Universal Semantic Parsing

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


Siva Reddy

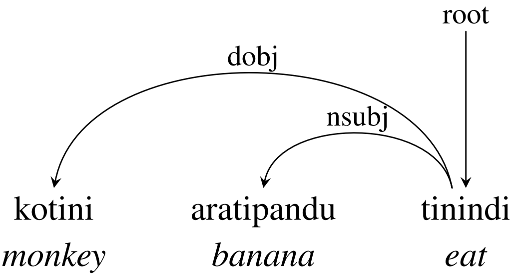
Dependency Trees help Semantics

kotini	aratipandu	tinindi
monkey	*banana*	*eat*

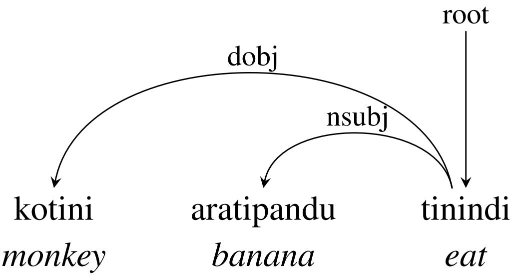
Dependency Trees help Semantics

kotini	aratipandu	tinindi
monkey	*banana*	*eat*

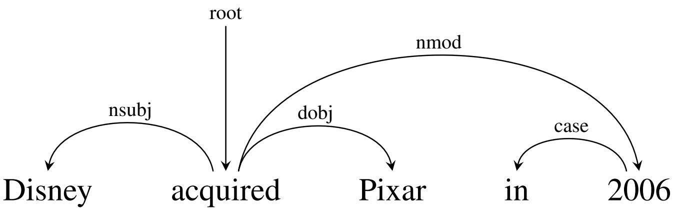
Dependency Trees help Semantics



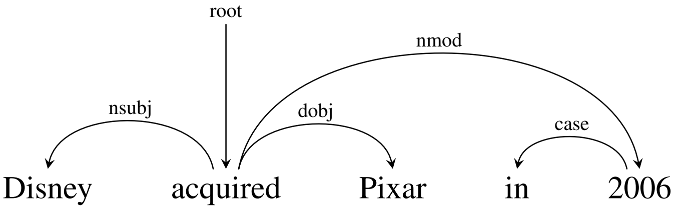
Dependency Trees help Semantics



Universal Dependencies

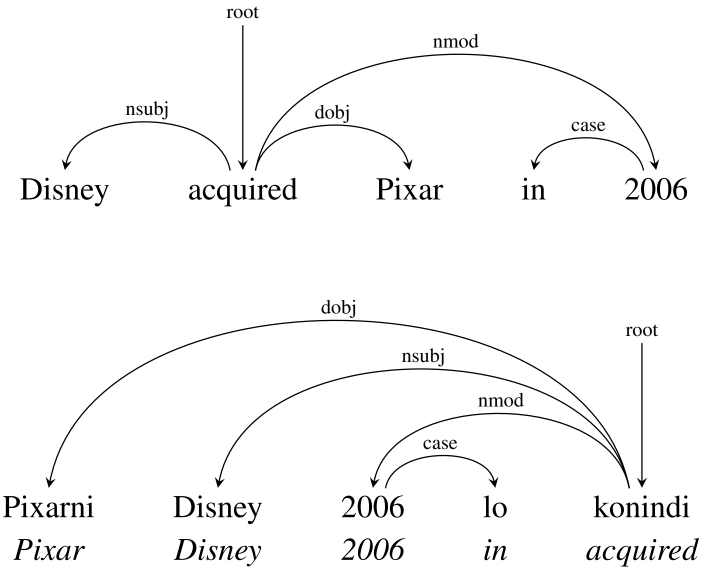


Universal Dependencies



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 Language-independent 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

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

CCG

Lambda Calculus

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

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

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

CCG

$$
\begin{array}{ccc}
\text{Disney} & \text{acquired} & \text{Pixar} \\
\overline{\quad NP \quad} & \overline{\qquad S\backslash NP/NP \qquad} & \overline{\qquad NP \qquad} \\
\text{Disney} & \lambda y \lambda x \lambda e.\ \text{acquired}(e) & \text{Pixar} \\
 & \wedge\ \text{arg}_1(e, x) & \\
 & \wedge\ \text{arg}_2(e, y) &
\end{array}
$$

CCG

$$
\begin{array}{ccc}
\text{Disney} & \text{acquired} & \text{Pixar} \\
\overline{NP} & \overline{S\backslash NP/NP} & \overline{NP} \\
\text{Disney} & \begin{array}{l}\lambda y \lambda x \lambda e.\ \text{acquired}(e) \\ \quad \wedge\ \text{arg}_1(e, x) \\ \quad \wedge\ \text{arg}_2(e, y)\end{array} & \text{Pixar} \\
 & \multicolumn{2}{c}{\overline{\qquad\qquad\qquad\qquad}{>}} \\
 & \multicolumn{2}{c}{S\backslash NP}
\end{array}
$$

CCG

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

$$
\xrightarrow{\hspace{6em}} >
$$

$$
S\backslash NP
$$

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

CCG

$$
\dfrac{\text{Disney}}{NP} \quad \dfrac{\text{acquired}}{S\backslash NP/NP} \quad \dfrac{\text{Pixar}}{NP}
$$

$$
\text{Disney} \quad \begin{array}{c}\lambda y \lambda x \lambda e.\ \text{acquired}(e) \\ \wedge\ \text{arg}_1(e, x) \\ \wedge\ \text{arg}_2(e, y)\end{array} \quad \text{Pixar}
$$

$$
\xrightarrow{\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad}>
$$

$$
S\backslash NP
$$

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

$$
\xleftarrow{\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad}<
$$

$$
S
$$

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

CCG

$$
\begin{array}{ccc}
\text{Disney} & \text{acquired} & \text{Pixar} \\
\hline
NP & S\backslash NP/NP & NP \\
\text{Disney} & \lambda y \lambda x \lambda e.\ \text{acquired}(e) & \text{Pixar} \\
 & \wedge\ \text{arg}_1(e, x) & \\
 & \wedge\ \text{arg}_2(e, y) & \\
\hline
 & \multicolumn{2}{c}{S\backslash NP \quad >} \\
 & \multicolumn{2}{c}{\lambda x \lambda e.\ \text{acquired}(e)} \\
 & \multicolumn{2}{c}{\wedge\ \text{arg}_1(e, x) \wedge\ \text{arg}_2(e, \text{Pixar})} \\
\hline
\multicolumn{3}{c}{S \quad <} \\
\multicolumn{3}{c}{\lambda e.\ \text{acquired}(e) \wedge \text{arg}_1(e, \text{Disney}) \wedge \text{arg}_2(e, \text{Pixar})}
\end{array}
$$

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

Universal Semantic Parsing

Application task: Freebase Question Answering

Results on English, German, and Spanish

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

Universal Semantic Parsing

Goals

1. Dependencies to logical forms
2. Compositional
3. Language-agnostic conversion
   - Dependency labels and postags dictate the semantics
   - No token-specific semantics

Logical Forms

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

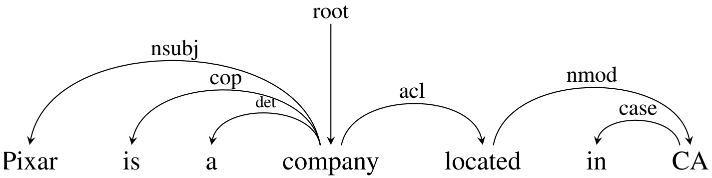
Simplification [Narayan & Gardent, 2014]

Paraphrasing [Pavlick et al., 2015]

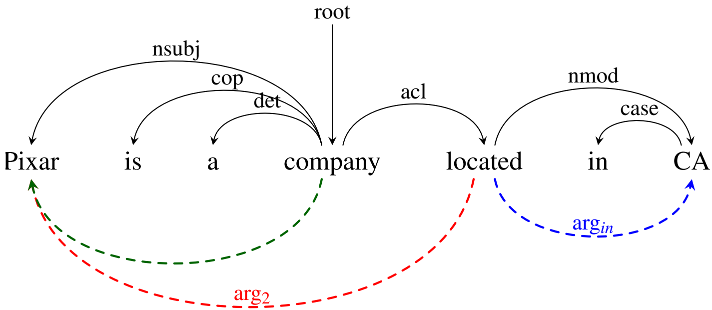
Information Extraction [Rocktäschel et al., 2015]

Summarization [Liu et al., 2015]

Compositional

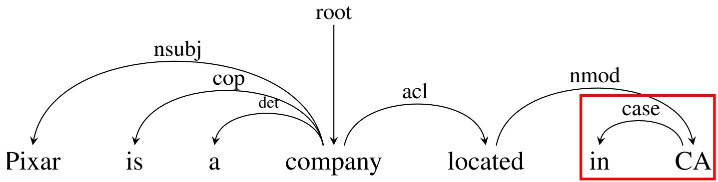


Compositional



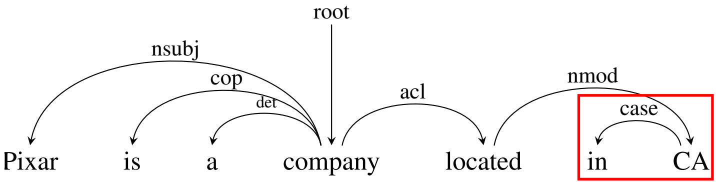
$$\lambda x.\exists yz.\,\text{located}(z_e) \wedge \text{Pixar}(x_a) \wedge \text{CA}(y_a) \wedge$$
$$\text{company}(x_a) \wedge \text{arg}_2(z_e, x_a) \wedge \text{arg}_{\text{in}}(z_e, y_a)$$

Compositional



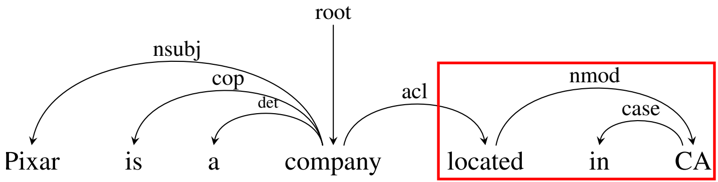
$(case \quad \text{CA} \quad \text{in})$

Compositional



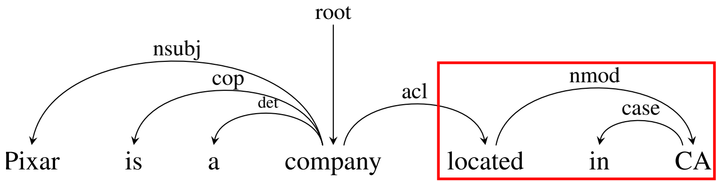
$$\frac{(\mathit{case} \qquad \mathrm{CA} \qquad \mathrm{in})}{\lambda x.\ \mathrm{CA}(x_a)}$$

Compositional



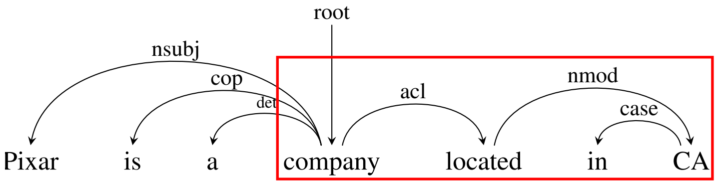
$$(nmod \quad \text{located} \quad \text{in CA})$$

Compositional



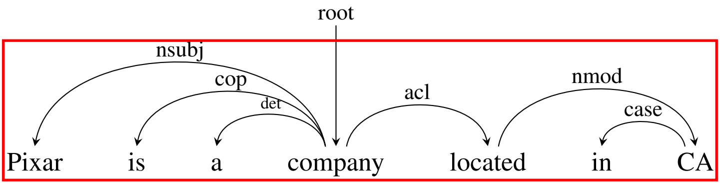
$$\frac{(\mathit{nmod} \qquad \text{located} \qquad \text{in CA})}{\lambda z.\text{located}(z_e) \land \text{CA}(x_a) \land \text{arg}_{\text{in}}(z_e, x_a)}$$

Compositional



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

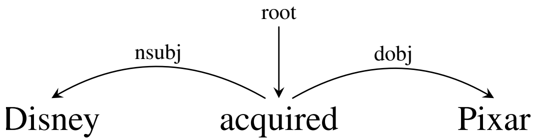
Compositional



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

Composition Order

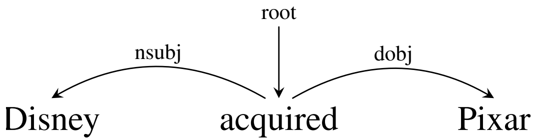
Binarization



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

Composition Order

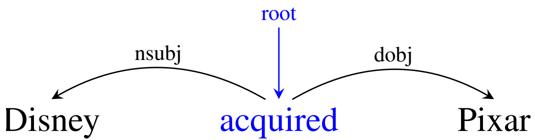
Binarization

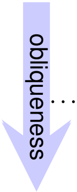


Dependency labels drive the composition

Composition Order

Binarization

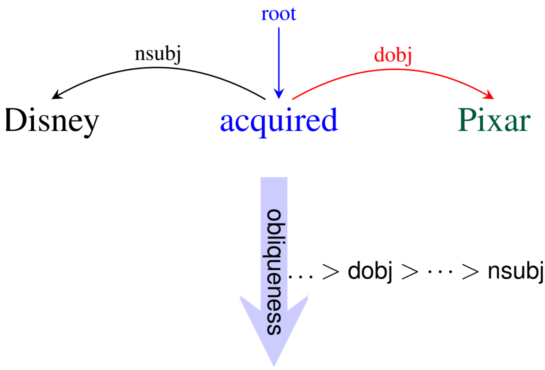




$\ldots >$ dobj $> \cdots >$ nsubj $> \ldots$

Composition Order

Binarization

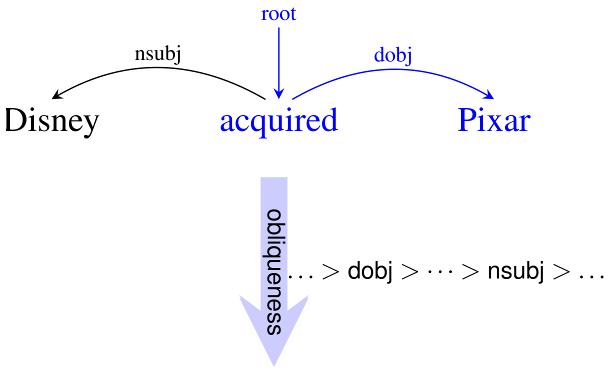


$\ldots >$ dobj $> \cdots >$ nsubj $> \ldots$

(dobj acquired Pixar)

Composition Order

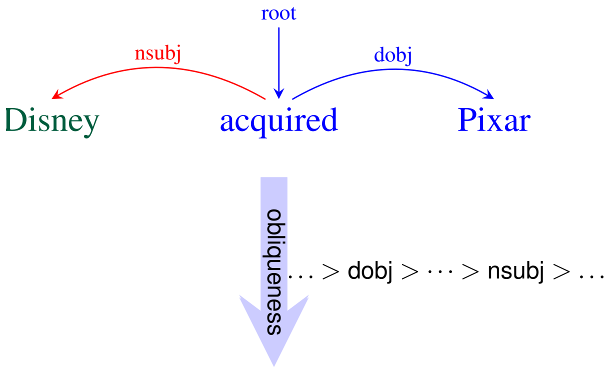
Binarization



(dobj acquired Pixar)

Composition Order

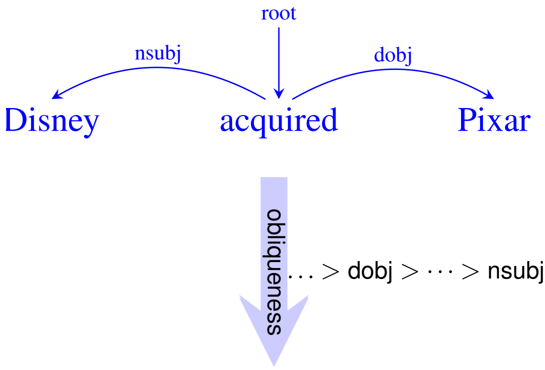
Binarization



(nsubj (dobj acquired Pixar) Disney)

Composition Order

Binarization

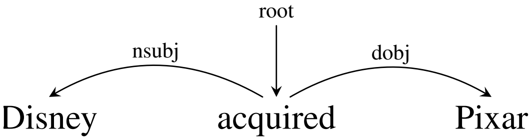


$\ldots >$ dobj $> \cdots >$ nsubj $> \ldots$

(nsubj (dobj acquired Pixar) Disney)

Composition Order

Binarization

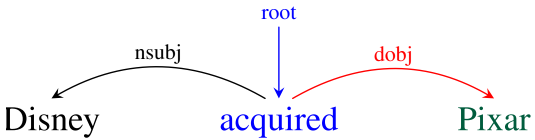


(nsubj (dobj acquired Pixar) Disney)

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

Language-agnostic Conversion

Substitution

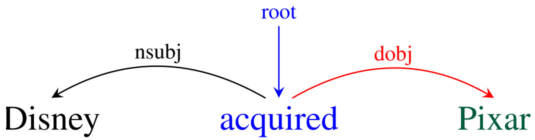


Lambda Calculus Basic Types

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

Language-agnostic Conversion

Substitution



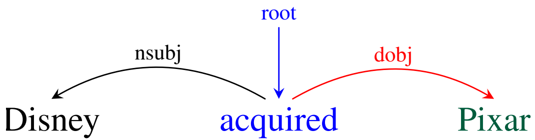
Lambda Expression for words

$$VERB \Rightarrow \lambda x.\,\mathsf{word}(x_e), e.g., \textsf{acquired} \Rightarrow \lambda x.\,\mathsf{acquired}(x_e)$$

$$PROPN \Rightarrow \lambda x.\,\mathsf{word}(x_a), e.g., \textsf{Pixar} \Rightarrow \lambda x.\,\mathsf{Pixar}(x_a)$$

Language-agnostic Conversion

Substitution

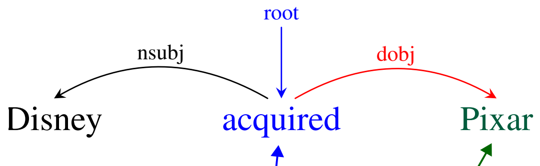


Lambda Expression for dependency labels

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

Language-agnostic Conversion

Substitution

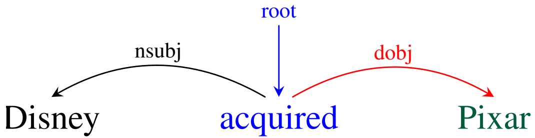


Lambda Expression for dependency labels

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

Dependencies to Logical Forms

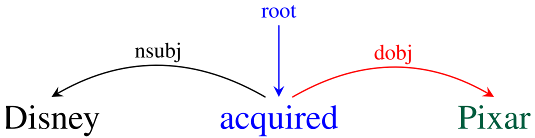
Composition



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

Dependencies to Logical Forms

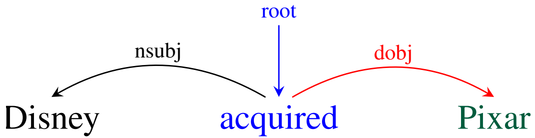
Composition



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

Dependencies to Logical Forms

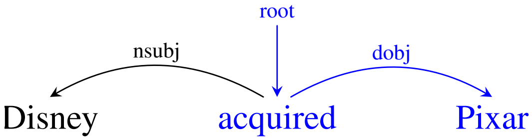
Composition



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

Dependencies to Logical Forms

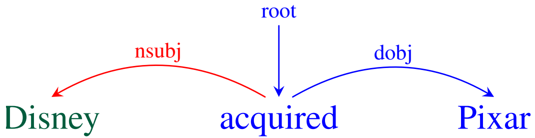
Composition



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

Dependencies to Logical Forms

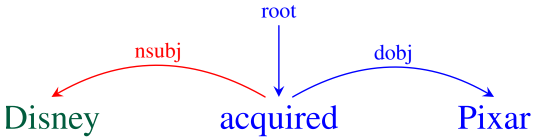
Composition



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

Dependencies to Logical Forms

Composition



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

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

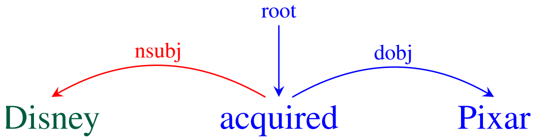
$$\frac{}{\lambda z.\ \exists y.\ \mathrm{acquired}(z_e) \wedge \mathrm{Pixar}(y_a) \wedge \mathrm{arg}_2(z_e, y_a)}$$

$$\lambda x.\mathrm{Disney}(x_a)$$

$$\frac{}{\lambda g \lambda z.\exists x y.\mathrm{acquired}(z_e) \wedge \mathrm{Pixar}(y_a) \wedge \mathrm{g}(x) \wedge \mathrm{arg}_1(z_e, x_a) \wedge \mathrm{arg}_2(z_e, y_a)}$$

Dependencies to Logical Forms

Composition



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

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

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

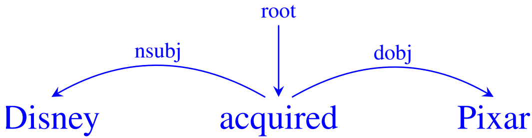
$$\lambda x.\mathrm{Disney}(x_a)$$

$$\frac{}{\lambda g \lambda z.\exists xy.\mathrm{acquired}(z_e) \wedge \mathrm{Pixar}(y_a) \wedge \mathrm{g}(x) \wedge \mathrm{arg}_1(z_e, x_a) \wedge \mathrm{arg}_2(z_e, y_a)}$$

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

Dependencies to Logical Forms

Composition



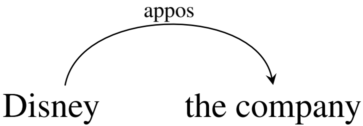
$$\frac{(\textbf{nsubj}\quad(\textbf{dobj}\quad\textbf{acquired}\quad\textbf{Pixar})\quad\textbf{Disney})}{\lambda z.\exists xy.\text{acquired}(z_e) \wedge \text{Pixar}(y_a) \wedge \text{Disney}(x_a) \wedge \text{arg}_1(z_e, x_a) \wedge \text{arg}_2(z_e, y_a)}$$

Dependencies to Logical Forms

Disney $\xrightarrow{\text{appos}}$ the company

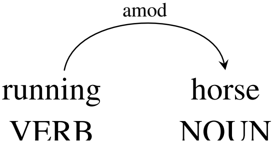
$$\textit{appos} = \lambda f \lambda g \lambda x . f(x) \wedge g(x)$$

Dependencies to Logical Forms



appos $=$

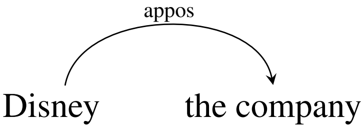
$$\lambda f \lambda g \lambda x. f(x) \wedge g(x)$$



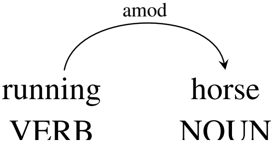
amod $=$

$$\lambda f \lambda g \lambda x. \exists z. f(x) \wedge g(z) \wedge \mathrm{amod}^i(z_e, x_a)$$

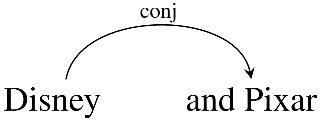
Dependencies to Logical Forms



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



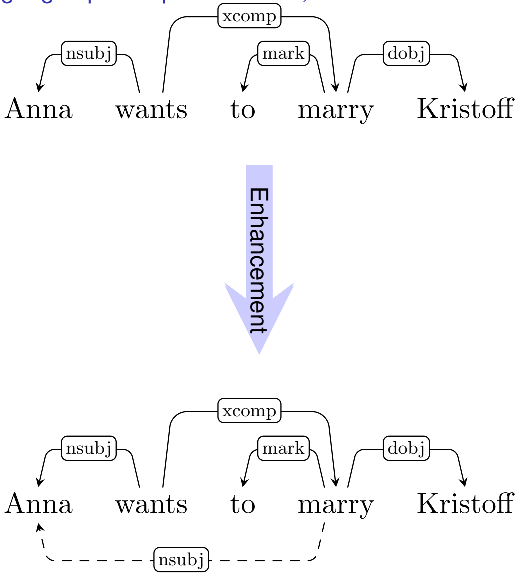
$$amod = \lambda f \lambda g \lambda x. \exists z. f(x) \wedge g(z) \wedge \mathrm{amod}^{i}(z_e, x_a)$$



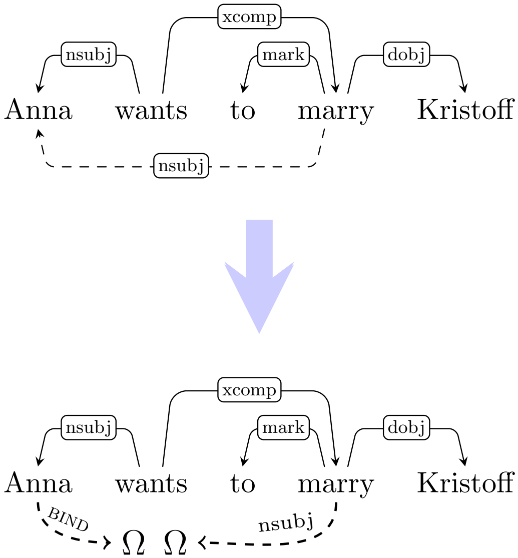
$$conj = \lambda f \lambda g \lambda z. \exists xy. f(x) \wedge g(y) \wedge \mathrm{coord}(z, x, y)$$

Enhancement

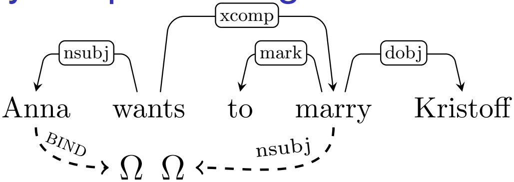
Long-distance, Language-specific phenomenon, Quantifiers



Dependency Graphs to Logical Forms



Dependency Graphs to Logical Forms



Substitution Expressions

$$\text{BIND} = \lambda f \lambda g \lambda x . f(x) \wedge g(x)$$

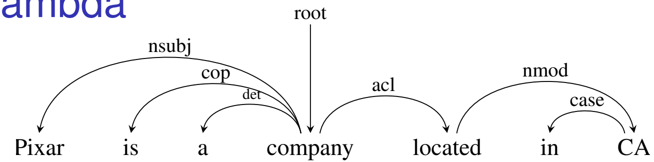
$$\text{xcomp} = \lambda f g x . \exists y . f(x) \wedge g(y) \wedge \text{xcomp}(x_e, y_e)$$

$$\omega = \lambda x . \text{EQ}(x, \omega)$$

Final Expression:

$$\begin{aligned} \lambda z . \exists x y w . & \, \text{wants}(z_e) \wedge \text{Anna}(x_a) \wedge \text{arg}_1(z_e, x_a) \\ & \wedge \text{marry}(y_e) \wedge \text{xcomp}(z_e, y_e) \wedge \text{arg}_1(y_e, x_a) \\ & \wedge \text{Kristoff}(w_a) \wedge \text{arg}_2(y_e, w_a) \, . \end{aligned}$$

UDepLambda



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

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

lambda expression composition

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

UDepLambda 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 UDepLambda

Freebase Semantic Parsing

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

Question

Who is the director of Titanic?

Answer

{James Cameron}

Freebase Semantic Parsing

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

Question

Who is the director of Titanic?

Grounded Logical Form

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

Latent

Answer

{James Cameron}

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

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

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]

Language to **ungrounded** logical form

Ungrounded logical form to **grounded** logical form

Question

Who is the director of Titanic?

Ungrounded Logical Form

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

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

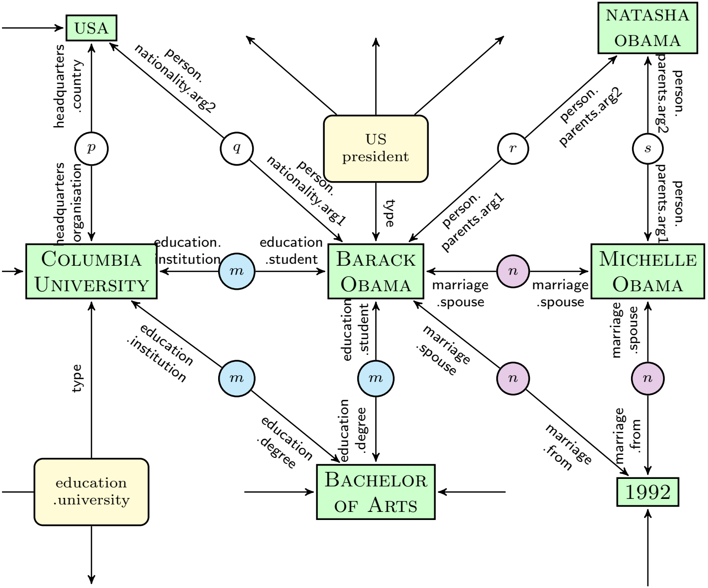
Training Data: Question and Answer Pairs

Evaluation: Question Answering on Freebase

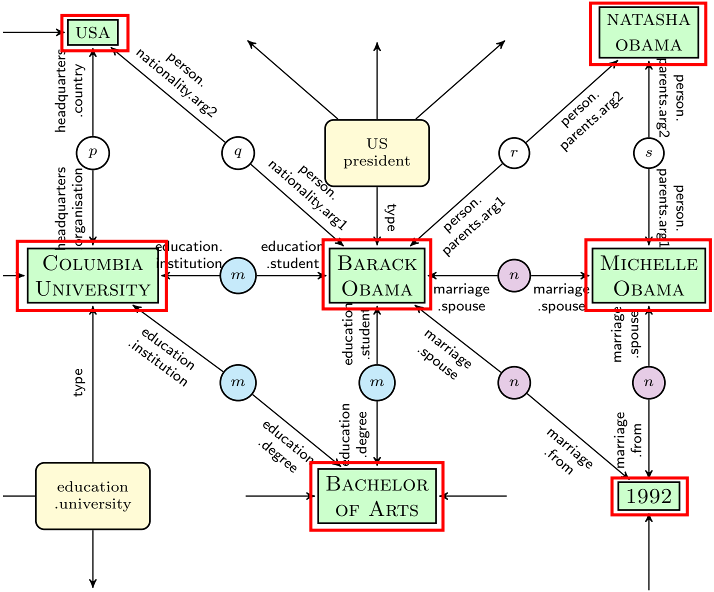
Resources: Dependency Parser, UDepLambda

Hypothesis: UDepLambda logical forms are useful

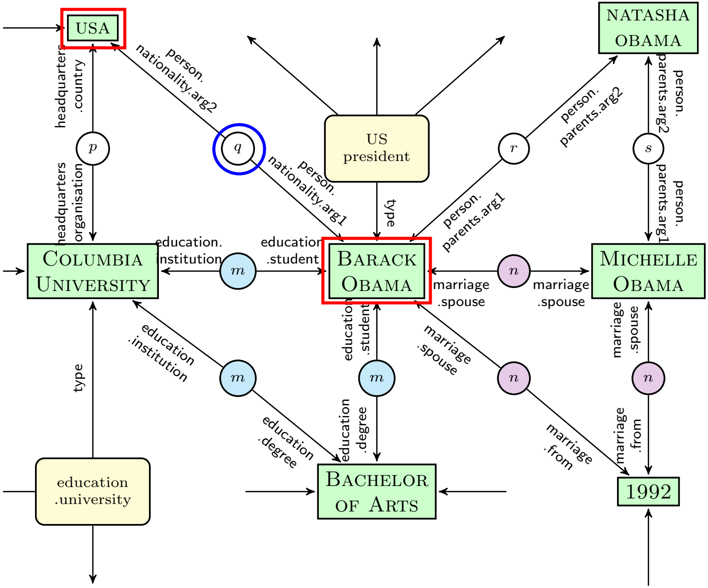
Freebase is a Graph



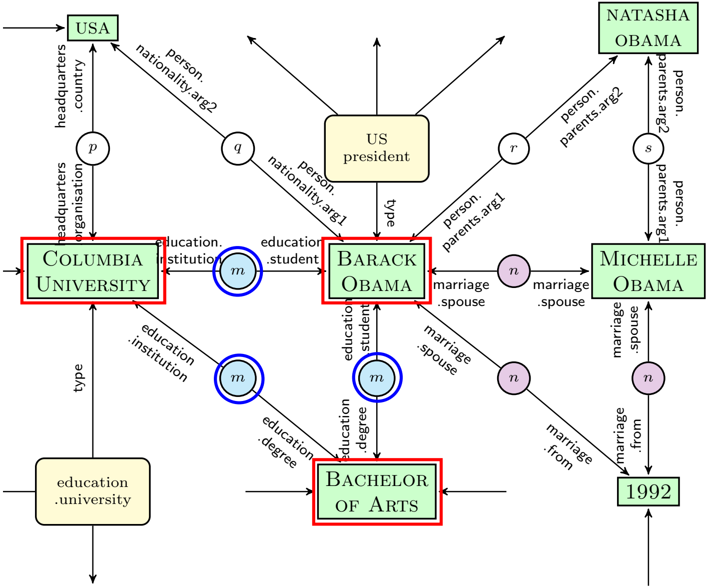
Freebase is a Graph



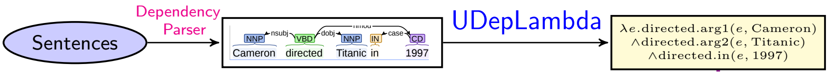
Freebase is a Graph



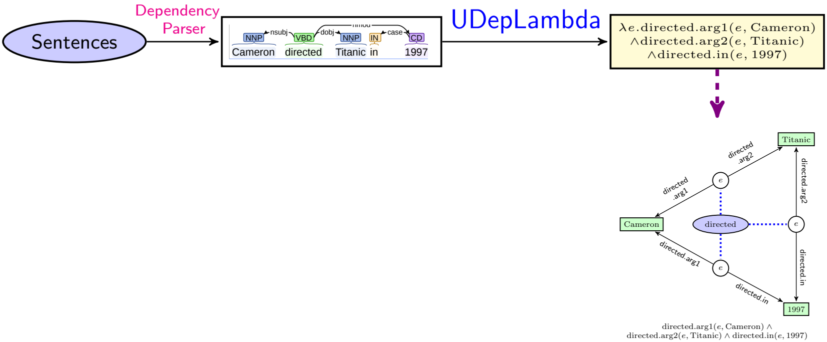
Freebase is a Graph



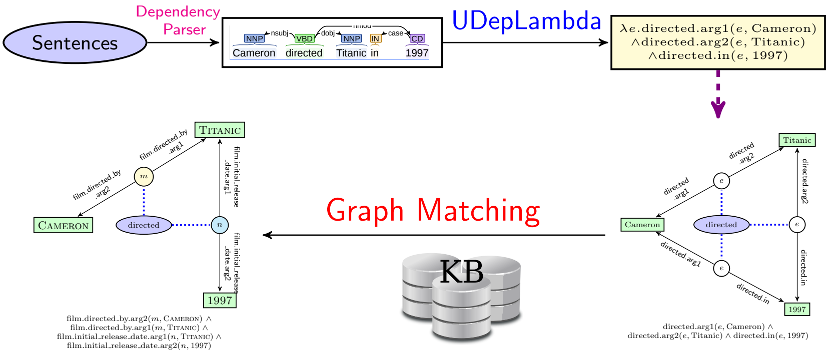
Our Approach



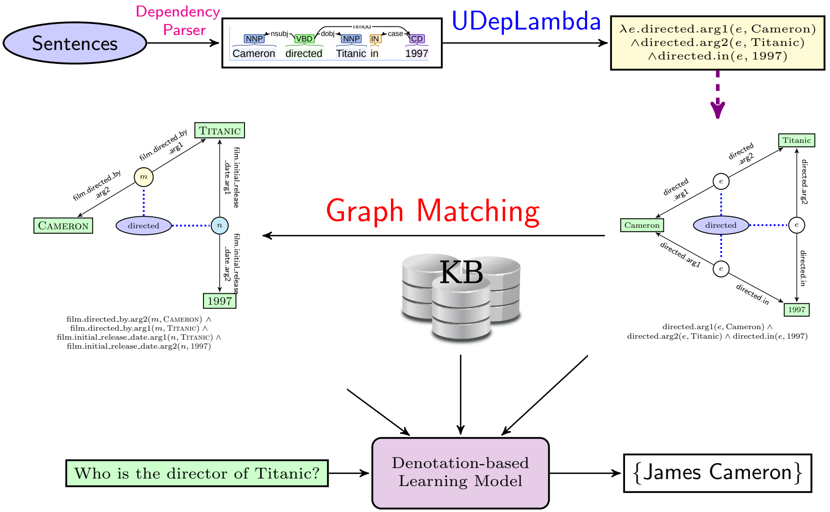
Our Approach



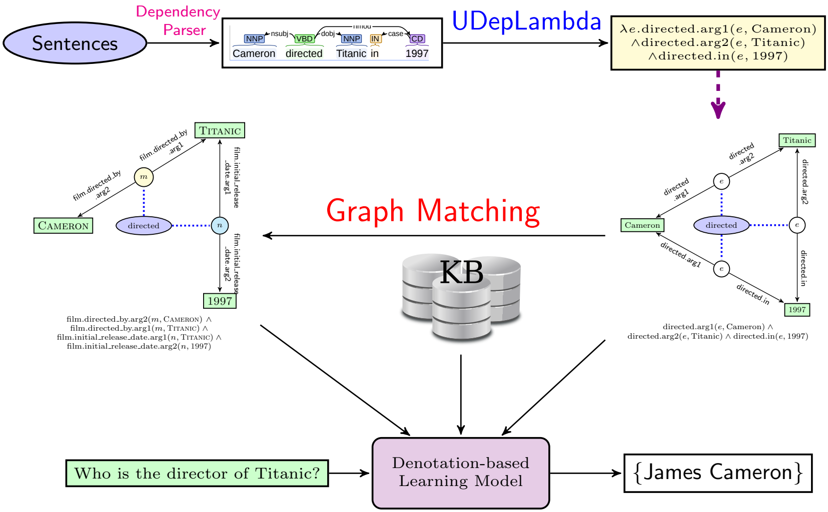
Our Approach



Our Approach



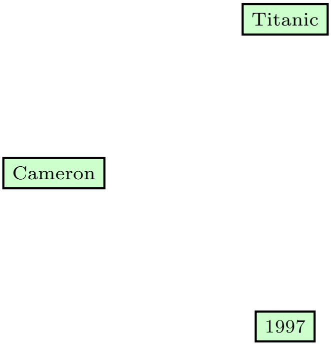
Our Approach



Logical Form to Ungrounded Graph

Cameron directed Titanic in 1997

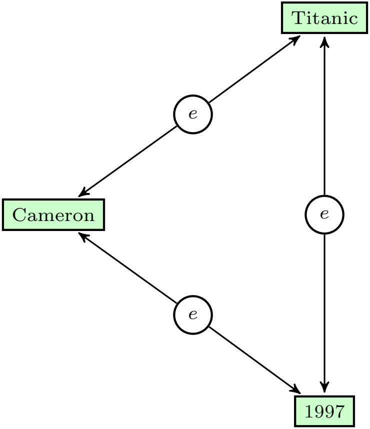
$$\lambda e.\text{directed.arg1}(e, \text{Cameron}) \wedge \text{directed.arg2}(e, \text{Titanic}) \wedge \text{directed.in}(e, 1997)$$



Logical Form to Ungrounded Graph

Cameron directed Titanic in 1997

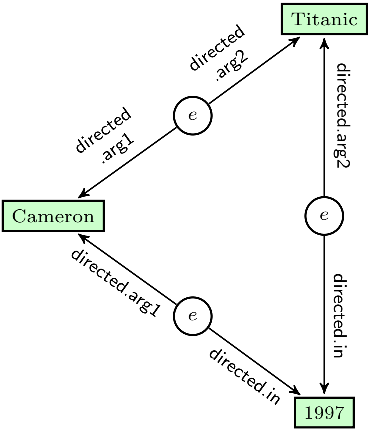
$$\lambda e.\text{directed.arg1}(\mathbf{e}, \texttt{Cameron}) \wedge \text{directed.arg2}(\mathbf{e}, \texttt{Titanic}) \wedge \text{directed.in}(\mathbf{e}, \texttt{1997})$$



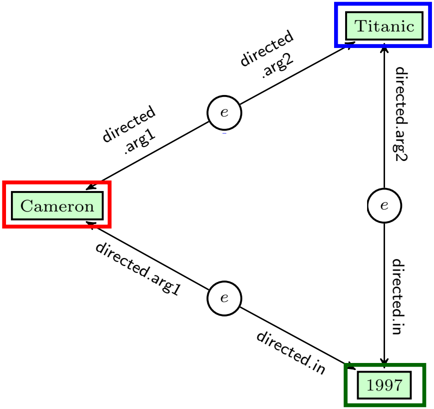
Logical Form to Ungrounded Graph

Cameron directed Titanic in 1997

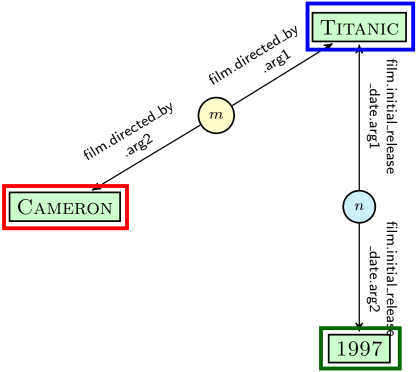
$$\lambda e.\text{directed.arg1}(e, \texttt{Cameron}) \wedge \text{directed.arg2}(e, \texttt{Titanic}) \wedge \text{directed.in}(e, \texttt{1997})$$



Graph Matching



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

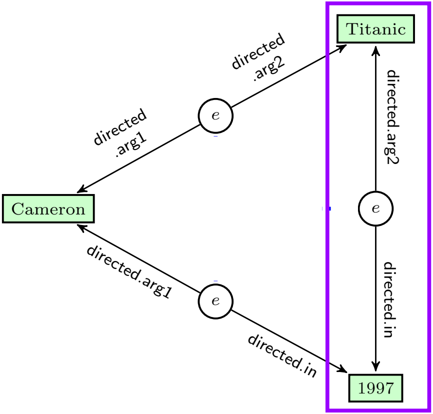


$\text{film.directed\_by.arg2}(m, \textsc{Cameron}) \wedge$
$\text{film.directed\_by.arg1}(m, \textsc{Titanic}) \wedge$
$\text{film.initial\_release\_date.arg1}(n, \textsc{Titanic}) \wedge$
$\text{film.initial\_release\_date.arg2}(n, 1997)$

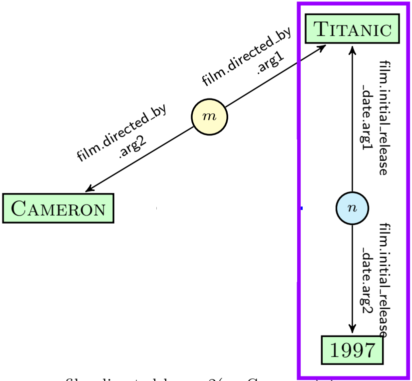
Ungrounded Graph

Grounded Graph

Graph Matching



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

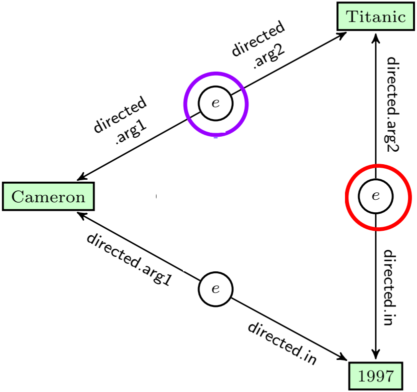


$$\text{film.directed\_by.arg2}(m, \textsc{Cameron}) \land$$
$$\text{film.directed\_by.arg1}(m, \textsc{Titanic}) \land$$
$$\text{film.initial\_release\_date.arg1}(n, \textsc{Titanic}) \land$$
$$\text{film.initial\_release\_date.arg2}(n, 1997)$$

Ungrounded Graph

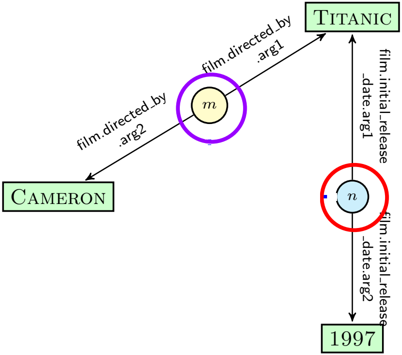
Grounded Graph

Graph Matching



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

Ungrounded Graph

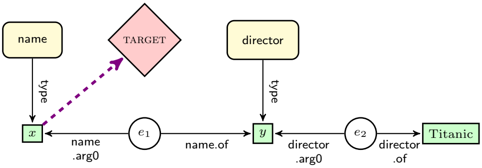


$$\text{film.directed\_by.arg2}(m, \textsc{Cameron}) \wedge$$
$$\text{film.directed\_by.arg1}(m, \textsc{Titanic}) \wedge$$
$$\text{film.initial\_release\_date.arg1}(n, \textsc{Titanic}) \wedge$$
$$\text{film.initial\_release\_date.arg2}(n, 1997)$$

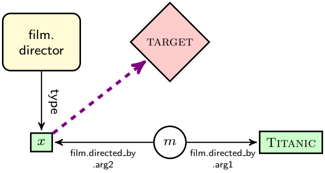
Grounded Graph

Graph Mismatch

What is the name of the director of Titanic?



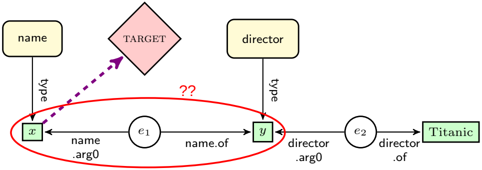
Ungrounded graph



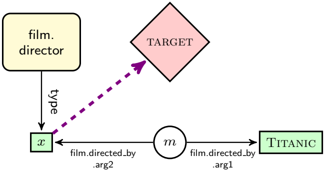
Grounded graph

Graph Mismatch

What is the name of the director of Titanic?



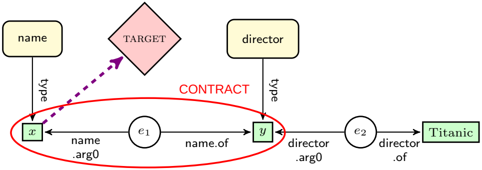
Ungrounded graph



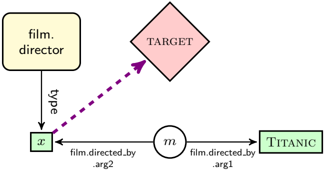
Grounded graph

Graph Mismatch

What is the name of the director of Titanic?



Ungrounded graph



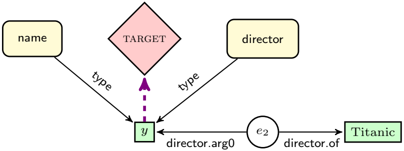
Grounded graph

- Paraphrasing is an alternative[†]

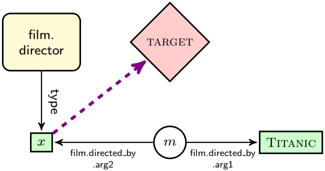
[†] Narayan, Reddy, Cohen (INLG 2016)

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

$$(\hat{g}, \hat{u}) = \arg\max_{g,u} \Phi(g, u, q, KB) \cdot \underbrace{\theta}_{\text{Weight Vector}}$$

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:

$$(g^+, u^+) = \underbrace{\arg\max_{(g,u)\in O(q)}}_{\text{Oracle Graphs}} \Phi(g, u, q, KB) \cdot \theta^t$$

Experimental Setup: UD

69 lambda calculus formulae

WebQuestions in English, German, and Spanish

BiLSTM Parser [Kipperwiser and Goldberg (2016)]

- English: 81.8
- German: 74.7
- Spanish: 82.2

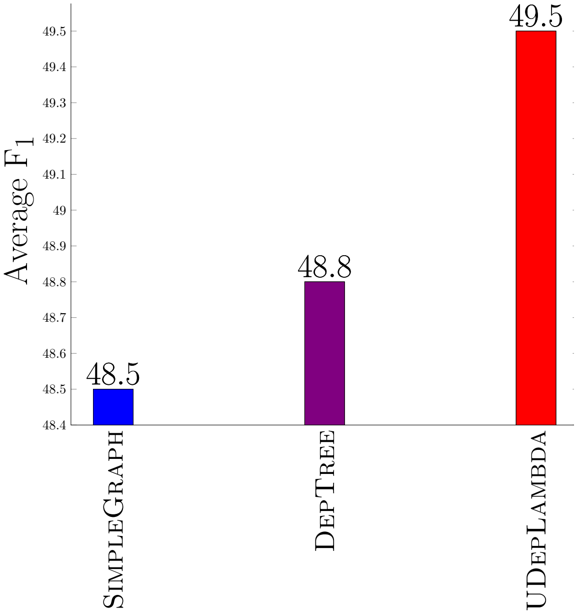
Baselines

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

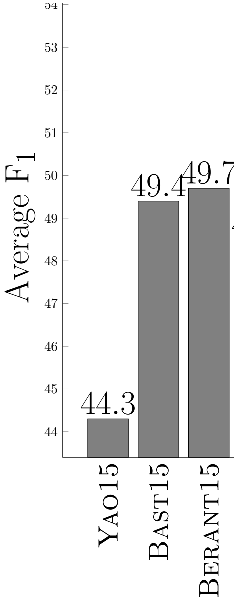
DepTree: Transduce a dependency tree to target graph

Results on Multilingual WebQuestions

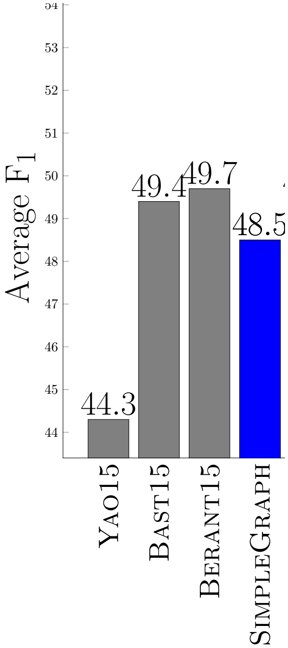
English



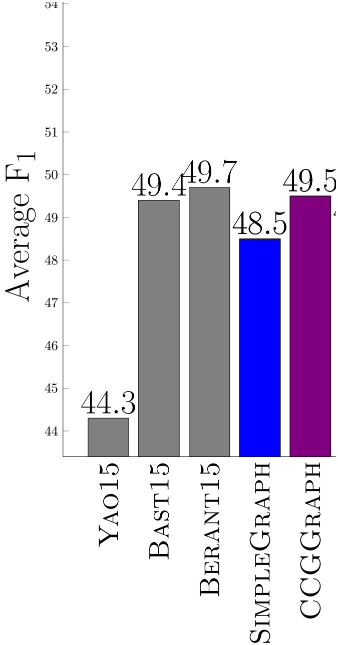
Results on WebQuestions



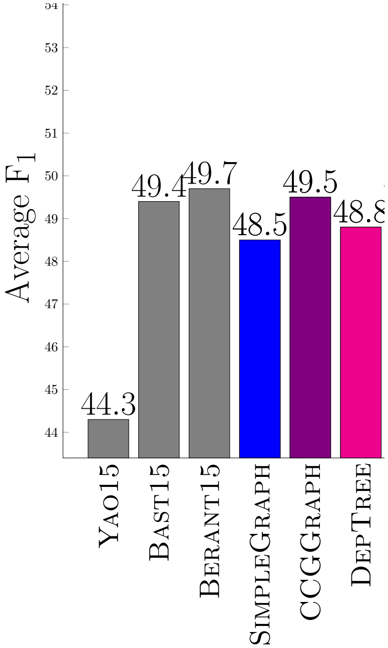
Results on WebQuestions



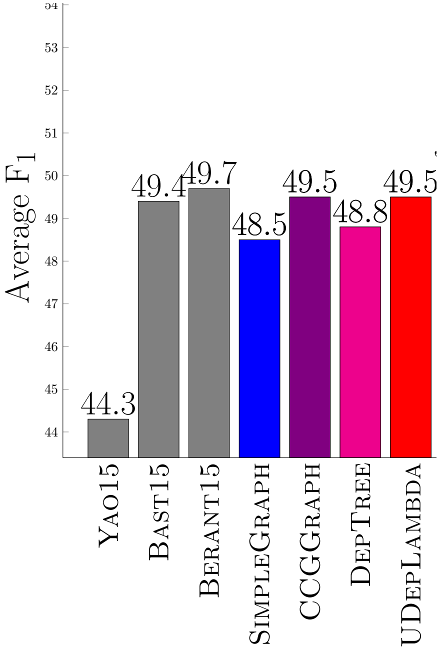
Results on WebQuestions



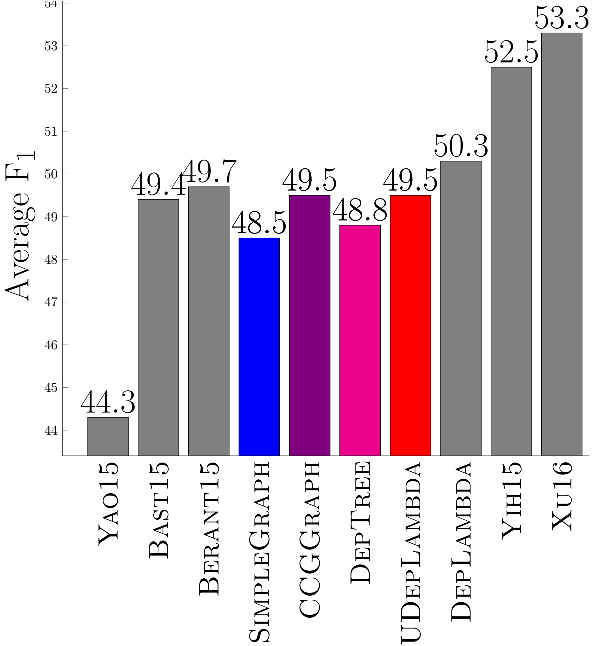
Results on WebQuestions



Results on WebQuestions

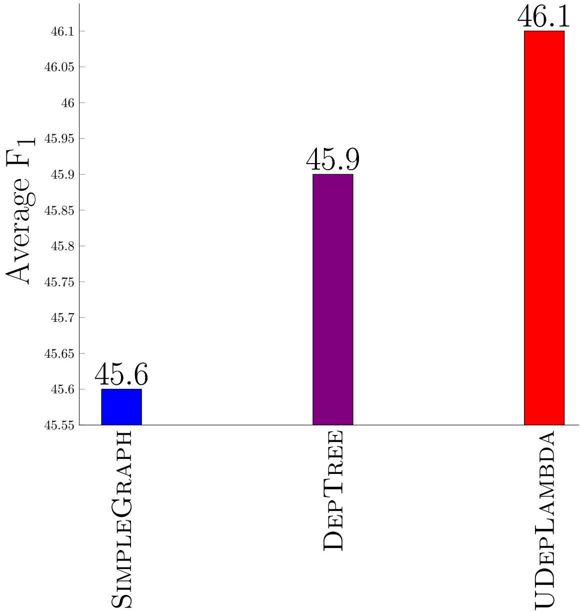


Results on WebQuestions



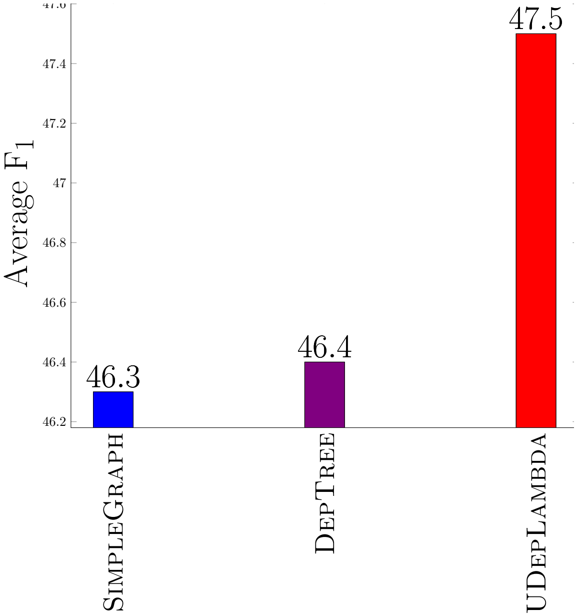
Results on Multilingual WebQuestions

German



Results on Multilingual WebQuestions

Spanish



Comparison with CCG

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

$$
\xrightarrow{\qquad\qquad\qquad\qquad\qquad\qquad} >
$$

$$
S\backslash NP
$$

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

$$
\xleftarrow{\qquad\qquad\qquad\qquad\qquad\qquad} <
$$

$$
S
$$

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

Comparison with CCG

CCG	UDepLambda
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	Universal

Limitations

Context-sensitive semantics of dependency labels, e.g., *nsubj* could mean either agent or patient

- John broke the window
- The window broke

Delexicalized context is not sufficient, e.g., quantifiers vs determiners

Quantifiers and Negation Scope

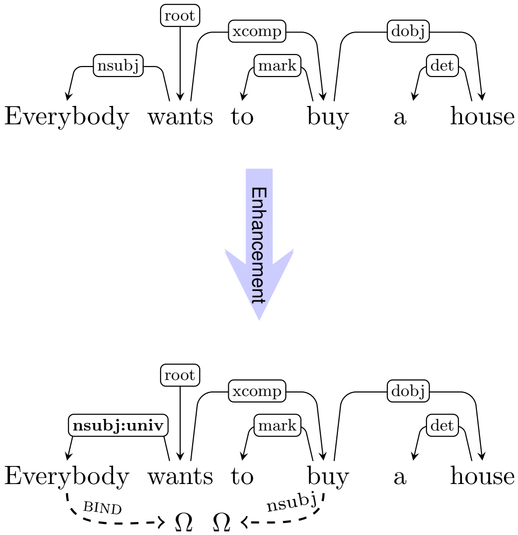
(Fancellu et al. 2017, Reddy et al. 2017)

Higher-order type system

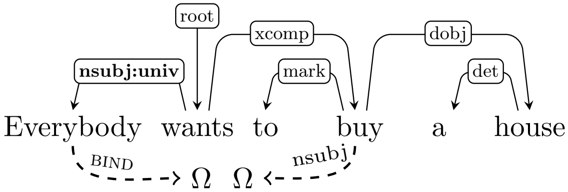
Fine-grained dependency labels

Quantifiers and Negation Scope

Fancellu et al. 2017, Reddy et al. 2017



Quantifiers and Negation Scope



Type System

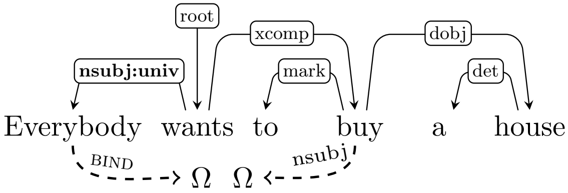
everybody $= \lambda x.\text{everybody}(x_a)$ [Old Type]

$= \lambda f.\forall x.\text{person}(x) \rightarrow f(x)$ [New Type]

wants $= \lambda x.\text{wants}(x_e)$ [Old Type]

$= \lambda f.\exists x.\text{wants}(x_e) \wedge f(x)$ [New Type]

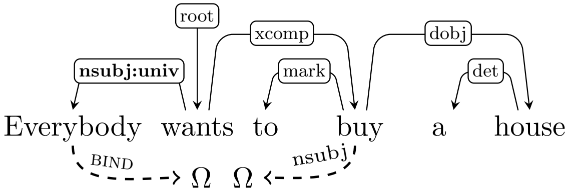
Quantifiers and Negation Scope



Type System

$$\begin{array}{lll}
\text{nsubj} & = \lambda fgx.\exists y.f(x) \wedge g(y) \wedge \arg_1(x_e, y_a) & \text{[Old]} \\
\text{nsubj:univ} & = \lambda PQf.Q(\lambda y.P(\lambda x.f(x) \wedge \arg_1(x_e, y_a))) & \text{[New]} \\
 & & \\
\text{dobj} & = \lambda fgx.\exists y.f(x) \wedge g(y) \wedge \arg_2(x_e, y_a) & \text{[Old]} \\
 & = \lambda PQf.P(\lambda x.f(x) \wedge Q(\lambda y.\arg_2(x_e, y_a))) & \text{[New]}
\end{array}$$

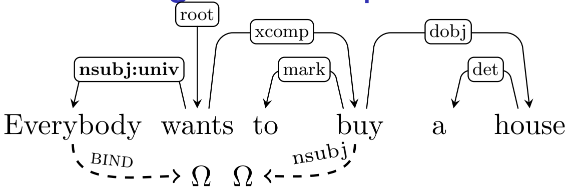
Quantifiers and Negation Scope



Type System

nsubj	$= \lambda fgx.\exists y.f(x) \wedge g(y) \wedge \mathrm{arg}_1(x_e, y_a)$	[Old]
nsubj:univ	$= \lambda PQf.Q(\lambda y.P(\lambda x.f(x) \wedge \mathrm{arg}_1(x_e, y_a)))$	[New]
dobj	$= \lambda fgx.\exists y.f(x) \wedge g(y) \wedge \mathrm{arg}_2(x_e, y_a)$	[Old]
	$= \lambda PQf.P(\lambda x.f(x) \wedge Q(\lambda y.\mathrm{arg}_2(x_e, y_a)))$	[New]

Quantifiers and Negation Scope



Old Expression:

(3) $\lambda z. \exists xyw. \mathrm{wants}(z_e) \wedge \mathrm{everybody}(x_a) \wedge \mathrm{arg}_1(z_e, x_a)$
$\wedge\ \mathrm{buy}(y_e) \wedge \mathrm{xcomp}(z_e, y_e) \wedge \mathrm{arg}_1(y_e, x_a)$
$\wedge\ \mathrm{arg}_1(x_e, y_a) \wedge \mathrm{house}(w_a) \wedge \mathrm{arg}_2(y_e, w_a)$.

New Expression:

(6) $\lambda f. \forall x\ . \mathrm{person}(x_a) \rightarrow$
$[\exists zyw. f(z) \wedge \mathrm{wants}(z_e) \wedge \mathrm{arg}_1(z_e, x_a) \wedge \mathrm{buy}(y_e)$
$\wedge\ \mathrm{xcomp}(z_e, y_e) \wedge \mathrm{house}(w_a)$
$\wedge\ \mathrm{arg}_1(z_e, x_a) \wedge \mathrm{arg}_2(z_e, w_a)]$.

UDepLambda: Present

Compositional Typed Semantics using Lambda Calculus

UDepLambda: Towards ...

Compositional Typed Semantics using Lambda Calculus

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

UDepLambda: Present

Compositional Typed Semantics using Lambda Calculus

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

Two universal types

UDepLambda: 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

UDepLambda: 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

UDepLambda: 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

Summary

Compositional Typed Semantic interface for Dependencies

Universal Semantic Parsing

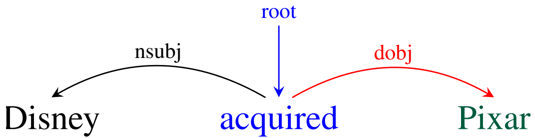
Freebase Semantic Parsing using Logical Forms

Demo at

https://sivareddy.in/udeplambda.html

Thank You!

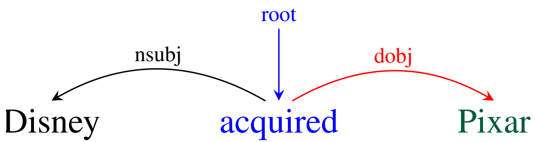
Single Type System



All constituents are of the same lambda expression type

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

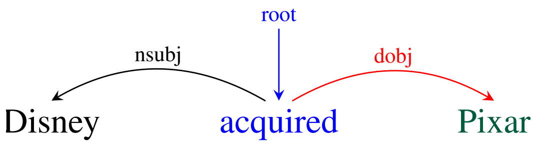
Single Type System



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

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

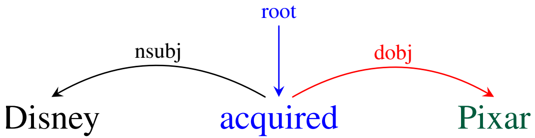
Single Type System



All constituents have a *lambda expression* of type η

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

Single Type System

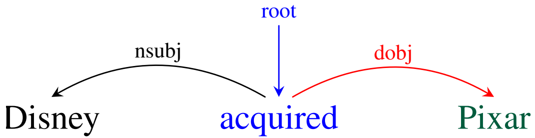


All constituents have a *lambda expression* of type η

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

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

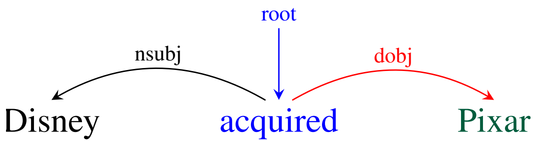
Single Type System



Lambda Expression for words

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

Single Type System



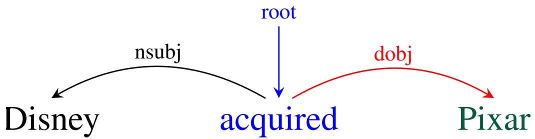
Lambda Expression for words

$$\text{acquired} \Rightarrow \lambda x_e.\,\mathsf{acquired}(x_e) \qquad \Rightarrow \textsf{TYPE} = \textbf{Event} \rightarrow \textbf{Bool}$$

$$\text{Pixar} \Rightarrow \lambda x_a.\,\mathsf{Pixar}(x_a) \qquad \Rightarrow \textsf{TYPE} = \textbf{Ind} \rightarrow \textbf{Bool}$$

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

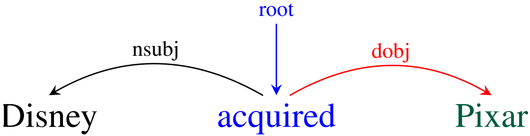
Single Type System



Lambda Expression for dependency labels

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

Single Type System

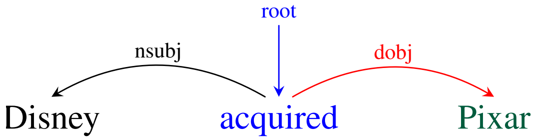


Lambda Expression for dependency labels

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

This operation mirrors the tree structure

Single Type System

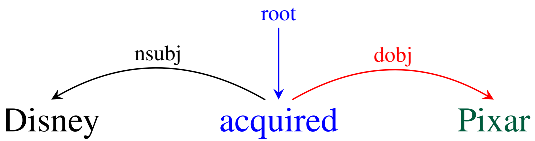


Lambda Expression for words

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

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

Single Type System



Lambda Expression for words

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

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

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

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

Logical Expression:

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

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

Logical Expression:

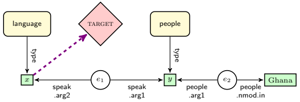
$$\lambda w.\exists xyz.\mathrm{Eminem}(x_a) \wedge \mathrm{coord}(w,y,z) \\ \wedge \mathrm{arg}_1(w_e,x_a) \wedge \mathrm{s\_to\_I}(y) \wedge \mathrm{d\_50}(z)$$

Post processing:

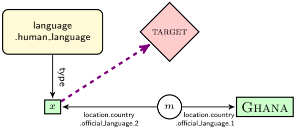
$$\lambda e.\exists xyz.\mathrm{Eminem}(x_a) \wedge \mathrm{arg}_1(y_e,x_a) \\ \wedge \mathrm{arg}_1(z_e,x_a) \wedge \mathrm{s\_to\_I}(y) \wedge \mathrm{d\_50}(z)$$

Graph Transformation: CONTRACT operation

What language do the people in Ghana speak?



Ungrounded graph



Grounded graph

Graph Mismatch: EXPAND operation

What to do Washington DC December?

Before EXPAND

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

After EXPAND

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