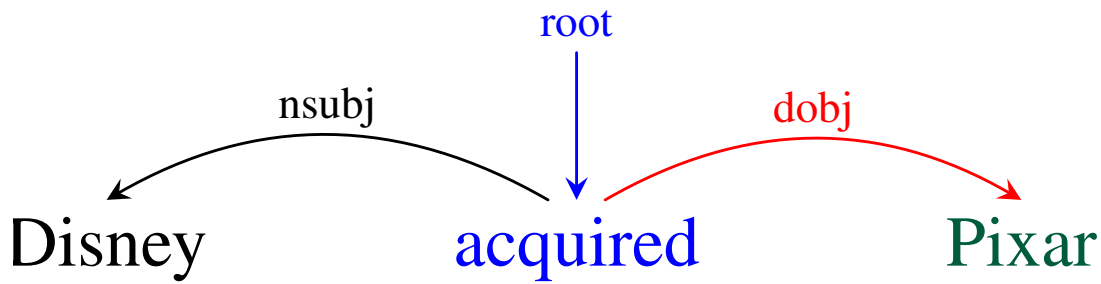


Lambda Expression for dependency labels

$$\text{dobj} \Rightarrow \lambda \mathbf{f}\ \lambda \mathbf{g}\ \lambda \mathbf{z}\ .\ \exists \mathbf{x}\ .\ \mathbf{f}(\mathbf{z}) \quad \wedge \quad \mathbf{g}(\mathbf{x}) \quad \wedge \quad \mathbf{arg_2}(\mathbf{z_e}, \mathbf{x_a})$$

Dependencies to Logical Forms

Lambda Calculus



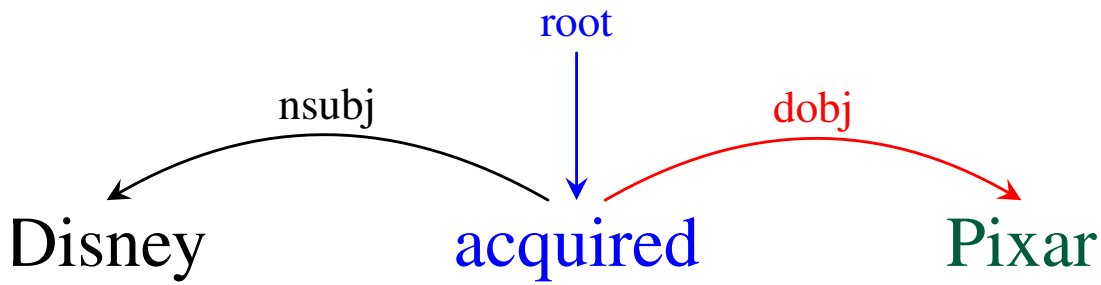
Lambda Expression for dependency labels

$$\text{dobj} \Rightarrow \lambda \mathbf{f}\ \lambda \mathbf{g}\ \lambda \mathbf{z}\ .\ \exists \mathbf{x}\ .\ \mathbf{f}(\mathbf{z}) \quad \wedge \quad \mathbf{g}(\mathbf{x}) \quad \wedge \quad \mathbf{arg_2}(\mathbf{z_e}, \mathbf{x_a})$$

This operation mirrors the tree structure

Dependencies to Logical Forms

Lambda Calculus for Single Type System



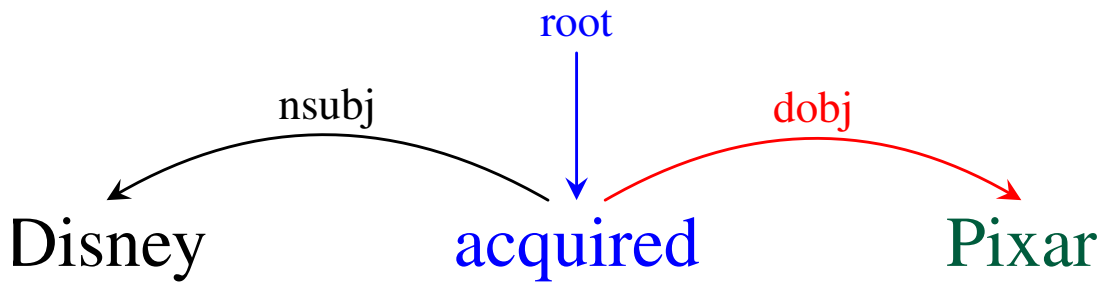
Lambda Expression for words

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

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

Dependencies to Logical Forms

Lambda Calculus for Single Type System



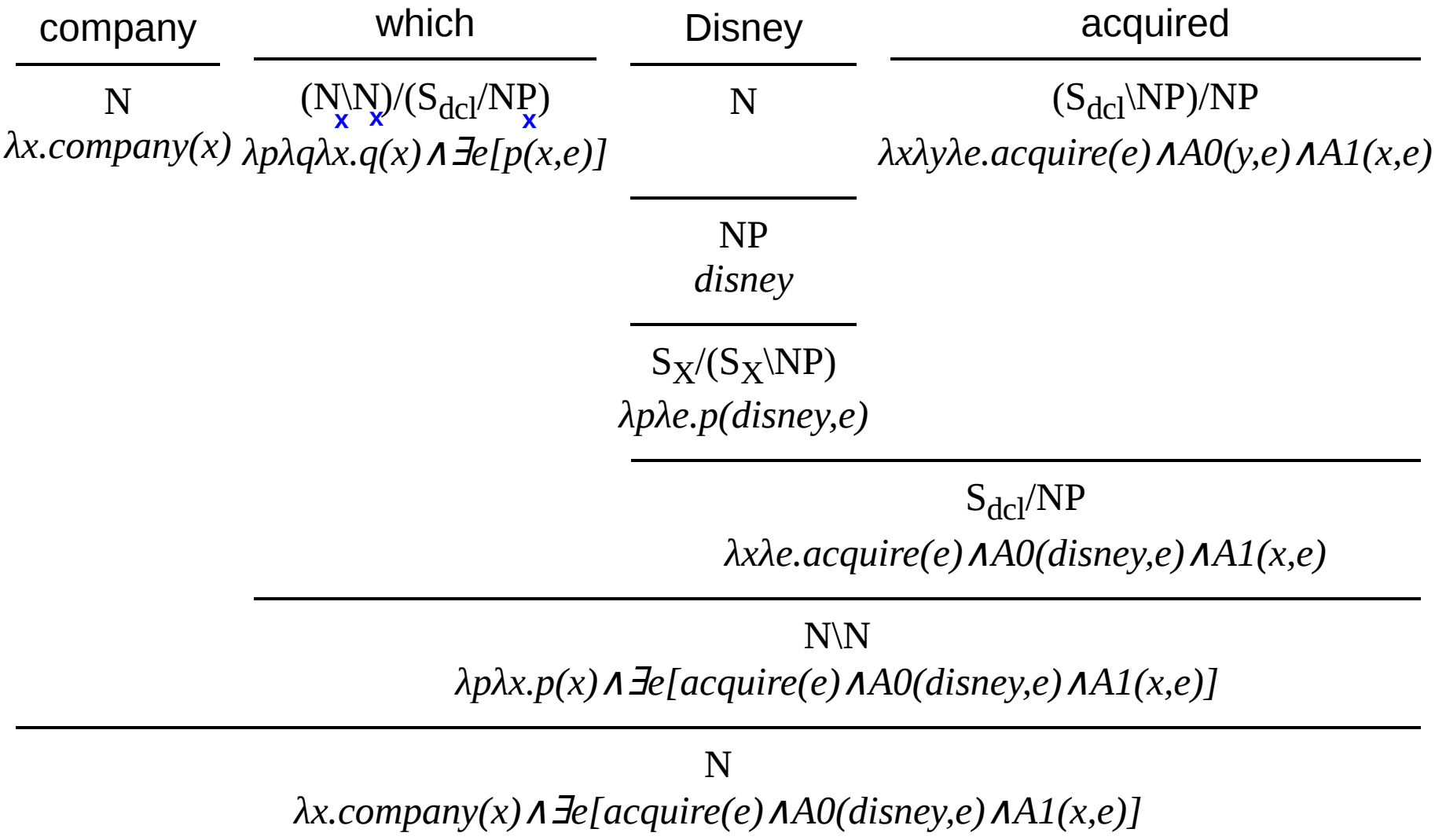
Lambda Expression for words

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

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

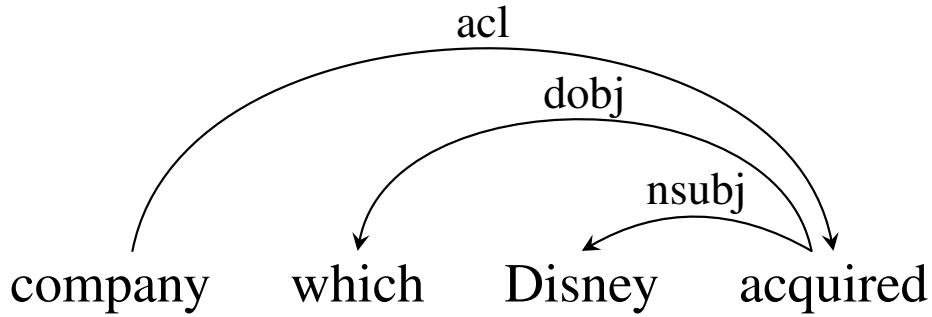
Here η = **TYPE**[acquired] = **TYPE**[Pixar] ✓

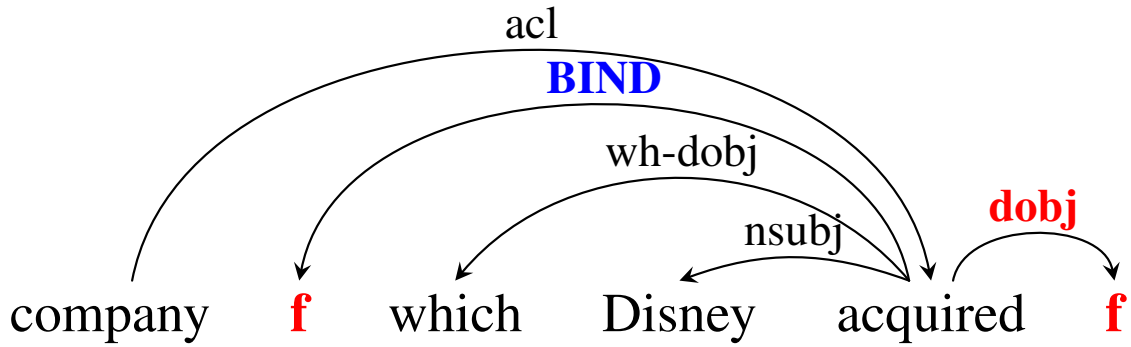
Relative Clause in CCG



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

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 $\Rightarrow \lambda fgx.\, \exists yz.f(y) \wedge g(z) \wedge \text{coord}(x,y,z)$

Logical Expression:

$$\lambda w.\, \exists xyz.\, \text{Eminem}(x_a) \wedge \text{coord}(w,y,z) \\ \wedge \text{arg}_1(w_e, x_a) \wedge \text{s\_to\_I}(y) \wedge \text{d\_50}(z)$$

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 $\Rightarrow \lambda fgx.\, \exists yz.f(y) \wedge g(z) \wedge \text{coord}(x,y,z)$

Logical Expression:

$$\lambda w.\, \exists xyz.\, \text{Eminem}(x_a) \wedge \text{coord}(w,y,z) \\ \wedge \text{arg}_1(w_e, x_a) \wedge \text{s\_to\_I}(y) \wedge \text{d\_50}(z)$$

Post processing:

$$\lambda e.\, \exists xyz.\, \text{Eminem}(x_a) \wedge \text{arg}_1(y_e, x_a) \\ \wedge \text{arg}_1(z_e, x_a) \wedge \text{s\_to\_I}(y) \wedge \text{d\_50}(z)$$

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:

$$\begin{array}{l}(\text{rcmod Apple} \\ \quad (\text{wh-dobj } (\text{BIND } f\ (\text{nsubj } (\text{dobj founded } f)\ \text{Jobs})) \\ \quad\quad \text{which}))\end{array}$$

Substitution:

wh-dobj $\Rightarrow \lambda fgz.f(z)$

rcmod $\Rightarrow \lambda fgz.f(z) \wedge g(z)$

Logical Expression:

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

Expressivity

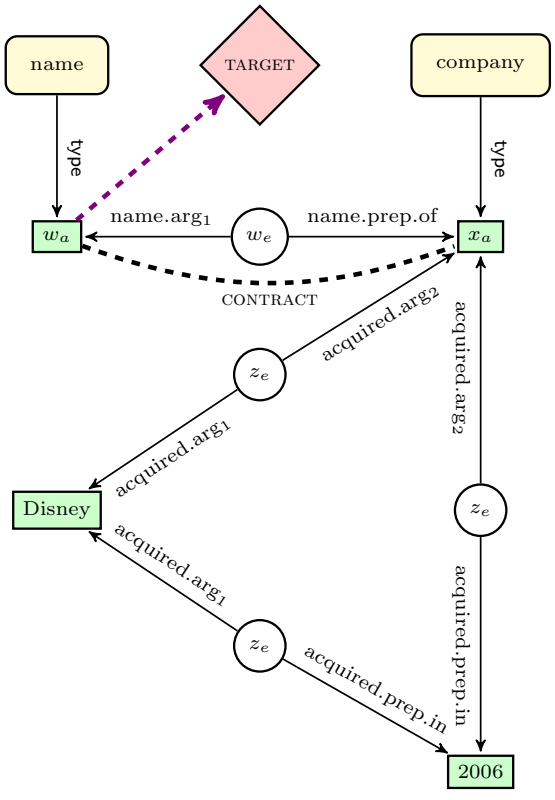
How isomorphic are the representations compared to Knowledge Graph?
Average Oracle F_1 Table.

Search Space

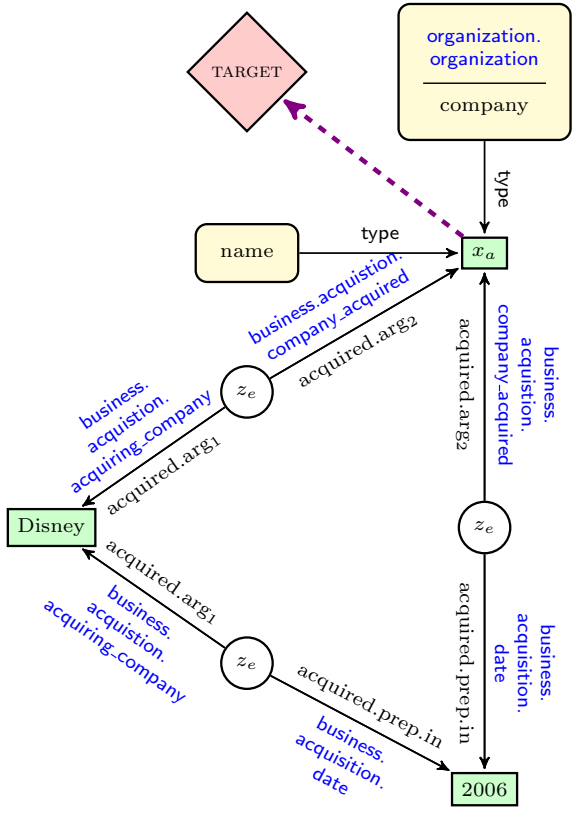
How many ways to reach an answer?
Average Oracle F_1 Table.

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

- $\lambda z.\, \exists xyw.\, \text{TARGET}(x_a) \wedge \text{do}(z_e) \wedge \text{arg}_1(z_e, x_a) \wedge$
  $\text{Washington\_DC}(y_a) \wedge \text{December}(w_a)$

After EXPAND

- $\lambda z.\, \exists xyw.\, \text{TARGET}(x_a) \wedge \text{do}(z_e) \wedge \text{arg}_1(z_e, x_a) \wedge$
  $\text{Washington\_DC}(y_a) \wedge \text{dep}(z_e, y_a) \wedge \text{December}(w_a) \wedge \text{dep}(z_e, w_a)$

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