Large-scale Semantic Parsing via Schema Matching and Lexicon Extension

Qingqing Cai and Alexander Yates
Temple University

What is semantic parsing?

Who directed the Titanic?

Semantic Parsing

λx. *directed_by* (*titanic*, x)

Combinatory Categorial Grammar (CCG) Semantic Parsing

CCG Category

Syntax

Semantic: lambda expression

Operation

Application

Composition

Type Shifting

… …

S :
λx. *directed_by*(*titanic*, x)

S/(S\NP) :
λf. λw. λz. f(w, z)

S\NP :
λx. *directed_by*(*titanic*, x)

Forward application

S\NP/NP :
λx. λy. *directed_by*(x,y)

NP : *titanic*

Laxical Entry

Who

directed

Titanic

The problem

Training Examples

Who is the director of Titanic?
λx. *directed_by*(*titanic*, x)
Who produced the film Spider-Man?
λx. *produced_by*(*spider_man*, x)
… …

Training: Movie Domain

Learning Algorithm UBL
(Kwiatkowski et al., 2010)

Named Entity List

8 Mile :- NP : *8_mile*

Liam Gallagher :- NP
: *liam_gallagher*
… …

CCG grammar

Lexicon (word :- syntax : semantics)

director :- S/NP\NP : λx. λy. *directed_by*(y, x)
produced:- S/NP/NP : λx. λy. *produced_by*(y, x)
who :- S/(S/NP) : λx. λf. f(x)
where was :- S/(S\NP) : λx. λf. f(x)
… …

Testing: Movie Domain

Test Question:
Who produced 8 Mile?

Semantic Output:
λx. *produced_by*(*8_mile*, x)

The problem

Training Examples

Who is the director of Titanic?
λx. *directed_by*(*titanic*, x)
Who produced the film Spider-Man?
λx. *produced_by*(*spider_man*, x)
… …

Training: Movie Domain

Learning Algorithm UBL
(Kwiatkowski et al., 2010)

Named Entity List

8 Mile :- NP : *8_mile*

Liam Gallagher :- NP
: *liam_gallagher*
… …

CCG grammar

Lexicon (word :- syntax : semantics)

director :- S/NP\NP : λx. λy. *directed_by*(y, x)
produced:- S/NP/NP : λx. λy. *produced_by*(y, x)
who :- S/(S/NP) : λx. λf. f(x)
where was :- S/(S\NP) : λx. λf. f(x)
… …

Testing: Person Domain

Test Question:
Where was Liam Gallagher born?

NOT FOUND!

Semantic Output:
NO PARSE!!!

The Problem

Where was Liam Gallagher born?

born |- S/NP : $\lambda x.\ \lambda y.\ place\_of\_birth(x,y)$

Not found in training data!!!

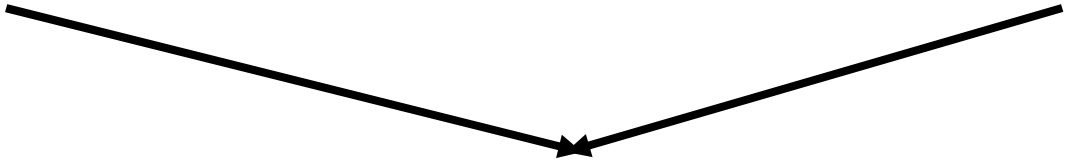
- **Why can't we correctly parse the questions?**
- **Why can't we get the correct lexical entry?**

Schema Matching

born :: *place_of_birth*

Lexicon Extension

born :: S/NP : $\lambda x.\ \lambda y.\ \lambda p.\ p(x,y)$



born |- S/NP : $\lambda x.\ \lambda y.\ place\_of\_birth(x,y)$

System

- **Schema Matching - Matcher**
- **Lexicon Extension - Lextender**

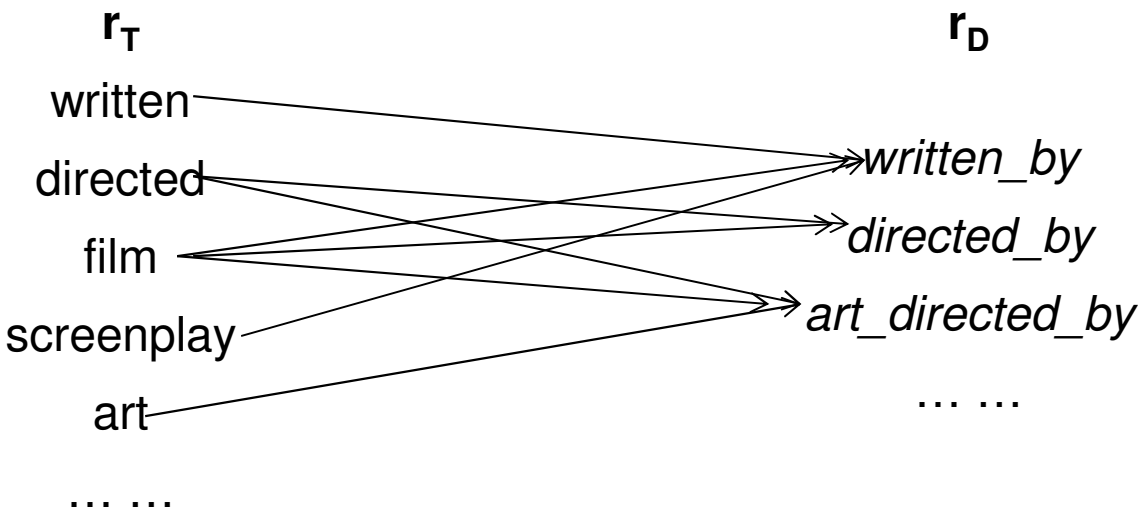
MATCHER - Task

- **Task**

  identify natural language words and phrases (r_T) which correspond with relations and entities in a fixed schema (r_D)

  Freebase

- **Example**



MATCHER - Method

- **Step1: Identify candidate matches**

- **Step2: Matching**
  - **Pattern-based matching selection**
  - **Instance-based matching selection**

Based on Logistic Regression
Trained on golden-standard matches

MATCHER – Identify candidate matches

r_D = "*film_music*"

Entities for "*film_music*", from Freebase	
arg1: *film*	**arg2: *contributer***
Spider-man	Danny Elfman
Twilight	Carter Burwell
Sleepless in Seattle	Marc Shaiman
Forrest Gump	Alan Silvestri
…	…

(1) Generate queries
Eg: "Spider-man" AND "Danny Elfman"

(2) Issue queries to the Web

Matching snippets for "*film_music*":

(1) ~~To~~ listen ~~***Spiderman***~~ Theme ~~***Danny Elfman***~~ music ~~just~~ click play ~~to~~ download.
(2) ~~***Twilight***~~ was directed ~~by~~ ~~***Catherine Hardwicke***~~ and composed ~~by~~ ~~***Carter Burwell***~~…
(3) … …

(1) Remove stop-words and entity-names;

(2) Rank remaining words by frequency;

Candidates for *film_music*

composed:: 0.4
directed :: 0.35
download :: 0.34
film :: 0.32
music:: 0.3
composers:: 0.21

MATCHER - Method

- **Step1: Identify candidate matches**

- **Step2: Matching**
  - **Pattern-based matching selection**
  - **Instance-based matching selection**

Based on Logistic Regression
Trained on golden-standard matches

MATCHER – Pattern-based match selection

- **Motivation**

  relation words appear in **simple**, **unambiguous** grammatical **patterns** that connects r_T with entities from r_D

  r_D = "*film_music*"

  r_T = {"composed", "directed", "download", …}

  e: an entity from r_D, with a specific type *contributor*

  #(composed by e) > 100*#(directed by e)

MATCHER – Pattern-based match selection

- **Pattern**

Pattern	Condition	Example
"r_T in E"	r_T ends with "-ed", and E has type *datetime* or *location*	"founded in 1989"
"r_T by E"	rT ends with "-ed"	"invented by Edison"
"r_T such as E"	rT ends with "-s"	"directors such as Tarantino"
"E is a(n) r_T"	all cases	"Paul Rudd is an acotr"

- **Why not use more patterns?**

  To cover all of Freebase, Machter needs **40 million** queries, or over **1.25 years** if issues 1 query per second

MATCHER – Pattern-based match selection

Candidates for *film_music*
composed:: 0.4
directed :: 0.35
download :: 0.34
film :: 0.32
music:: 0.3
composers:: 0.21

Pattern
"r_T in E"
"r_T by E"
"rat such as E"
"E is a(n) r_T"

(1) Generate pattern queries
Eg: "composed by Danny Elfman"
"composers such as Alan Silvestri"
"Alan Silvestri is a film"

(2) Issue pattern queries to the Web

pattern / (r_T, r_D)	r_T in E	r_T by E	r_T such as E	r_T is a(n) E	total	POS of r_T	…
(composed, *film_music*)	0	343	0	0	343	V	…
(directed, *film_music*)	0	30	0	0	20	V	…
(film, *film_music*)	0	0	0	431	431	N	…
(composed, *directed_by*)	0	21	0	0	21	V	…
…	…	…	…	…	…	…	…

$c(p, r_D, r_T)$: number of instances where r_T appears in **p** that involves entities from r_D

Logistic Regression Model

Candidates for *film_music*
composed:: 0.5
directed :: 0.4
film :: 0.4
composers:: 0.37
music:: 0.25
download :: 0.14

MATCHER - Method

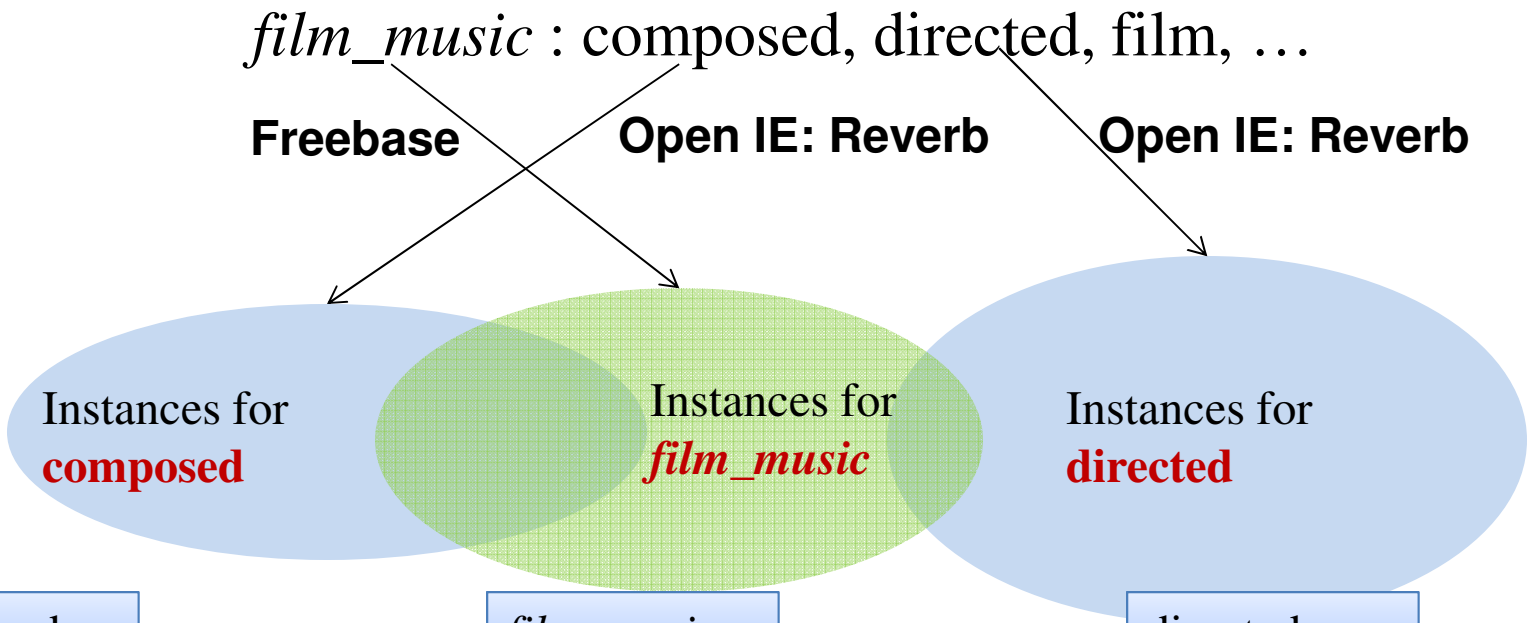
- **Step1: Identify candidate matches**

- **Step2: Matching**
  - **Pattern-based matching selection**
  - **Instance-based matching selection**

Based on Logistic Regression
Trained on golden-standard matches

MATCHER – Instance-based match selection

- **Motivation**



composed

film	*contributor*
Million Dollar Tree	Clint Eastwood
The Crack	Camilo Sanabria
…	…
…	…

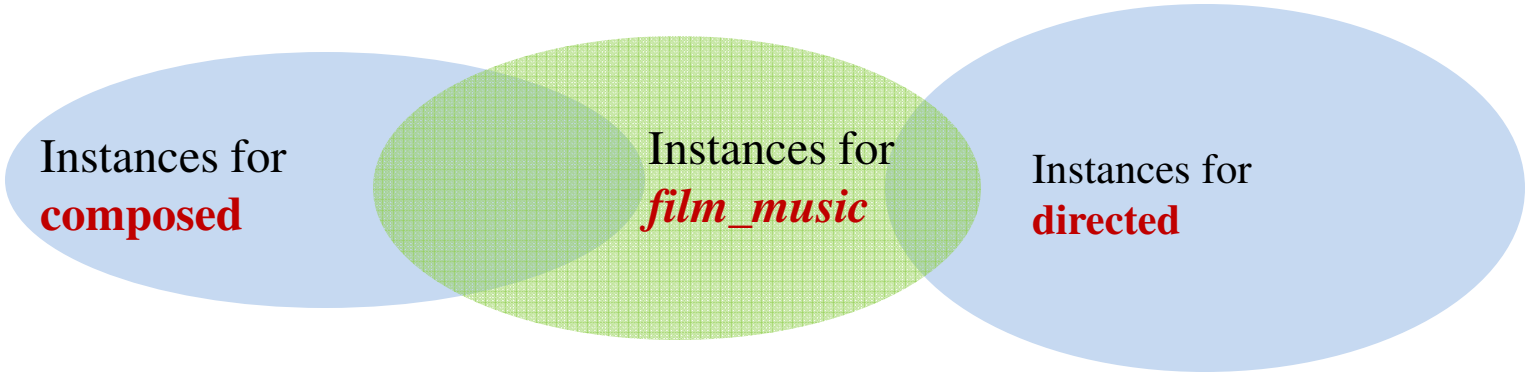
film_music

film	*contributor*
Million Dollar Tree	Clint Eastwood
The Crack	Camilo Sanabria
George Harrison	Water
…	…

directed

film	*contributor*
Million Dollar Tree	Clint Eastwood
The Brown Bunny	Mr. Gallo
Titanic	James Cameron
…	…

MATCHER – Instance-based match selection



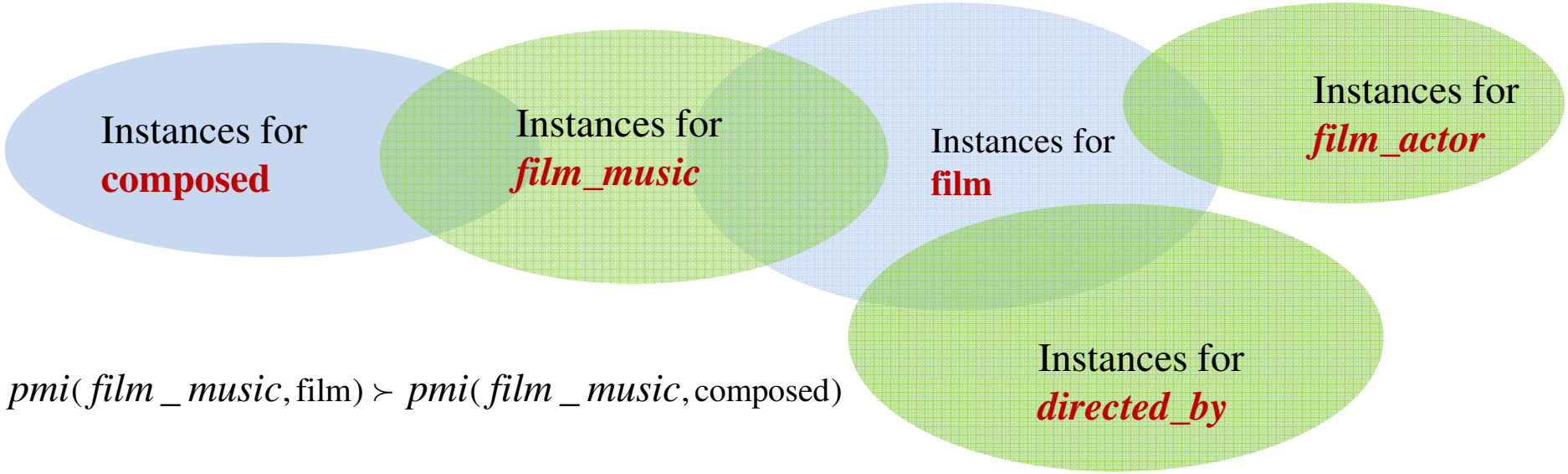
$$PMI(r_D, r_T) = \frac{|I_D(r_D) \cap I_T(r_T)|}{|I_D(r_D)| \cdot |I_T(r_T)|}$$

$I_D(r_D)$: instances extracted from database relations

$I_T(r_T)$: instances extracted from textual relations

$$pmi(film\_music, \text{composed}) \succ pmi(film\_music, \text{directed})$$

MATCHER – Instance-based match selection

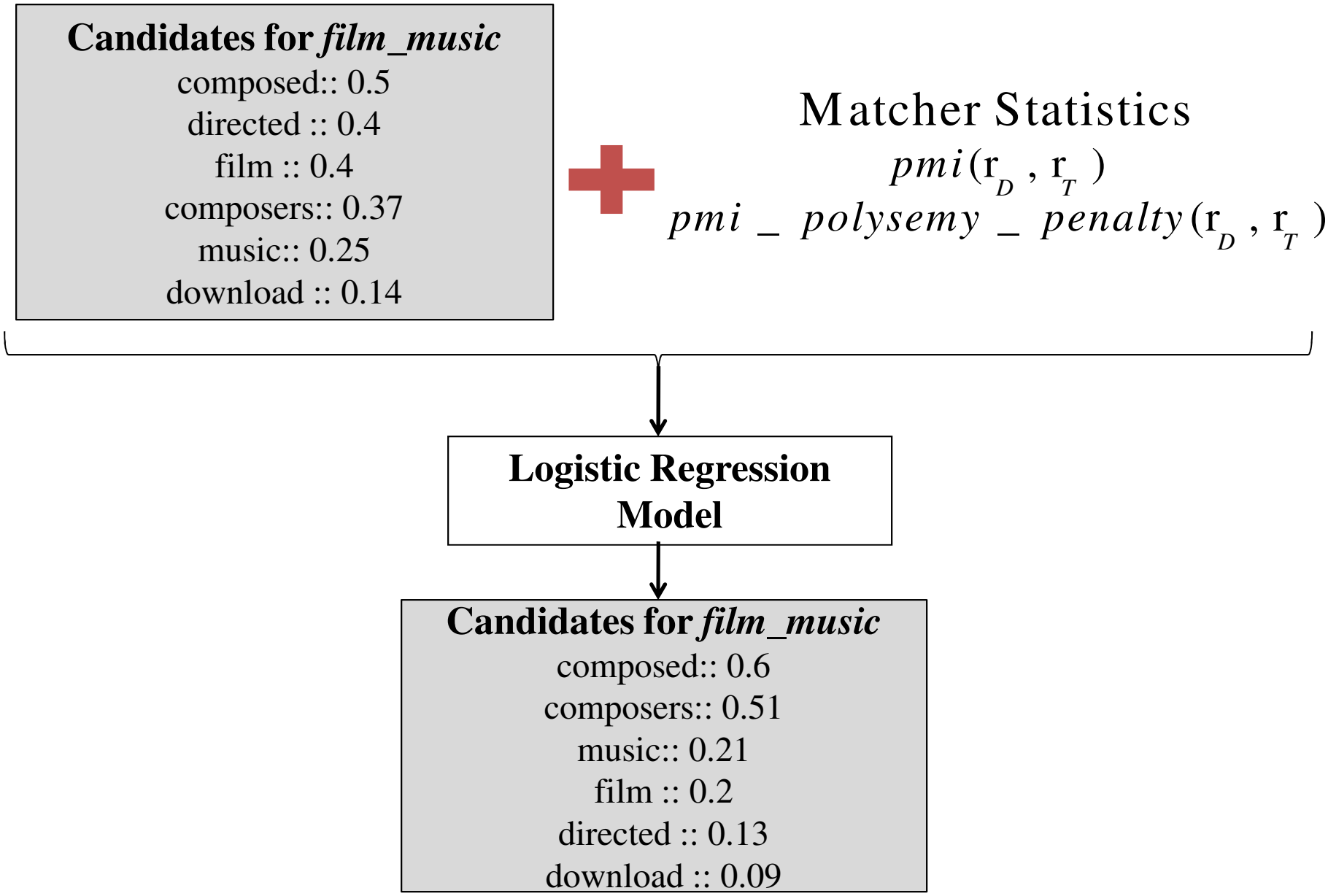


$$pmi(film\_music, \text{film}) \succ pmi(film\_music, \text{composed})$$

$$pmi\_polysemy\_penalty(r_D, r_T) = \frac{pmi(r_D, r_T)}{\sum_{r'_D} pmi(r'_D, r_T)}$$

$$pmi\_polysemy\_penalty(film\_music, \text{composed}) \succ pmi\_polysemy\_penalty(film\_music, \text{film})$$

MATCHER – Instance-based match selection



Our System's Architecture

- **Schema Matching - Matcher**
- **Lexicon Extension - Lextender**

LEXTENDER – Task

- **Input**

  Matching **M** together with **scores** for each pair (r_D, r_T) in M

  Lexical entries learned from UBL

- **Output**

  Predict the syntactic category ***Syn***, lambda-calculus semantics ***Sem***, and **weight** for a full lexical entry **L** for each pair in M

LEXTENDER – Method

(r_D, r_T)
(*located_in*, city)
(*award_winner_time*, won)

features: pos of r_T, suffix of r_T, #arguments of r_D, argument types of r_D

Naïve Bayes Classifier

features	matching_score	classification_score	…
(*located_in*, city) : S\NP/NP : λp λx λy. p(x, y)	0.7	2.1	…
(*award_winner_time*, won) (S|NP|NP)/NP : λp λx λy. p(x, y, z)	0.81	3.12	…

(r_D, r_T) : syn : sem
(*…ed_in*, city) : λp λx λy. p(x, y)
(*award_winner_time*, won) (S|NP|NP)/NP : λp λx λy. p(x, y, z)

(r_D, r_T) : syn : sem : weight
(*…ed_in*, city) : λp λx λy. p(x, y) : 0.217
(*award_winner_time*, won) (S|NP|NP)/NP : λp λx λy. p(x, y, z): 0.217

features	pos of r_T	# args	type of arg1	type of arg2	…
(*located_in*, city)	N	2	*location*	*location*	…
(*award_winner_time*, won)	V	3	*award*	*Award_winner*	…

Data Set

- **Dataset**
  - 917 natural language questions and logical forms
  - 81 domains and 635 relations

- **Split into three groups for cross-domain validation**

Group1		Group2		Group3	
Film	49	business	46	Tv	34
people	29	location	29	award	32
book	21	organization	24	medicine	24
…	...	…	…	…	…
Total	300	total	310	total	307

Experiment 1 – Matcher Test

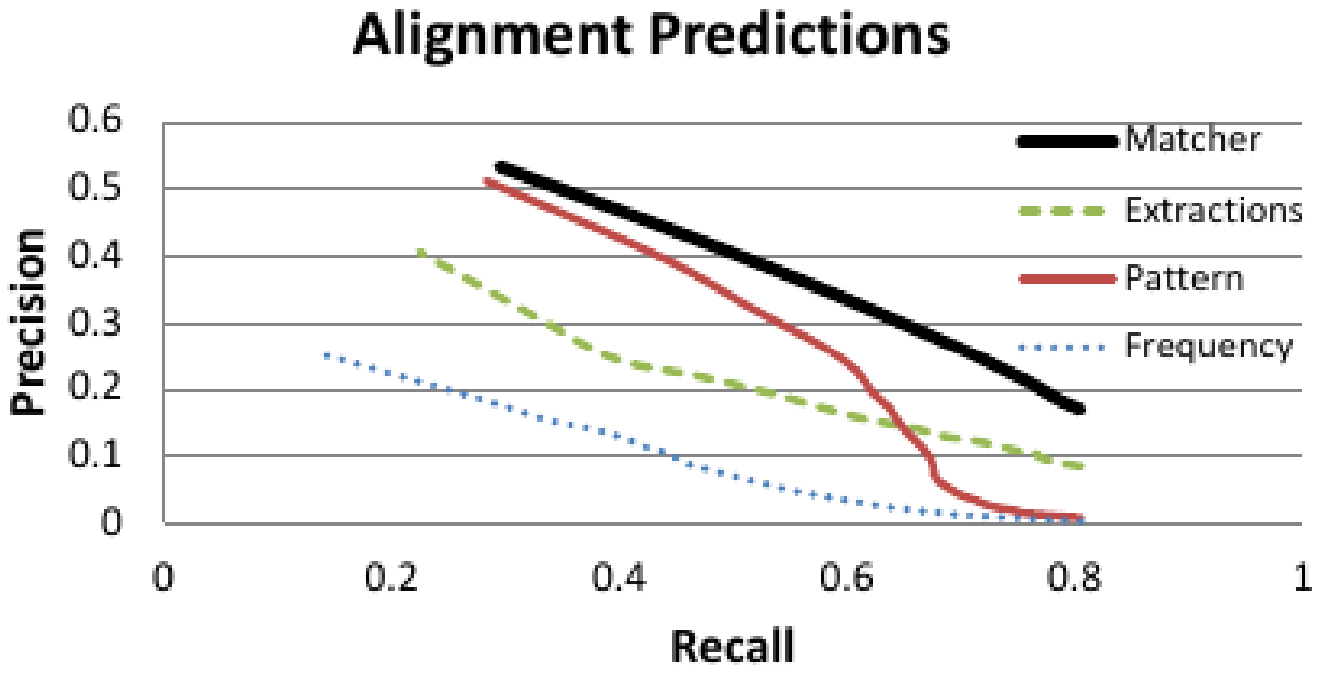
- **Setting**
  Frequency: get candidates
  Pattern: pattern-based matching selection
  Extractions: instance-based matching selection
  Matcher: Pattern + Extractions Selection

- **Evaluation**
  M: (r_D, r_T) matches predicted by models
  G: matches in the gold-standard manual data

$$\text{precision}: P = \frac{|M \cap G|}{|M|} \qquad \text{recall}: R = \frac{|M \cap G|}{|G|}$$

Experiment 1 – Matcher Test

- **Performance**



- **Lessons**

  (1) Pattern, Extractions, Matcher > Frequency

  (2) Matcher yields the best precision at all levels of recall

Experiment 2 – Semantic Parsing Test

- **Setting**

  70/30 split and cross-domain validation

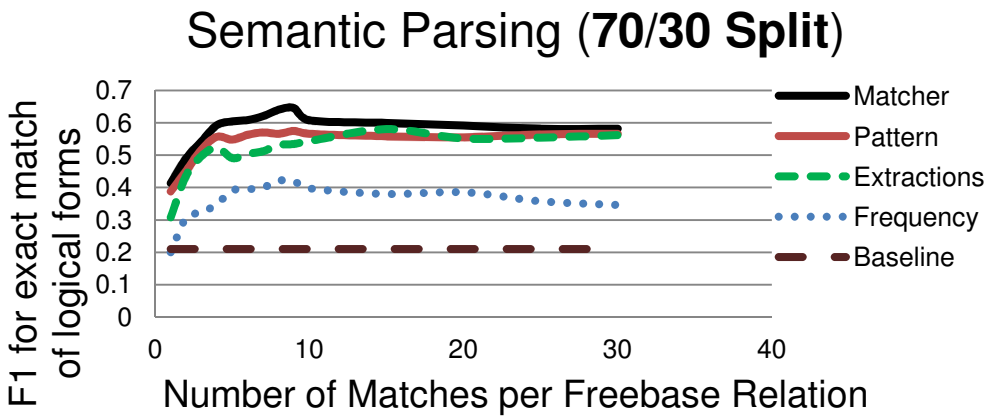
  Baseline: UBL (supervised semantic parsing)
  Frequency: get candidates + LEXTENDER + UBL
  Pattern: pattern-based matching selection + LEXTENDER + UBL
  Extractions: instance-based matching selection + LEXTENDER + UBL
  Matcher: Pattern + Extractions + LEXTENDER + UBL

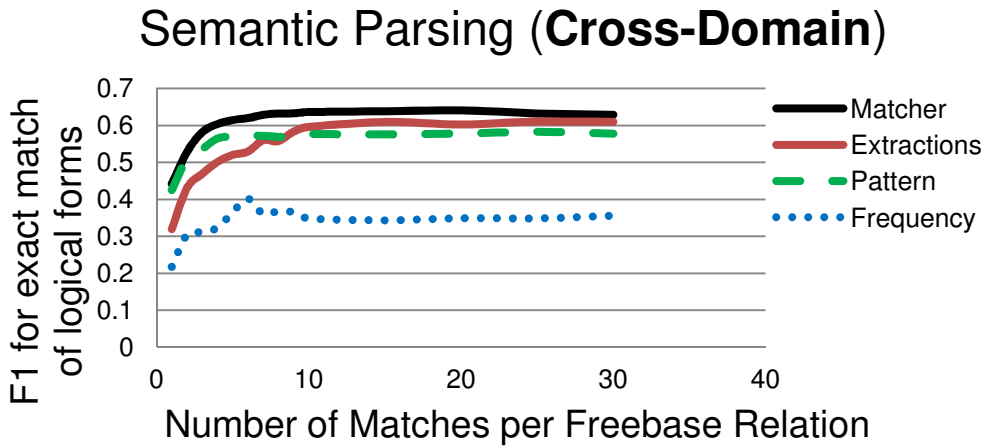
- **Evaluation**

  Exact match between top-scoring and manually-produced logical forms

Experiment 2 – Semantic Parsing Test

- **Performance**





Lessons:

(1) Matcher, Pattern, Extractions, Frequency > Baseline;

(2) Matcher > Frequency, Pattern, Extractions

Conclusion

- **Problem**
  (1) How to handle testing questions which are **significantly different** from labeled training questions

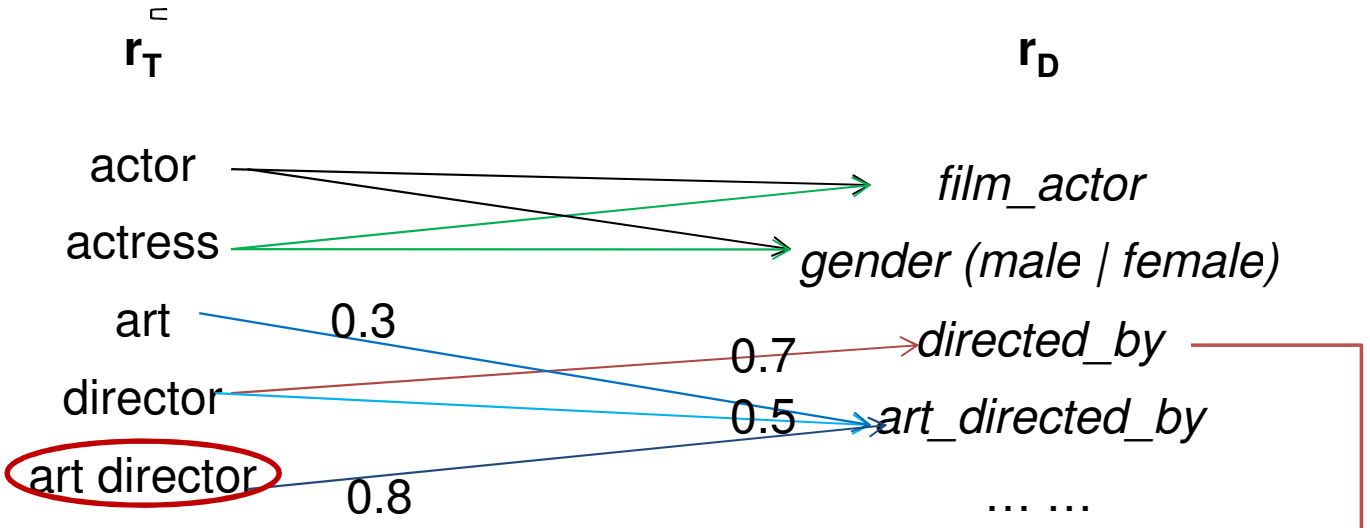
- **Solution**
  MATCHER + LEXTENDER
  (1) MATCHER: find correspondences between words and database symbols
  (2) LEXTENDER: integrate (word, database symbol) correspondences into a semantic parsing lexicon

- **Result**
  MATCHER + LEXTENDER + UBL: semantic parsers can handle Freebase questions on a large variety of domains with an F1 of **0.63**

Future Work

- **Future work**

  (1) Handle more complex matches between database and textual relations



  (2) Handle more complex natural language questions

  who is the **art director** of the Titanic?

  who is the **director** of the Titanic?

Thanks for your attention!