Macro Grammars and Holistic Triggering for Efficient Semantic Parsing

Yuchen Zhang and Panupong Pasupat and Percy Liang

EMNLP 2017

Year	Competition	Venue	Position	Event	Notes
Representing Poland					
2001	World Youth Championships	Debrecen, Hungary	2nd	400 m	47.12
			1st	Medley relay	1:50.46
	European Junior Championships	Grosseto, Italy	1st	4x400 m relay	3:06.12
2003	European Junior Championships	Tampere, Finland	3rd	400 m	46.69
			2nd	4x400 m relay	3:08.62
2005	European U23 Championships	Erfurt, Germany	11th (sf)	400 m	46.62
			1st	4x400 m relay	3:04.41
	Universiade	Izmir, Turkey	7th	400 m	46.89
			1st	4x400 m relay	3:02.57
2006	World Indoor Championships	Moscow, Russia	2nd (h)	4x400 m relay	3:06.10
	European Championships	Gothenburg, Sweden	3rd	4x400 m relay	3:01.73
2007	European Indoor Championships	Birmingham, United Kingdom	3rd	4x400 m relay	3:08.14
	Universiade	Bangkok, Thailand	7th	400 m	46.85
			1st	4x400 m relay	3:02.05
2008	World Indoor Championships	Valencia, Spain	4th	4x400 m relay	3:08.76
	Olympic Games	Beijing, China	7th	4x400 m relay	3:00.32
2009	Universiade	Belgrade, Serbia	2nd	4x400 m relay	3:05.69

In what city did Piotr's last 1st place finish occur?

Year	Competition	Venue	Position	Event	Notes
Representing Poland					
2001	World Youth Championships	Debrecen, Hungary	2nd	400 m	47.12
			1st	Medley relay	1:50.46
	European Junior Championships	Grosseto, Italy	1st	4x400 m relay	3:06.12

How long did it take this competitor to finish the 4x400 meter relay at Universiade in 2005?

Where was the competition held immediately before the one in Turkey?

How many times has this competitor placed 5th or better in competition?

	Olympic Games	Beijing, China	7th	4x400 m relay	3:00.32
2009	Universiade	Belgrade, Serbia	2nd	4x400 m relay	3:05.69

In what city did Piotr's last 1st place finish occur?

Semantic Parsing

Parse utterances into executable logical forms

"Who ranked right after Turkey?"

Rank	Nation	Gold	Silver	Bronze
1	France	3	1	1
2	Turkey	2	0	1
3	Sweden	2	0	0

Semantic Parsing

Parse utterances into executable logical forms

"Who ranked right after Turkey?"

NationOf.NextOf.HasNation.Turkey

Rank	Nation	Gold	Silver	Bronze
1	France	3	1	1
2	Turkey	2	0	1
3	Sweden	2	0	0

Semantic Parsing

Parse utterances into executable logical forms

"Who ranked right after Turkey?"

NationOf.NextOf.HasNation.Turkey

Rank	Nation	Gold	Silver	Bronze
1	France	3	1	1
2	Turkey	2	0	1
3	Sweden	2	0	0

Semantic Parsing

Parse utterances into executable logical forms

"Who ranked right after Turkey?"

NationOf.NextOf.HasNation.Turkey

Rank	Nation	Gold	Silver	Bronze
1	France	3	1	1
2	Turkey	2	0	1
3	Sweden	2	0	0

Semantic Parsing

Parse utterances into executable logical forms

"Who ranked right after Turkey?"

NationOf.NextOf.HasNation.Turkey

Rank	Nation	Gold	Silver	Bronze
1	France	3	1	1
2	Turkey	2	0	1
3	Sweden	2	0	0

Semantic Parsing

Parse utterances into executable logical forms

"Who ranked right after Turkey?"

NationOf.NextOf.HasNation.Turkey

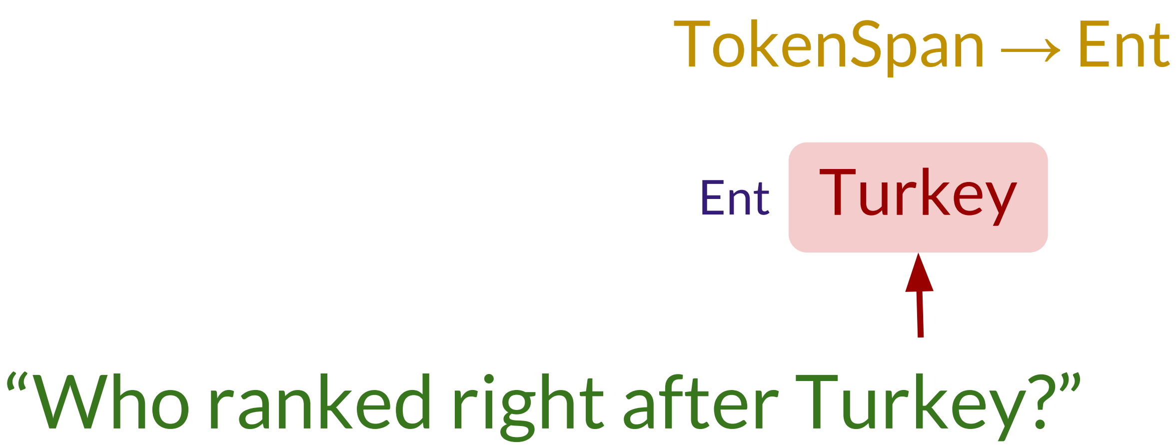
Rank	Nation	Gold	Silver	Bronze
1	France	3	1	1
2	Turkey	2	0	1
3	$\boxed{\text{Sweden}}$	2	0	0

Denotation

Floating Parser

Given an utterance, the parser composes logical forms using a grammar

- Terminal rules generate terminal tokens



Floating Parser

Given an utterance, the parser composes logical forms using a grammar

- Terminal rules generate terminal tokens

$\varnothing \longrightarrow$ Rel

Rel Nation

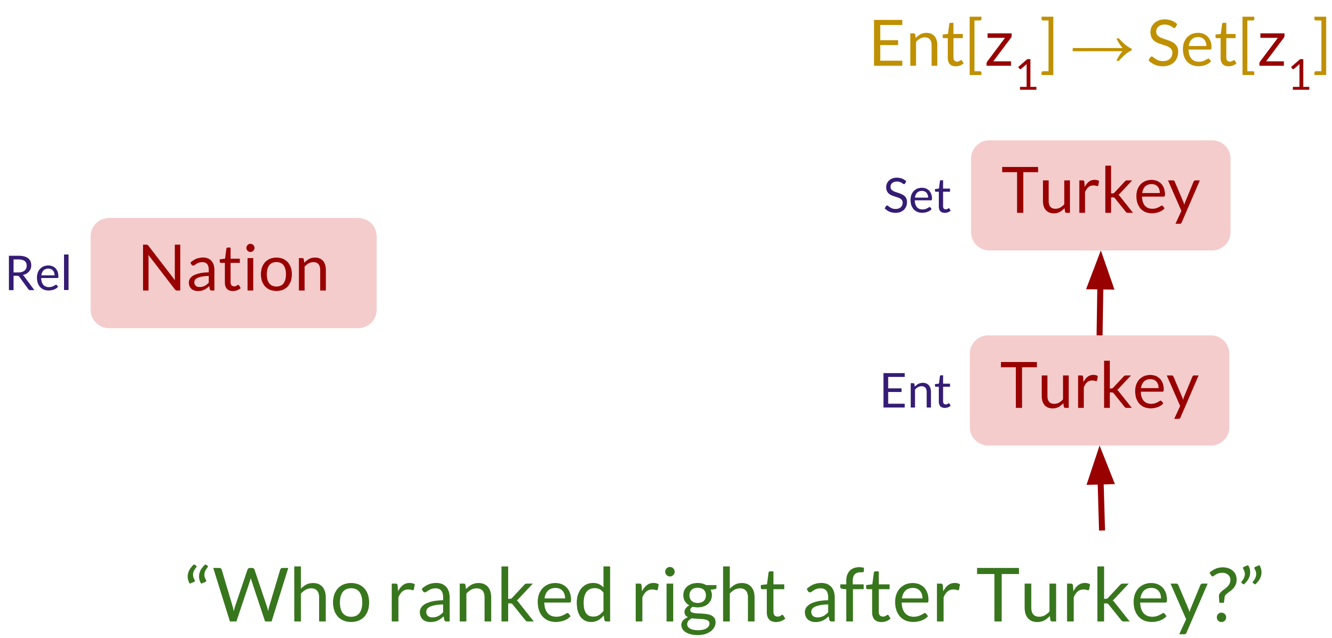
Ent Turkey

"Who ranked right after Turkey?"

Floating Parser

Given an utterance, the parser composes logical forms using a grammar

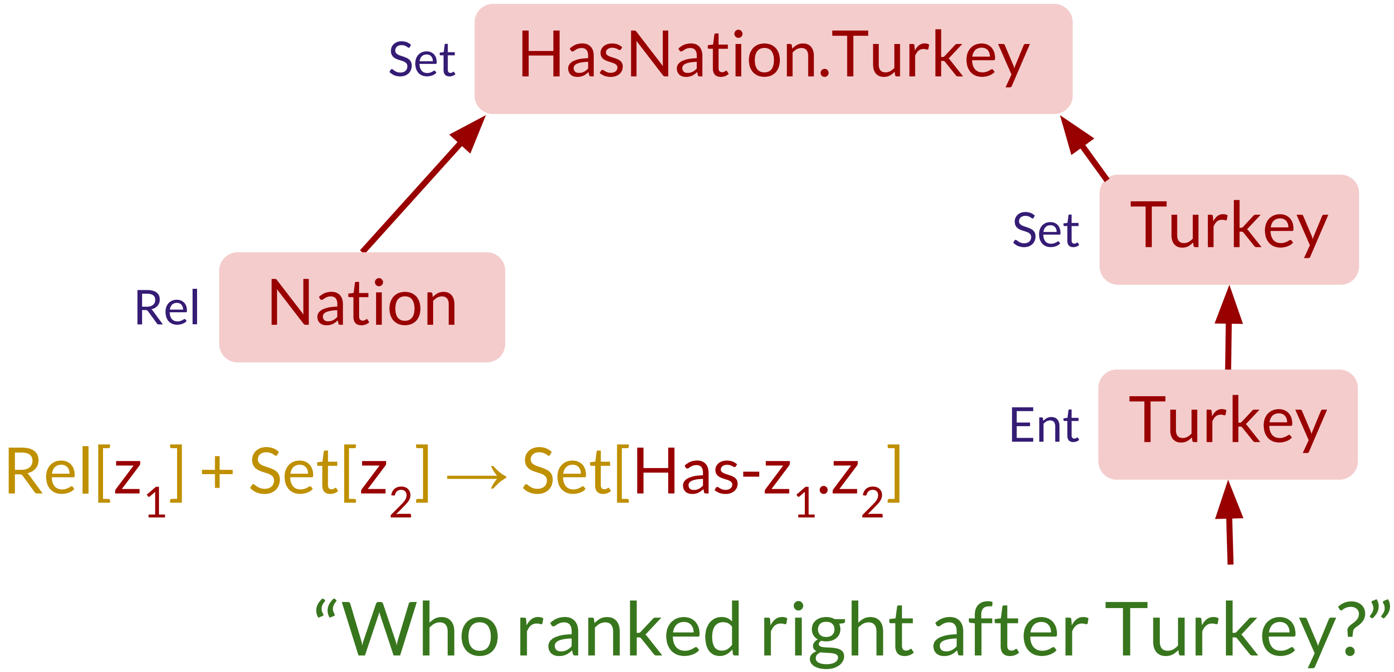
- Compositional rules combine parts



Floating Parser

Given an utterance, the parser composes logical forms using a grammar

- Compositional rules combine parts




Root
NationOf.NextOf.HasNation.Turkey
Set
NationOf.NextOf.HasNation.Turkey
Set
NextOf.HasNation.Turkey
Set
HasNation.Turkey
Rel
Nation
Set
Turkey
Ent
Turkey
“Who ranked right after Turkey?”

Training a Semantic Parser

Setup: Each training example has an utterance, a table, and the target denotation

- The logical form is latent

"Who ranked right after Turkey?"

Sweden

Rank	Nation	Gold	Silver	Bronze
1	France	3	1	1
2	Turkey	2	0	1
3	Sweden	2	0	0

Training a Semantic Parser

Given a training example:

1. Generate a bunch of logical forms (beam search)
2. Featurize the logical forms and score them

Rank	Nation	Gold	Silver	Bronze
1	France	3	1	1
2	Turkey	2	0	1
3	Sweden	2	0	0

NationOf.NextOf.HasNation.Turkey

NationOf.HasNext.HasNation.Turkey

count(HasNation.Turkey)

"Who ranked right after Turkey?"

Training a Semantic Parser

Given a training example:

1. Generate a bunch of logical forms (beam search)
2. Featurize the logical forms and score them
3. Execute the logical forms to identify the ones that are consistent with the target denotation
4. Gradient update toward consistent logical forms

Rank	Nation	Gold	Silver	Bronze
1	France	3	1	1
2	Turkey	2	0	1
3	Sweden	2	0	0

NationOf.NextOf.HasNation.Turkey

NationOf.HasNext.HasNation.Turkey

count(HasNation.Turkey)

"Who ranked right after Turkey?"

Training a Semantic Parser

Given a training example:

1. Generate a bunch of logical forms (beam search)
2. Featurize the logical forms and score them
3. Execute the logical forms to identify the ones that are consistent with the target denotation
4. Gradient update toward consistent logical forms

Rank	Nation	Gold	Silver	Bronze
1	France	3	1	1
2	Turkey	2	0	1
3	Sweden	2	0	0

NationOf.NextOf.HasNation.Turkey

NationOf.HasNext.HasNation.Turkey

count(HasNation.Turkey)

"Who ranked right after Turkey?"

Main Problem: Speed

Depending on the generality of the grammar, the number of generated partial logical forms can grow exponentially

count(NextOf.HasNation.Turkey)

sum(IndexOf.HasNation.Turkey)

argmax(NextOf.HasNation.Turkey, Index)

- Many partial logical forms are also useless

Main Problem: Speed

Depending on the generality of the grammar, the number of generated partial logical forms can grow exponentially

- To reach 40% accuracy, each example:
  - Generates ~ 13700 partial logical forms
  - Takes ~ 1.1 seconds (2.6 GHz machine)
  - 3 epochs on 14K examples → 12 hours

Main Problem: Speed

Depending on the generality of the grammar, the number of generated partial logical forms can grow exponentially

- To reach 40% accuracy, each example:
  - Generates ~ 13700 partial logical forms
  - Takes ~ 1.1 seconds (2.6 GHz machine)
  - 3 epochs on 14K examples → 12 hours

Our contribution: 11x speedup

Main Ideas

Idea 1: Macros

- Good logical forms share common patterns (“macro”)
- Restrict the generation to such macros

Idea 2: Holistic Triggering

- There are still too many macros
- Only use macros from logical forms with similar utterances

Idea 1: Macros

Good logical forms usually share useful patterns ("macros")

NationOf.NextOf.HasNation.Turkey

Idea 1: Macros

Good logical forms usually share useful patterns ("macros")

NationOf.NextOf.HasNation.Turkey

{REL1}Of.NextOf.Has{REL1}.{ENT2}

~ What {REL1} comes after {ENT2}

Idea 1: Macros

Good logical forms usually share useful patterns ("macros")

NationOf.NextOf.HasNation.Turkey

{REL1}Of.NextOf.Has{REL1}.{ENT2}

~ What {REL1} comes after {ENT2}

- When we find a consistent logical form in one example, we want to cache and reuse its macro in other examples

Training Algorithm

Given a training example:

- Try applying macros found in previous examples to generate logical forms
- If a consistent logical form is found:
  - Do gradient update as usual
- Otherwise:
  - Fall back to the full compositional search

Macro Grammar

We encode the macros as grammar rules ("macro rules") so that we can use the same beam search algorithm to generate logical forms from macros

NationOf.NextOf.HasNation.Turkey

{REL1}Of.NextOf.Has{REL1}.{ENT2}

Macro Grammar

We encode the macros as grammar rules ("macro rules") so that we can use the same beam search algorithm to generate logical forms from macros

NationOf.NextOf.HasNation.Turkey

{REL1}Of.NextOf.Has{REL1}.{ENT2}

$$\mathrm{Rel}[z_1] + \mathrm{Ent}[z_2] \rightarrow \mathrm{Root}[z_1\text{-}\mathrm{Of.NextOf.Has}\text{-}z_1.z_2]$$

(Rel and Ent are built by terminal rules)

Training Algorithm

Given a training example:

- Try applying macros found in previous examples
- If a consistent logical form is found:
  - Do gradient update as usual
- Otherwise:
  - Fall back to the full compositional search

Training Algorithm Revised

Maintain a list R of macro rules

Given a training example:

- Apply beam search on R + terminal rules
- If a consistent logical form is found:
  - Do gradient update as usual
- Otherwise:
  - Fall back to beam search on the base grammar
  - If a consistent logical form is found, extract its macro and augment R

Decomposed Macro Rules

Some macros share parts

max(RankOf.HasGold.>.2)

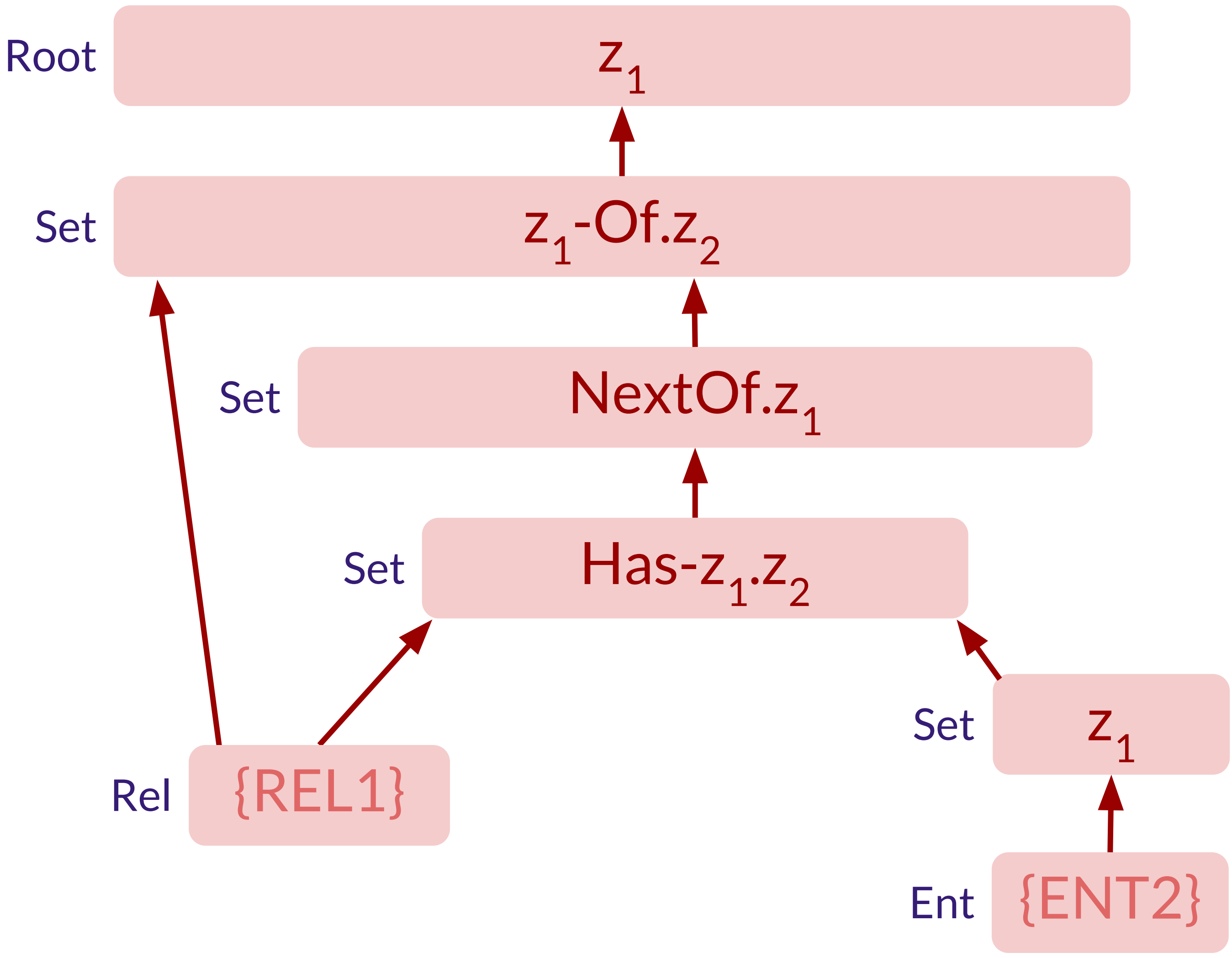
max({REL1}Of.Has{REL2}.>.{ENT3})

NationOf.argmin(HasSilver.>.2, Index)

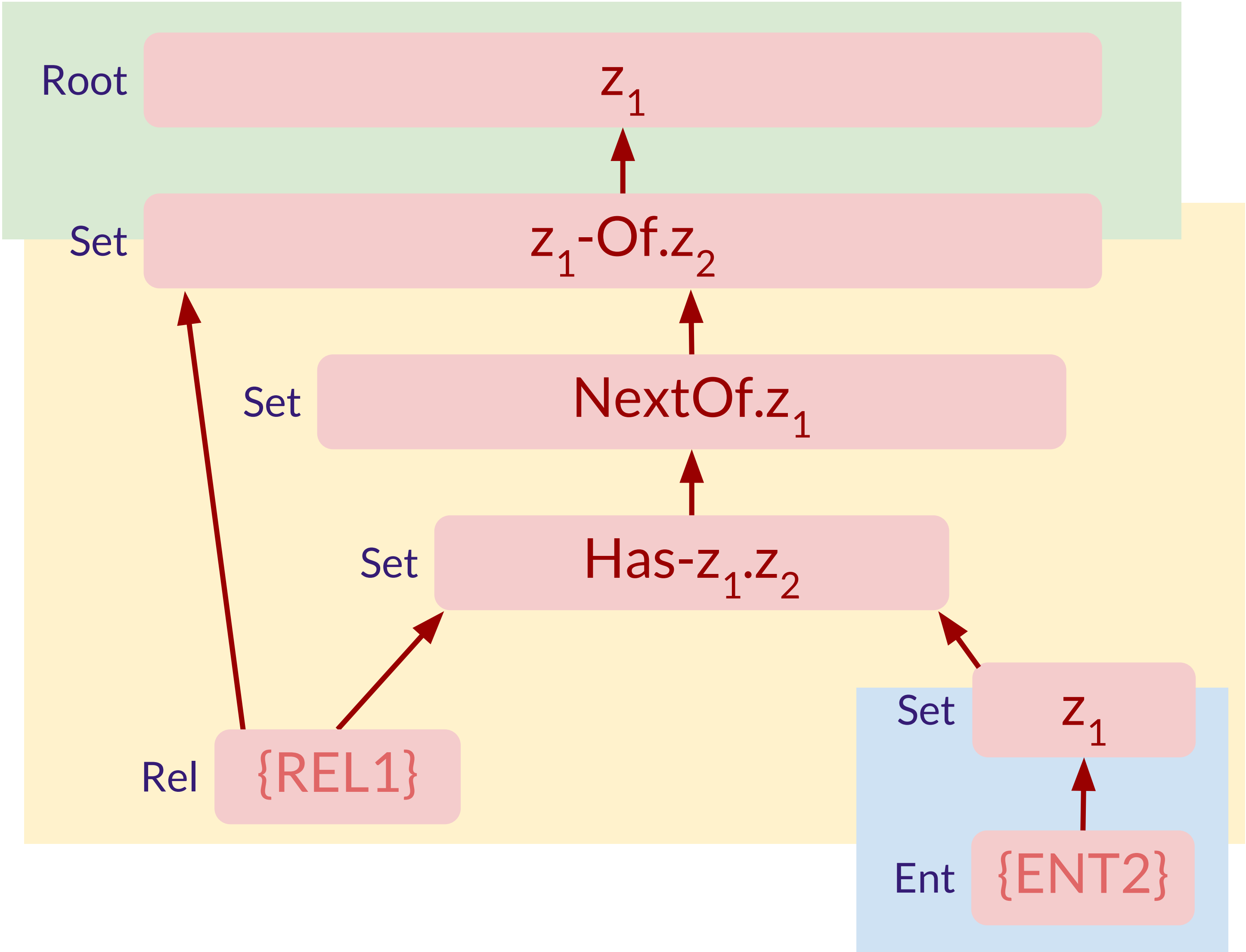
{REL1}Of.argmin(Has{REL2}.>.{ENT3}, Index)

If we need to try both macros, it would be nice to have to featurize the shared part only once

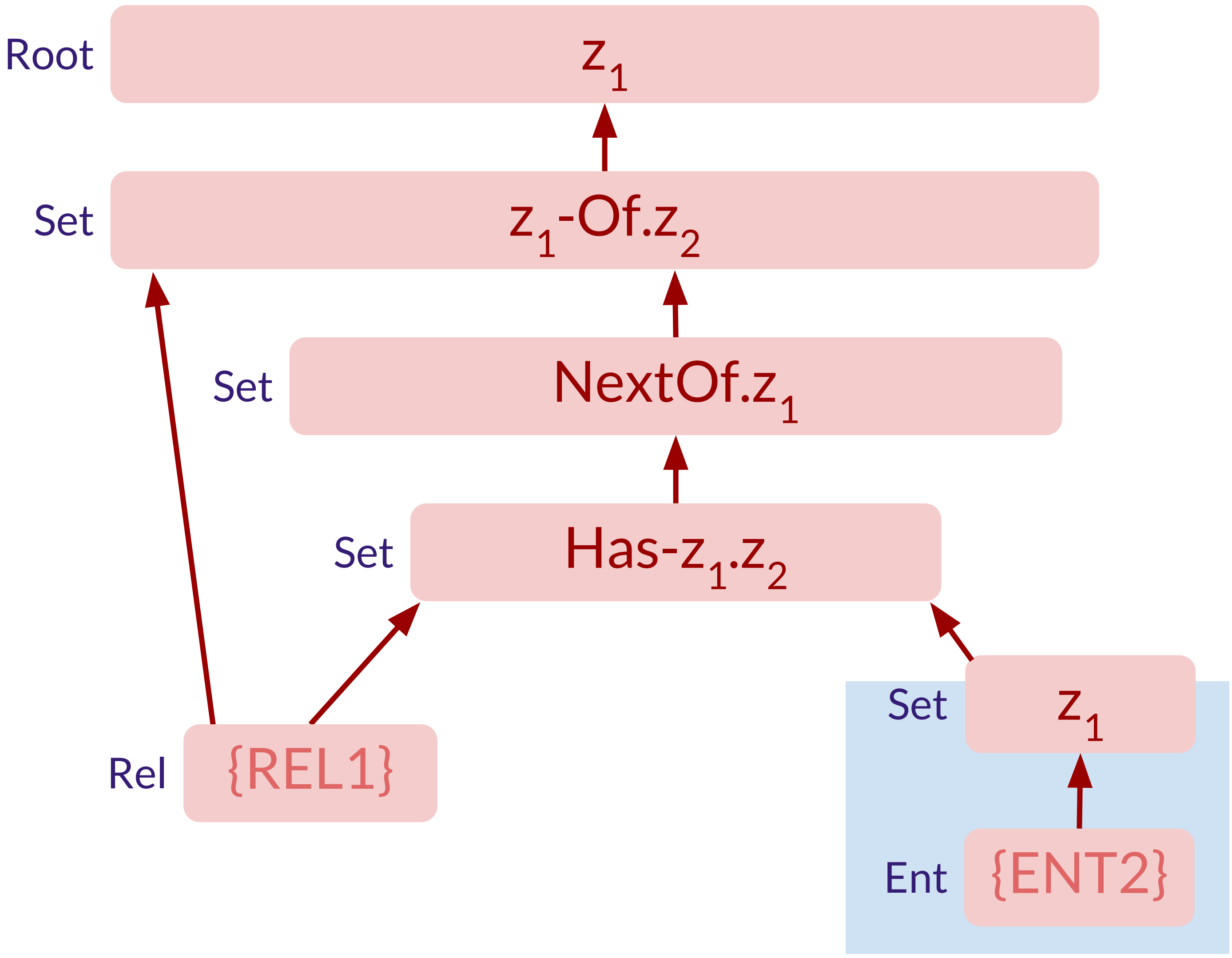
Root
NationOf.NextOf.HasNation.Turkey
Set
NationOf.NextOf.HasNation.Turkey
Set
NextOf.HasNation.Turkey
Set
HasNation.Turkey
Set
Turkey
Rel
Nation
Ent
Turkey
“Who ranked right after Turkey?”



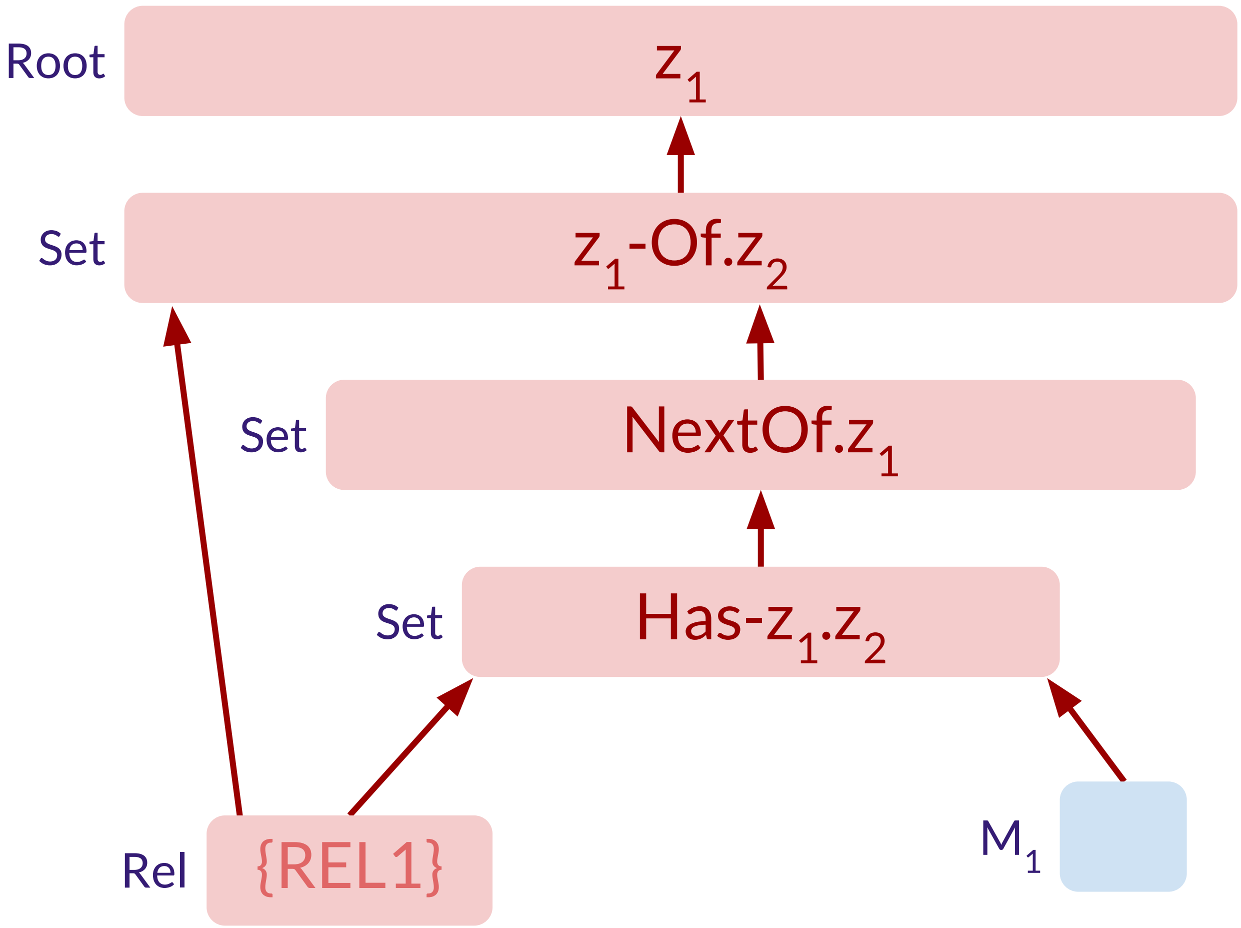
Root
z_1
Set
z_1-Of.z_2
Set
NextOf.z_1
Set
Has-z_1.z_2
Set
z_1
Rel
{REL1}
Ent
{ENT2}



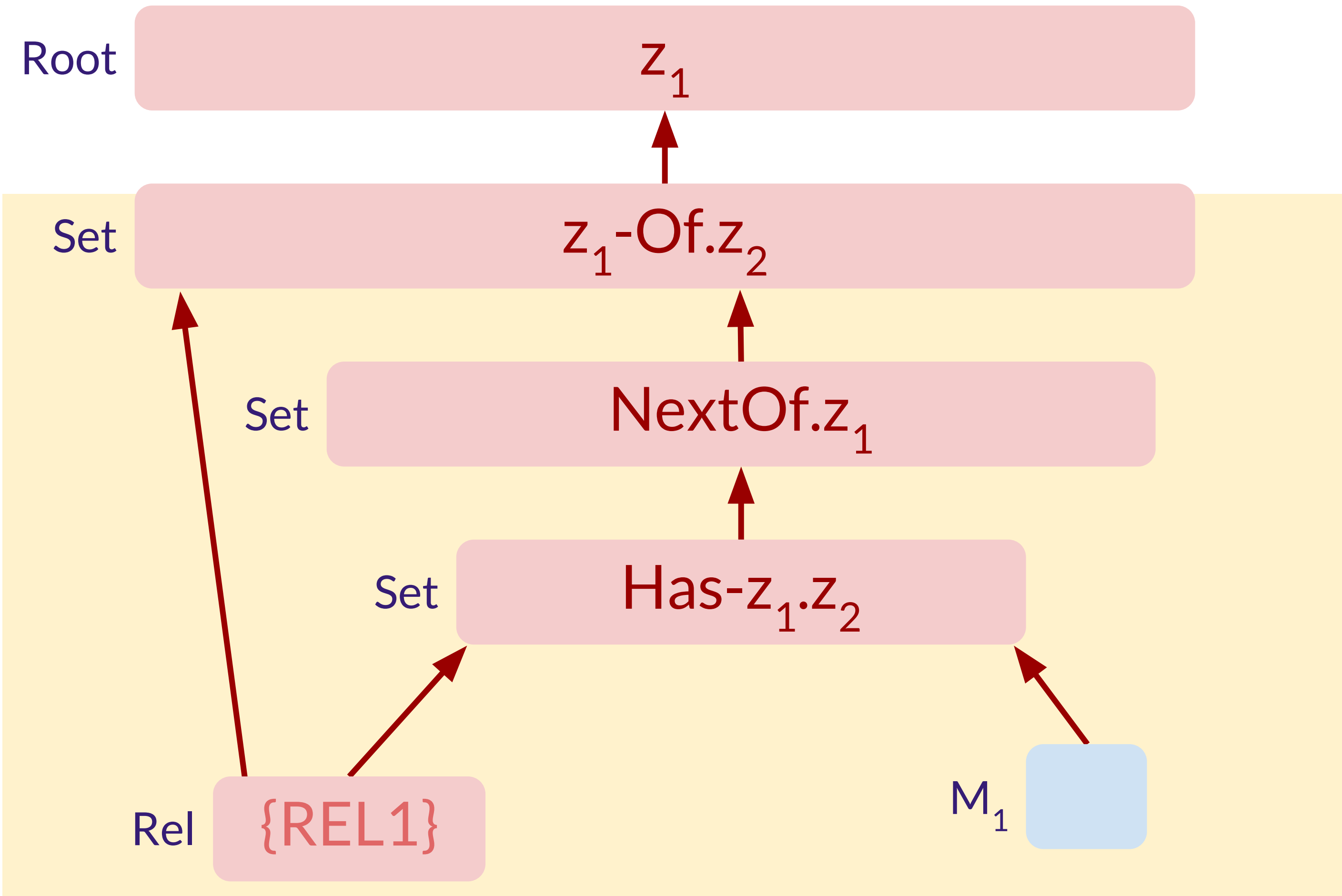
Root
z_1
Set
z_1-Of.z_2
Set
NextOf.z_1
Set
Has-z_1.z_2
Rel
{REL1}
Set
z_1
Ent
{ENT2}



$$\mathrm{Ent}[z_1] \rightarrow M_1[z_1]$$

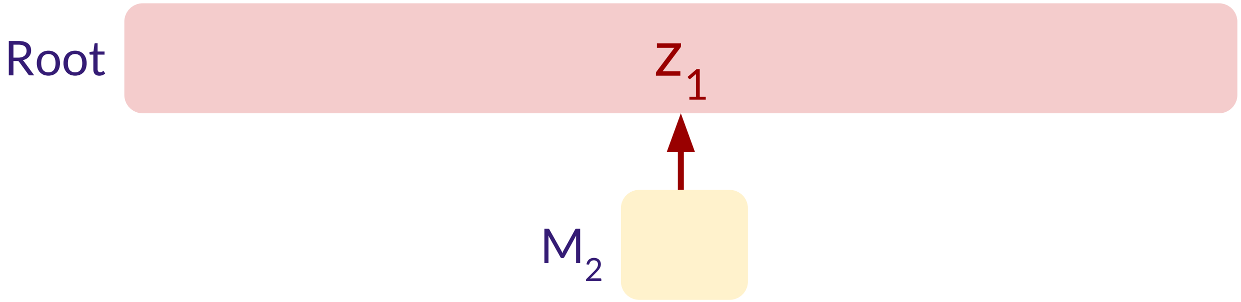


$$\mathrm{Ent}[z_1] \rightarrow M_1[z_1]$$



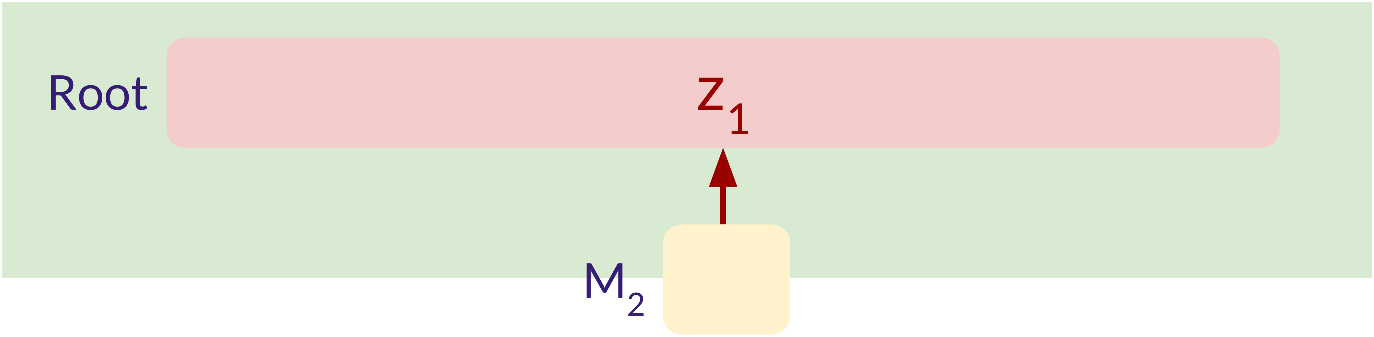
$$\mathrm{Rel}[z_1] + M_1[z_2] \rightarrow M_2[z_1\text{-Of.NextOf.Has-}z_1.z_2]$$

$$\mathrm{Ent}[z_1] \rightarrow M_1[z_1]$$



$$\mathrm{Rel}[z_1] + M_1[z_2] \longrightarrow M_2[z_1\text{-Of.NextOf.Has-}z_1.z_2]$$

$$\mathrm{Ent}[z_1] \longrightarrow M_1[z_1]$$



$$M_2[z_1] \rightarrow \mathrm{Root}[z_1]$$

$$\mathrm{Rel}[z_1] + M_1[z_2] \rightarrow M_2[z_1\text{-Of.NextOf.Has-}z_1.z_2]$$

$$\mathrm{Ent}[z_1] \rightarrow M_1[z_1]$$

Training Algorithm Revised

Maintain a list R of macro rules

Given a training example:

- Apply beam search on R + terminal rules
- If a consistent logical form is found:
  - Do gradient update as usual
- Otherwise:
  - Fall back to beam search on the base grammar
  - If a consistent logical form is found, extract its macro and augment R with decomposed rules

Training Algorithm Revised

Maintain a list R of macro rules

Given a training example:

- Apply beam search on R + terminal rules
- If a consistent logical form is found:
  - Do gradient update as usual
- Otherwise:
  - Fall back to beam search on the base grammar
  - If a consistent logical form is found, extract its macro and augment R with decomposed rules

New Problem: R grows with the number of examples

Idea 2: Holistic Triggering

Instead of using all macro rules, use only a subset

An ideal subset R' of macro rules should:

- be able to generate consistent logical form
- be small (to save time)

How do we choose such a subset?

Idea 2: Holistic Triggering

Observation: Similar utterances tend to give logical forms with identical or similar macros

"Who ranked right after Turkey?"

NationOf.NextOf.HasNation.Turkey

{REL1}Of.NextOf.Has{REL1}.{ENT2}

"Who took office right after Uriah Forrest?"

NameOf.NextOf.HasName.UriahForrest

{REL1}Of.NextOf.Has{REL1}.{ENT2}

Idea 2: Holistic Triggering

We select which macro rules to use based on utterance similarity:

- Compute edit distances between the current utterance and utterances in previous examples
  - Word-level Levenshtein after removing determiners and infrequent nouns
- Get the K = 40 nearest neighbors
- Get the macro rules from the consistent logical forms found in those examples

Final Training Algorithm

Maintain a list R of macro rules

Given a training example:

- Holistic triggering $\rightarrow$ macro rule subset R'
- Apply beam search on R' + terminal rules
- If a consistent logical form is found:
  - Do gradient update as usual
- Otherwise:
  - Fall back to beam search on the base grammar
  - If a consistent logical form is found, extract its macro and augment R with decomposed rules

Final Training Algorithm

Maintain a list R of macro rules

Given a training example:

- Holistic triggering → macro rule subset R'
- Apply beam search on R' + terminal rules
- If a consistent logical form is found:
  - Do gradient update as usual
- Otherwise, if it is the first epoch:
  - Fall back to beam search on the base grammar with early stopping (found a consistent LF or generated 5000 LFs)
  - If a consistent logical form is found, extract its macro and augment R with decomposed rules

Prediction

Maintain a list R of macro rules

Given a test example:

- Holistic triggering $\longrightarrow$ macro rule subset R'
- Apply beam search on R' + terminal rules
- Return the logical form with the highest score

Related work

UBL (Unification Based Learning)

(Kwiatkowski et al., 2010; 2011)

next_to(ny, vt)

New York borders Vermont

Related work

UBL (Unification Based Learning)

(Kwiatkowski et al., 2010; 2011)

ny	λx.λy.next_to(x, y)	vt
New York	borders	Vermont

Related work

UBL (Unification Based Learning)

(Kwiatkowski et al., 2010; 2011)

ny λx.λy.next_to(x, y) vt

New York borders Vermont

borders λx.λy.next_to(x, y)

Related work

UBL (Unification Based Learning)

(Kwiatkowski et al., 2010; 2011)

borders $\lambda x.\lambda y.next\_to(x, y)$

UBL / Factored UBL	Our Work
Learn templates and the possible words to trigger them	Learn templates; choose templates with holistic triggering
Templates are for lexicon learning	Templates are for speed

Experiment: Accuracy

	Dev	Test
SEMPRE 2015 (Pasupat and Liang, 2015)	37.0	37.1
Neural Programmer (Neelakantan et al., 2016)	37.5	37.7
Neural Multi-Step Reasoning (Haug et al., 2017)	-	38.7

(averaged over 3 dev splits)

Experiment: Accuracy

	Dev	Test
SEMPRE 2015 (Pasupat and Liang, 2015)	37.0	37.1
Neural Programmer (Neelakantan et al., 2016)	37.5	37.7
Neural Multi-Step Reasoning (Haug et al., 2017)	-	38.7
Base Grammar (SEMPRE 2015 ++)	40.6	42.7

- Improved the TokenSpan → Ent rule
- Added a few compositional rules
- Changed the objective function to “first good vs first bad” instead of log-likelihood

Experiment: Accuracy

	Dev	Test
SEMPRE 2015 (Pasupat and Liang, 2015)	37.0	37.1
Neural Programmer (Neelakantan et al., 2016)	37.5	37.7
Neural Multi-Step Reasoning (Haug et al., 2017)	-	38.7
Base Grammar (SEMPRE 2015 ++)	40.6	42.7
Macro Grammar	40.4	43.7

Experiment: Accuracy

	Dev	Test
SEMPRE 2015 (Pasupat and Liang, 2015)	37.0	37.1
Neural Programmer (Neelakantan et al., 2016)	37.5	37.7
Neural Multi-Step Reasoning (Haug et al., 2017)	-	38.7
Base Grammar (SEMPRE 2015 ++)	40.6	42.7
Macro Grammar	40.4	43.7
Krishnamurthy et al., 2017	42.7 (5 dev splits)	43.3
Krishnamurthy et al., 2017 (ensemble)	-	45.9

Experiment: Speed

(averaged over 3 dev splits)

	Accuracy	Time (ms/example)	
		Train	Predict
SEMPRE 2015	37.0	619	645
Base Grammar	**40.6**	1117	1150
Macro Grammar	40.4	**99**	**70**

Experiment: Speed

(averaged over 3 dev splits)

	Accuracy	Time (ms/example)	
		Train	Predict
SEMPRE 2015	37.0	619	645
Base Grammar	**40.6**	1117	1150
Macro Grammar	40.4	**99**	**70**
Macro Grammar No macro decomposition	40.3	177	159
Macro Grammar No holistic triggering	40.1	361	369

Experiment: Coverage

	Found a consistent LF
SEMPRE 2015	76.6%
Base Grammar	81.0%
Macro Grammar	75.6%

- Restricted search space → Smaller chance to get a consistent LF

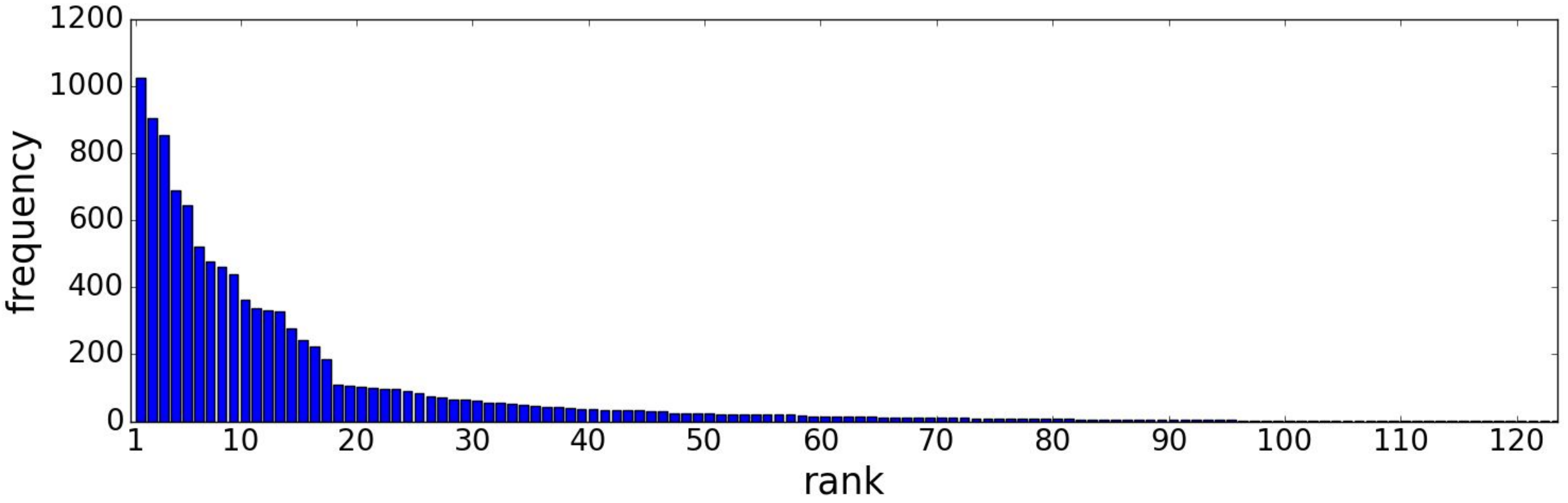
Experiment: Coverage

(over 300 examples)

	Found a consistent LF	**Top consistent LF is semantically correct**
SEMPRE 2015	76.6%	-
Base Grammar	81.0%	48.7%
Macro Grammar	75.6%	48.7%

- Restricted search space → Smaller chance to get a consistent LF
- But the ability to find a semantically correct LF remains the same

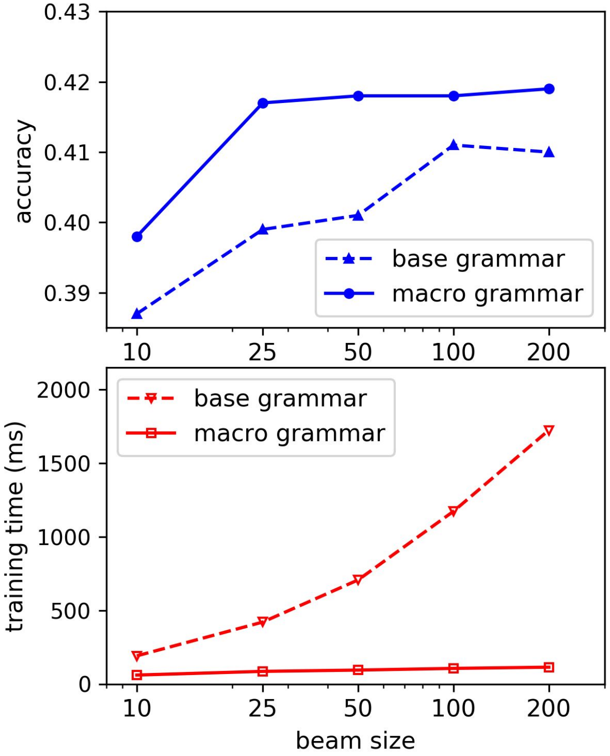
Experiment: Coverage



- Extracted 123 macros
- Top 34 macros cover 90% of the consistent logical forms found
- Most of the top 34 macros have clear semantics

Experiment: Tradeoffs

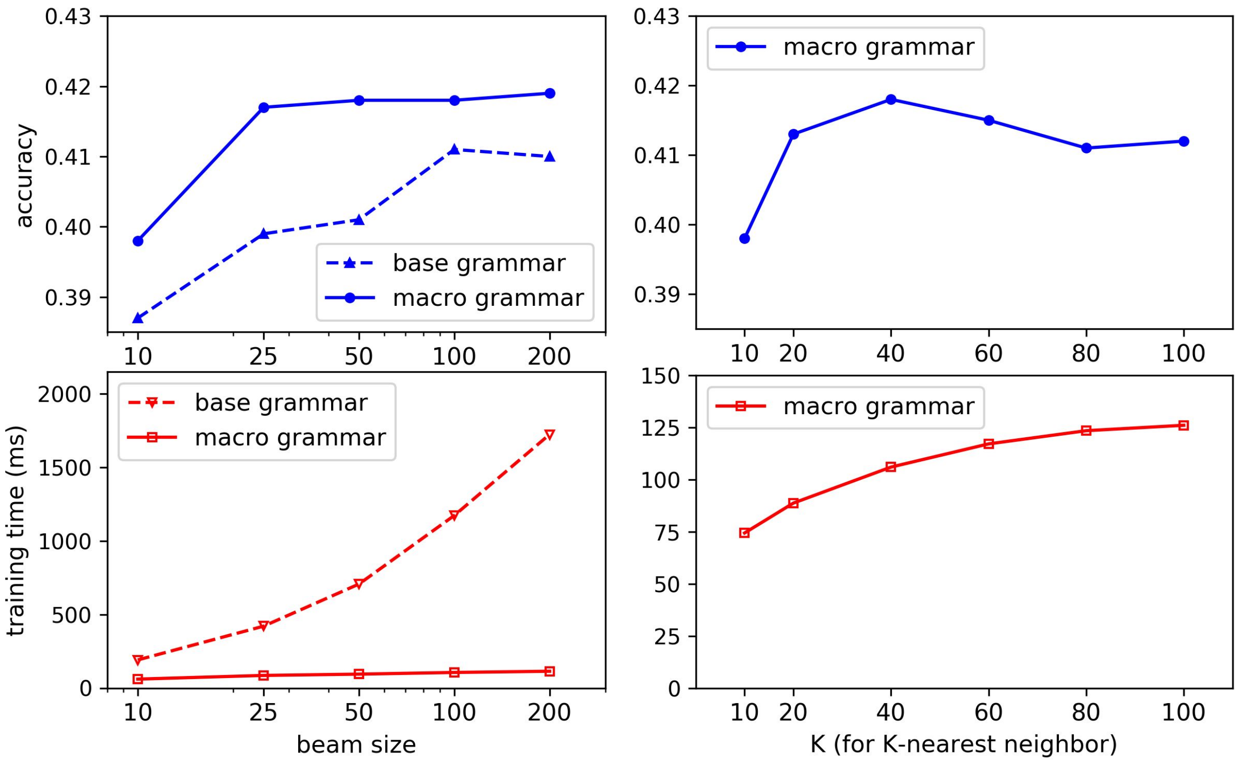
(first dev split)



- Macro grammar is fast even with larger beam sizes

Experiment: Tradeoffs

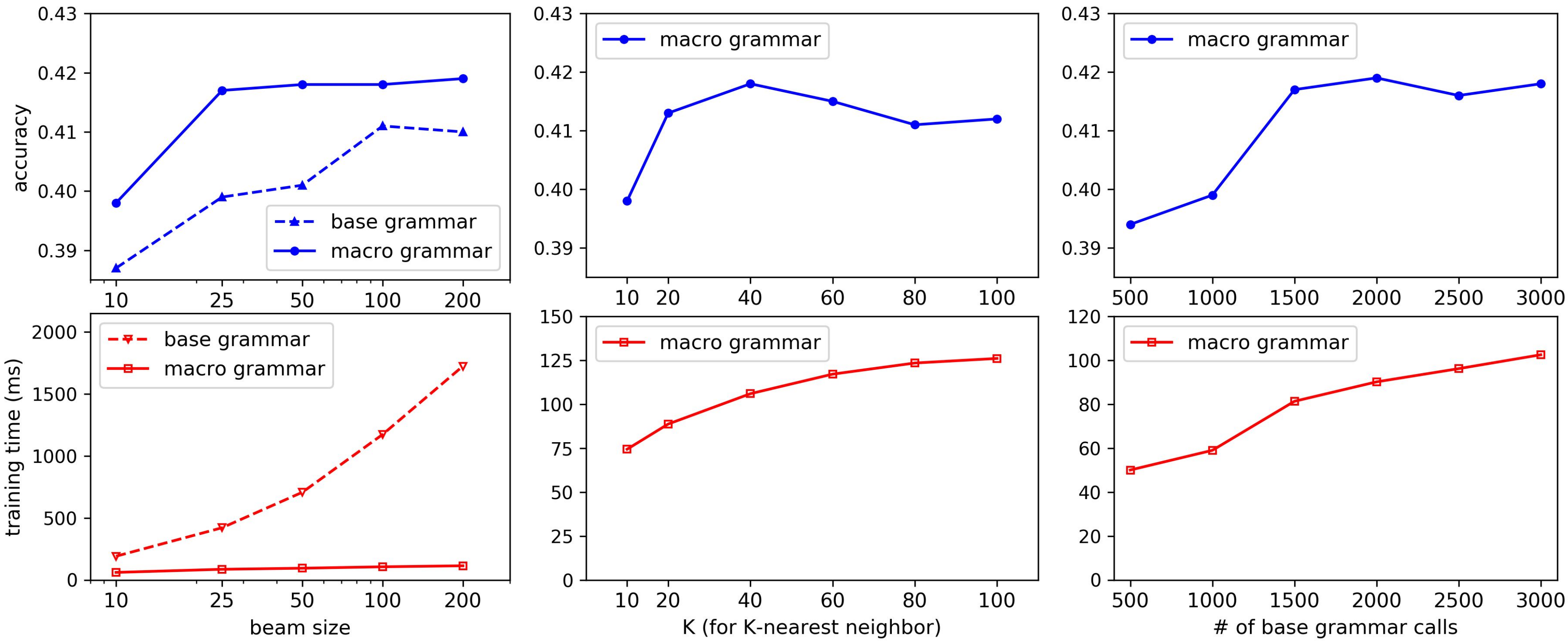
(first dev split)



- Macro grammar is fast even with larger beam sizes
- The number of neighbors should be tuned

Experiment: Tradeoffs

(first dev split)



- Macro grammar is fast even with larger beam sizes
- The number of neighbors should be tuned
- We can reach 42% accuracy even with only 1500 fallback calls to the base grammar (~ 1500 times we augment the macro grammar)

Summary

Method for speeding up semantic parsing

- Why is it faster? Because we search over a restricted space of relevant logical forms
- Still maintain coverage by falling back to the base grammar when needed
- The speed allows us to add more bells and whistles (rules and features) to the model