Large Language Models & Symbolic Structures

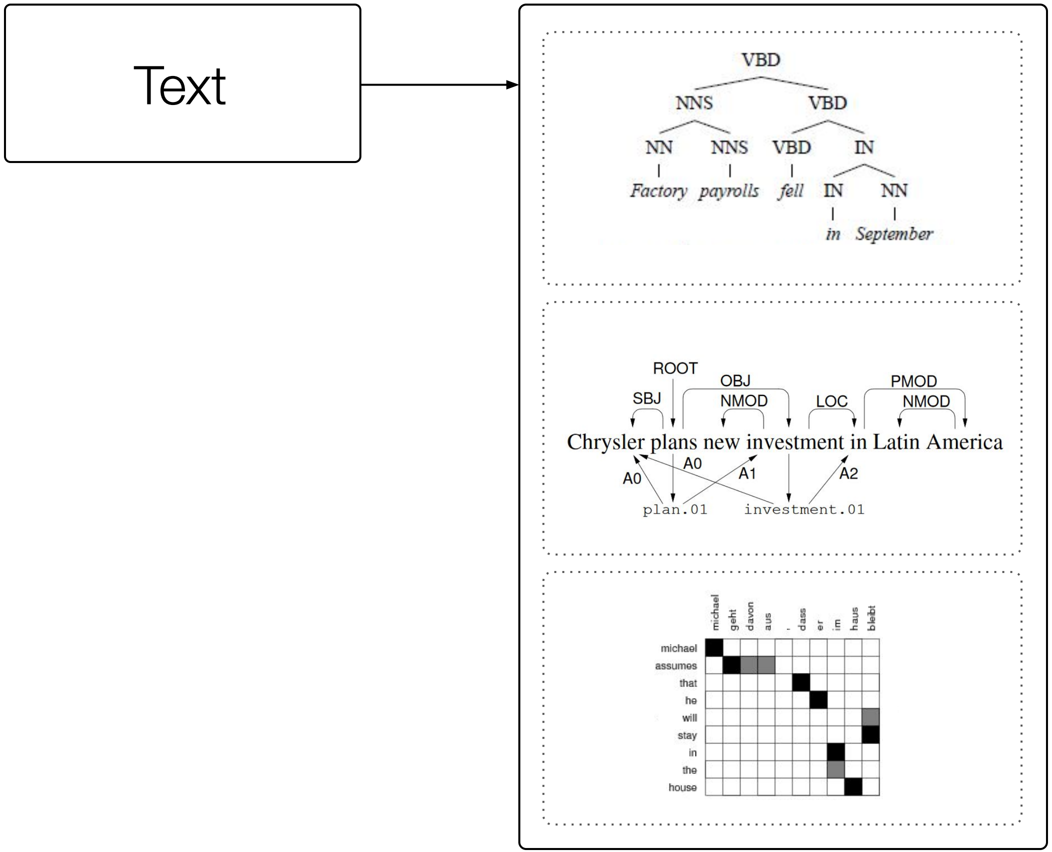
Yoon Kim
MIT

Classic Statistical NLP [1980-2013]

Text

Classic Statistical NLP [1980-2013]

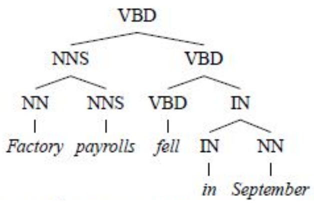
Symbolic Structures

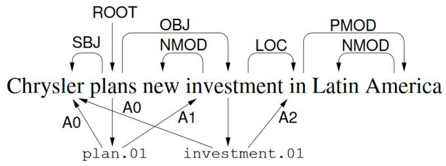


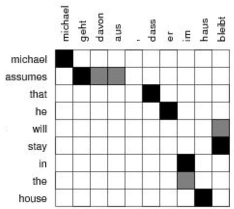
Classic Statistical NLP [1980-2013]

Text

Symbolic Structures







Applications

Machine Translation

Question Answering

Semantic Parsing

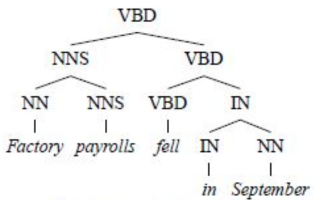
Dialogue

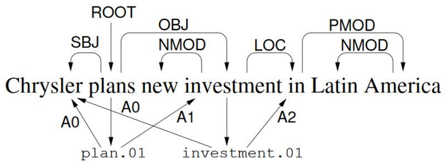
⋮

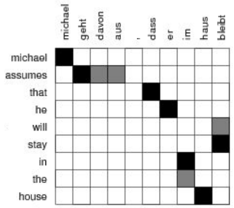
Classic Statistical NLP [1980-2013]

Text

Symbolic Structures







Applications

- Machine Translation
- Question Answering
- Semantic Parsing
- Dialogue
- ⋮

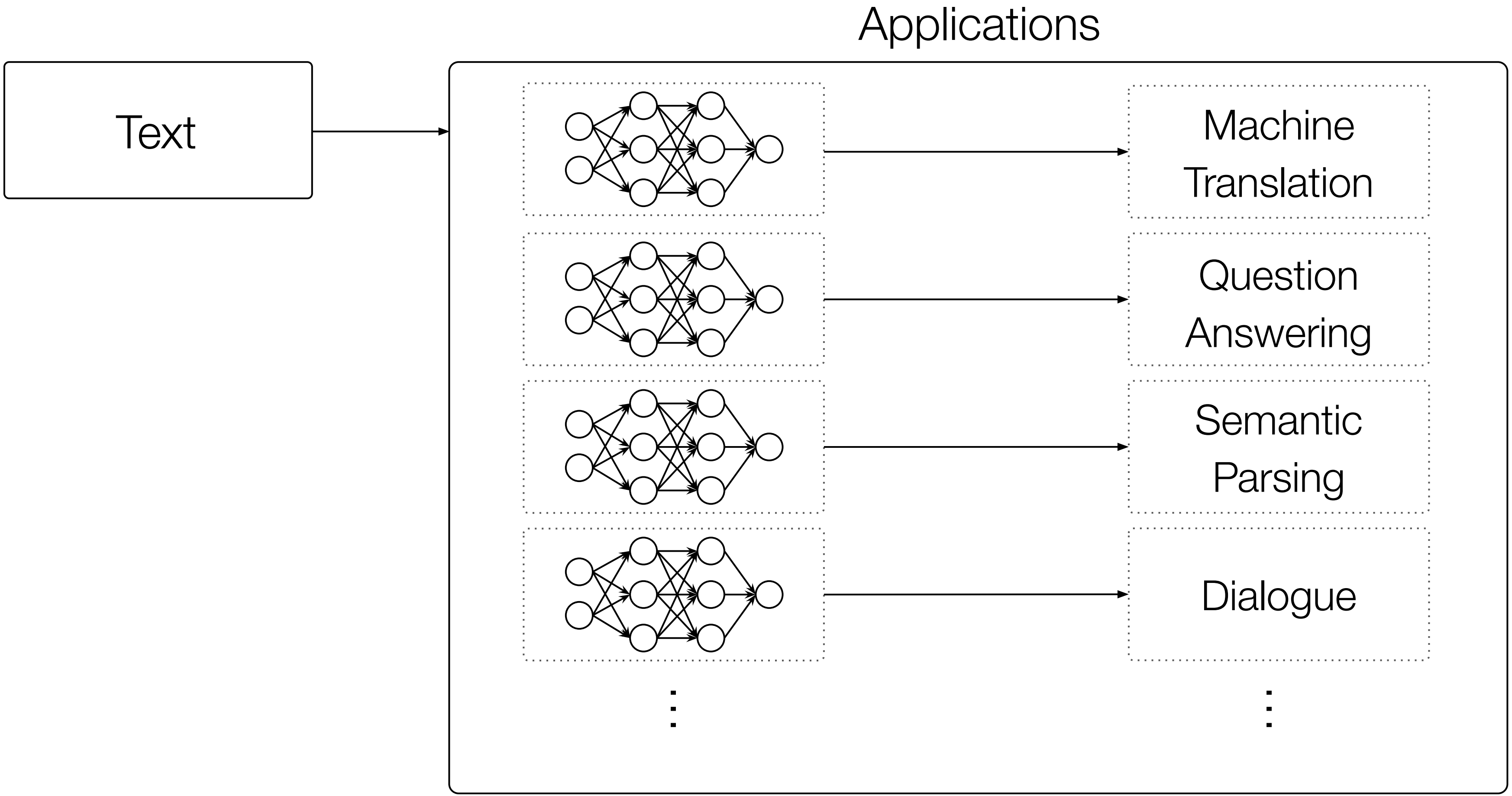
Intermediate symbolic structures for downstream tasks.

Focus on building models of symbolic structures.

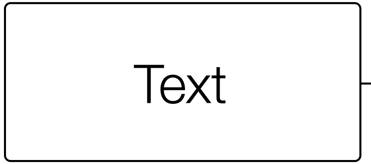
Neural NLP [2013-2018]

Text

Neural NLP [2013-2018]

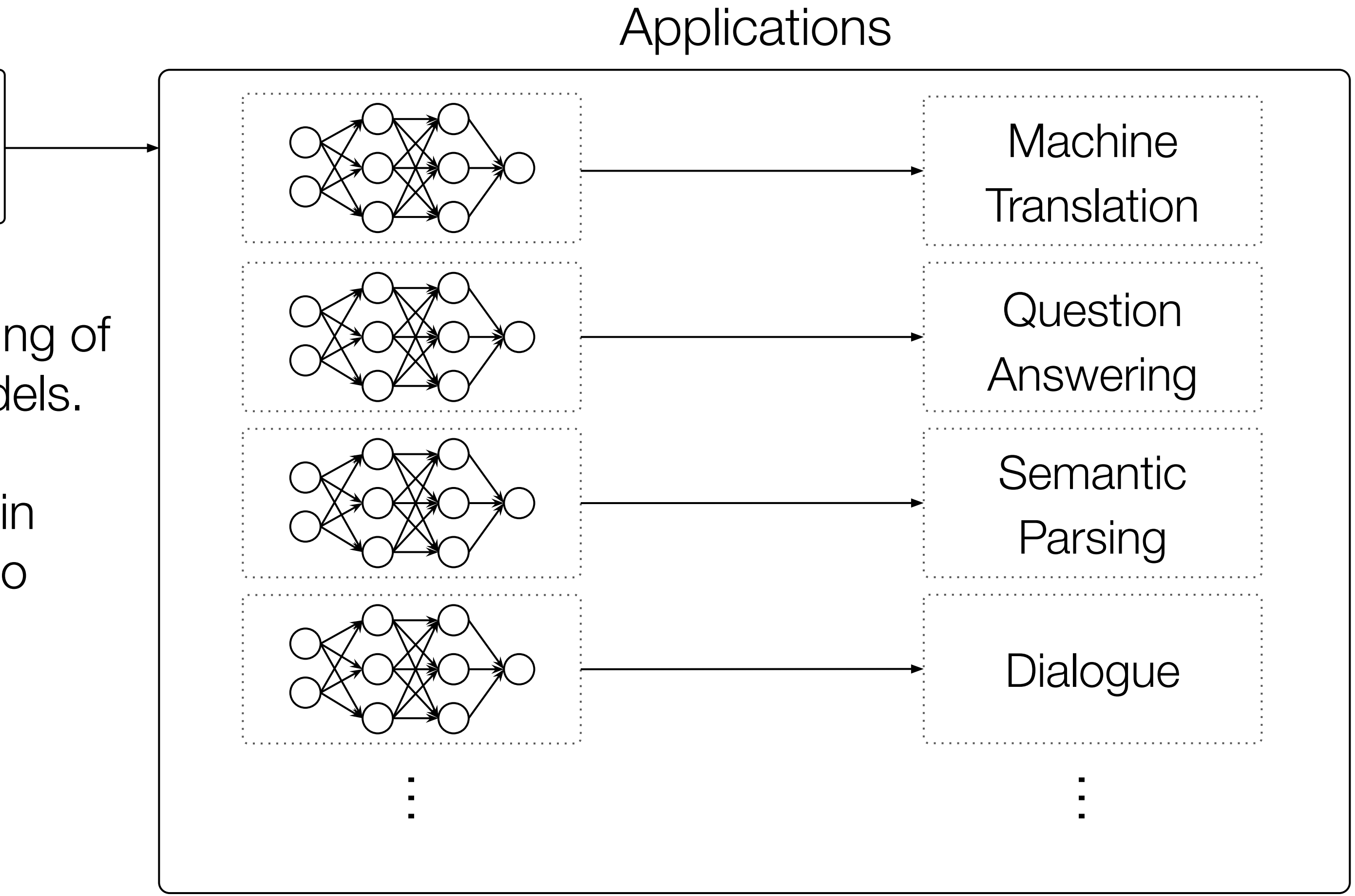


Neural NLP [2013-2018]



End-to-end learning of task-specific models.

Focus on baking in inductive biases to deep learning architectures.



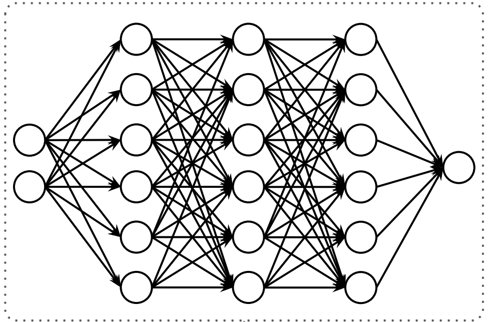
Large Language Models (LLM) [2018-Present]

Text

Large Language Models (LLM) [2018-Present]

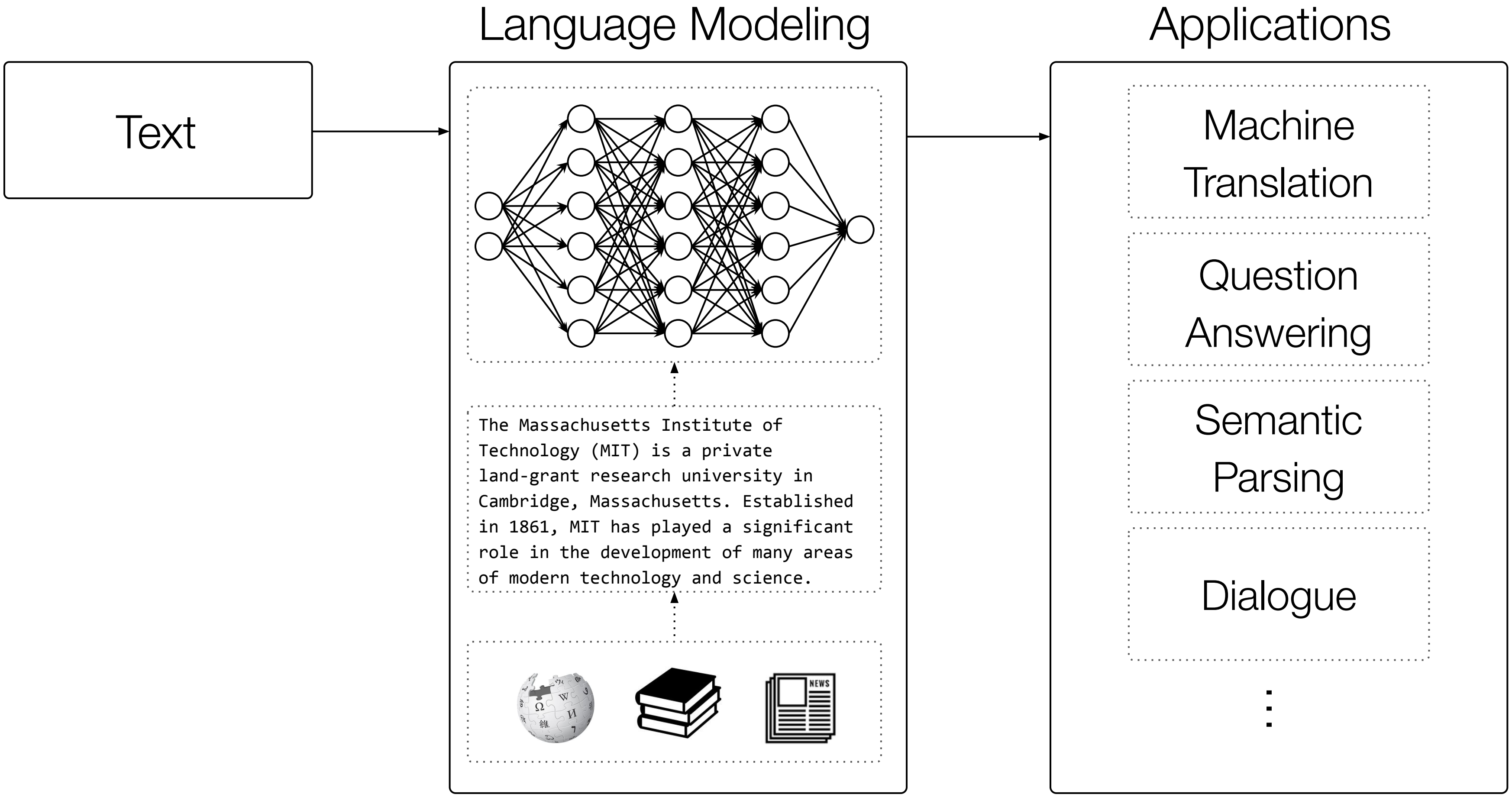
Text

Language Modeling



The Massachusetts Institute of Technology (MIT) is a private land-grant research university in Cambridge, Massachusetts. Established in 1861, MIT has played a significant role in the development of many areas of modern technology and science.

Large Language Models (LLM) [2018-Present]

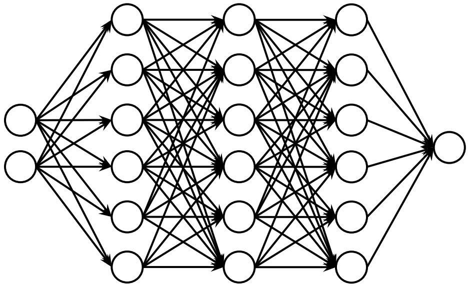


Large Language Models (LLM) [2018-Present]

Language Modeling

Applications

Text



```
The Massachusetts Institute of
Technology (MIT) is a private
land-grant research university in
Cambridge, Massachusetts. Established
in 1861, MIT has played a significant
role in the development of many areas
of modern technology and science.
```

Machine Translation

Question Answering

Semantic Parsing

Dialogue

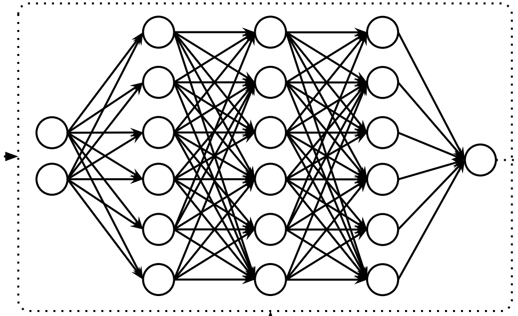
⋮

Language model pretraining for learning linguistic/world knowledge.

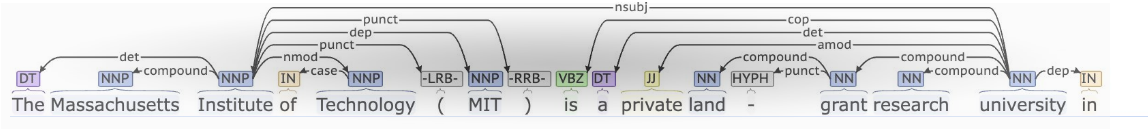
Focus on adapting pretrained language models (finetuning, prompting, etc.).

LLMs & Implicit Structures

```
The Massachusetts Institute
of Technology (MIT) is a
private land-grant research
university in ???
```



```
Cambridge
```



The language modeling (next-word prediction) objective forces the LLM to *implicitly* encode useful structure/knowledge.

LLMs & *Explicit* Structures?

Is language modeling “all you need”?

$$\arg\max_{\theta} \; P_{\theta}(\boldsymbol{x}_t \mid \boldsymbol{x}_{<t})$$

LLMs & *Explicit* Structures?

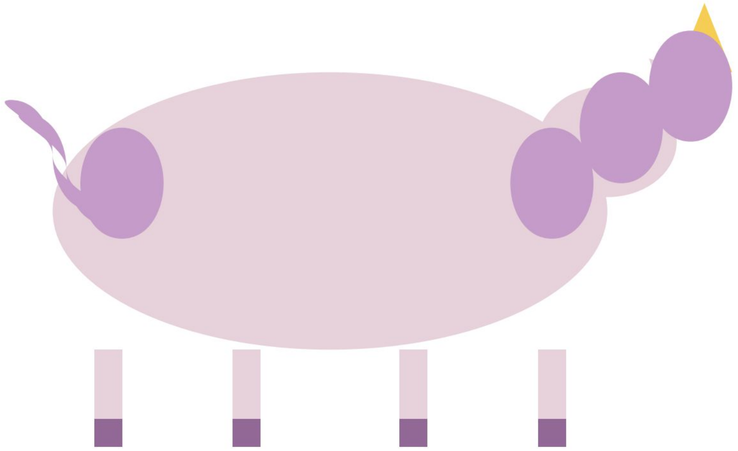
Is language modeling "all you need"?

$$\arg\max_{\theta} \; P_{\theta}(\boldsymbol{x}_t \mid \boldsymbol{x}_{<t})$$

Maybe…

Prompt: Draw a unicorn in TiKZ.

GPT-4: [Produces LaTeX compiling to following picture.]



[Bubeck et al. '23]

LLMs & *Explicit* Structures?

Is language modeling "all you need"?

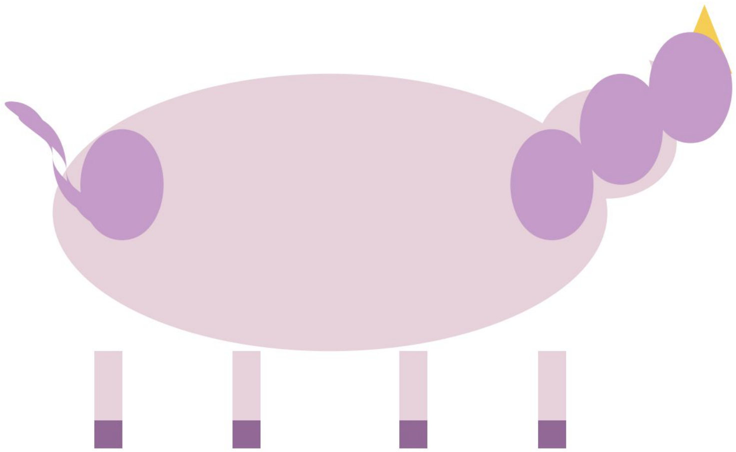
$$\arg\max_{\theta} \; P_\theta(\boldsymbol{x}_t \mid \boldsymbol{x}_{<t})$$

Maybe…

What role (if any) can explicit symbolic structures play in the era of LLMs?

Prompt: Draw a unicorn in TiKZ.

GPT-4: [Produces LaTeX compiling to following picture.]

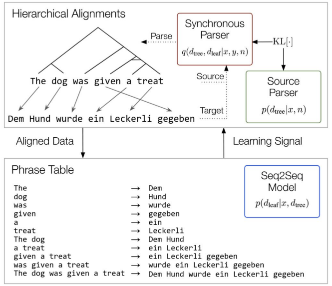


[Bubeck et al. '23]

Symbolic Structures for Controlling & Augmenting LLMs

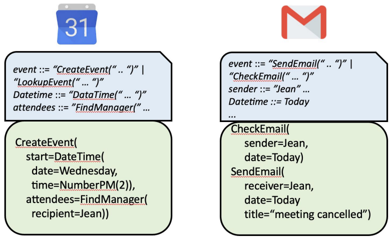
Hierarchical Phrase-based Sequence-to-Sequence Learning

Bailin Wang, Ivan Titov, Jacob Andreas, Yoon Kim
EMNLP '22



Grammar Prompting for DSL Generation with Large Language Models

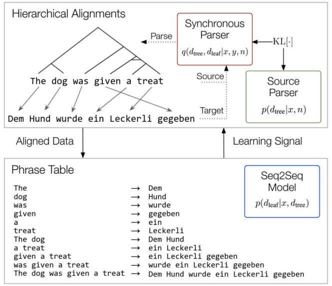
Bailin Wang, Zi Wang, Rif A. Saurous, Xuezhi Wang, Yuan Cao, Yoon Kim
NeurIPS '23



Symbolic Structures for Controlling & Augmenting LLMs

Hierarchical Phrase-based Sequence-to-Sequence Learning

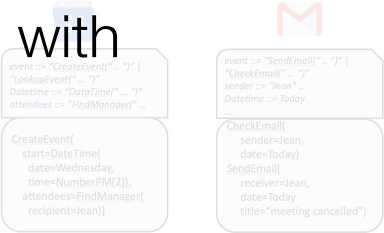
Bailin Wang, Ivan Titov, Jacob Andreas, Yoon Kim
EMNLP '22



TL;DR. Combine latent symbolic structures with pretrained LMs for machine translation.

Grammar Prompting for Large Language Models

Bailin Wang, Zi Wang, Rif A. Saurous, Xuezhi Wang, Yuan Cao, Yoon Kim
Ongoing work



Encoder-Decoder "Language Model" Pretraining

Unaligned multilingual text

```
It is a far, far better thing that I do, than I
have ever done; it is a far, far better rest that
I go to than I have ever known.

The Massachusetts Institute of Technology (MIT)
is a private land-grant research university in
Cambridge, Massachusetts.

보스턴과 찰스 강 하나를 사이에 두고 있으며, 실제로
처음에는 보스턴에 위치하였으나 학교가 너무 커져 신규
부지가 필요해져서 1916년 찰스 강변을 메운 현재의
부지에 신축하고 이동하였다

दिल्ली, आधिकारिक तौर पर राष्ट्रीय राजधानी क्षेत्र दिल्ली
भारत की राजधानी और एक केंद्र-शासित प्रदेश है। इसमें नई
दिल्ली सम्मिलित है जो भारत की राजधानी है। दिल्ली राजधानी
होने के नाते केंद्र सरकार की तीनों इकाइयों
```

⋮

[Lewis et al. '19, Raffel et al. '19, Liu et al. '20, Xue et al. '20]

Encoder-Decoder “Language Model” Pretraining

Unaligned multilingual text

```
It is a far, far better thing that I do, than I
have ever done; it is a far, far better rest that
I go to than I have ever known.
```

The Massachusetts Institute of Technology (MIT) is a private land-grant research university in Cambridge, Massachusetts.

보스턴과 찰스 강 하나를 사이에 두고 있으며, 실제로 처음에는 보스턴에 위치하였으나 학교가 너무 커져 신규 부지가 필요해져서 1916년 찰스 강변을 메운 현재의 부지에 신축하고 이동하였다

दिल्ली, आधिकारिक तौर पर राष्ट्रीय राजधानी क्षेत्र दिल्ली भारत की राजधानी और एक केंद्र-शासित प्रदेश है। इसमें नई दिल्ली सम्मिलित है जो भारत की राजधानी है। दिल्ली राजधानी होने के नाते केंद्र सरकार की तीनों इकाइयों

⋮

Create synthetic input-output pairs

Input

<EN> The Massachusetts [MASK] (MIT) is a [MASK] research university in [MASK].

Output

<EN> Institute of Technology [/s] private land-grand [/s] Cambridge, Massachusetts [/s]

[Lewis et al. ’19, Raffel et al. ’19, Liu et al. ’20, Xue et al. ’20]

Encoder-Decoder “Language Model” Pretraining

Unaligned multilingual text

It is a far, far better thing that I do, than I have ever done; it is a far, far better rest that I go to than I have ever known.

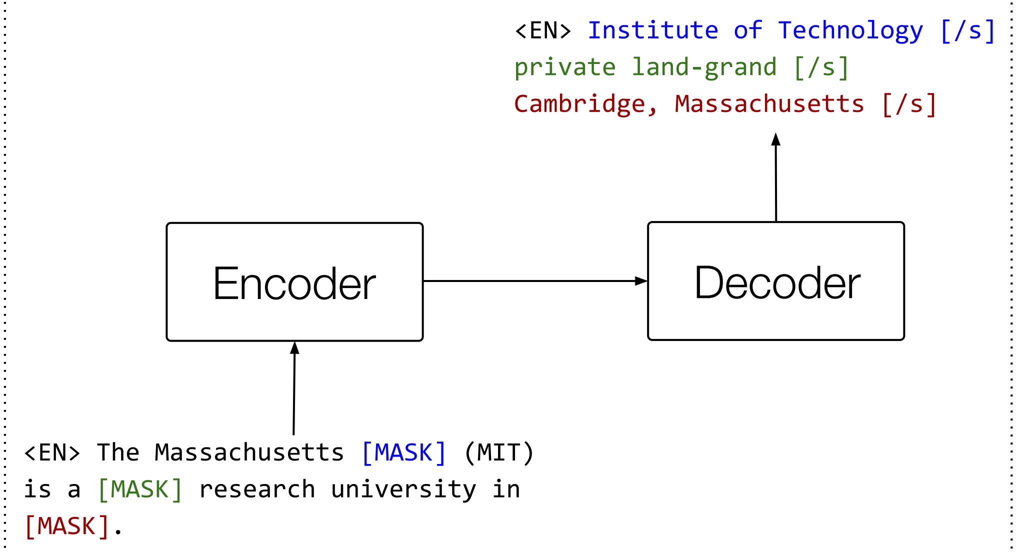
The Massachusetts Institute of Technology (MIT) is a private land-grant research university in Cambridge, Massachusetts.

보스턴과 찰스 강 하나를 사이에 두고 있으며, 실제로 처음에는 보스턴에 위치하였으나 학교가 너무 커져 신규 부지가 필요해져서 1916년 찰스 강변을 메운 현재의 부지에 신축하고 이동하였다

दिल्ली, आधिकारिक तौर पर राष्ट्रीय राजधानी क्षेत्र दिल्ली भारत की राजधानी और एक केंद्र-शासित प्रदेश है। इसमें नई दिल्ली सम्मिलित है जो भारत की राजधानी है। दिल्ली राजधानी होने के नाते केंद्र सरकार की तीनों इकाइयों

⋮

Pretrain as a denoising autoencoder

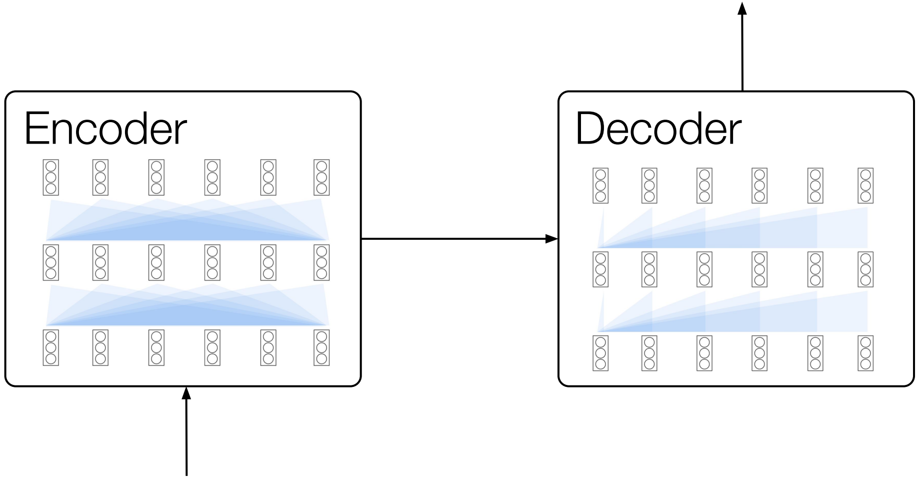


[Lewis et al. ’19, Raffel et al. ’19, Liu et al. ’20, Xue et al. ’20]

“Language Model” Pretraining for Machine Translation

Transfer learning: After pretraining, finetune on (smaller) set of aligned sentences

<KO> 대규모 언어 모델을 정확하게 제어하기 위한 메커니즘을 개발해야 할 긴급한 필요성이 있습니다.



<EN> There is a pressing need to develop mechanisms for precisely controlling large language models.

[Lewis et al. ’19, Raffel et al. ’19, Liu et al. ’20, Xue et al. ’20]

Hasn’t GPT-{3, 3.5, 4} made finetuning obsolete?

	German ⇒ English	Romanian ⇒ English	Chinese ⇒ English
Google NMT	45.0	50.1	31.7
GPT-4	43.7	45.0	24.7

Pretrain-then-finetuning paradigm still performant and cost-effective!

[Jiao et al. ’23]

Classic Hierarchical Phrase-based MT

Classic Hierarchical Phrase-based MT

Source (English)

S $\boxed{10}$

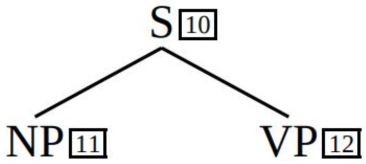
Target (Japanese)

S $\boxed{10}$

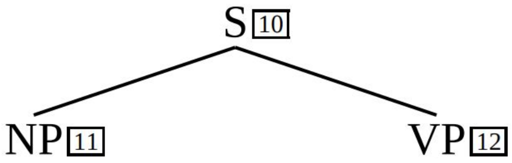
[Chiang ’05]

Classic Hierarchical Phrase-based MT

Source (English)



Target (Japanese)



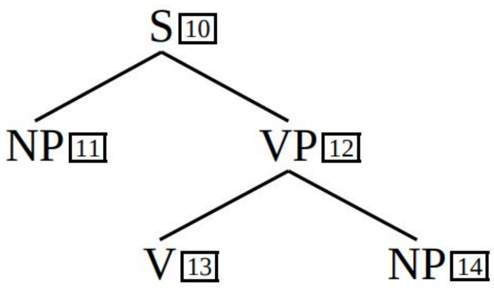
$$S \rightarrow \langle NP\boxed{1}\ VP\boxed{2}, NP\boxed{1}\ VP\boxed{2} \rangle$$

Synchronous context-free grammar rules

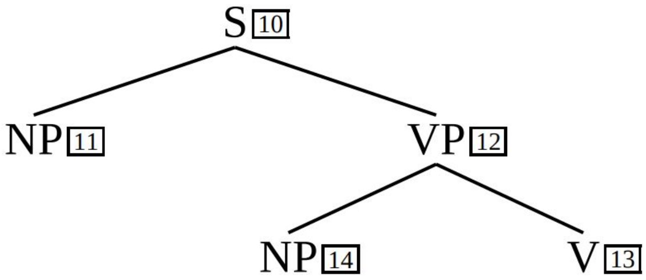
[Chiang ’05]

Classic Hierarchical Phrase-based MT

Source (English)



Target (Japanese)



$$S \rightarrow \langle NP_{\boxed{1}}\ VP_{\boxed{2}}, NP_{\boxed{1}}\ VP_{\boxed{2}} \rangle$$

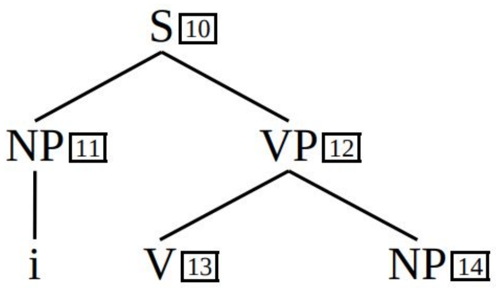
$$VP \rightarrow \langle V_{\boxed{1}}\ NP_{\boxed{2}}, NP_{\boxed{2}}\ V_{\boxed{1}} \rangle$$

Synchronous context-free grammar rules

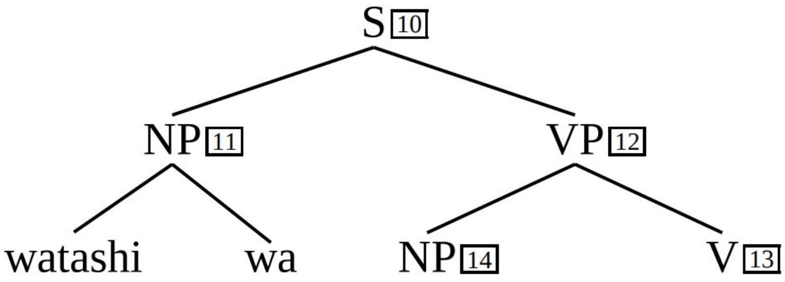
[Chiang '05]

Classic Hierarchical Phrase-based MT

Source (English)



Target (Japanese)



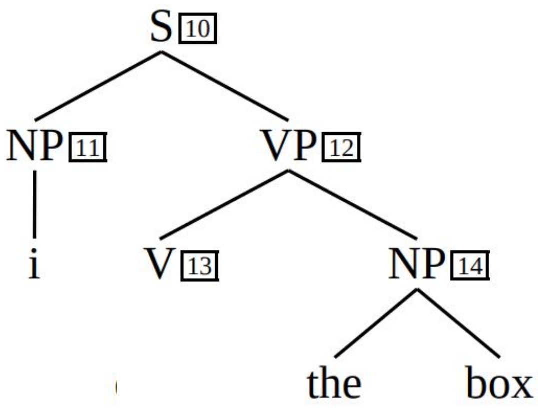
Synchronous context-free grammar rules

$$\begin{aligned}
\mathrm{S} &\rightarrow \langle \mathrm{NP}\boxed{1}\ \mathrm{VP}\boxed{2}, \mathrm{NP}\boxed{1}\ \mathrm{VP}\boxed{2} \rangle \\
\mathrm{VP} &\rightarrow \langle \mathrm{V}\boxed{1}\ \mathrm{NP}\boxed{2}, \mathrm{NP}\boxed{2}\ \mathrm{V}\boxed{1} \rangle \\
\mathrm{NP} &\rightarrow \langle \mathrm{i}, \mathrm{watashi\ wa} \rangle
\end{aligned}$$

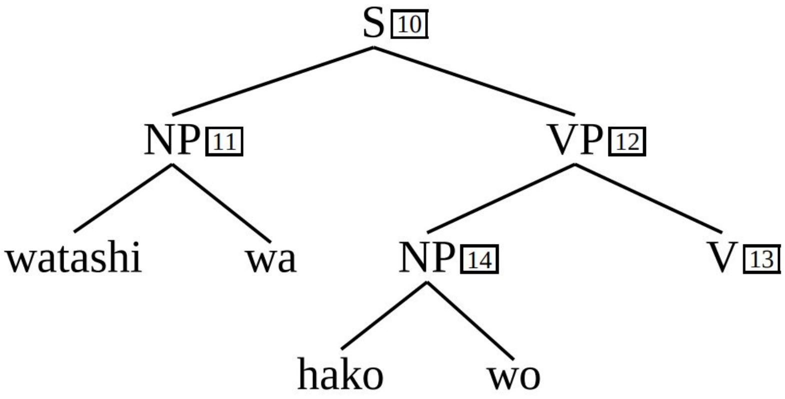
[Chiang '05]

Classic Hierarchical Phrase-based MT

Source (English)



Target (Japanese)



Synchronous context-free grammar rules

$$\mathrm{S} \rightarrow \langle \mathrm{NP}\boxed{1}\ \mathrm{VP}\boxed{2}, \mathrm{NP}\boxed{1}\ \mathrm{VP}\boxed{2} \rangle$$

$$\mathrm{VP} \rightarrow \langle \mathrm{V}\boxed{1}\ \mathrm{NP}\boxed{2}, \mathrm{NP}\boxed{2}\ \mathrm{V}\boxed{1} \rangle$$

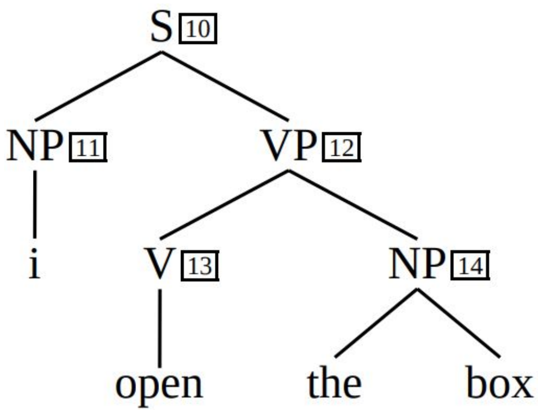
$$\mathrm{NP} \rightarrow \langle \text{i, watashi wa} \rangle$$

$$\mathrm{NP} \rightarrow \langle \text{the box, hako wo} \rangle$$

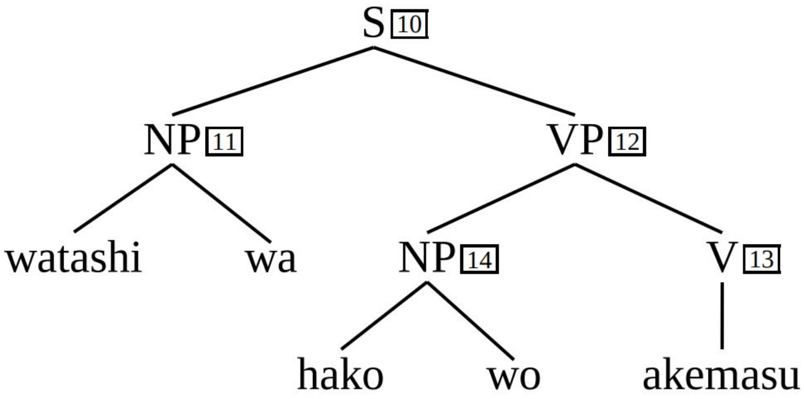
[Chiang ’05]

Classic Hierarchical Phrase-based MT

Source (English)



Target (Japanese)



Synchronous context-free grammar rules

$$S \rightarrow \langle NP\boxed{1}\ VP\boxed{2},\ NP\boxed{1}\ VP\boxed{2} \rangle$$

$$VP \rightarrow \langle V\boxed{1}\ NP\boxed{2},\ NP\boxed{2}\ V\boxed{1} \rangle$$

$$NP \rightarrow \langle \text{i, watashi wa} \rangle$$

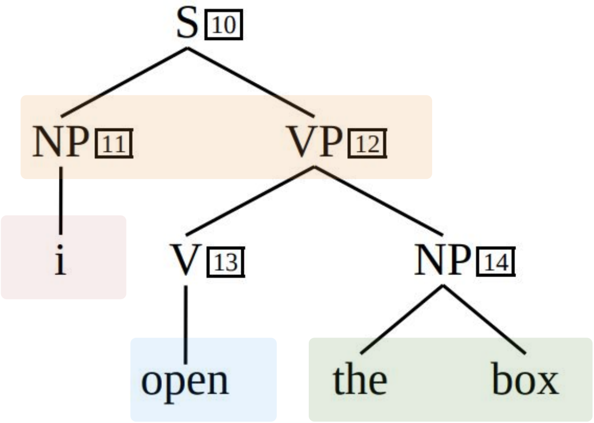
$$NP \rightarrow \langle \text{the box, hako wo} \rangle$$

$$V \rightarrow \langle \text{open, akemasu} \rangle$$

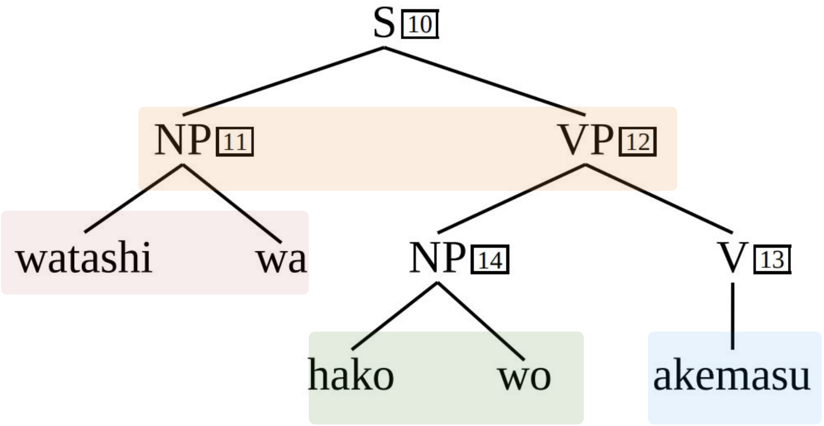
[Chiang ’05]

Classic Hierarchical Phrase-based MT

Source (English)



Target (Japanese)



Synchronous context-free grammar rules

$S \rightarrow \langle NP\boxed{1}\ VP\boxed{2}, NP\boxed{1}\ VP\boxed{2} \rangle$

$VP \rightarrow \langle V\boxed{1}\ NP\boxed{2}, NP\boxed{2}\ V\boxed{1} \rangle$

$NP \rightarrow \langle \text{i, watashi wa} \rangle$

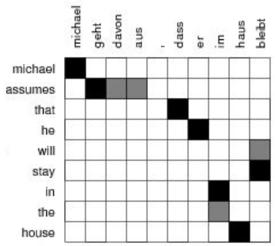
$NP \rightarrow \langle \text{the box, hako wo} \rangle$

$V \rightarrow \langle \text{open, akemasu} \rangle$

[Chiang '05]

Classic Hierarchical Phrase-based MT

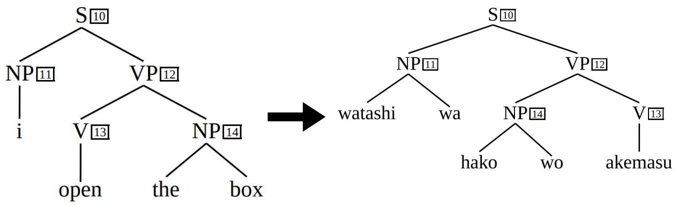
Extract phrase alignments from parallel sentences



Induce synchronous grammar rules and weights

$$
\begin{array}{rll}
\text{S} & \rightarrow \langle \text{NP}\boxed{1}\ \text{VP}\boxed{2}, \text{NP}\boxed{1}\ \text{VP}\boxed{2} \rangle & s_1 \\
\text{VP} & \rightarrow \langle \text{V}\boxed{1}\ \text{NP}\boxed{2}, \text{NP}\boxed{2}\ \text{V}\boxed{1} \rangle & s_2 \\
\text{NP} & \rightarrow \langle \text{i, watashi wa} \rangle & s_3 \\
\text{NP} & \rightarrow \langle \text{the box, hako wo} \rangle & s_4 \\
\text{V} & \rightarrow \langle \text{open, akemasu} \rangle & s_5
\end{array}
$$

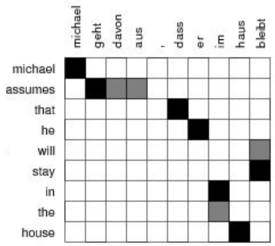
Translation as (monolingual) parsing



[Chiang '05]

Classic Hierarchical Phrase-based MT

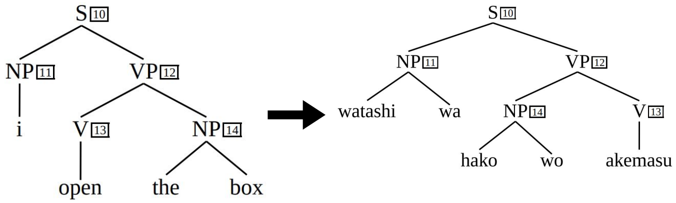
Extract phrase alignments from parallel sentences



Induce synchronous grammar rules and weights

$$
\begin{array}{lll}
S & \rightarrow \langle NP\boxed{1}\ VP\boxed{2}, NP\boxed{1}\ VP\boxed{2} \rangle & s_1 \\
VP & \rightarrow \langle V\boxed{1}\ NP\boxed{2}, NP\boxed{2}\ V\boxed{1} \rangle & s_2 \\
NP & \rightarrow \langle \text{i, watashi wa} \rangle & s_3 \\
NP & \rightarrow \langle \text{the box, hako wo} \rangle & s_4 \\
V & \rightarrow \langle \text{open, akemasu} \rangle & s_5
\end{array}
$$

Translation as (monolingual) parsing



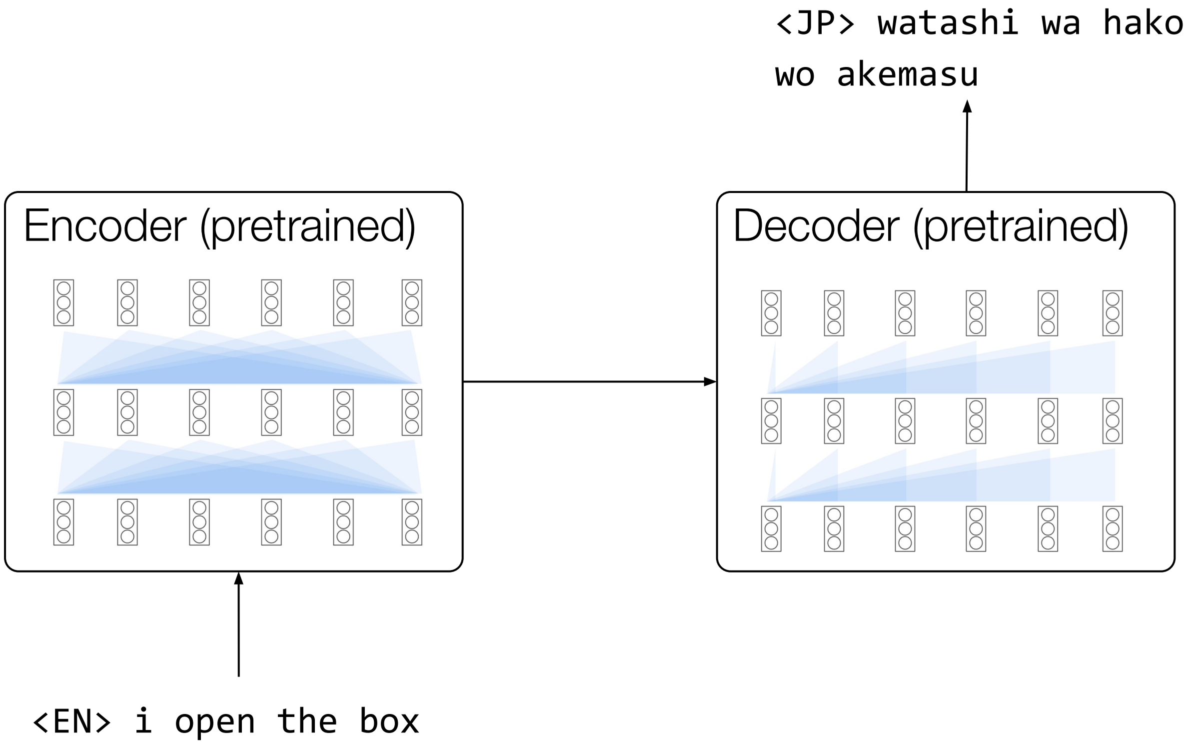
✗ Error propagation due to “pipelining”.

✗ Inflexible model due to (too) strong assumptions.

✗ Cannot (directly) be applied on top of pretrained LMs.

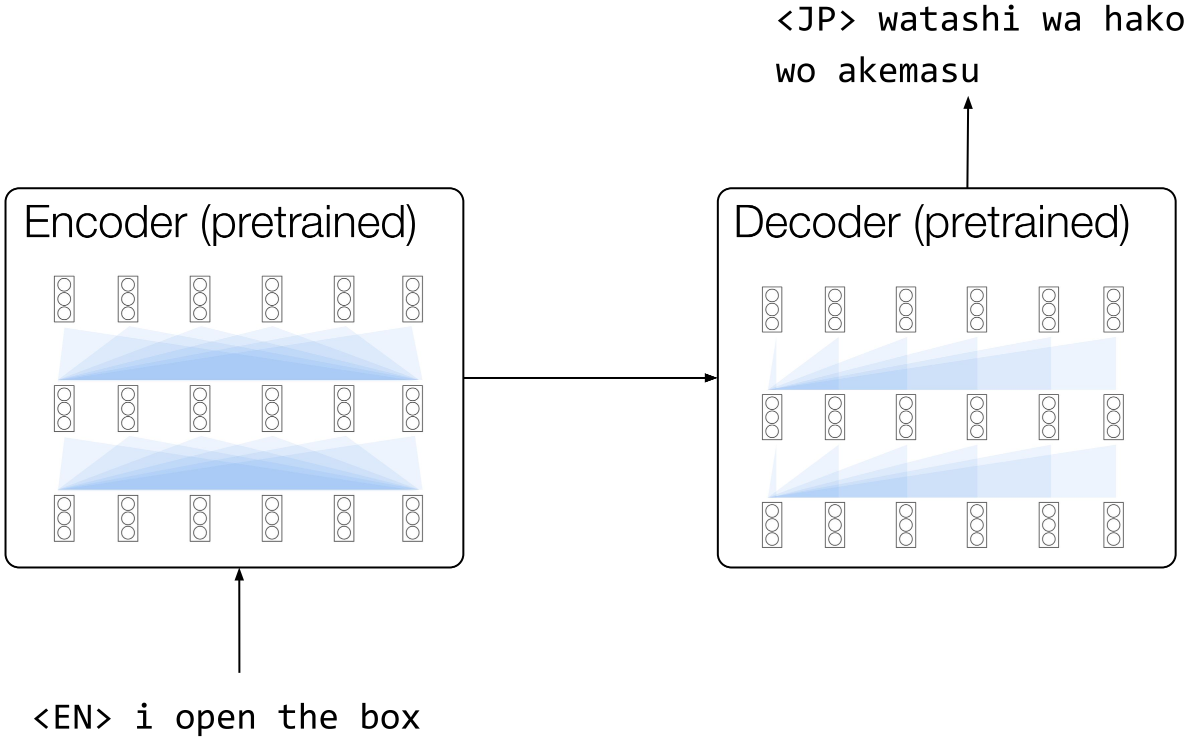
[Chiang ’05]

Neural MT with Pretraining



[Sutkever et al. ’14, Cho et al. ’14, Bahdanau et al. ’15, Vaswani et al. ’17]

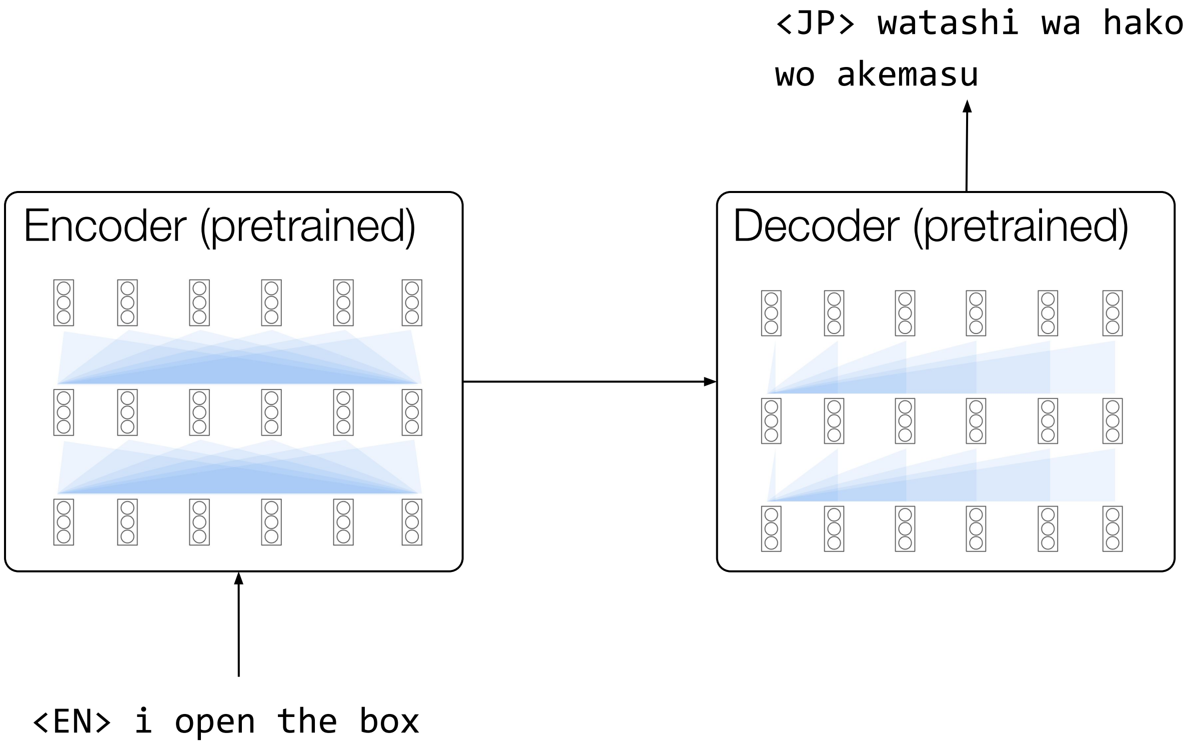
Neural MT with Pretraining



✔ End-to-end learning: all phenomena captured implicitly in the hidden layers.

[Sutkever et al. ’14, Cho et al. ’14, Bahdanau et al. ’15, Vaswani et al. ’17]

Neural MT with Pretraining



✔ End-to-end learning: all phenomena captured implicitly in the hidden layers.

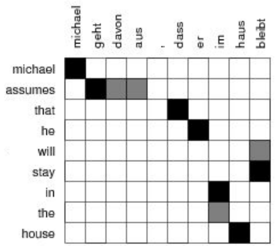
✗ Too flexible of a model ⇒ data hungry and prone to overfitting.

✗ End-to-end generation ⇒ hard to control/interpret translation process.

[Sutkever et al. ’14, Cho et al. ’14, Bahdanau et al. ’15, Vaswani et al. ’17]

Classic Hierarchical Phrase-based MT

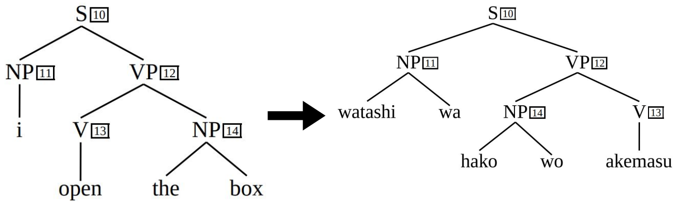
Extract phrase alignments from parallel sentences



Induce synchronous grammar rules and weights

$$
\begin{array}{rll}
S & \rightarrow \langle NP\boxed{1}\ VP\boxed{2}, NP\boxed{1}\ VP\boxed{2} \rangle & s_1 \\
VP & \rightarrow \langle V\boxed{1}\ NP\boxed{2}, NP\boxed{2}\ V\boxed{1} \rangle & s_2 \\
NP & \rightarrow \langle \text{i, watashi wa} \rangle & s_3 \\
NP & \rightarrow \langle \text{the box, hako wo} \rangle & s_4 \\
V & \rightarrow \langle \text{open, akemasu} \rangle & s_5
\end{array}
$$

Translation as (monolingual) parsing



✔ Explicit symbolic rules ⇒ controllable & interpretable generation.

$$
\begin{array}{rl}
NP & \rightarrow \langle \text{the student, gakusei-ga} \rangle \\
NP & \rightarrow \langle \text{the teacher, sensei-ga} \rangle \\
V & \rightarrow \langle \text{danced, odotta} \rangle \\
V & \rightarrow \langle \text{said, itta} \rangle
\end{array}
$$

✔ Strong inductive bias for compositionality.

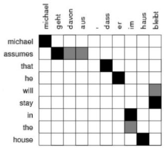
✔ Data-efficient.

[Chiang '05]

Classic MT + Modern Pretraining?

Can we combine the controllability and the compositional inductive bias of classic methods with the flexibility of pretrained LMs?

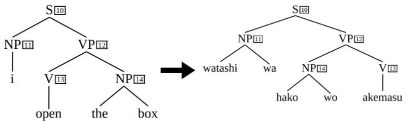
Extract phrase alignments from parallel sentences



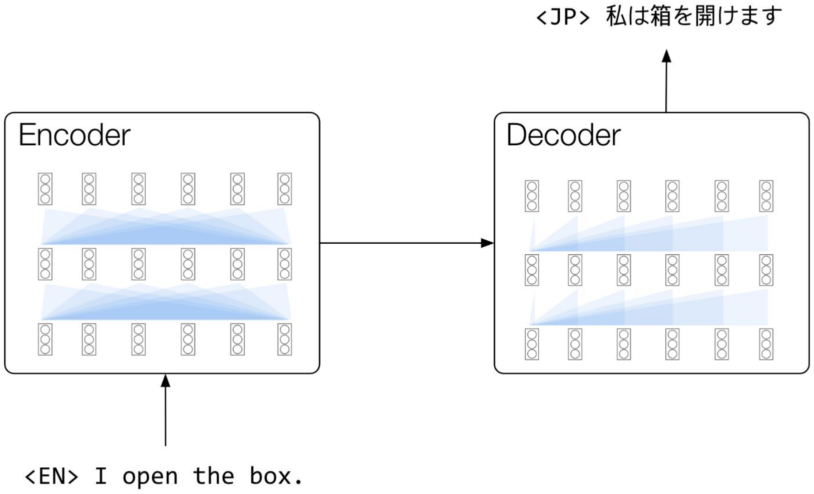
Induce synchronous grammar rules and weights

$$S \rightarrow \langle NP\boxed{1}\ VP\boxed{2}, NP\boxed{1}\ VP\boxed{2} \rangle \quad s_1$$
$$VP \rightarrow \langle V\boxed{1}\ NP\boxed{2}, NP\boxed{2}\ V\boxed{1} \rangle \quad s_2$$
$$NP \rightarrow \langle \text{i, watashi wa} \rangle \quad s_3$$
$$NP \rightarrow \langle \text{the box, hako wo} \rangle \quad s_4$$
$$V \rightarrow \langle \text{open, akemasu} \rangle \quad s_5$$

Translation as (monolingual) parsing

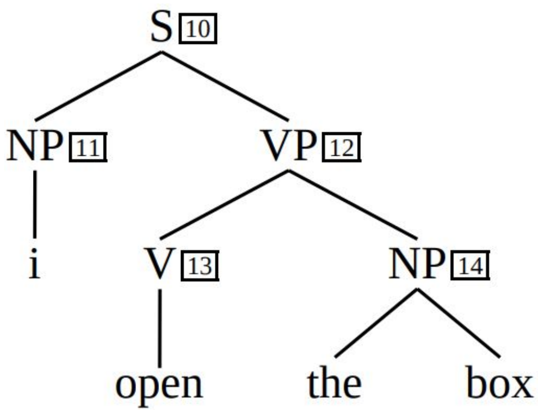


+

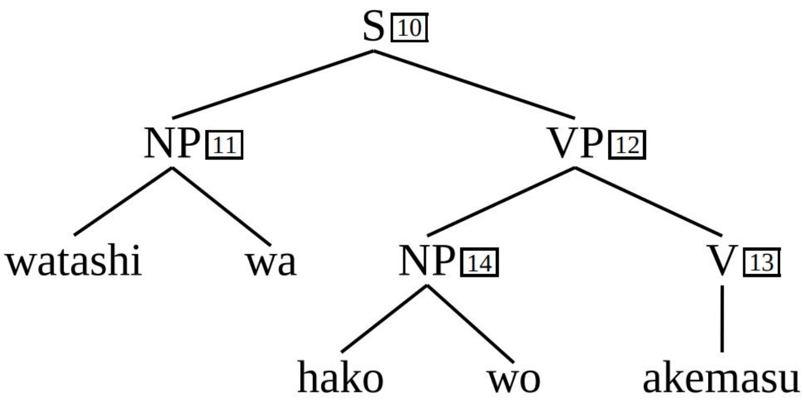


A “Neural” Parameterization of Grammar Rules

Source



Target



Synchronous context-free grammar rules

$$S \rightarrow \langle NP_{\boxed{1}}\ VP_{\boxed{2}}, NP_{\boxed{1}}\ VP_{\boxed{2}} \rangle$$

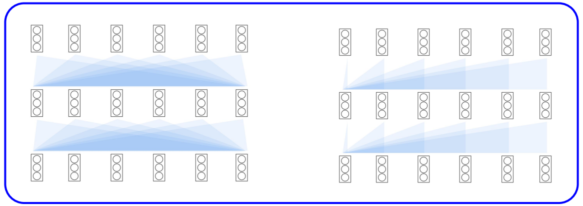
$$VP \rightarrow \langle V_{\boxed{1}}\ NP_{\boxed{2}}, NP_{\boxed{2}}\ V_{\boxed{1}} \rangle$$

$$NP \rightarrow \langle \text{i, watashi wa} \rangle$$

$$\boxed{NP \rightarrow \langle \text{the box, hako wo} \rangle}$$

$$V \rightarrow \langle \text{open, akemasu} \rangle$$

From a pretrained LM!



A Neural Synchronous Grammar

```
i    open    the    box
```

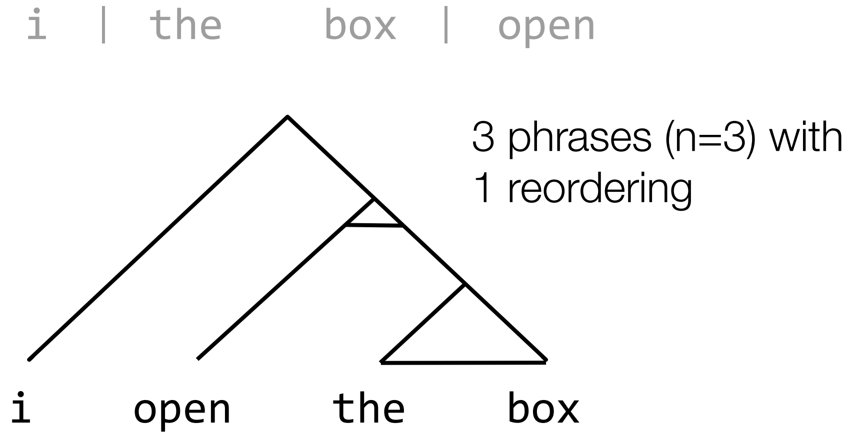
A Neural Synchronous Grammar

`i    open    the    box`

$$P_{\mathsf{BTG}}(\boldsymbol{t} \mid \boldsymbol{x})$$

Bracketing Transduction Grammar (BTG) segments and reorders source sentence.

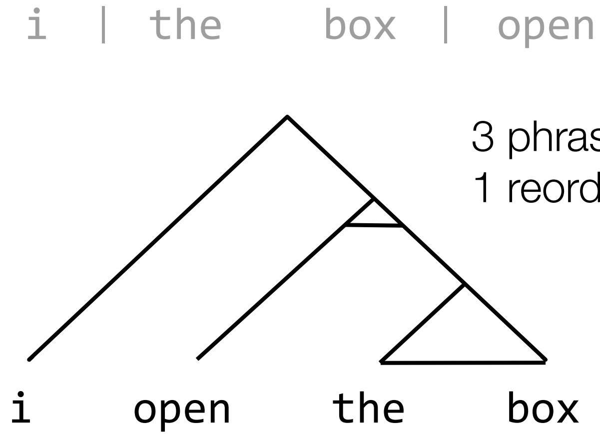
A Neural Synchronous Grammar



$$P_{\mathsf{BTG}}(\boldsymbol{t} \mid \boldsymbol{x})$$

Bracketing Transduction Grammar (BTG) segments and reorders source sentence.

A Neural Synchronous Grammar



3 phrases (n=3) with
1 reordering

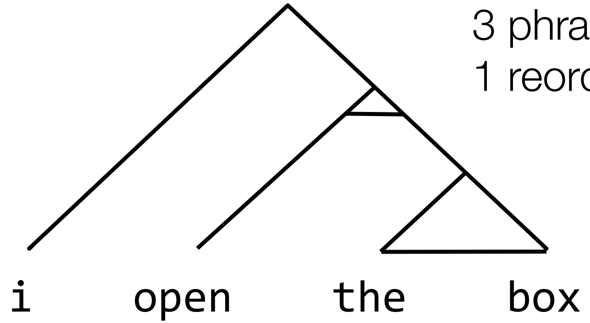
$$P_{\mathsf{BTG}}(\boldsymbol{t} \mid \boldsymbol{x}) \propto P_{\mathsf{seg}}(n \mid \boldsymbol{x}) \prod_{1 \leq i < j \leq |\boldsymbol{x}|} e_S^{\top} f(\mathbf{h}_i, \mathbf{h}_j)$$

$P_{\mathsf{BTG}}(\boldsymbol{t} \mid \boldsymbol{x})$

Bracketing Transduction Grammar (BTG) segments and reorders source sentence.

A Neural Synchronous Grammar

i | the box | open



3 phrases (n=3) with
1 reordering

$P_{\mathsf{BTG}}(\boldsymbol{t} \mid \boldsymbol{x})$

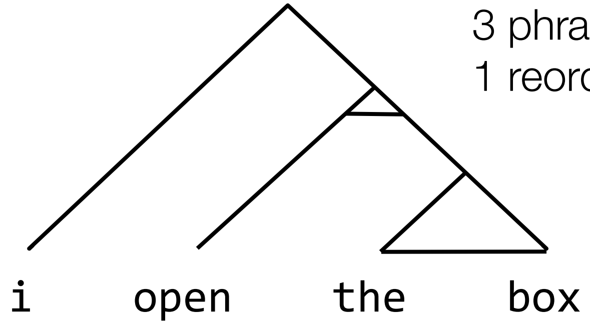
Bracketing Transduction Grammar (BTG) segments and reorders source sentence.

Distribution over number of phrase segments

$$P_{\mathsf{BTG}}(\boldsymbol{t} \mid \boldsymbol{x}) \propto \boxed{P_{\mathsf{seg}}(n \mid \boldsymbol{x})} \prod_{1 \le i < j \le |\boldsymbol{x}|} e_S^\top f(\mathbf{h}_i, \mathbf{h}_j)$$

A Neural Synchronous Grammar

i | the box | open



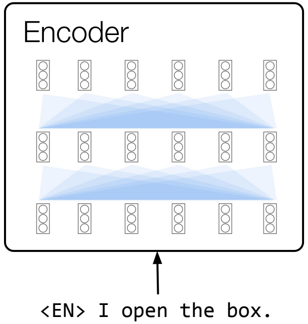
3 phrases (n=3) with
1 reordering

$$P_{\mathsf{BTG}}(\boldsymbol{t} \mid \boldsymbol{x})$$

Bracketing Transduction Grammar (BTG) segments and reorders source sentence.

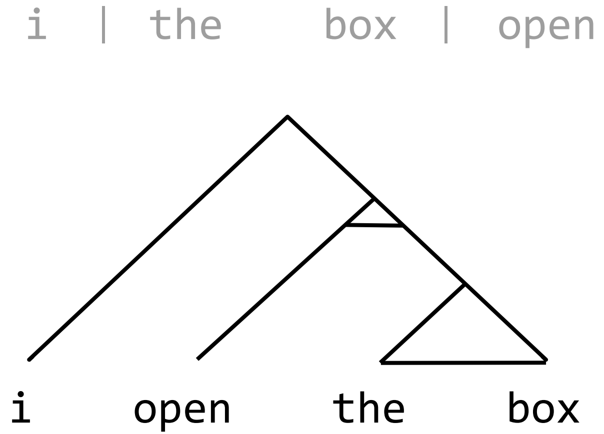
Distribution over number of phrase segments

$$P_{\mathsf{BTG}}(\boldsymbol{t} \mid \boldsymbol{x}) \propto P_{\mathsf{seg}}(n \mid \boldsymbol{x}) \prod_{1 \leq i < j \leq |\boldsymbol{x}|} e_S^{\top} f(\mathbf{h}_i, \mathbf{h}_j)$$



CRF parser over pretrained encoder

A Neural Synchronous Grammar



$$P_{\mathsf{BTG}}(\boldsymbol{t} \mid \boldsymbol{x})$$

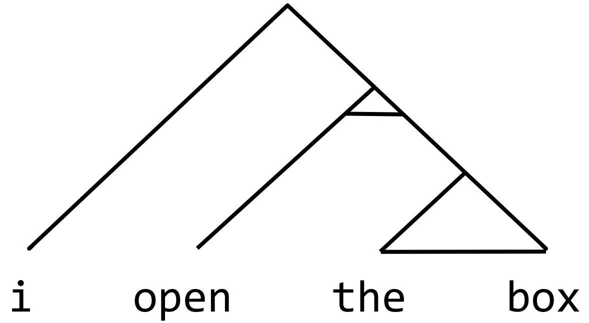
Bracketing Transduction Grammar (BTG) segments and reorders source sentence.

$$P_{\mathsf{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t})$$

Seq2seq model translates segmented source phrases one-by-one.

A Neural Synchronous Grammar

`i | the box | open`



$$P_{\mathsf{BTG}}(\boldsymbol{t} \mid \boldsymbol{x})$$

Bracketing Transduction Grammar (BTG) segments and reorders source sentence.

`watashi wa hako wo akemasu`

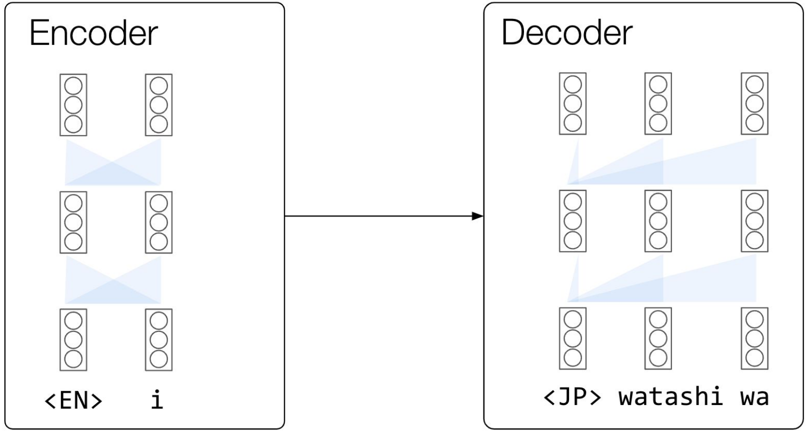
$$P_{\mathsf{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t})$$

Seq2seq model translates segmented source phrases one-by-one.

A Neural Synchronous Grammar

$$P_{\text{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t}) = P_{\text{LM}}(\text{“watashi wa”} \mid \text{“i”}) \times$$

i | the box | open



watashi wa

$$P_{\text{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t})$$

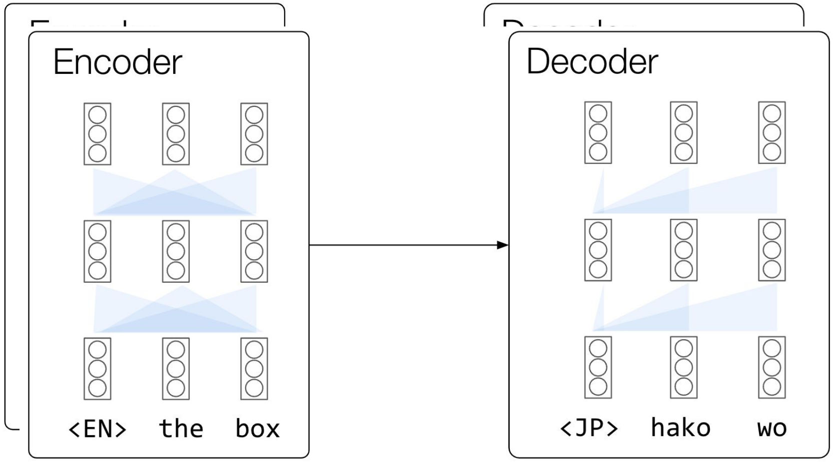
Bracketing Transduction Grammar (BTG) segments and reorders source sentence.

Seq2seq model translates segmented source phrases one-by-one.

A Neural Synchronous Grammar

$$P_{\text{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t}) = P_{\text{LM}}(\text{“watashi wa”} \mid \text{“i”}) \times P_{\text{LM}}(\text{“hako wo”} \mid \text{“the box”}) \times$$

`i | the box | open`



`watashi wa` `hako wo`

$$P_{\text{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t})$$

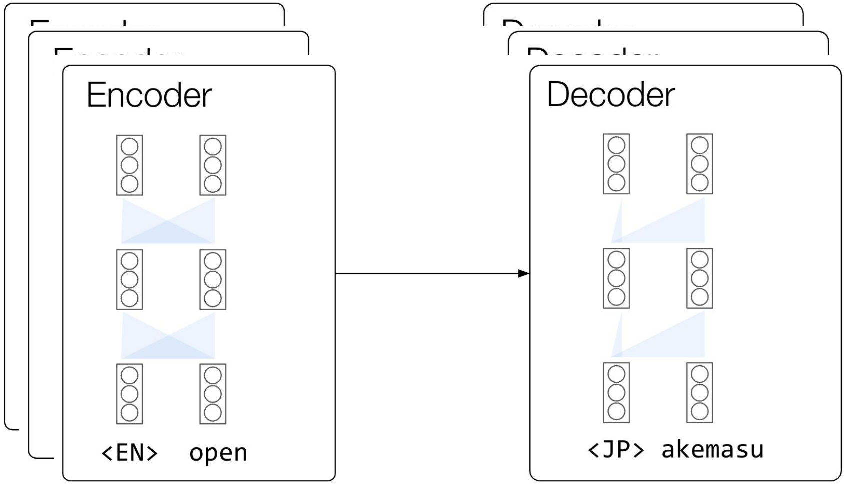
Bracketing Transduction Grammar (BTG) segments and reorders source sentence.

Seq2seq model translates segmented source phrases one-by-one.

A Neural Synchronous Grammar

$$P_{\text{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t}) = P_{\text{LM}}(\text{“watashi wa”} \mid \text{“i”}) \times P_{\text{LM}}(\text{“hako wo”} \mid \text{“the box”}) \times P_{\text{LM}}(\text{“akemasu”} \mid \text{“open”}) \times$$

i | the box | open



watashi wa hako wo akemasu

$$P_{\text{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t})$$

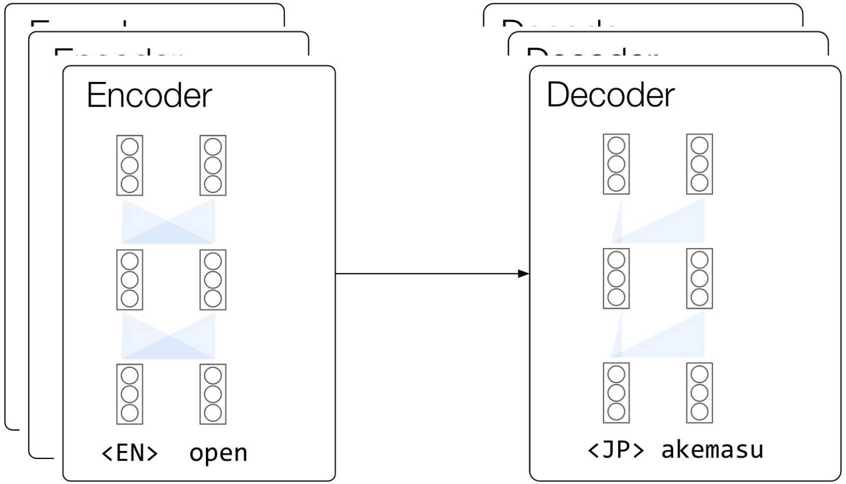
Bracketing Transduction Grammar (BTG) segments and reorders source sentence.

Seq2seq model translates segmented source phrases one-by-one.

A Neural Synchronous Grammar

$$P_{\text{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t}) = P_{\text{LM}}(\text{“watashi wa”} \mid \text{“i”}) \times P_{\text{LM}}(\text{“hako wo”} \mid \text{“the box”}) \times P_{\text{LM}}(\text{“akemasu”} \mid \text{“open”}) \times$$

i | the box | open



Same pretrained LM (which will be finetuned) to translate at all phrase scales.

watashi wa | hako wo | akemasu

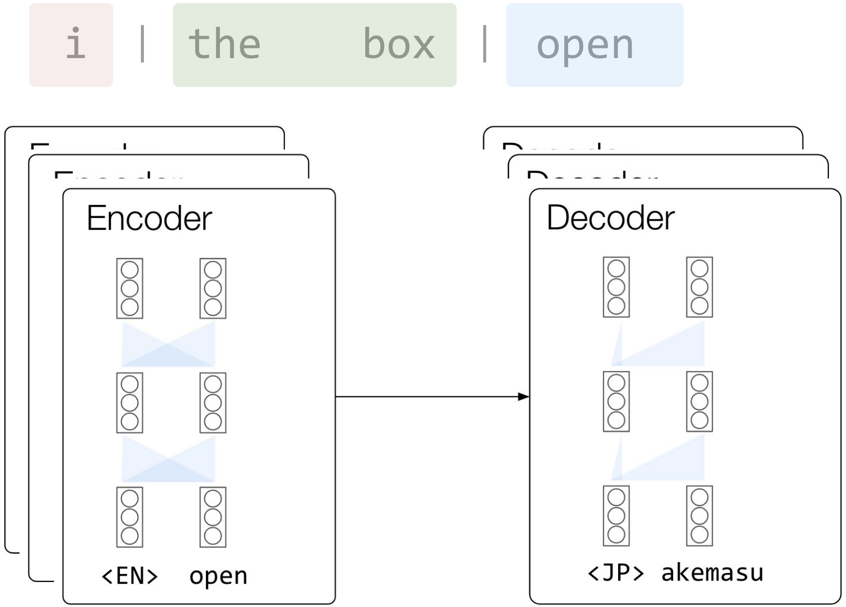
$$P_{\text{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t})$$

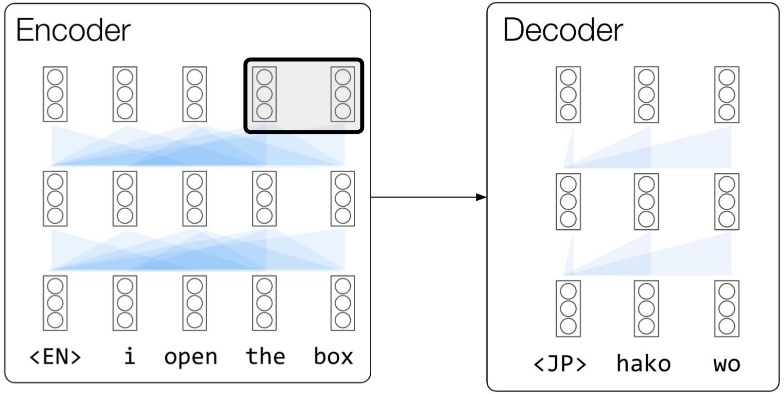
Seq2seq model translates segmented source phrases one-by-one.

Bracketing Transduction Grammar (BTG) segments and reorders source sentence.

A Neural Synchronous Grammar

$$P_{\text{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t}) = P_{\text{LM}}(\text{“watashi wa”} \mid \text{“i”}) \times P_{\text{LM}}(\text{“hako wo”} \mid \text{“the box”}) \times P_{\text{LM}}(\text{“akemasu”} \mid \text{“open”}) \times$$





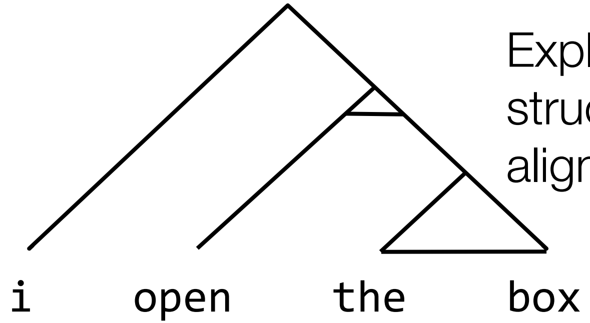
Bracketing Transduction Grammar (BTG) segments and reorders source sentence.

Actually use **contextualized** word representations on the source side.

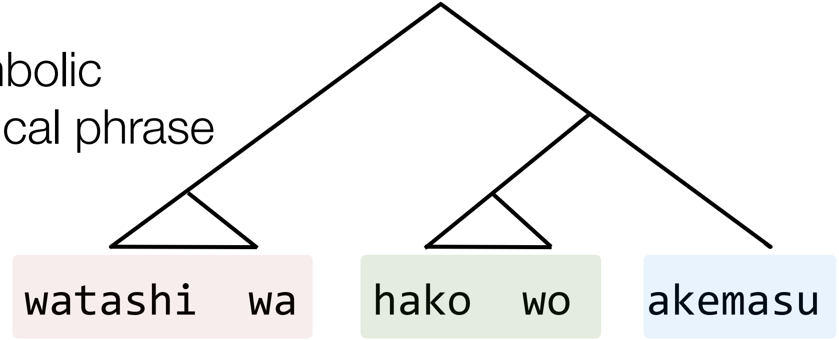
A Neural Synchronous Grammar

$$P_{\text{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t}) = P_{\text{LM}}(\text{“watashi wa”} \mid \text{“i”}) \times P_{\text{LM}}(\text{“hako wo”} \mid \text{“the box”}) \times P_{\text{LM}}(\text{“akemasu”} \mid \text{“open”}) \times$$

`i | the box | open`



Explicit use of symbolic structure (hierarchical phrase alignments)!



$P_{\text{BTG}}(\boldsymbol{t} \mid \boldsymbol{x})$

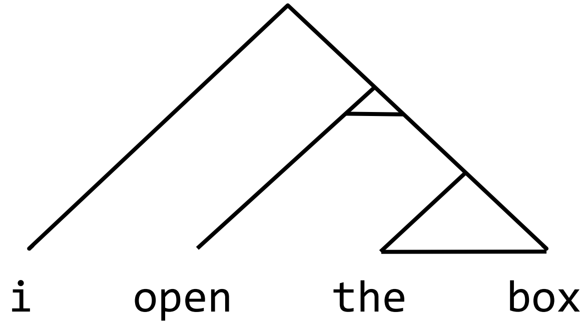
$P_{\text{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t})$

Bracketing Transduction Grammar (BTG) segments and reorders source sentence.

Seq2seq model translates segmented source phrases one-by-one.

A Neural Synchronous Grammar

`i | the box | open`



$P_{\mathsf{BTG}}(\boldsymbol{t} \mid \boldsymbol{x})$

Bracketing Transduction Grammar (BTG) segments and reorders source sentence.

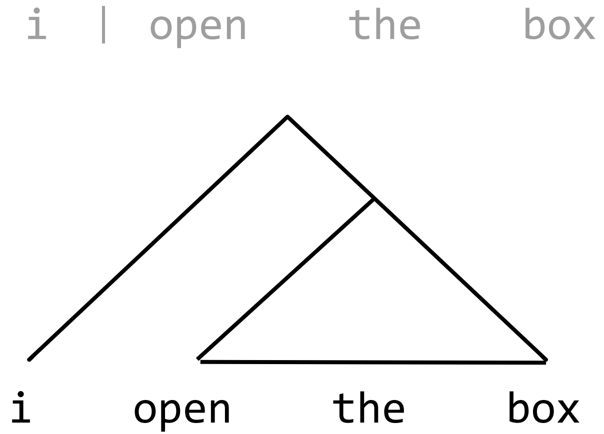
Distribution over number of phrase segments

$$P_{\mathsf{BTG}}(\boldsymbol{t} \mid \boldsymbol{x}) \propto P_{\mathsf{seg}}(n \mid \boldsymbol{x}) \prod_{1 \le i < j \le |\boldsymbol{x}|} e_S^\top f(\mathbf{h}_i, \mathbf{h}_j)$$

This example: 3 phrase segments (n=3) with 1 reordering.

What if the number of phrase segments is different (e.g., n=2)?

A Neural Synchronous Grammar



$P_{\text{BTG}}(\boldsymbol{t} \mid \boldsymbol{x})$

Bracketing Transduction Grammar (BTG) segments and reorders source sentence.

Distribution over number of phrase segments

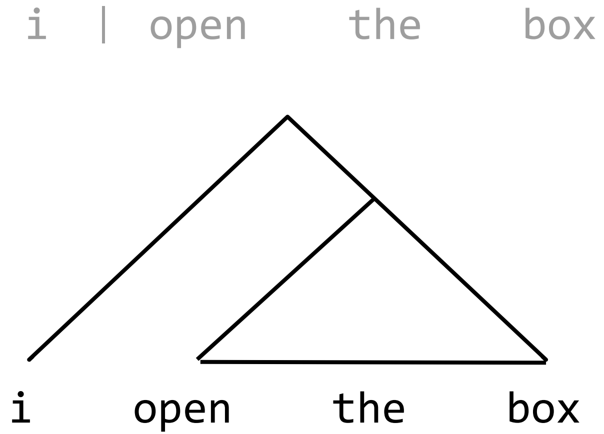
$$P_{\text{BTG}}(\boldsymbol{t} \mid \boldsymbol{x}) \propto P_{\text{seg}}(n \mid \boldsymbol{x}) \prod_{1 \leq i < j \leq |\boldsymbol{x}|} e_S^\top f(\mathbf{h}_i, \mathbf{h}_j)$$

This example: 3 phrase segments (n=3) with 1 reordering.

What if the number of phrase segments is different (e.g., n=2)?

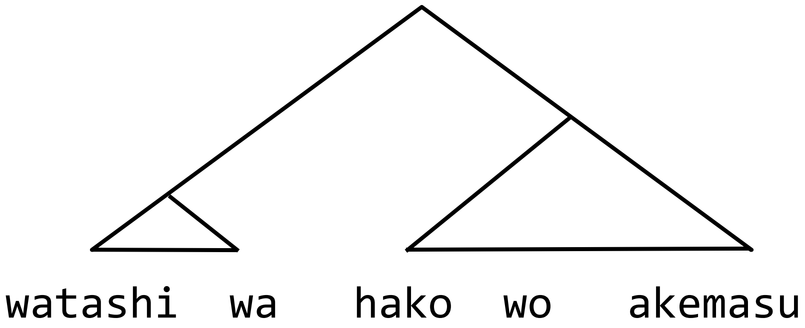
A Neural Synchronous Grammar

$$P_{\mathsf{seq2seq}}(\boldsymbol{y} \,|\, \boldsymbol{x}, \boldsymbol{t}) = P_{\mathsf{LM}}(\text{“watashi wa”} \mid \text{“i”}) \times$$
$$P_{\mathsf{LM}}(\text{“hako wo akemasu”} \mid \text{“open the box”})$$



$P_{\mathsf{BTG}}(\boldsymbol{t} \,|\, \boldsymbol{x})$

Bracketing Transduction Grammar (BTG) segments and reorders source sentence.

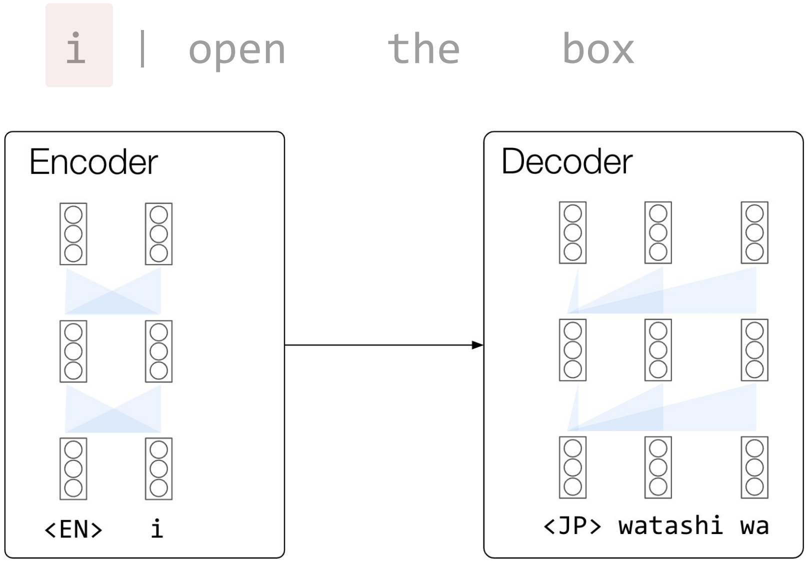


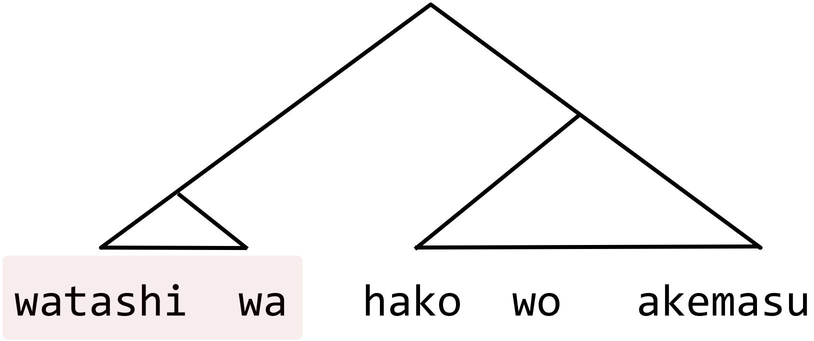
$P_{\mathsf{seq2seq}}(\boldsymbol{y} \,|\, \boldsymbol{x}, \boldsymbol{t})$

Seq2seq model translates segmented source phrases one-by-one.

A Neural Synchronous Grammar

$$P_{\text{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t}) = P_{\text{LM}}(\text{“watashi wa”} \mid \text{“i”}) \times$$
$$P_{\text{LM}}(\text{“hako wo akemasu”} \mid \text{“open the box”})$$





$$P_{\text{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t})$$

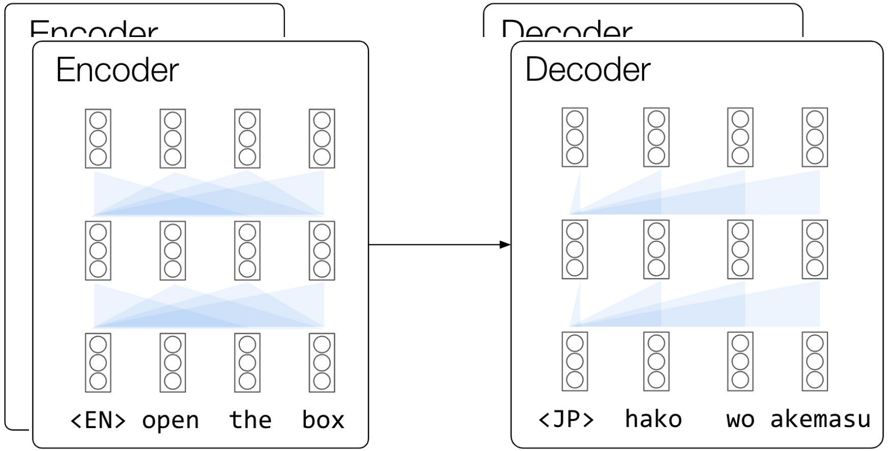
Bracketing Transduction Grammar (BTG) segments and reorders source sentence.

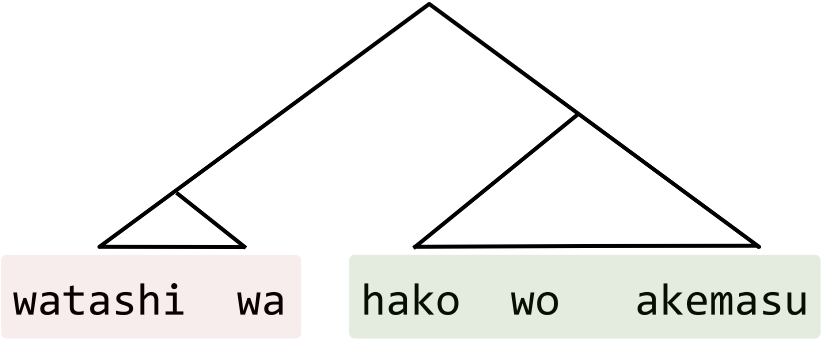
Seq2seq model translates segmented source phrases one-by-one.

A Neural Synchronous Grammar

$$P_{\text{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t}) = P_{\text{LM}}(\text{“watashi wa”} \mid \text{“i”}) \times P_{\text{LM}}(\text{“hako wo akemasu”} \mid \text{“open the box”})$$

i | open the box





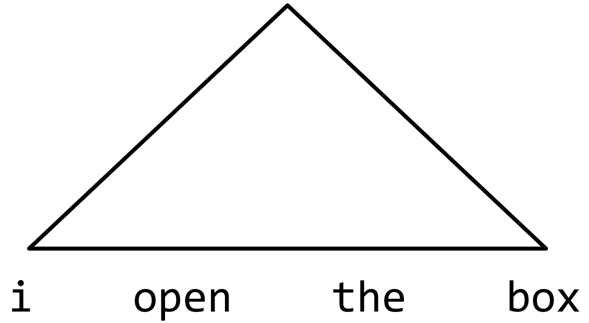
$$P_{\text{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t})$$

Pretrained LM has to (learn) to capture reorderdings in this case!

Seq2seq model translates segmented source phrases one-by-one.

A Neural Synchronous Grammar

i open the box

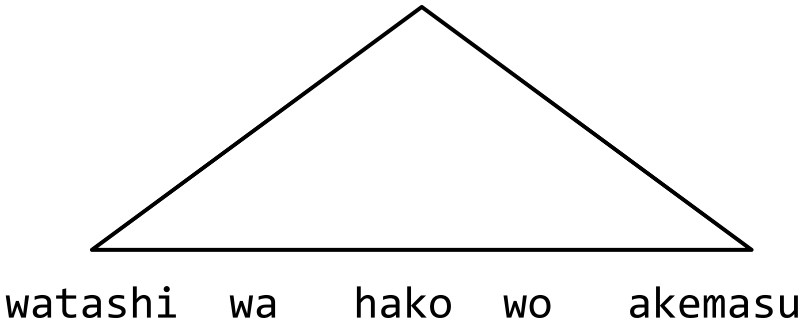


$$P_{\text{BTG}}(\boldsymbol{t} \mid \boldsymbol{x})$$

Bracketing Transduction Grammar (BTG) segments and reorders source sentence.

$$P_{\text{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t}) =$$

$$P_{\text{LM}}(\text{“watashiwa hako wo akemasu”} \mid \text{“i open the box”})$$



$$P_{\text{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t})$$

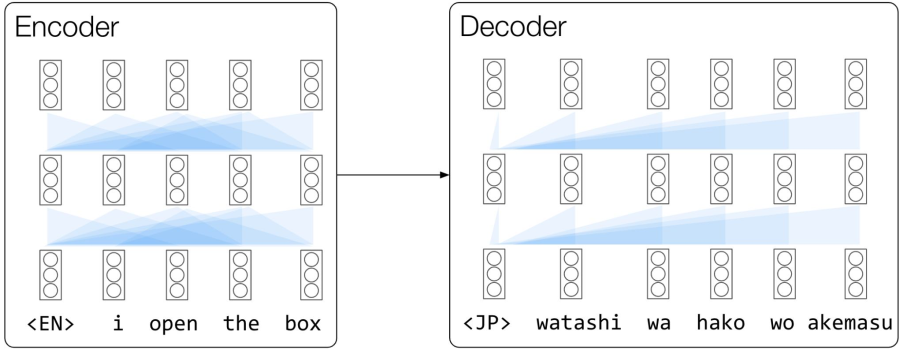
Seq2seq model translates segmented source phrases one-by-one.

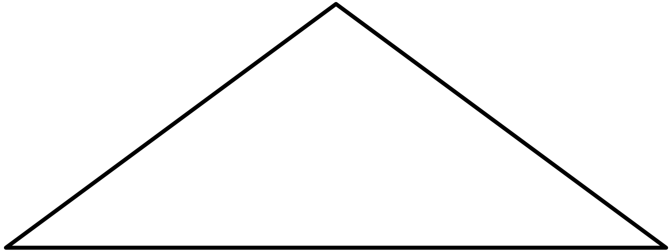
A Neural Synchronous Grammar

i open the box

$$P_{\text{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t}) =$$

P_{LM}(“watashiwa hako wo akemasu” | “i open the box”)





watashi wa hako wo akemasu

$$P_{\text{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t})$$

The setting with n=1 reduces to standard neural MT finetuning with pretrained LMs!

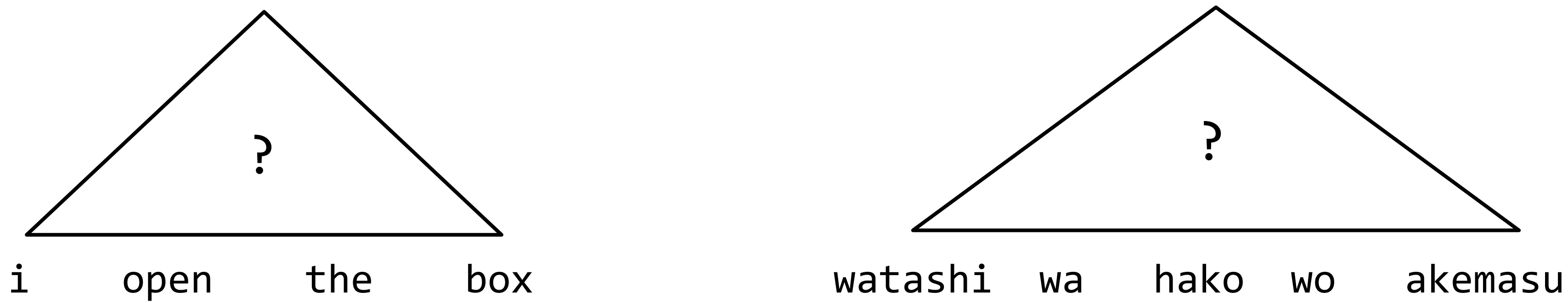
Seq2seq model translates segmented source phrases one-by-one.

LM-based Neural Synchronous Grammar

Synchronous grammar whose rule probabilities are given by a pretrained LM.

LM-based Neural Synchronous Grammar

Synchronous grammar whose rule probabilities are given by a pretrained LM.



All we have are source/target sentences.

No ground-truth structure ⇒ treat the hierarchical phrase alignments as latent variables.

LM-based Neural Synchronous Grammar

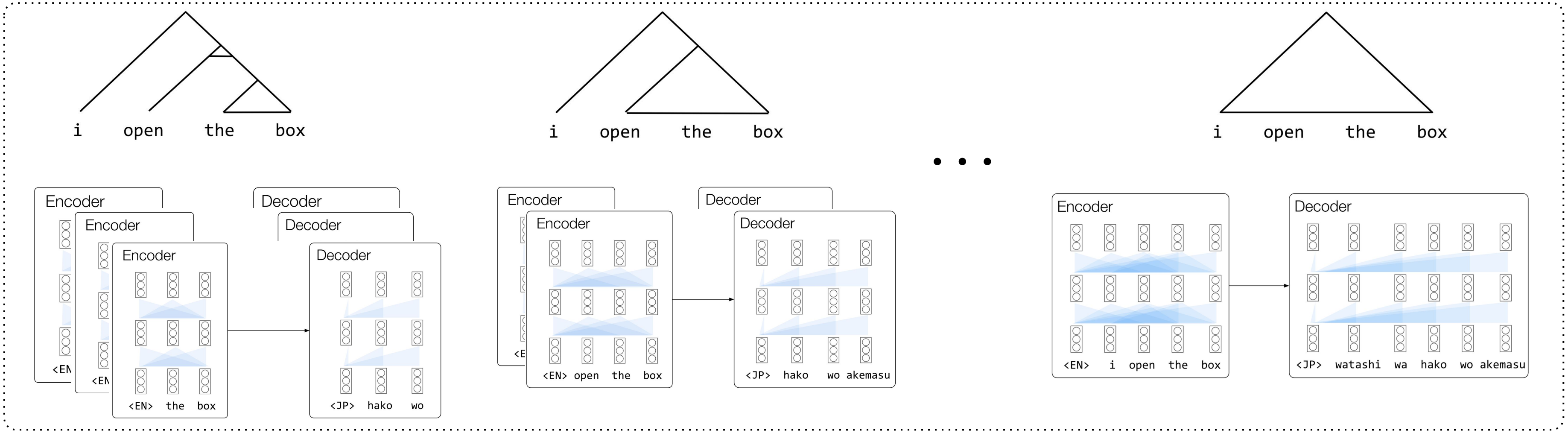
Synchronous grammar whose rule probabilities are given by a pretrained LM.

Learning/Finetuning: $\log P(\boldsymbol{y} \mid \boldsymbol{x}) = \log \left(\sum_{\boldsymbol{t} \in \mathcal{T}(\boldsymbol{x})} P_{\mathsf{BTG}}(\boldsymbol{t} \mid \boldsymbol{x}) \, P_{\mathsf{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t}) \right)$

LM-based Neural Synchronous Grammar

Synchronous grammar whose rule probabilities are given by a pretrained LM.

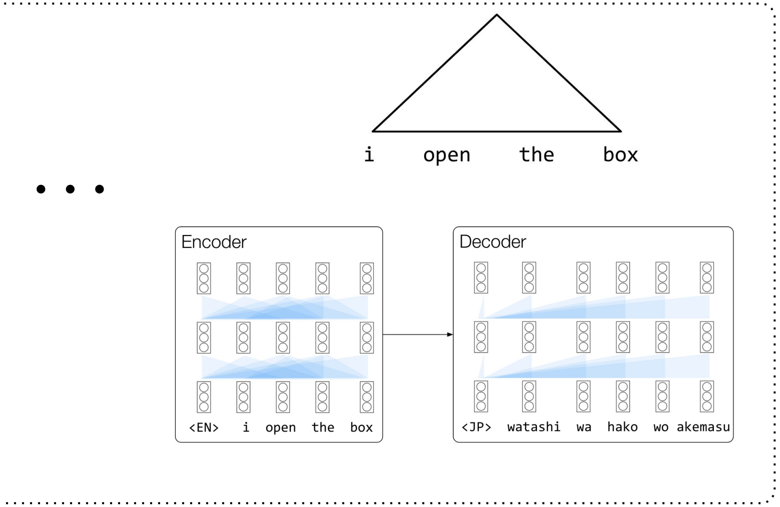
Learning/Finetuning: $\log P(\boldsymbol{y} \mid \boldsymbol{x}) = \log \left(\sum_{\boldsymbol{t} \in \mathcal{T}(\boldsymbol{x})} P_{\mathsf{BTG}}(\boldsymbol{t} \mid \boldsymbol{x}) \, P_{\mathsf{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t}) \right)$



LM-based Neural Synchronous Grammar

Synchronous grammar whose rule probabilities are given by a pretrained LM.

Learning/Finetuning: $\log P(\boldsymbol{y} \mid \boldsymbol{x}) = \log \left(\underbrace{\sum_{\boldsymbol{t} \in \mathcal{T}(\boldsymbol{x})} P_{\mathsf{BTG}}(\boldsymbol{t} \mid \boldsymbol{x}) \, P_{\mathsf{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t})} \right)$



Marginalization is tractable in theory but requires a $\mathcal{O}(L^7)$ dynamic program, where $L = \min\{|\boldsymbol{x}|, |\boldsymbol{y}|\}$

⇒ Variational inference to perform approximate inference in $\mathcal{O}(L^3)$

Learning

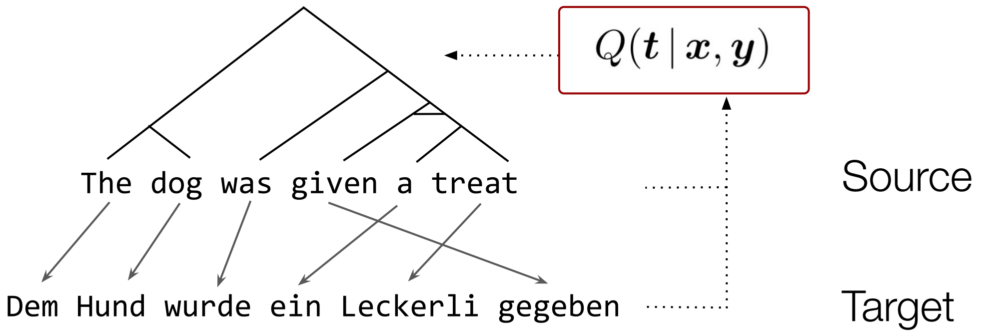
`The dog was given a treat` Source

`Dem Hund wurde ein Leckerli gegeben` Target

Learning

$Q(\boldsymbol{t} \mid \boldsymbol{x}, \boldsymbol{y})$: Variational Synchronous Parser. CRF over pretrained encoder/decoder representations.

ϕ



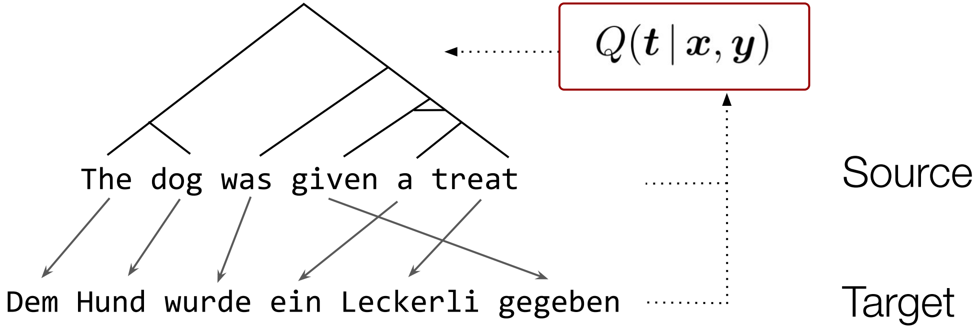
Learning

$Q(\boldsymbol{t} \mid \boldsymbol{x}, \boldsymbol{y})$: Variational Synchronous Parser. CRF over pretrained encoder/decoder representations.

ϕ

$P_{\text{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t})$: Seq2seq model initialized with pretrained encoder/decoder. Finetuned on sampled source phrases.

θ



Hierarchically aligned phrases

$P_{\text{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t})$

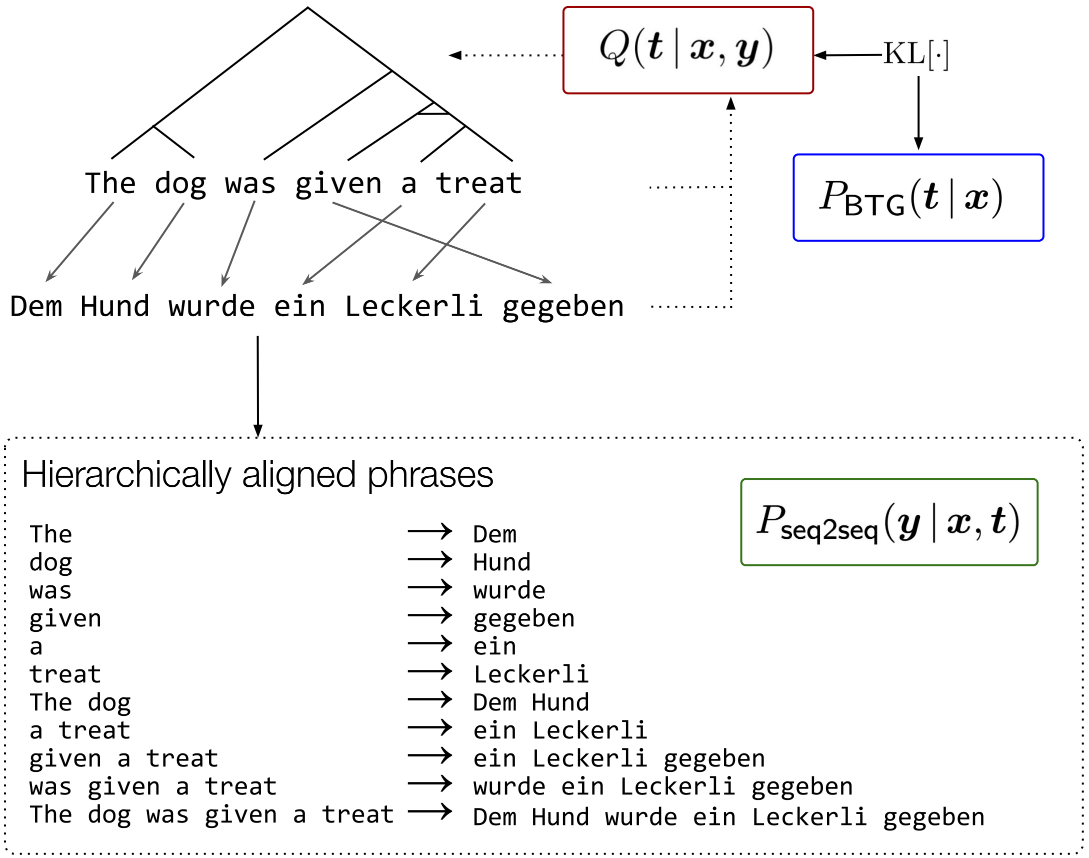
```
The                      ⟶ Dem
dog                      ⟶ Hund
was                      ⟶ wurde
given                    ⟶ gegeben
a                        ⟶ ein
treat                    ⟶ Leckerli
The dog                  ⟶ Dem Hund
a treat                  ⟶ ein Leckerli
given a treat            ⟶ ein Leckerli gegeben
was given a treat        ⟶ wurde ein Leckerli gegeben
The dog was given a treat ⟶ Dem Hund wurde ein Leckerli gegeben
```

Learning

$Q(\boldsymbol{t} \mid \boldsymbol{x}, \boldsymbol{y})$: Variational Synchronous Parser. CRF over pretrained encoder/decoder representations. ϕ

$P_{\text{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t})$: Seq2seq model initialized with pretrained encoder/decoder. Finetuned on sampled source phrases. θ

$P_{\text{BTG}}(\boldsymbol{t} \mid \boldsymbol{x})$: “Prior” BTG parser. CRF over pretrained encoder/decoder representations. π



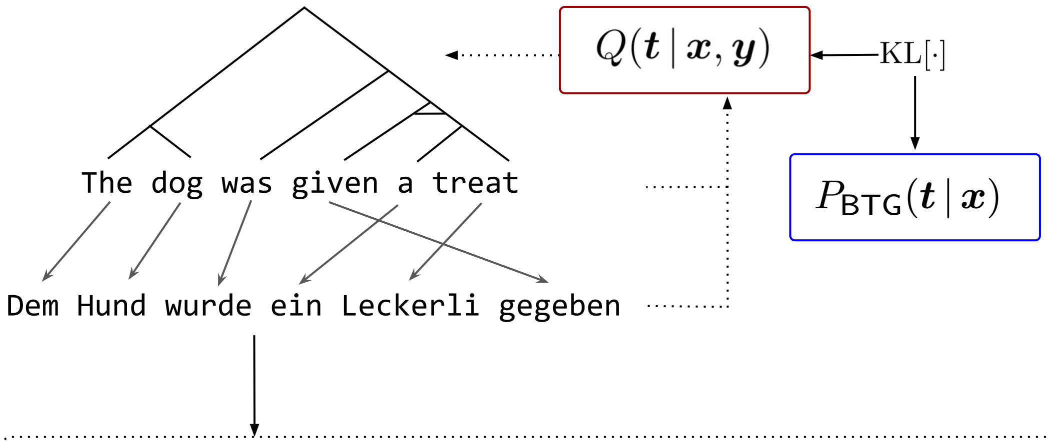
Learning

$Q(\boldsymbol{t} \mid \boldsymbol{x}, \boldsymbol{y})$: Variational Synchronous Parser. CRF over pretrained encoder/decoder representations. ϕ

$P_{\text{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t})$: Seq2seq model initialized with pretrained encoder/decoder. Finetuned on sampled source phrases. θ

$P_{\text{BTG}}(\boldsymbol{t} \mid \boldsymbol{x})$: “Prior” BTG parser. CRF over pretrained encoder/decoder representations. π

$$\max_{\theta,\phi,\pi} \mathbb{E}_{Q(\boldsymbol{t} \mid \boldsymbol{x},\boldsymbol{y};\, \phi)}[\log P_{\text{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t};\, \theta)] - \mathrm{KL}[Q(\boldsymbol{t} \mid \boldsymbol{x}, \boldsymbol{y};\, \phi) \,\|\, P_{\text{BTG}}(\boldsymbol{t} \mid \boldsymbol{x};\, \pi)]$$



Hierarchically aligned phrases

$P_{\text{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t})$

The	$\longrightarrow$	Dem
dog	$\longrightarrow$	Hund
was	$\longrightarrow$	wurde
given	$\longrightarrow$	gegeben
a	$\longrightarrow$	ein
treat	$\longrightarrow$	Leckerli
The dog	$\longrightarrow$	Dem Hund
a treat	$\longrightarrow$	ein Leckerli
given a treat	$\longrightarrow$	ein Leckerli gegeben
was given a treat	$\longrightarrow$	wurde ein Leckerli gegeben
The dog was given a treat	$\longrightarrow$	Dem Hund wurde ein Leckerli gegeben

Learning

$Q(\boldsymbol{t} \mid \boldsymbol{x}, \boldsymbol{y})$: Variational Synchronous Parser. CRF over pretrained encoder/decoder representations.

ϕ

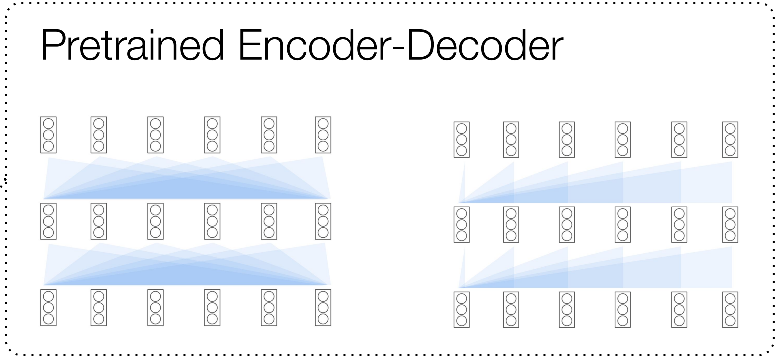
$P_{\text{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t})$: Seq2seq model initialized with pretrained encoder/decoder. Finetuned on sampled source phrases.

θ

$P_{\text{BTG}}(\boldsymbol{t} \mid \boldsymbol{x})$: "Prior" BTG parser. CRF over pretrained encoder/decoder representations.

π

$$\max_{\theta,\phi,\pi} \; \mathbb{E}_{Q(\boldsymbol{t} \mid \boldsymbol{x},\boldsymbol{y};\, \phi)}[\log P_{\text{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t};\, \theta)] - \mathrm{KL}[Q(\boldsymbol{t} \mid \boldsymbol{x}, \boldsymbol{y};\, \phi) \,\|\, P_{\text{BTG}}(\boldsymbol{t} \mid \boldsymbol{x};\, \pi)]$$



Encoder-decoder parameter-sharing across all three models.

Encoder-decoder initialized (and finetuned) from the same backbone LM (mBART/mT5).

(Comparatively) Few additional parameters on top of the shared model.

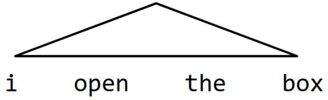
Inference: Neural Grammar + Single-segment Decoding

Mode 1: Discard $P_{\mathsf{BTG}}(\boldsymbol{t} \mid \boldsymbol{x})$ and decode with one segment (n=1)

Inference: Neural Grammar + Single-segment Decoding

Mode 1: Discard $P_{\mathsf{BTG}}(\boldsymbol{t}\,|\,\boldsymbol{x})$ and decode with one segment (n=1)

$$\arg\max_{\boldsymbol{y}\in\mathcal{Y}} \; P_{\mathsf{seq2seq}}(\boldsymbol{y}\,|\,\boldsymbol{x}, \boldsymbol{t} =$$



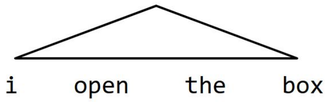
$$)$$

$$= \arg\max_{\boldsymbol{y}\in\mathcal{Y}} \; P_{\mathsf{LM}}(\boldsymbol{y}\,|\,\boldsymbol{x})$$

Inference: Neural Grammar + Single-segment Decoding

Mode 1: Discard $P_{\mathsf{BTG}}(\boldsymbol{t} \mid \boldsymbol{x})$ and decode with one segment (n=1)

$$\arg\max_{\boldsymbol{y}\in\mathcal{Y}} \; P_{\mathsf{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t} =$$



$$)$$

$$= \arg\max_{\boldsymbol{y}\in\mathcal{Y}} \; P_{\mathsf{LM}}(\boldsymbol{y} \mid \boldsymbol{x})$$

Reduces to regular beam search in neural MT ⇒ Fast inference.

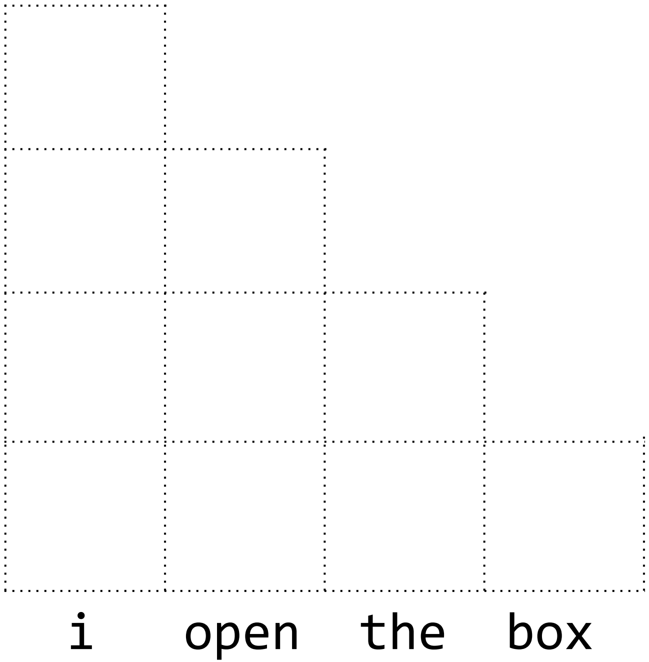
Use latent symbolic structures to softly guide (overly) flexible neural networks.

Inference: Neural Grammar + CKY Decoding

Mode 2: Jointly parse and translate. $\arg\max_{\boldsymbol{t},\boldsymbol{y}} \; P_{\mathsf{BTG}}(\boldsymbol{t} \mid \boldsymbol{x}) \, P_{\mathsf{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t})$

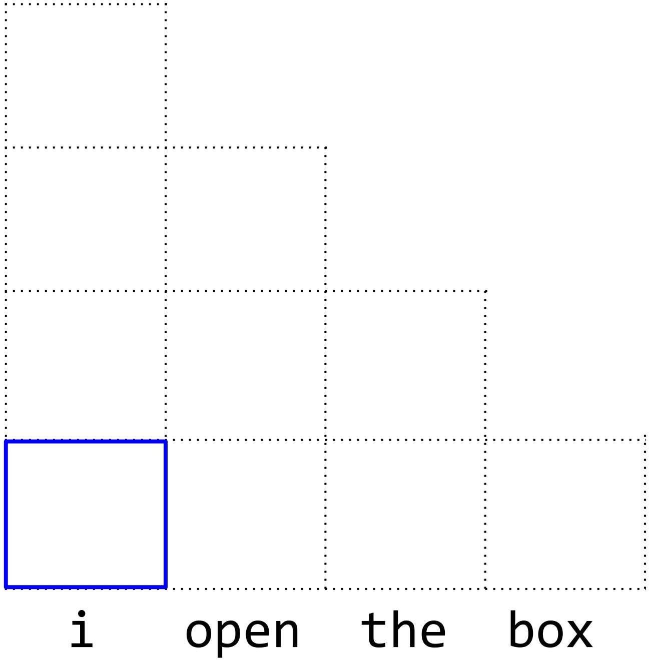
Inference: Neural Grammar + CKY Decoding

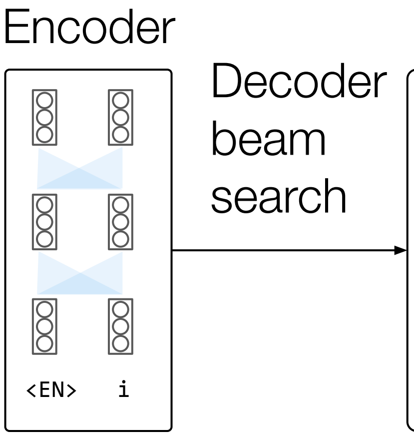
Mode 2: Jointly parse and translate. $\underset{\boldsymbol{t},\boldsymbol{y}}{\arg\max}\; P_{\mathsf{BTG}}(\boldsymbol{t}\mid\boldsymbol{x})\, P_{\mathsf{seq2seq}}(\boldsymbol{y}\mid\boldsymbol{x},\boldsymbol{t})$



Inference: Neural Grammar + CKY Decoding

Mode 2: Jointly parse and translate. $\underset{\boldsymbol{t},\boldsymbol{y}}{\arg\max}\ P_{\mathsf{BTG}}(\boldsymbol{t} \mid \boldsymbol{x})\, P_{\mathsf{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t})$



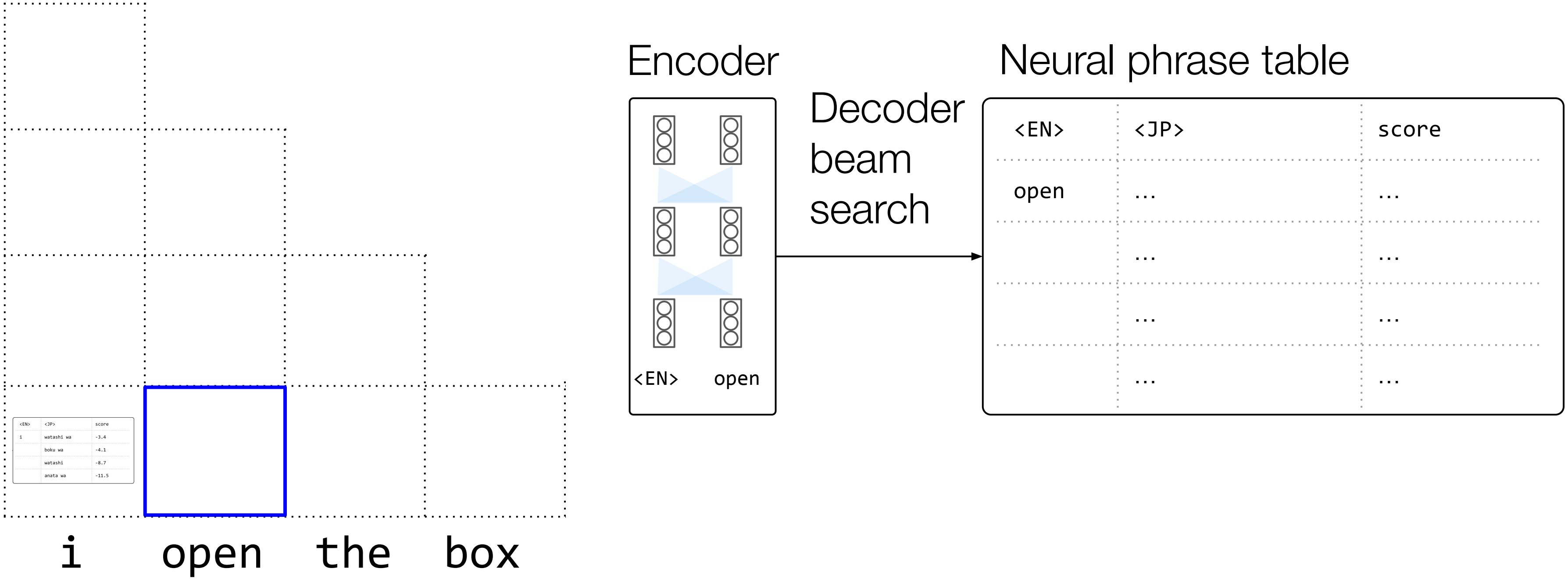


Neural phrase table

<EN>	<JP>	score
i	watashi wa	-3.4
	boku wa	-4.1
	watashi	-8.7
	anata wa	-11.5

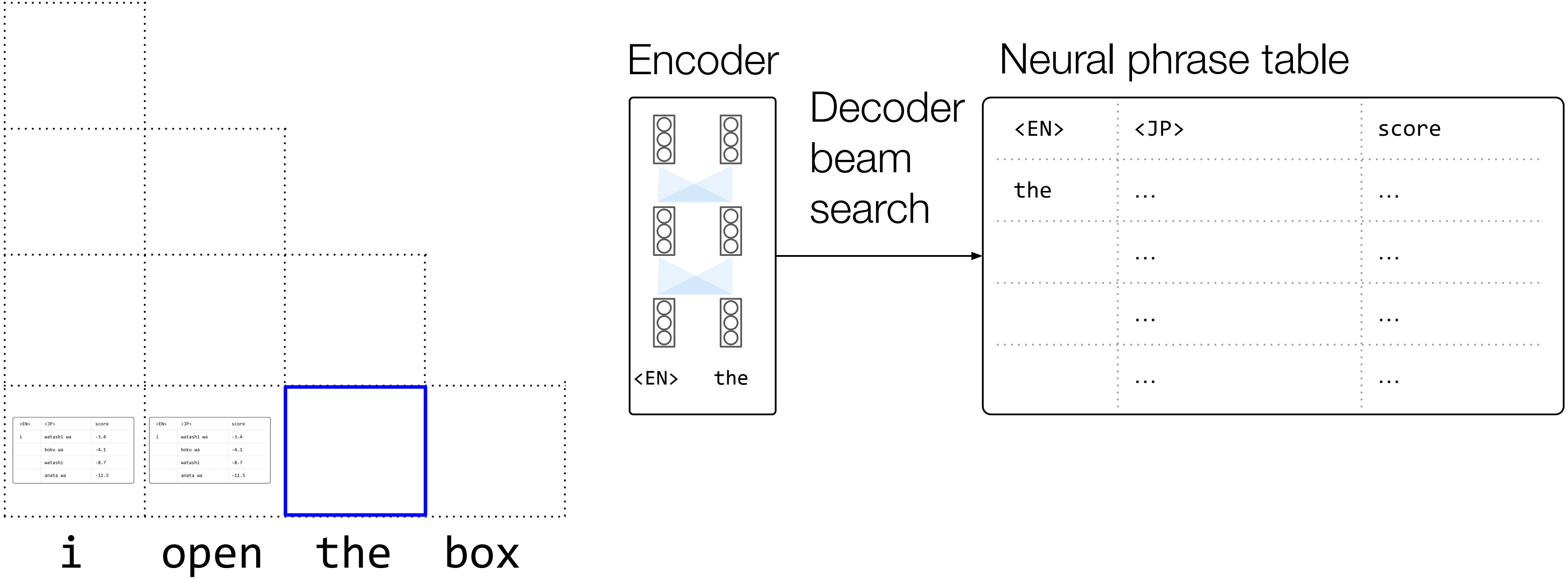
Inference: Neural Grammar + CKY Decoding

Mode 2: Jointly parse and translate. $\arg\max_{\boldsymbol{t},\boldsymbol{y}} \; P_{\mathsf{BTG}}(\boldsymbol{t} \mid \boldsymbol{x}) \, P_{\mathsf{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t})$



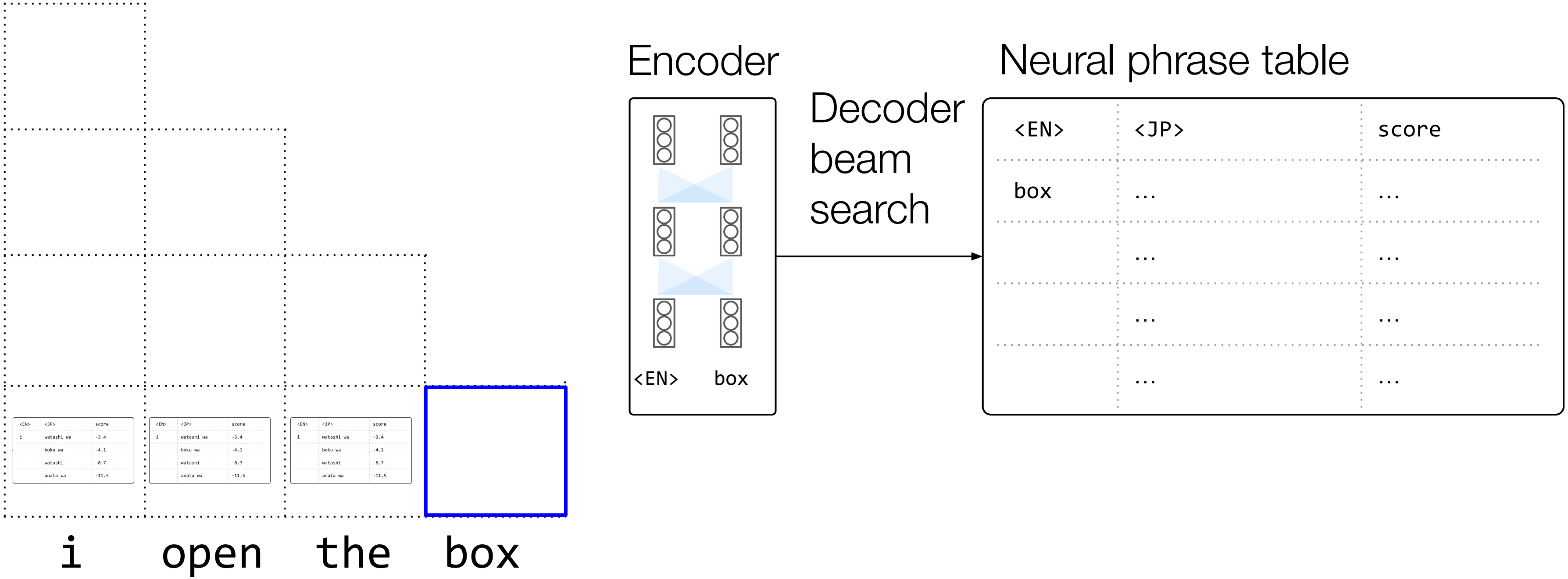
Inference: Neural Grammar + CKY Decoding

Mode 2: Jointly parse and translate. $\arg\max_{\boldsymbol{t},\boldsymbol{y}} \; P_{\mathsf{BTG}}(\boldsymbol{t} \mid \boldsymbol{x}) \, P_{\mathsf{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t})$



Inference: Neural Grammar + CKY Decoding

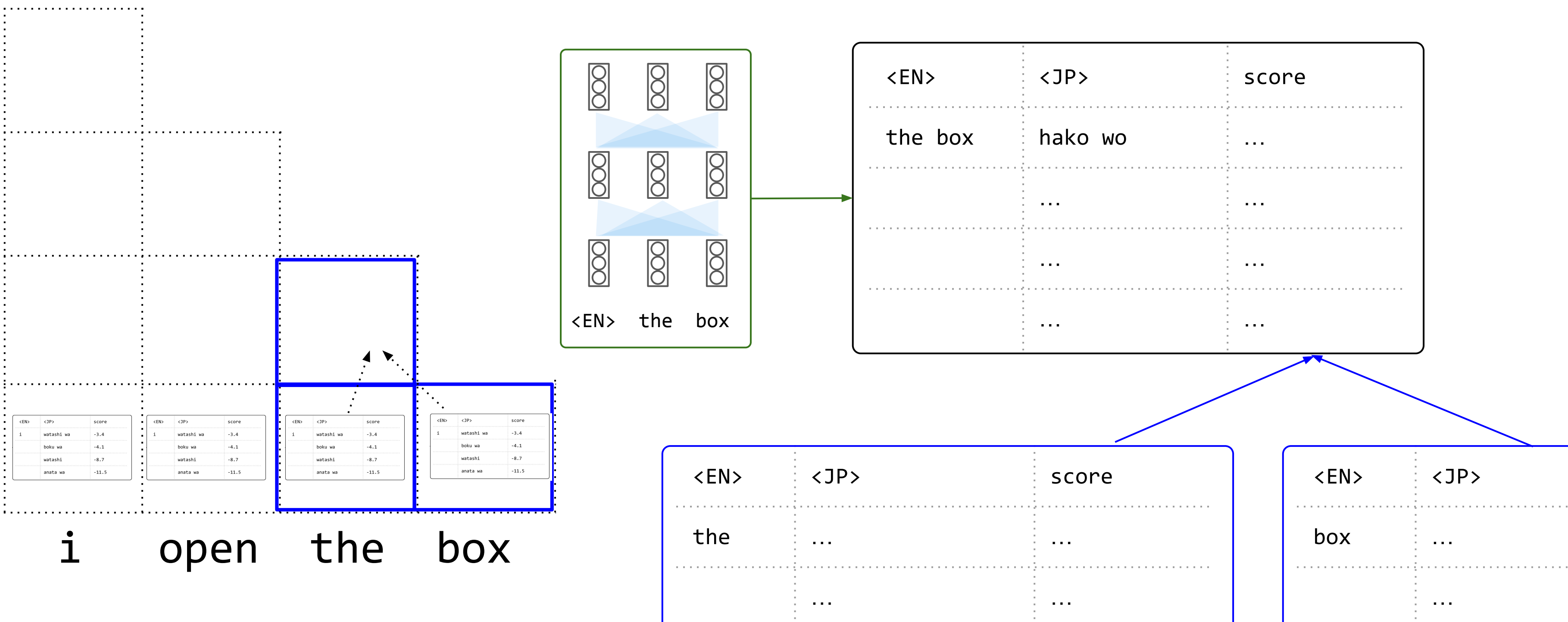
Mode 2: Jointly parse and translate. $\arg\max_{\boldsymbol{t},\boldsymbol{y}} \; P_{\mathsf{BTG}}(\boldsymbol{t} \mid \boldsymbol{x}) \, P_{\mathsf{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t})$



Inference: Neural Grammar + CKY Decoding

Mode 2: Jointly parse and translate.

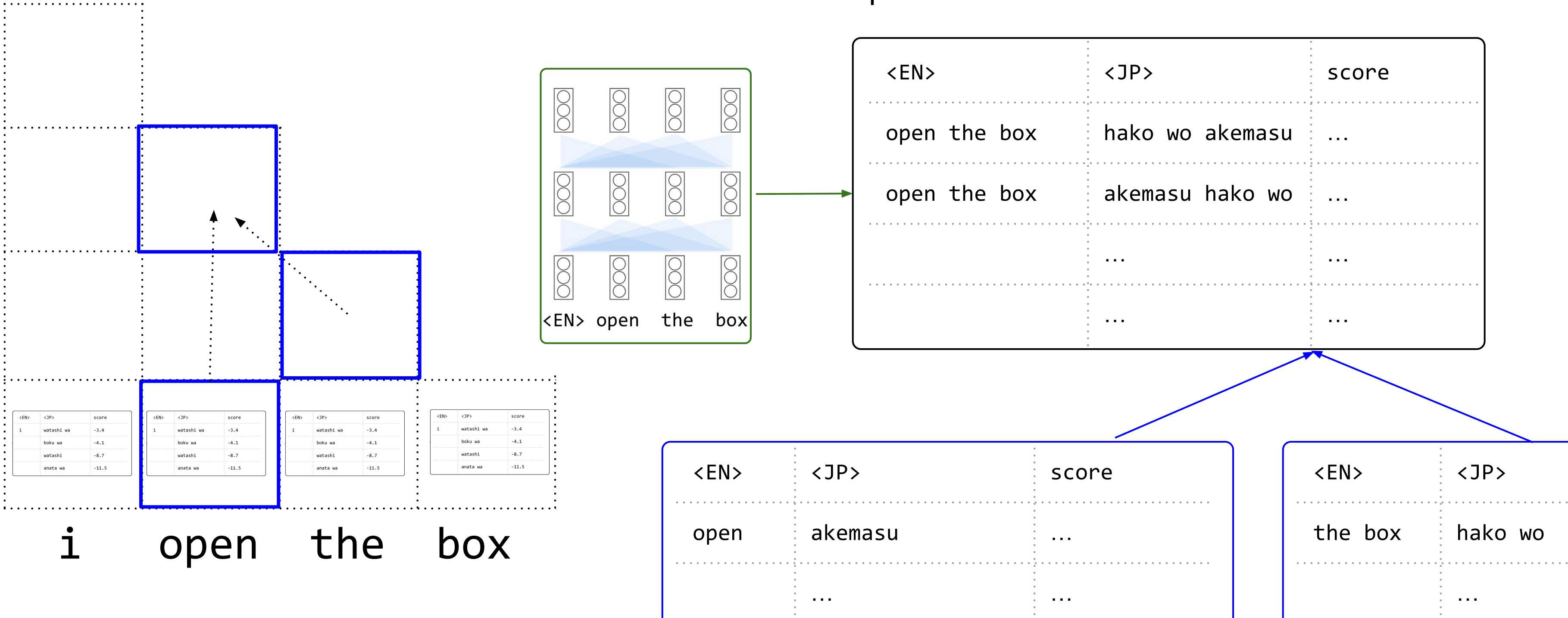
phrase-level translation score +
merging children score



Inference: Neural Grammar + CKY Decoding

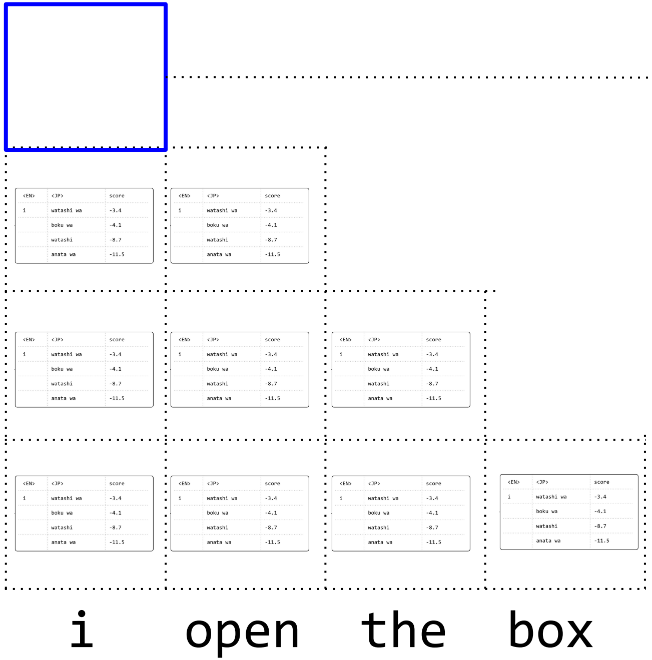
Mode 2: Jointly parse and translate.

phrase-level translation score +
merging children score +
phrase inversion score



Inference: Neural Grammar + CKY Decoding

Mode 2: Jointly parse and translate. $\arg\max_{\boldsymbol{t},\boldsymbol{y}} \; P_{\mathsf{BTG}}(\boldsymbol{t} \mid \boldsymbol{x}) \, P_{\mathsf{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t})$



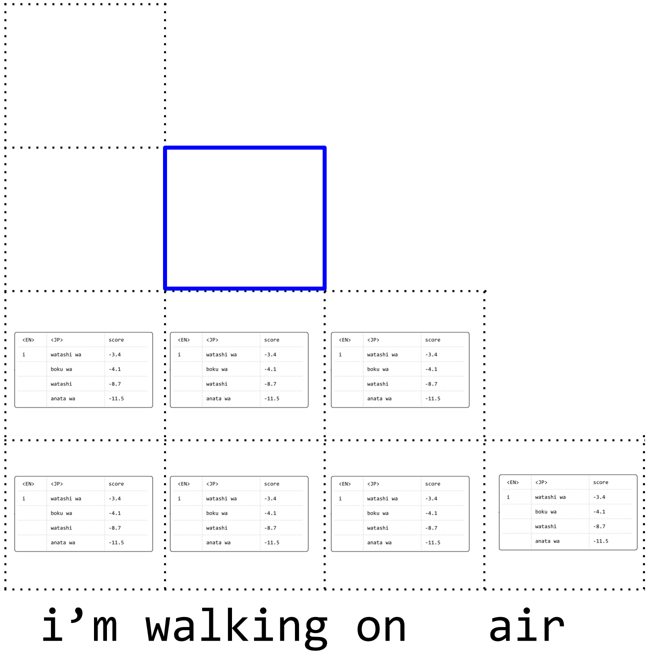
<EN>	<JP>	score
i open the box	watashi wa hako wo akemasu	…
	boku wa hako wo akemasu	…
	…	…
	…	…

CKY-style bottom up dynamic programming + beam search from LM.

"Cube-pruned" CKY [Huang and Chiang '05]

Inference: Neural Grammar + CKY Decoding

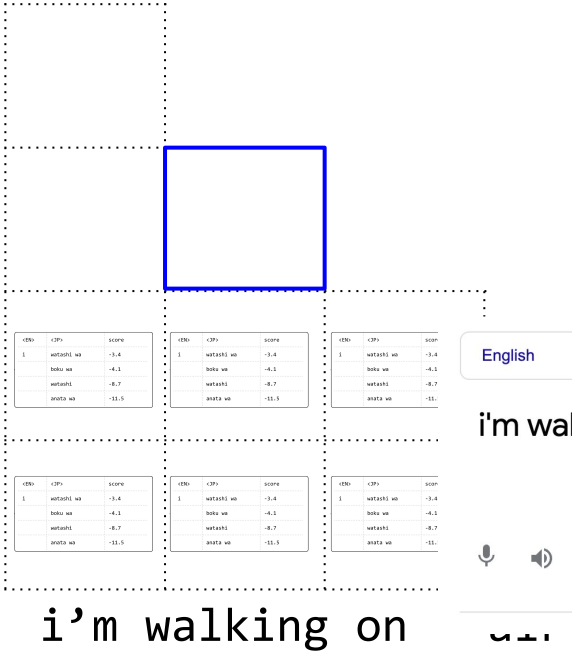
Mode 2: Jointly parse and translate. $\arg\max_{\boldsymbol{t},\boldsymbol{y}} \; P_{\mathsf{BTG}}(\boldsymbol{t} \mid \boldsymbol{x})\, P_{\mathsf{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t})$



<EN>	<JP>	score
walking on air	kuuchuu o aruite imasu	-4.2
	kuuchuu o aruite iru	-6.5
	…	…
	…	…

Inference: Neural Grammar + CKY Decoding

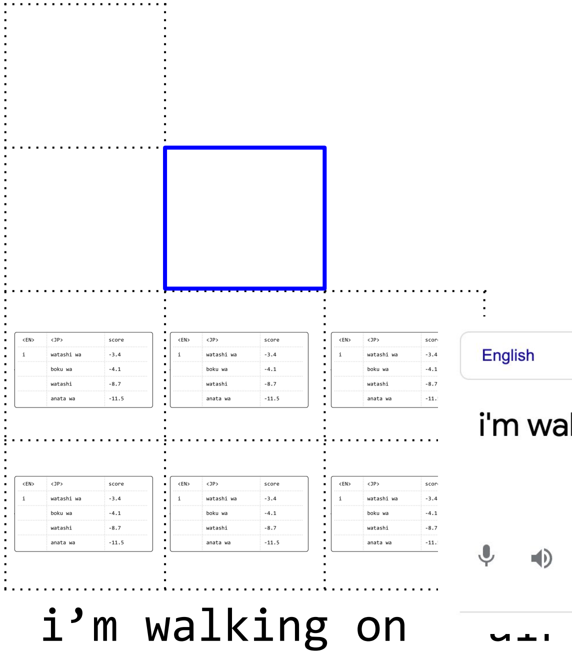
Mode 2: Jointly parse and translate. $\arg\max_{\boldsymbol{t},\boldsymbol{y}} \; P_{\mathsf{BTG}}(\boldsymbol{t} \mid \boldsymbol{x}) \, P_{\mathsf{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t})$



<EN>	<JP>	score
walking on air	kuuchuu o aruite imasu	-4.2
	kuuchuu o aruite iru	-6.5
	...	...

Inference: Neural Grammar + CKY Decoding

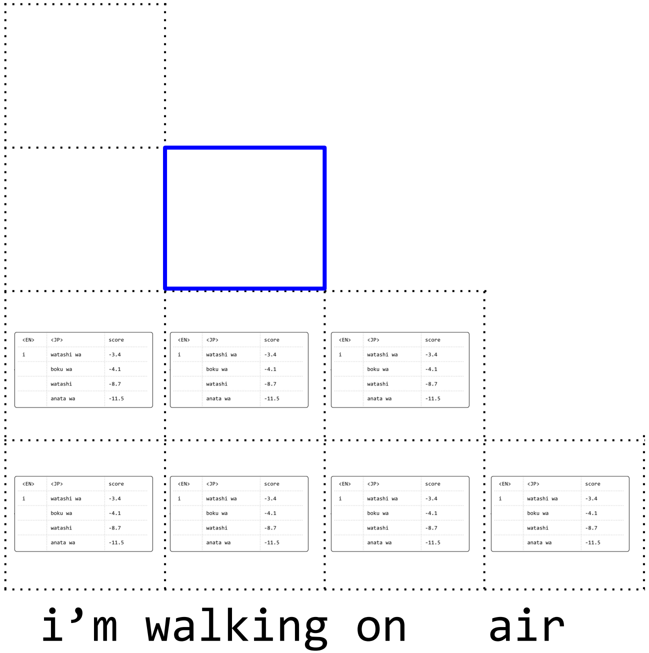
Mode 2: Jointly parse and translate. $\arg\max_{\boldsymbol{t},\boldsymbol{y}} \; P_{\mathsf{BTG}}(\boldsymbol{t} \mid \boldsymbol{x}) \, P_{\mathsf{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t})$



<EN>	<JP>	score
walking on air	~~kuuchuu o aruite imasu~~	~~-4.2~~
	kuuchuu o aruite iru	-6.5
	...	...

Inference: Neural Grammar + CKY Decoding

Mode 2: Jointly parse and translate. $\arg\max_{\boldsymbol{t},\boldsymbol{y}} \; P_{\mathsf{BTG}}(\boldsymbol{t} \mid \boldsymbol{x}) \, P_{\mathsf{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t})$

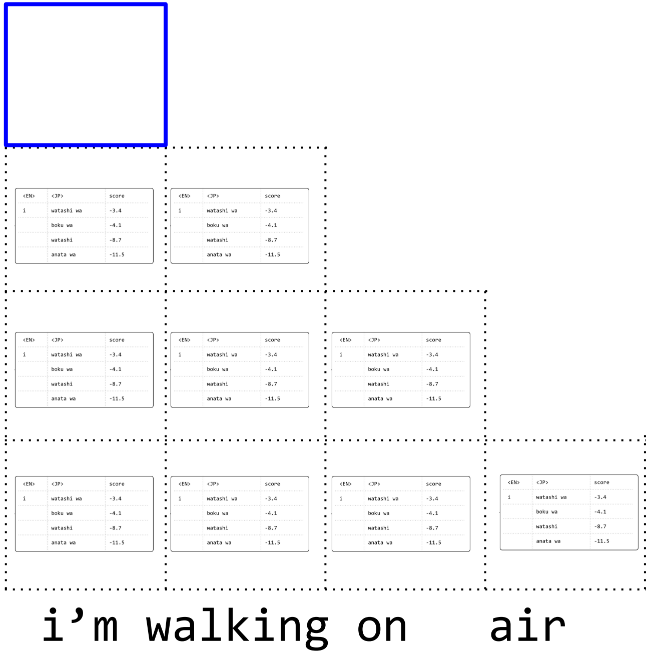


<EN>	<JP>	score
walking on air	totemo wakuwaku shiteimasu	0
	kuuchuu o aruite imasu	-4.2
	kuuchuu o aruite iru	-6.5
	…	…

$P(\text{“totemo wakuwaku shiteimasu”} \mid \text{“walking on air”}) = 1$

Inference: Neural Grammar + CKY Decoding

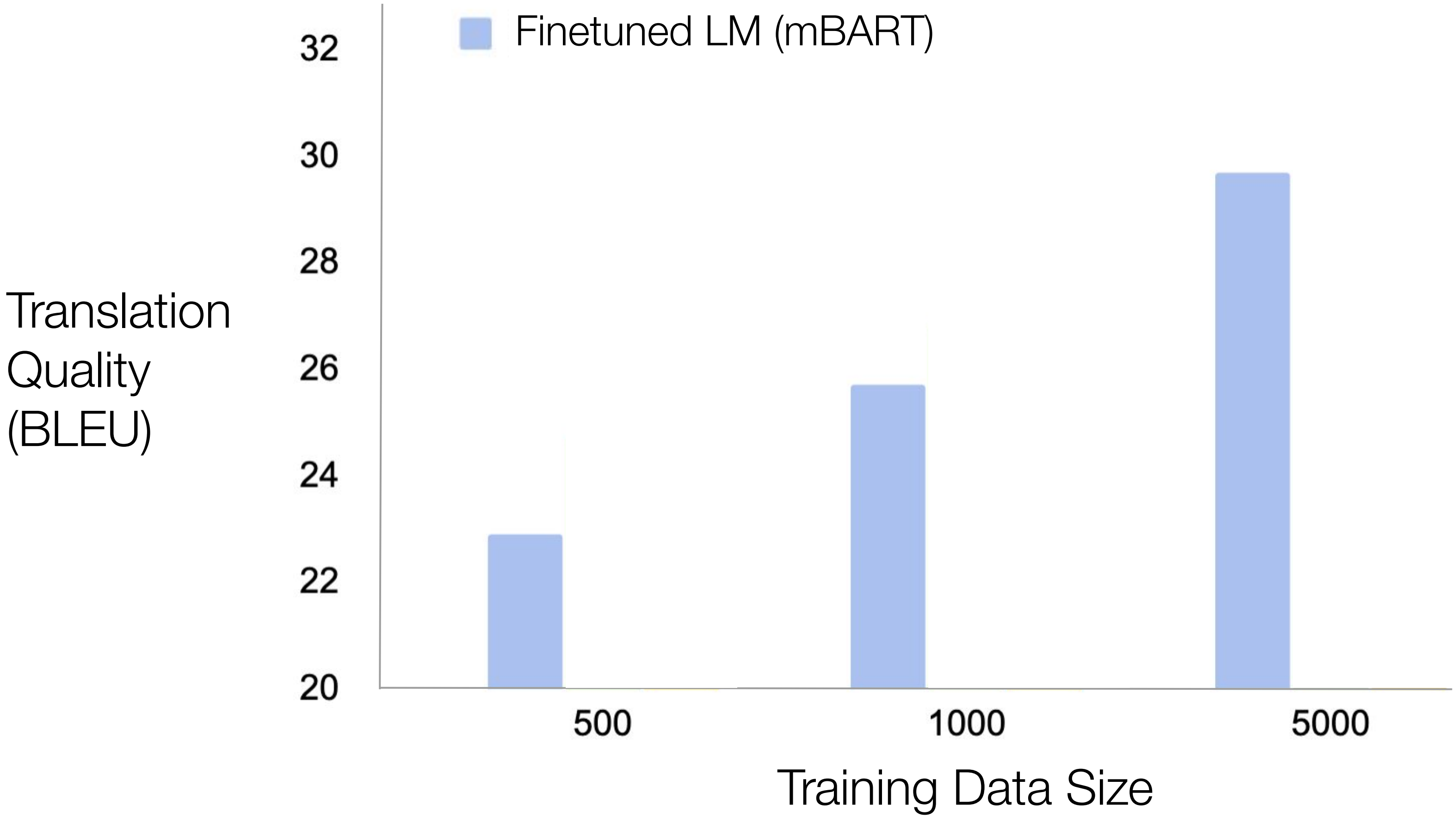
Mode 2: Jointly parse and translate. $\arg\max_{\boldsymbol{t},\boldsymbol{y}} \; P_{\mathsf{BTG}}(\boldsymbol{t} \mid \boldsymbol{x})\, P_{\mathsf{seq2seq}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{t})$



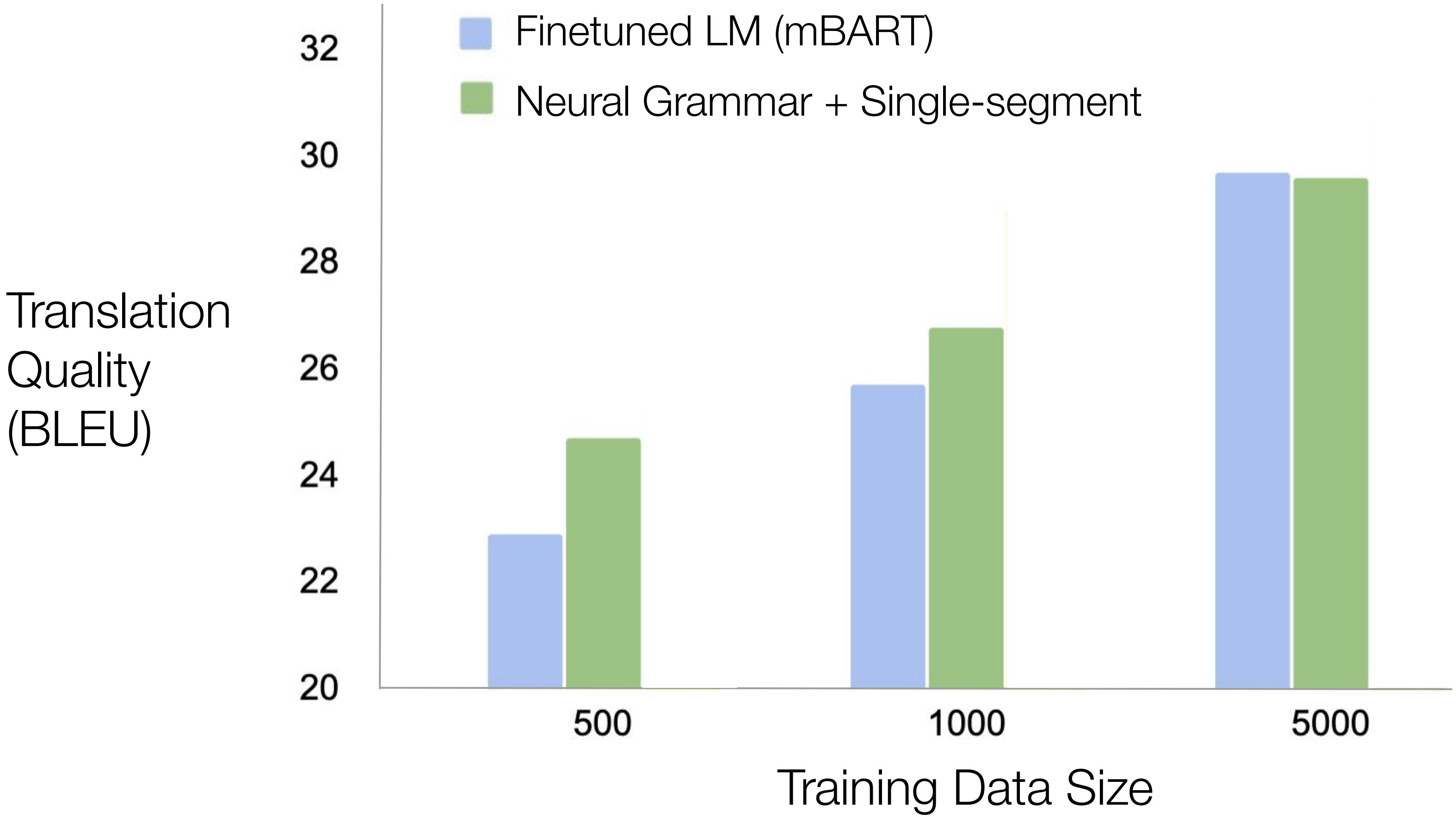
More expensive than single-segment decoding, but can incorporate new translation rules during inference for idioms & transliterations.

Results: German → English

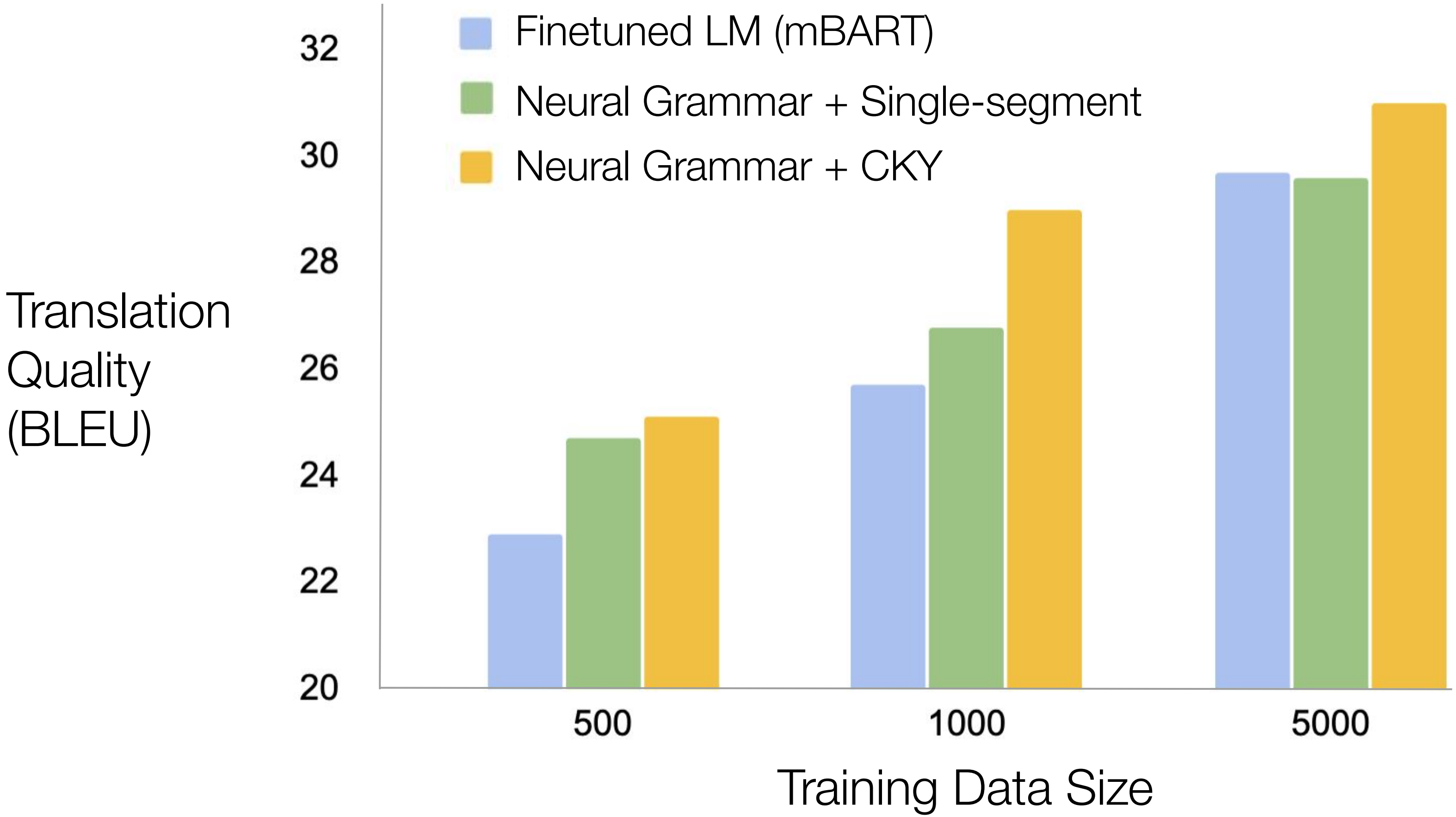
Results: German → English



Results: German → English



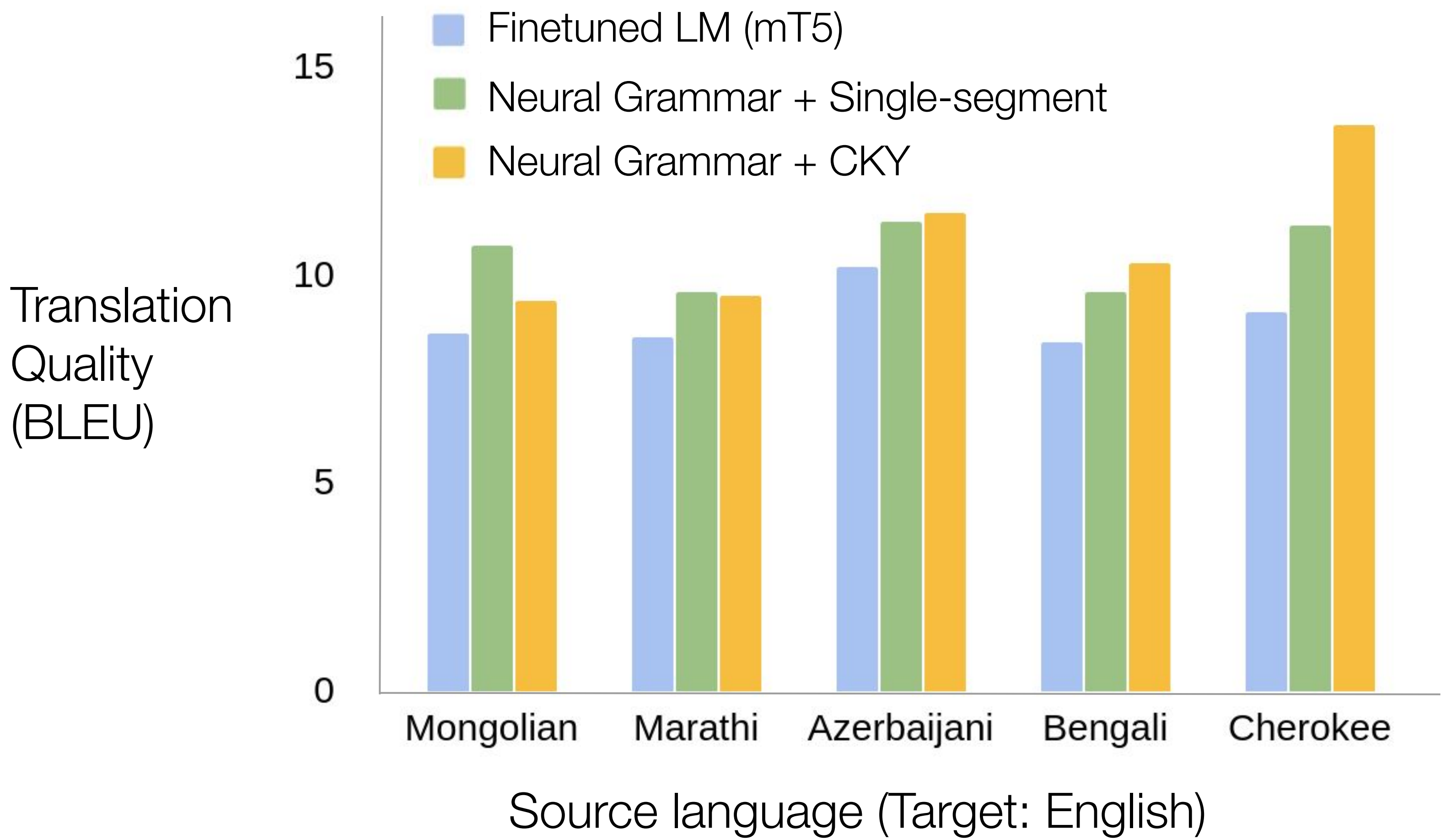
Results: German → English



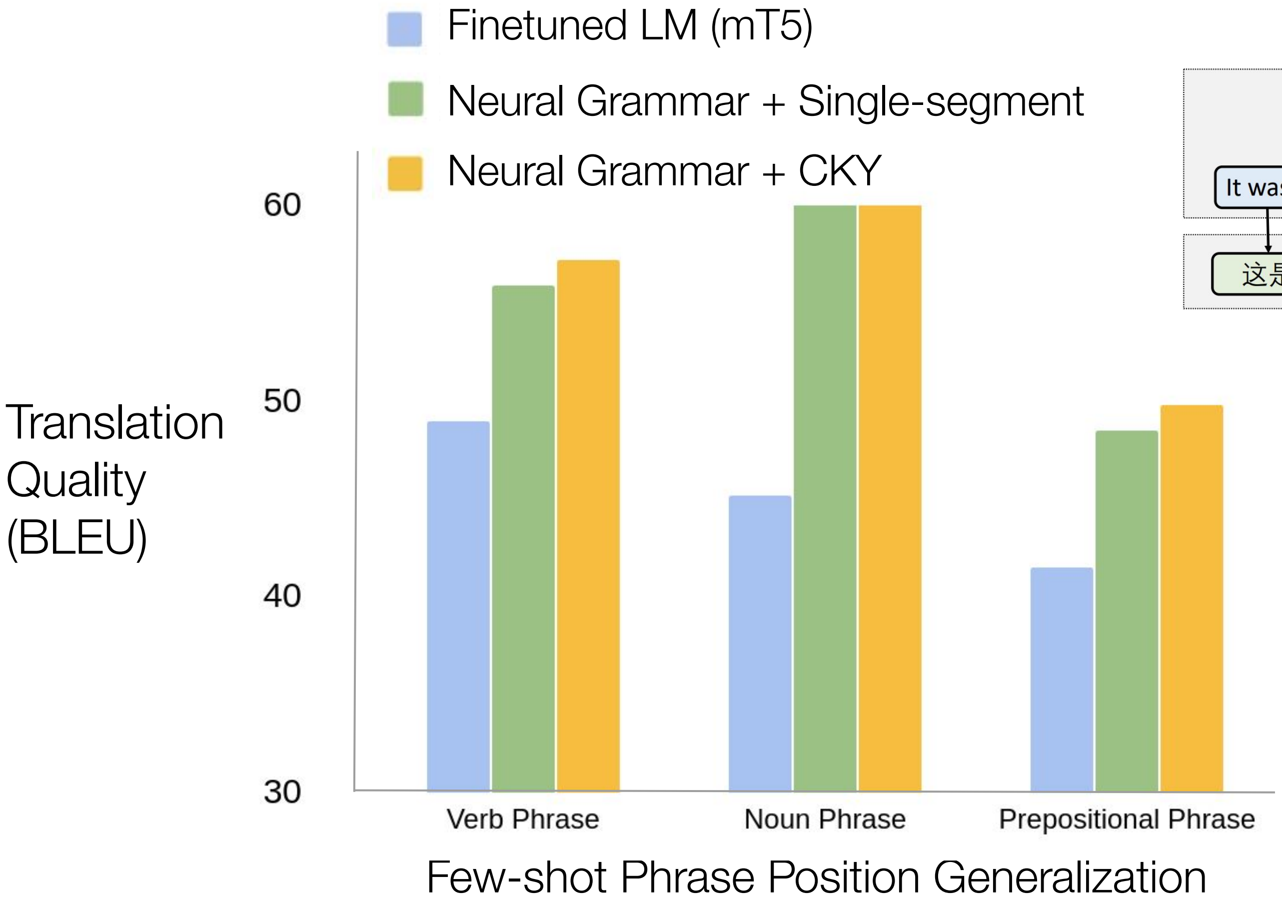
Results: German → English

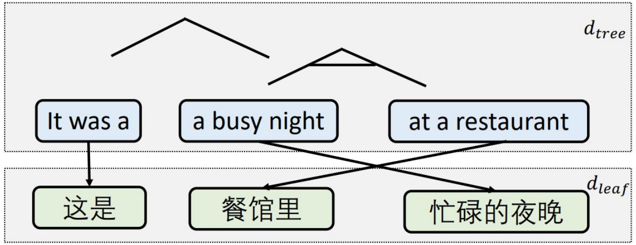
Input:	Er hat das Unternehmen *von Grund auf* aufgebaut
New translation rule:	von Grund auf → from scratch
Original Prediction:	He built the company from the ground up
New Prediction:	He built the company from scratch
Reference:	He built the company from scratch
Google Translate:	He built the company from the ground up
Input:	Die europäische Krise *schließt den Kreis*
New translation rule:	schließt den Kreis → is coming full circle
Original Prediction:	The euro crisis closes the loop
New Prediction:	The European crisis is coming full circle
Reference:	The European crisis is coming full circle
Google Translate:	The European crisis closes the circle
Input:	Die ARD strahlte gestern um 20:15 “Aus dem Nichts" aus
New translation rule:	“Aus dem Nichts" → “In the Fade", 20:15 → 8.15 pm
Original Prediction:	The ARD aired yesterday at 20:15 “Out of Nowhere"
New Prediction:	The ARD aired yesterday 8.15 pm “In the Fade"
Reference:	ARD broadcast “In the Fade” yesterday at 8.15 pm
Google Translate:	Yesterday at 8:15 p.m., ARD broadcast "Out of Nowhere".

Results: Low Resource MT



Results: “Compositional” MT on English→Chinese





Summary

Latent symbolic structures (hierarchical phrase alignments) can be used in conjunction with pretrained LMs.

Neural grammar can incorporate “hard” translation rules during inference.

Still requires finetuning the pretrained model… what if we can’t?

Translate the following sentence to English: *Pada dasarnya, hal tersebut terbagi ke dalam dua kategori: Anda bekerja sambil mengadakan perjalanan atau mencoba mencoba atau membatasi pengeluaran Anda. Artikel ini berfokus pada hal yang terakhir.*
In this context, the word "sambil" means "while"; the word "membatasi" means "limiting", "restrict", "limit".
The full translation to English is: *Basically, they fall into two categories: Either work while you travel or try and limit your expenses. This article is focused on the latter.*

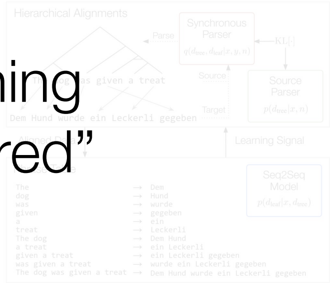
Translate the following sentence to English: *Ia melakukan pembuatan bel pintu dengan teknologi WiFi, katanya.*
In this context, the word "pembuatan" means "creation"; the word "bel" means "buzzer", "bell"; the word "pintu" means "door", "doors".
The full translation to English is:

“Dictionary Prompting” [Ghazvininejad et al. ’23]

Symbolic Structures for Controlling & Augmenting LLMs

Hierarchical Phrase-based Sequence-to-Sequence Learning

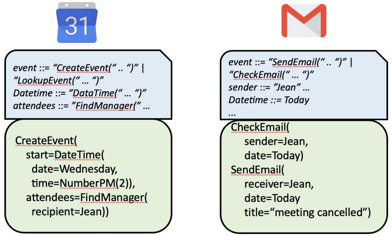
Bailin Wang, Ivan Titov, Jacob Andreas, Yoon Kim
EMNLP '22



TL;DR. Use grammars within in-context learning to improve generation of strings in a "structured" language (DSL).

Grammar Prompting for DSL generation with Large Language Models

Bailin Wang, Zi Wang, Rif A. Saurous, Xuezhi Wang, Yuan Cao, Yoon Kim
NeurIPS '23

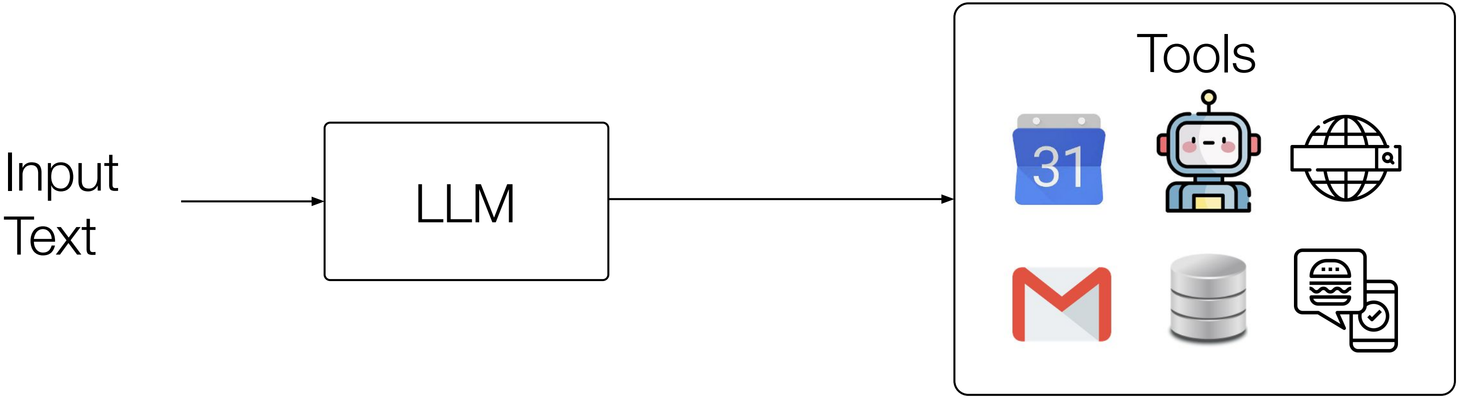


LLMs & The External World

Input Text → LLM → Output Text

$$P_\theta(\boldsymbol{x}_t \mid \boldsymbol{x}_{<t})$$

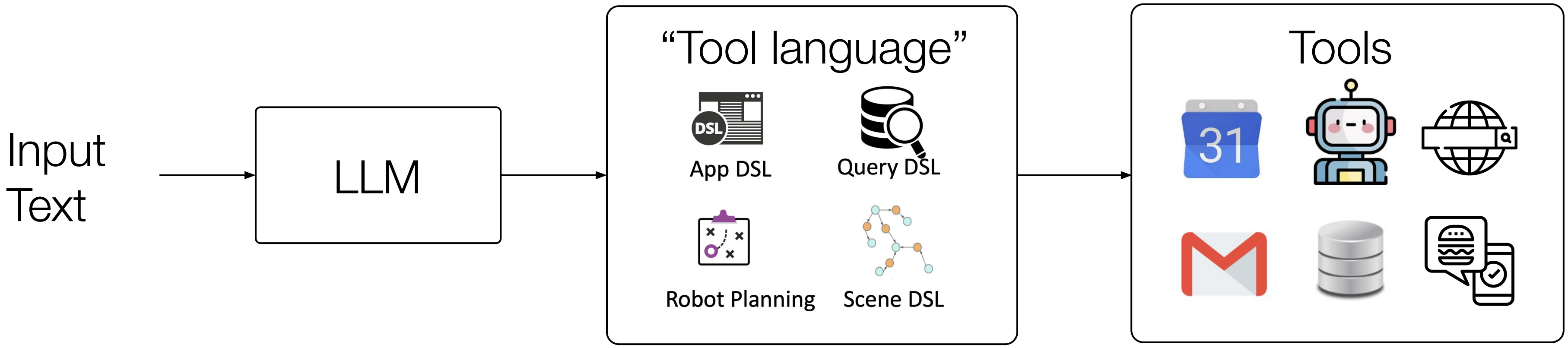
LLMs & The External World



How can we get LLMs to

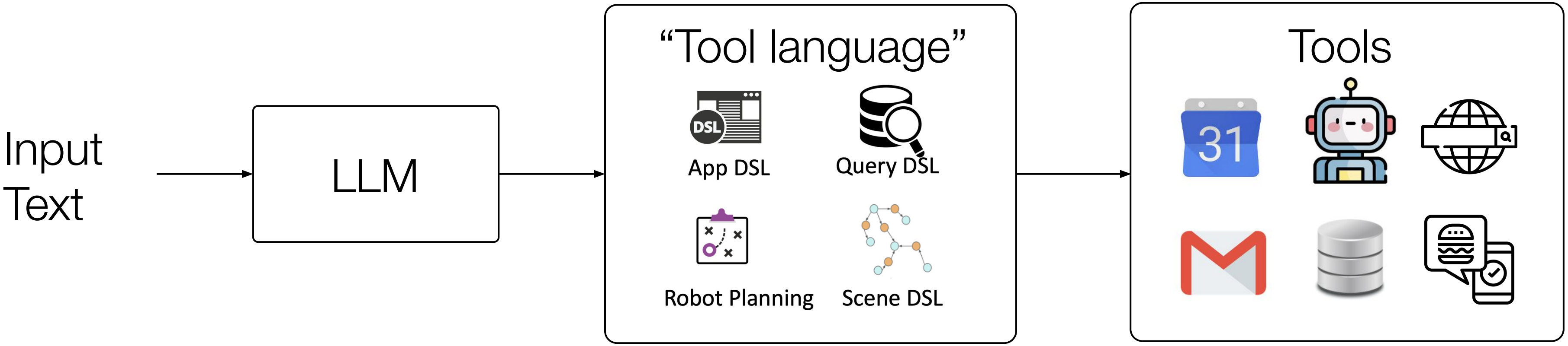
book appointments?
send emails?
search?
affect the external world?

LLMs & Tools



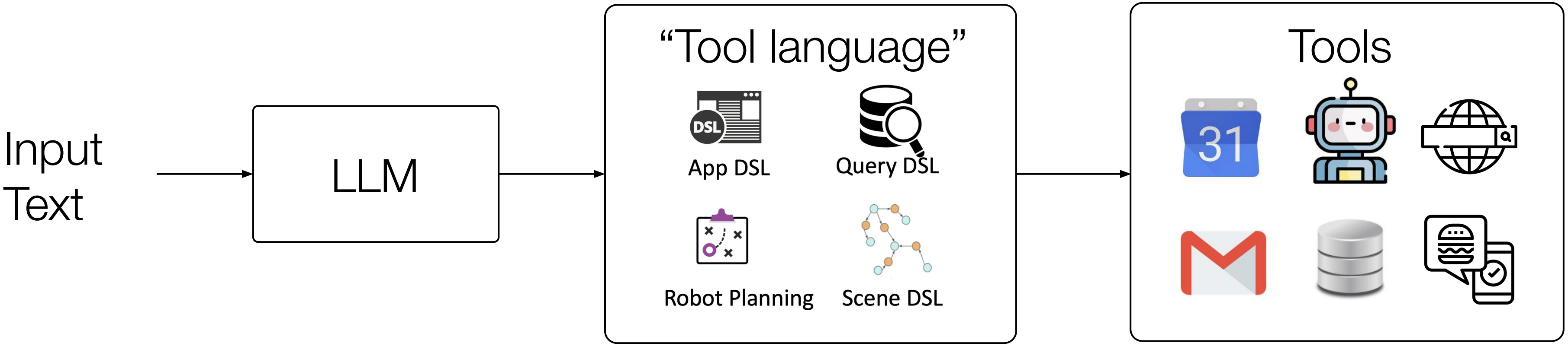
… through domain-specific tool languages (DSL)!

LLMs & Tools



Add meeting with
Jean’s manager on
Wednesday at 2 PM

LLMs & Tools

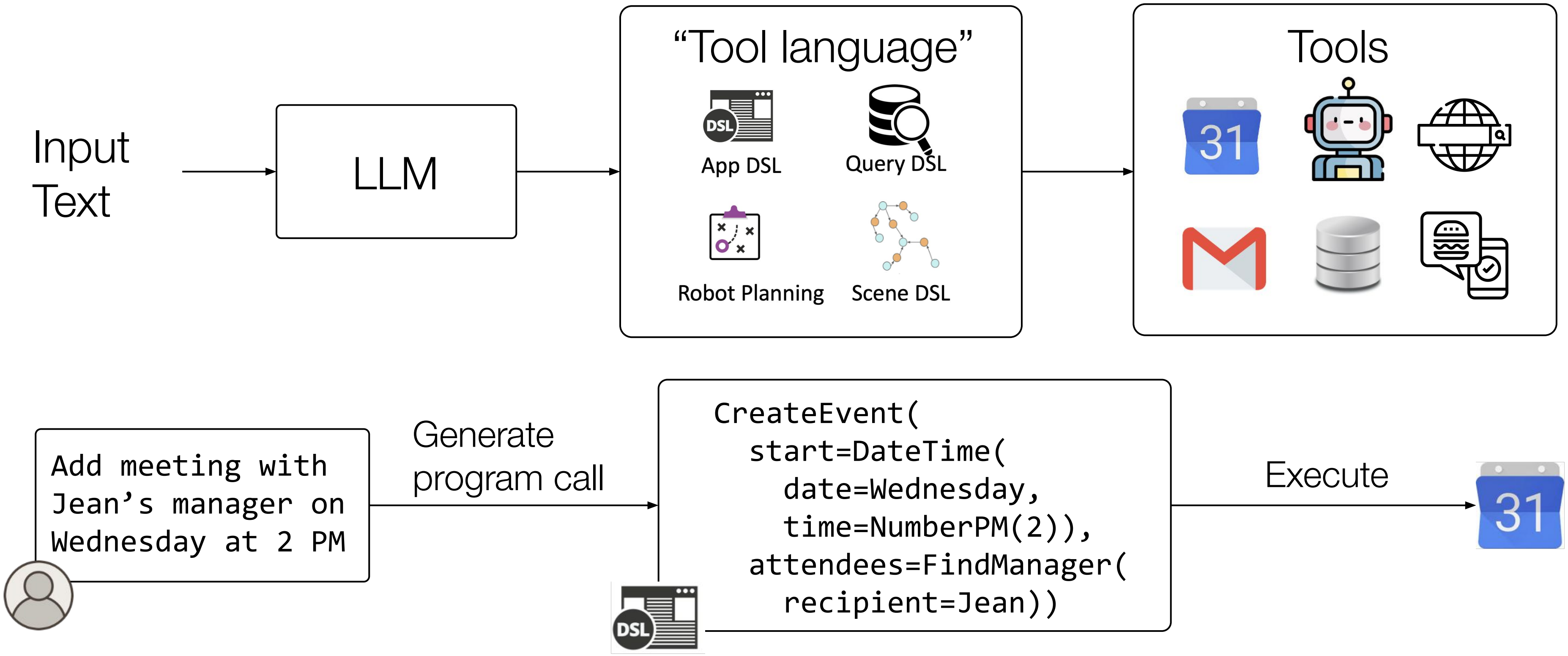



```
Add meeting with
Jean’s manager on
Wednesday at 2 PM
```

Generate
program call

```
CreateEvent(
  start=DateTime(
    date=Wednesday,
    time=NumberPM(2)),
  attendees=FindManager(
    recipient=Jean))
```

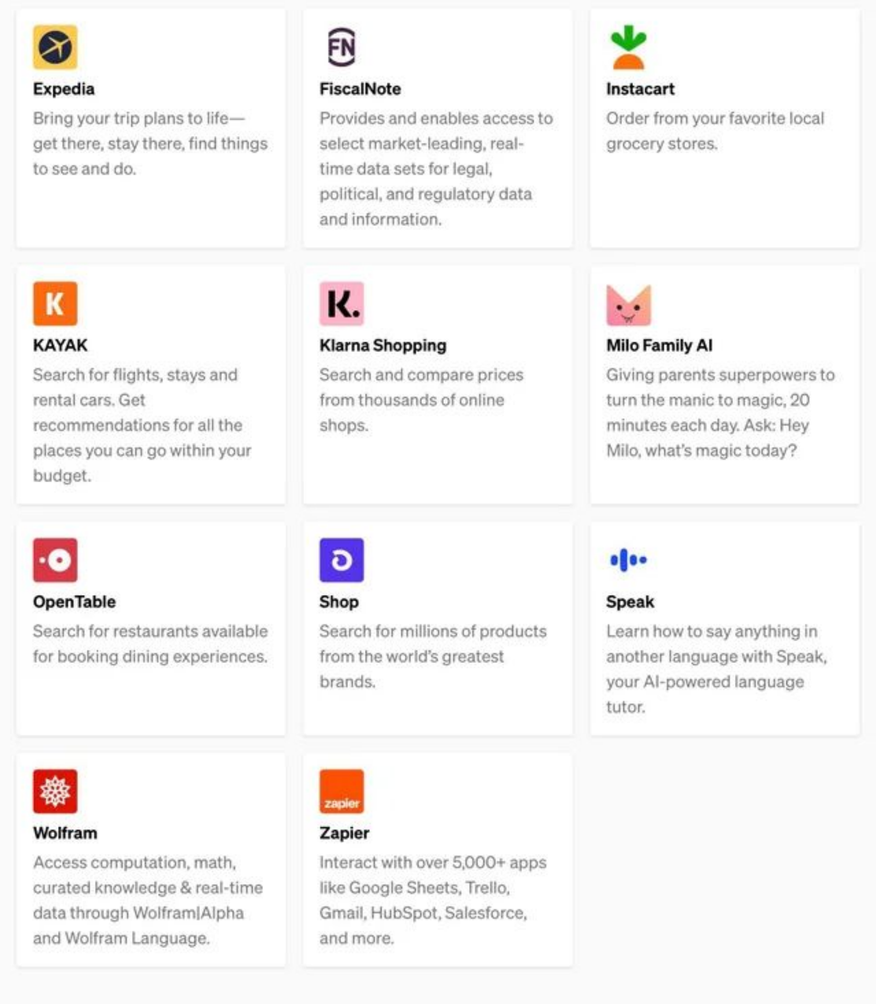
LLMs & Tools


Input
Text
LLM
“Tool language”
App DSL
Query DSL
Robot Planning
Scene DSL
Tools
31
Add meeting with
Jean’s manager on
Wednesday at 2 PM
Generate
program call
CreateEvent(
start=DateTime(
date=Wednesday,
time=NumberPM(2)),
attendees=FindManager(
recipient=Jean))
DSL
Execute
31

LLMs & Tools

ChatGPT plugins

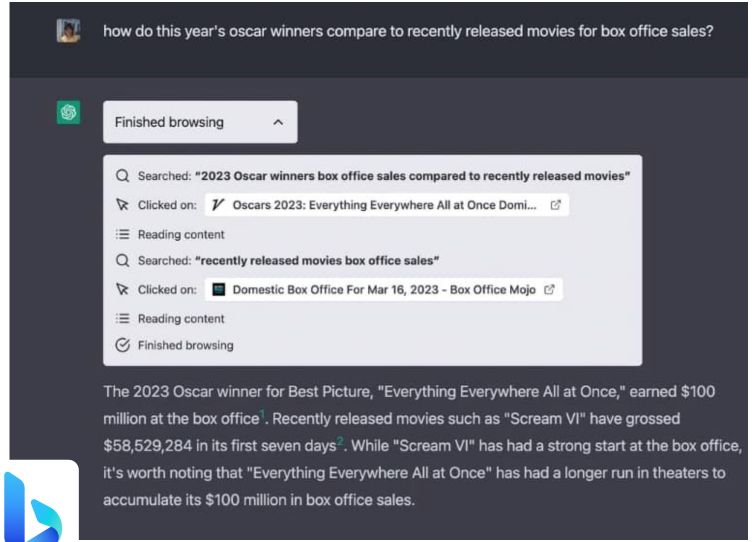
We've implemented initial support for plugins in ChatGPT. Plugins are tools designed specifically for language models with safety as a core principle, and help ChatGPT access up-to-date information, run computations, or use third-party services.



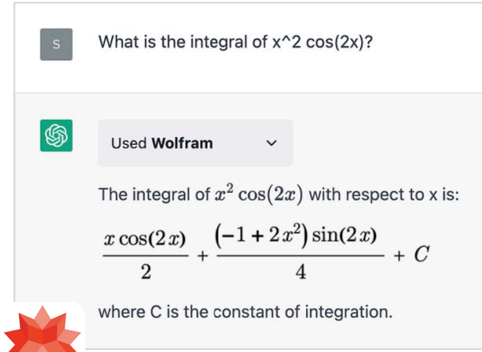
[OpenAI]

LLMs & Tools

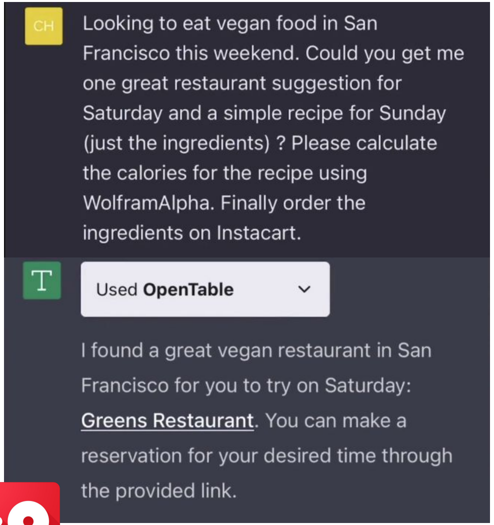
```
find_restaurant(
  type=vegan
  location=San Francisco,
  time=Saturday)
```



Search



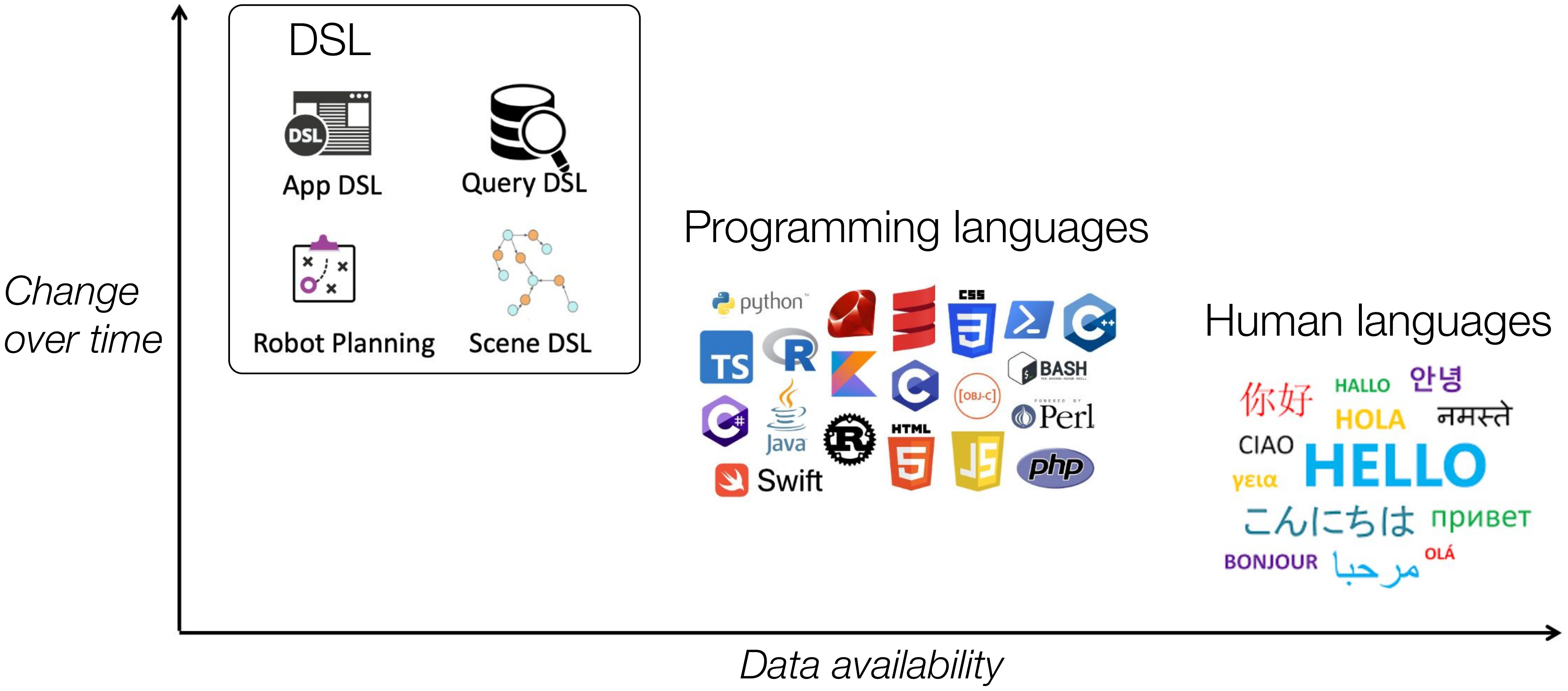
Math



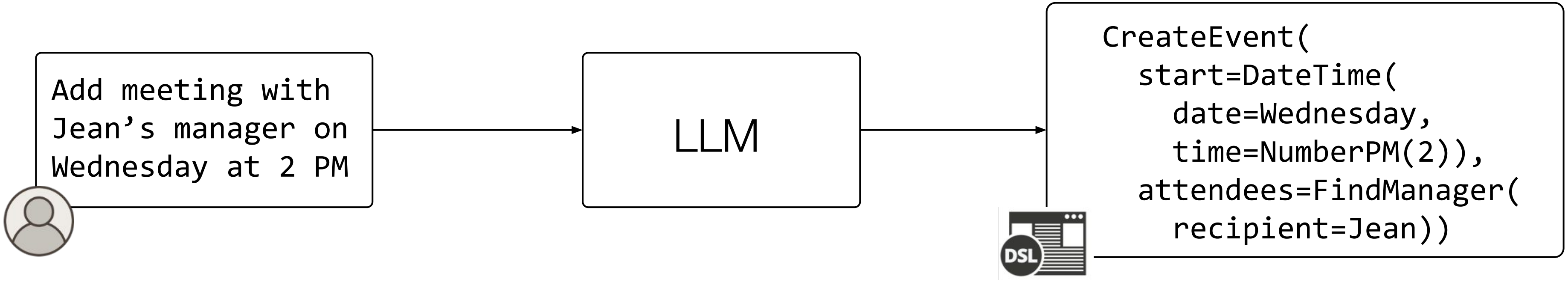
Restaurant reservation

[OpenAI]

LLMs & DSL: Challenges



LLMs for Generating DSL (Semantic Parsing)



LLMs for Generating DSL (Semantic Parsing)

"In-context" demonstrations (from training)

Input 1

```
What time on Tuesday is my planning
meeting?
```

Output 1

```
start(findEvent(EventSpec( name='planning',
  start=DateTimeSpec(weekday=tuesday))))
```

Input 2

```
Can you remind me to go to the
airport tomorrow morning at 8am?
```

Output 2

```
createCommitEventWrapper(
  createPreflightEventWrapper(
  EventBuilder(subject='go to the airport',
  start=dateAtTime( date=tomorrow(),
  time=numberAM(8)))))
```

⋮

User input

```
Add meeting with Jean's manager on
Wednesday at 2 PM
```

LLMs for Generating DSL (Semantic Parsing)

"In-context" demonstrations (from training)

```
What time on Tuesday is my planning
meeting?
```

```
start(findEvent(EventSpec( name='planning',
  start=DateTimeSpec(weekday=tuesday))))
```

```
Can you remind me to go to the
airport tomorrow morning at 8am?
```

```
createCommitEventWrapper(
  createPreflightEventWrapper(
  EventBuilder(subject='go to the airport',
  start=dateAtTime( date=tomorrow(),
  time=numberAM(8)))))
```

⋮

User input

```
Add meeting with Jean's manager on
Wednesday at 2 PM
```

LLM

```
CreateEvent(start=DateTime(
  date=Wednesday,
  time=NumberPM(2)),
  attendees=FindManager(
  recipient=Jean))
```

LLMs for Generating DSL (Semantic Parsing)

“In-context” demonstrations (from training)

What time on Tuesday is my planning meeting?

```
start(findEvent(EventSpec( name='planning',
  start=DateTimeSpec(weekday=tuesday))))
```

Can you remind me to go to the airport tomorrow morning at 8am?

```
createCommitEventWrapper(
  createPreflightEventWrapper(
  EventBuilder(subject='go to the airport',
  start=dateAtTime( date=tomorrow(),
  time=numberAM(8)))))
```

⋮

User input

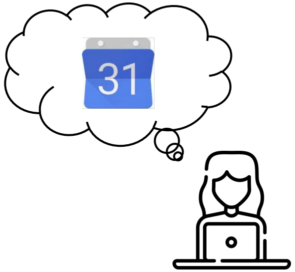
Add meeting with Jean’s manager on Wednesday at 2 PM

LLM

```
CreateEvent(start=DateTime(
  date=Wednesday,
  time=NumberPM(2)),
  attendees=FindManager(
  recipient=Jean))
```

Where do these outputs come from?

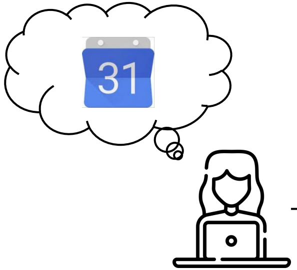
DSL for a Scheduling Assistant



Domain expert

DSL for a Scheduling Assistant

```
call ::= event | "(Yield"  org ")"  | "(Yield (size" org "))" | "(Yield"  event ")"  | "(Yield"  weather ")" |
        "(Yield (> (size"  event ")" number "))" | "(do" datetime call ")" | "(do" call  call ")" | "(do" org  call ")"

event ::= "(CreateCommitEventWrapper (CreatePreflightEventWrapper" event_constraint "))" |
          "(FindEventWrapperWithDefaults" event_constraint ")" |
          "(QueryEventResponse.results (FindEventWrapperWithDefaults" event_constraint "))" | "(singleton" event ")" |
          "(FindNumNextEvent" event_constraint number ")" | "(FindLastEvent" event_constraint ")" |
          "(Execute (refer (^(Dynamic) ActionIntensionConstraint)))" |
          "(Execute (refer (extensionConstraint (^(Event) EmptyStructConstraint))))" |
          "(^(Event) EmptyStructConstraint)"

event_constraint ::= "(&" event_constraint event_constraint ")" | "(Event.subject_? (?=" string "))" | "(Event.subject_? (?~=" string "))" |
                     "(Event.start_?" datetime_constraint  ")" | "(Event.end_?" datetime_constraint  ")" |
                     "(Event.location_?" location_constraint ")" | "(Event.duration_?"  duration_constraint ")" |
                     "(Event.showAs_?" status_constraint ")" | "(Event.attendees_?" attendee_constraint ")" |
                     "(EventAllDayForDateRange" event_constraint date_range ")" | "(EventAllDayOnDate" event_constraint date ")"

                                    ⋮
```



DSL Grammar

Domain expert

DSL for a Scheduling Assistant

```
call ::= event | "(Yield"  org ")"  | "(Yield (size" org "))" | "(Yield"  event ")"  | "(Yield"  weather ")" |
        "(Yield (> (size"  event ")" number "))" | "(do" datetime call ")" | "(do" call  call ")" | "(do" org  call ")"

event ::= "(CreateCommitEventWrapper (CreatePreflightEventWrapper" event_constraint "))" |
          "(FindEventWrapperWithDefaults" event_constraint ")" |
          "(QueryEventResponse.results (FindEventWrapperWithDefaults" event_constraint "))" | "(singleton" event ")" |
          "(FindNumNextEvent" event_constraint number ")" | "(FindLastEvent" event_constraint ")" |
          "(Execute (refer (^(Dynamic) ActionIntensionConstraint)))" |
          "(Execute (refer (extensionConstraint (^(Event) EmptyStructConstraint))))" |
          "(^(Event) EmptyStructConstraint)"

event_constraint ::= "(&" event_constraint event_constraint ")" | "(Event.subject_? (?=" string "))" | "(Event.subject_? (?~=" string "))" |
                     "(Event.start_?" datetime_constraint  ")" | "(Event.end_?" datetime_constraint  ")" |
                     "(Event.location_?" location_constraint ")" | "(Event.duration_?"  duration_constraint ")" |
                     "(Event.showAs_?" status_constraint ")" | "(Event.attendees_?" attendee_constraint ")" |
                     "(EventAllDayForDateRange" event_constraint date_range ")" | "(EventAllDayOnDate" event_constraint date ")"

                                              ⋮
```

```
Invite my boss and his team to
a party tomorrow
```

→

```
(Yield (CreateCommitEventWrapper (CreatePreflightEventWrapper (&
(& (Event.subject_? (?= "party")) (Event.start_?
(DateTime.date_? (?= (Tomorrow))))) (Event.attendees_? (&
(AttendeeListHasRecipient (FindManager (toRecipient
(CurrentUser))) (AttendeeListHasPeople (FindTeamOf (FindManager
(toRecipient (CurrentUser)))))))))))
```

DSL for a Scheduling Assistant

```
call ::= event | "(Yield"  org ")"  | "(Yield (size" org "))" | "(Yield"  event ")"  | "(Yield"  weather ")" |
        "(Yield (> (size"  event ")" number "))" | "(do" datetime call ")" | "(do" call  call ")" | "(do" org  call ")"

event ::= "(CreateCommitEventWrapper (CreatePreflightEventWrapper" event_constraint "))" |
          "(FindEventWrapperWithDefaults" event_constraint ")" |
          "(QueryEventResponse.results (FindEventWrapperWithDefaults" event_constraint "))" | "(singleton" event ")" |
          "(FindNumNextEvent" event_constraint number ")" | "(FindLastEvent" event_constraint ")" |
          "(Execute (refer (^(Dynamic) ActionIntensionConstraint)))" |
          "(Execute (refer (extensionConstraint (^(Event) EmptyStructConstraint))))" |
          "(^(Event) EmptyStructConstraint)"

event_constraint ::= "(&" event_constraint event_constraint ")" | "(Event.subject_? (?=" string "))" | "(Event.subject_? (?~=" string "))" |
                     "(Event.start_?" datetime_constraint  ")" | "(Event.end_?" datetime_constraint  ")" |
                     "(Event.location_?" location_constraint ")" | "(Event.duration_?"  duration_constraint ")" |
                     "(Event.showAs_?" status_constraint ")" | "(Event.attendees_?" attendee_constraint ")" |
                     "(EventAllDayForDateRange" event_constraint date_range ")" | "(EventAllDayOnDate" event_constraint date ")"

                                                ⋮
```

How can we enable LLMs to use the expert knowledge embedded within the symbolic structure/rules of the DSL?

LLMs & Grammars

Invite my boss and his team to a party tomorrow

LLM

```
(Yield (CreateCommitEventWrapper
(CreatePreflightEventWrapper (& (& (Event.subject_? (?=
"party")) (Event.start_? (DateTime.date_? (?=
(Tomorrow))))) (Event.attendees_? (&
(AttendeeListHasRecipient (FindManager (toRecipient
(CurrentUser)))) (AttendeeListHasPeople (FindTeamOf
(FindManager (toRecipient (CurrentUser))))))))))))
```

LLMs & Grammars

Full DSL grammar

```
call ::= event | "(Yield"  org ")"  | "(Yield (size" org "))" | "(Yield"  event ")"  | "(Yield"  weather ")" |
             "(Yield (> (size"  event ")" number "))" | "(do" datetime call ")" | "(do" call  call ")" | "(do" org  call ")"
event ::= "(CreateCommitEventWrapper (CreatePreflightEventWrapper" event_constraint "))" |
          "(FindEventWrapperWithDefaults" event_constraint ")" |
          "(QueryEventResponse.results (FindEventWrapperWithDefaults" event_constraint "))" | "(singleton" event ")" |
                "(FindNumNextEvent" event_constraint number ")" | "(FindLastEvent" event_constraint ")" |
          "(Execute (refer (^(Dynamic) ActionIntensionConstraint)))" |
          "(Execute (refer (extensionConstraint (^(Event) EmptyStructConstraint))))" |
                "(^(Event) EmptyStructConstraint)"
event_constraint ::= "(&" event_constraint event_constraint ")" | "(Event.subject_? (?=" string "))" | "(Event.subject_? (?~=" string "))" |
                                "(Event.start_?" datetime_constraint  ")" | "(Event.end_?" datetime_constraint  ")" |
                                "(Event.location_?" location_constraint ")" | "(Event.duration_?"  duration_constraint ")" |
                                "(Event.showAs_?" status_constraint ")" | "(Event.attendees_?" attendee_constraint ")" |
                                "(EventAllDayForDateRange" event_constraint date_range ")" | "(EventAllDayOnDate" event_constraint date ")"
                        ⋮
```

Minimal grammar for generating the output program

```
call ::= "(Yield" event ")"
event ::= "(CreateCommitEventWrapper (CreatePreflightEventWrapper"
  event_constraint "))"
event_constraint ::= "(&" event_constraint event_constraint ")" |
  "(Event.subject_? (?=" string "))" | "(Event.start_?" datetime_constraint
")" | "(Event.attendees_?"
   attendee_constraint ")"
string ::= "\"party\""
datetime_constraint ::= "(DateTime.date_?" datetime_constraint ")"
  | "(?" OP datetime ")"
OP ::= "=" datetime ::= date
                     ⋮
```

```
Invite my boss and his team to a
party tomorrow
```

LLM

```
(Yield (CreateCommitEventWrapper
(CreatePreflightEventWrapper (& (& (Event.subject_? (?=
"party")) (Event.start_? (DateTime.date_? (?=
(Tomorrow))))) (Event.attendees_? (&
(AttendeeListHasRecipient (FindManager (toRecipient
(CurrentUser)))) (AttendeeListHasPeople (FindTeamOf
(FindManager (toRecipient (CurrentUser))))))))))
```

LLMs & Grammars

Full DSL grammar

```
call ::= event | "(Yield"  org ")"  | "(Yield (size" org "))" | "(Yield"  event ")"  | "(Yield"  weather ")" |
              "(Yield (> (size"  event ")" number "))" | "(do" datetime call ")" | "(do" call  call ")" | "(do" org  call ")"
event ::= "(CreateCommitEventWrapper (CreatePreflightEventWrapper" event_constraint "))" |
          "(FindEventWrapperWithDefaults" event_constraint ")" |
          "(QueryEventResponse.results (FindEventWrapperWithDefaults" event_constraint "))" | "(singleton" event ")" |
                "(FindNumNextEvent" event_constraint number ")" | "(FindLastEvent" event_constraint ")" |
          "(Execute (refer (^(Dynamic) ActionIntensionConstraint)))" |
          "(Execute (refer (extensionConstraint (^(Event) EmptyStructConstraint))))" |
                "(^(Event) EmptyStructConstraint)"
event_constraint ::= "(&" event_constraint event_constraint ")" | "(Event.subject_? (?=" string "))" | "(Event.subject_? (?~=" string "))" |
                                "(Event.start_?" datetime_constraint  ")" | "(Event.end_?" datetime_constraint  ")" |
                                "(Event.location_?" location_constraint ")" | "(Event.duration_?"  duration_constraint ")" |
                                "(Event.showAs_?" status_constraint ")" | "(Event.attendees_?" attendee_constraint ")" |
                                "(EventAllDayForDateRange" event_constraint date_range ")" | "(EventAllDayOnDate" event_constraint date ")"
                                                        ⋮
```

```
(Yield (CreateCommitEventWrapper
(CreatePreflightEventWrapper (& (& (Event.subject_? (?=
"party")) (Event.start_? (DateTime.date_? (?=
(Tomorrow))))) (Event.attendees_? (&
(AttendeeListHasRecipient (FindManager (toRecipient
(CurrentUser)))) (AttendeeListHasPeople (FindTeamOf
(FindManager (toRecipient (CurrentUser))))))))))
```

Parsing!

```
call ::= "(Yield" event ")"
event ::= "(CreateCommitEventWrapper (CreatePreflightEventWrapper"
  event_constraint "))"
event_constraint ::= "(&" event_constraint event_constraint ")" |
  "(Event.subject_? (?=" string "))" | "(Event.start_?" datetime_constraint
")" | "(Event.attendees_?"
   attendee_constraint ")"
string ::= "\"party\""
datetime_constraint ::= "(DateTime.date_?" datetime_constraint ")"
  | "(?" OP datetime ")"
OP ::= "=" datetime ::= date
                                ⋮
```

Training output program

Minimal grammar for generating output program

Standard Prompting with LLMs

"In-context" demonstrations (from training)

```
What time on Tuesday is my planning
meeting?
```

```
start(findEvent(EventSpec( name='planning',
  start=DateTimeSpec(weekday=tuesday))))
```

```
Can you remind me to go to the
airport tomorrow morning at 8am?
```

```
createCommitEventWrapper(
  createPreflightEventWrapper(
  EventBuilder(subject='go to the airport',
  start=dateAtTime( date=tomorrow(),
  time=numberAM(8)))))
```

⋮

User input

```
Add meeting with Jean's manager on
Wednesday at 2 PM
```

LLM

Grammar Prompting for LLMs

"In-context" demonstrations (from training)

Input example

```
What time on Tuesday is my planning meeting?
```

Minimal DSL grammar

```
call ::= "(Yield" query ")"
query ::= "start(" & event & )" ...
```

Output example

```
start(findEvent(EventSpec( name='planning',
  start=DateTimeSpec(weekday=tuesday))))
```

```
Can you remind me to go to the airport
tomorrow morning at 8am?
```

```
call ::= "(Yield" event ")"
event ::= "createCommitEventWrapper("constraint")" ...
```

```
createCommitEventWrapper(
  createPreflightEventWrapper(
  EventBuilder(subject='go to the airport',
  start=dateAtTime( date=tomorrow(),
  time=numberAM(8)))))
```

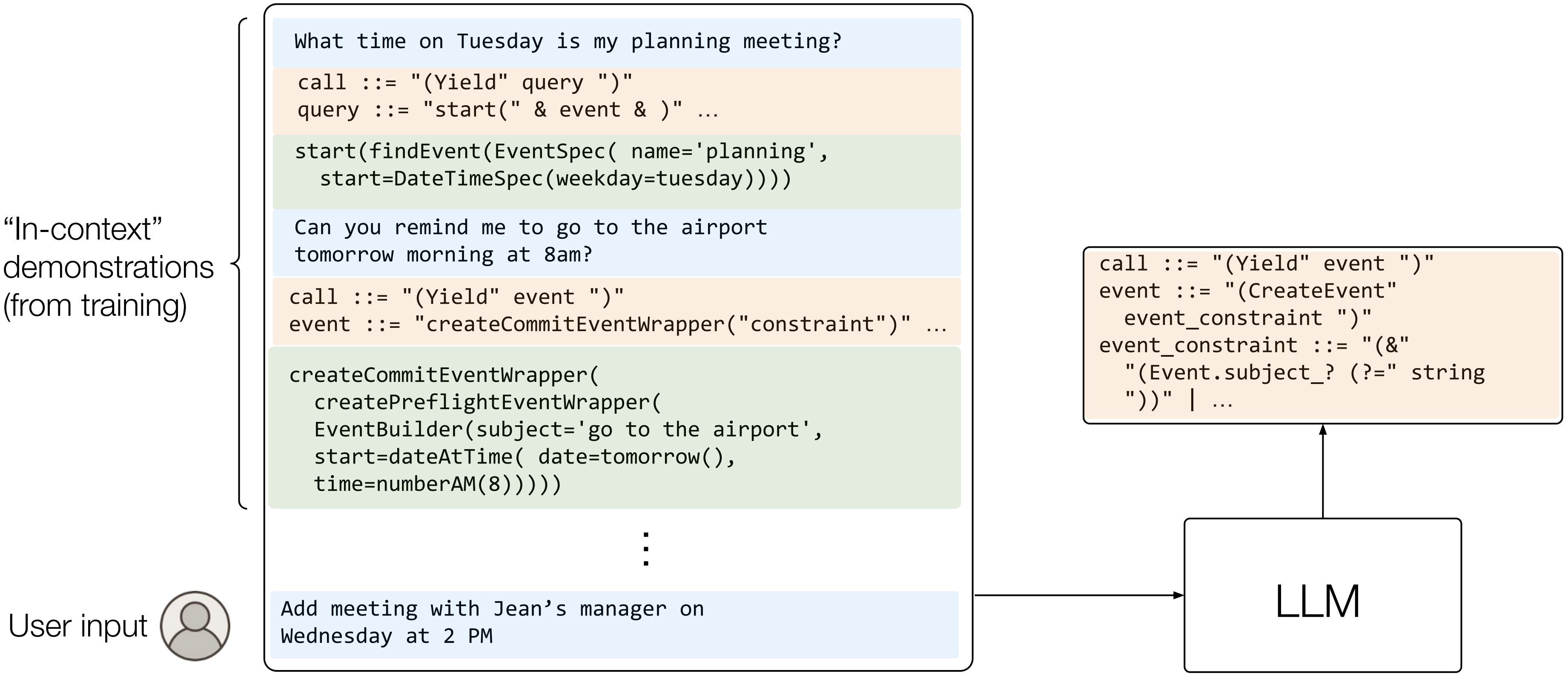
⋮

User input

```
Add meeting with Jean's manager on
Wednesday at 2 PM
```

LLM

Grammar Prompting for LLMs



Grammar Prompting for LLMs

```
call ::= event | "(Yield"  org ")"  | "(Yield (size" org "))" | "(Yield"  event ")"  | "(Yield"  weather ")" |
              "(Yield (> (size"  event ")" number "))" | "(do" datetime call ")" | "(do" call  call ")" | "(do" org  call ")"

event ::= "(CreateCommitEventWrapper (CreatePreflightEventWrapper" event_constraint "))" |
          "(FindEventWrapperWithDefaults" event_constraint ")" |
          "(QueryEventResponse.results (FindEventWrapperWithDefaults" event_constraint "))" | "(singleton" event ")" |
                "(FindNumNextEvent" event_constraint number ")" | "(FindLastEvent" event_constraint ")" |
          "(Execute (refer (^(Dynamic) ActionIntensionConstraint)))" |
          "(Execute (refer (extensionConstraint (^(Event) EmptyStructConstraint))))" |
                "(^(Event) EmptyStructConstraint)"

event_constraint ::= "(&" event_constraint event_constraint ")" | "(Event.subject_? (?=" string "))" | "(Event.subject_? (?~=" string "))" |
                               "(Event.start_?" datetime_constraint  ")" | "(Event.end_?" datetime_constraint  ")" |
                               "(Event.location_?" location_constraint ")" | "(Event.duration_?"  duration_constraint ")" |
                               "(Event.showAs_?" status_constraint ")" | "(Event.attendees_?" attendee_constraint ")" |
                               "(EventAllDayForDateRange" event_constraint date_range ")" | "(EventAllDayOnDate" event_constraint date ")"
                                                        ⋮
```

$\supseteq$

```
call ::= "(Yield" event ")"
event ::= "(CreateEvent"
  event_constraint ")"
event_constraint ::= "(&"
  "(Event.subject_? (?=" string
  "))" | …
```

LLM

User input

```
Add meeting with Jean's manager on
Wednesday at 2 PM
```

Grammar Prompting for LLMs

Constrain the minimal grammar output to be a subset of the full DSL grammar

$$P(x_t \mid \boldsymbol{x}_{<t}) \propto P_{\mathsf{LLM}}(x_t \mid \boldsymbol{x}_{<t}) \times \mathbb{1}\{x_t \boldsymbol{x}_{<t} \text{ is a valid rule in the DSL grammar}\}$$

```
call ::= event | "(Yield"  org ")"  | "(Yield (size" org "))" | "(Yield"  event ")"  | "(Yield"  weather ")" |
             "(Yield (> (size"  event ")" number "))" | "(do" datetime call ")" | "(do" call  call ")" | "(do" org  call ")"

event ::= "(CreateCommitEventWrapper (CreatePreflightEventWrapper" event_constraint "))" |
          "(FindEventWrapperWithDefaults" event_constraint ")" |
          "(QueryEventResponse.results (FindEventWrapperWithDefaults" event_constraint "))" | "(singleton" event ")" |
               "(FindNumNextEvent" event_constraint number ")" | "(FindLastEvent" event_constraint ")" |
          "(Execute (refer (^(Dynamic) ActionIntensionConstraint)))" |
          "(Execute (refer (extensionConstraint (^(Event) EmptyStructConstraint))))" |
               "(^(Event) EmptyStructConstraint)"

event_constraint ::= "(&" event_constraint event_constraint ")" | "(Event.subject_? (?=" string "))" | "(Event.subject_? (?~=" string "))" |
                          "(Event.start_?" datetime_constraint  ")" | "(Event.end_?" datetime_constraint  ")" |
                          "(Event.location_?" location_constraint ")" | "(Event.duration_?"  duration_constraint ")" |
                          "(Event.showAs_?" status_constraint ")" | "(Event.attendees_?" attendee_constraint ")" |
                          "(EventAllDayForDateRange" event_constraint date_range ")" | "(EventAllDayOnDate" event_constraint date ")"
                                              ⋮
```

$\supseteq$

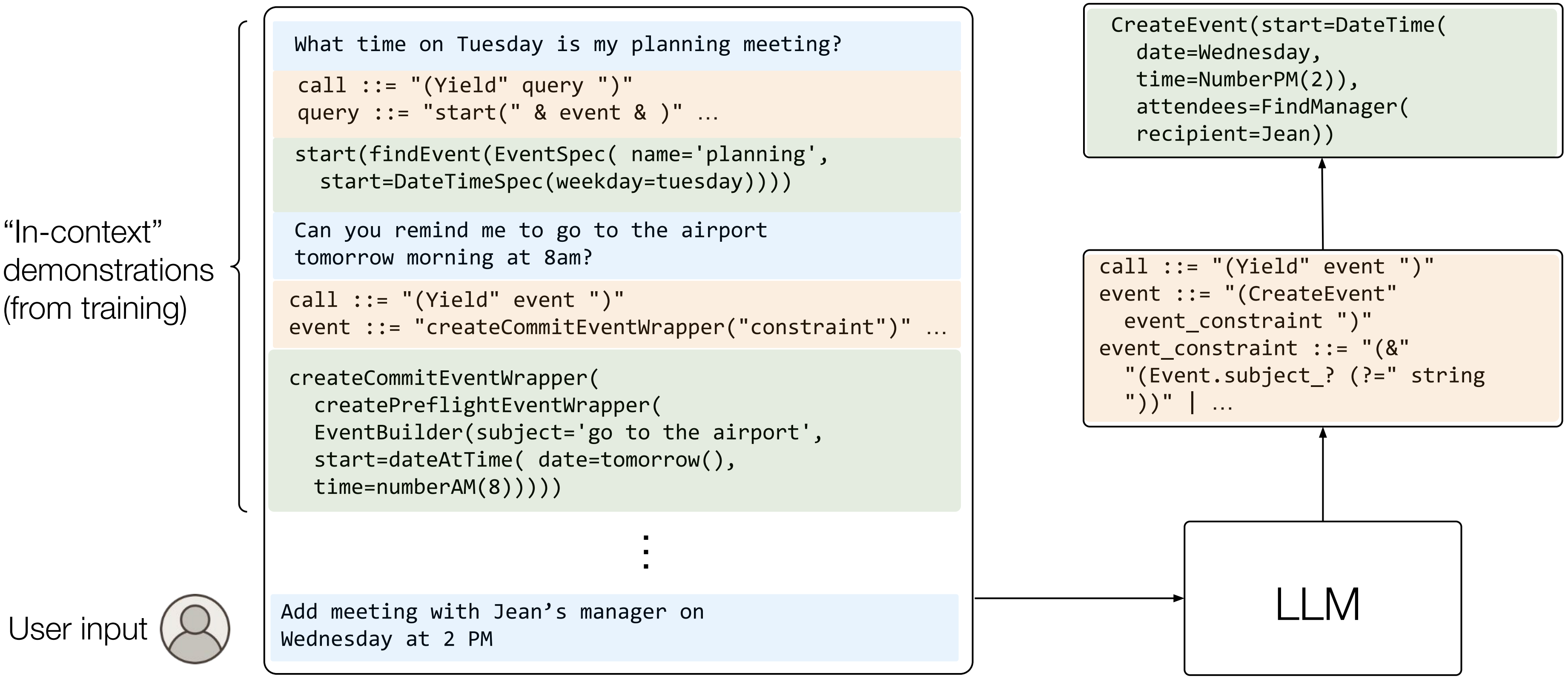
```
call ::= "(Yield" event ")"
event ::= "(CreateEvent"
  event_constraint ")"
event_constraint ::= "(&"
  "(Event.subject_? (?=" string
  "))" | …
```

User input

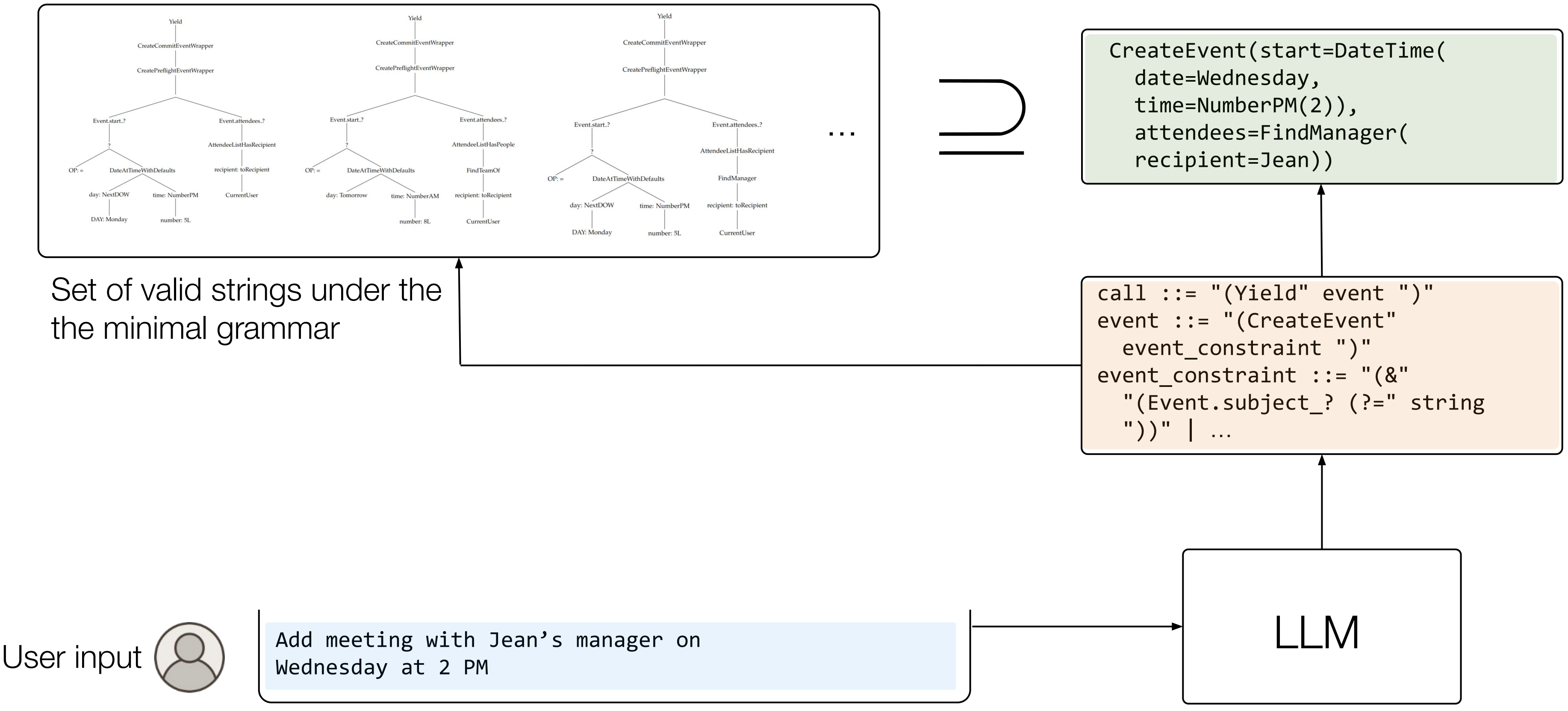
```
Add meeting with Jean's manager on
Wednesday at 2 PM
```

LLM

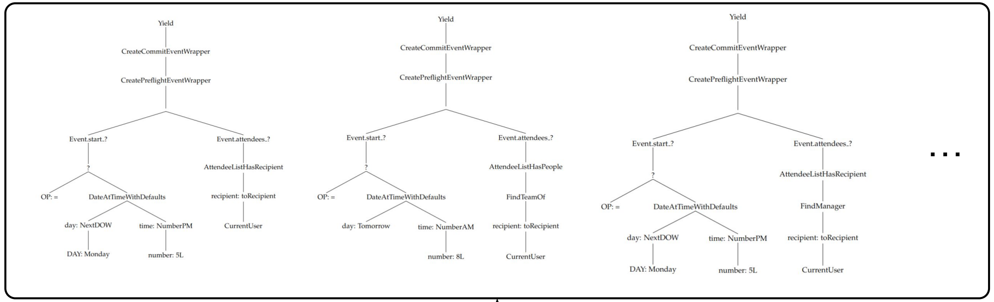
Grammar Prompting for LLMs



Grammar Prompting for LLMs



Grammar Prompting for LLMs



$\supseteq$

Set of valid strings under the the minimal grammar

```
CreateEvent(start=DateTime(
  date=Wednesday,
  time=NumberPM(2)),
  attendees=FindManager(
  recipient=Jean))
```

```
call ::= "(Yield" event ")"
event ::= "(CreateEvent"
  event_constraint ")"
event_constraint ::= "(&"
  "(Event.subject_? (?=" string
  "))" | …
```

$$P(x_t \mid \boldsymbol{x}_{<t}) \propto P_{\mathsf{LLM}}(x_t \mid \boldsymbol{x}_{<t}) \times \mathbb{1}\{x_t \boldsymbol{x}_{<t} \text{ is licensed by conditional DSL grammar}\}$$

Minimal grammar → Earley parser → Left-to-right parsing

```
Add meeting with Jean's manager on
Wednesday at 2 PM
```

LLM

Constrained Decoding

Actual algorithm:

1) Decode from LM
2) Check with Earley parser
3) Continue from earliest point of failure

Algorithm 1 Earley-based Constrained Generation

Input: Test input $\boldsymbol{x}$, predicted grammar $\widehat{G}$

Output: Program $\boldsymbol{y} \in L(\widehat{G})$

$t \leftarrow 0, \quad \hat{\boldsymbol{y}}^{(0)} \leftarrow \epsilon$ ▷ initialize to empty string

while True **do**

  $t \leftarrow t + 1$ ▷ unconstrained greedy decoding

  $\bar{\boldsymbol{y}} \leftarrow \text{decode}\left(P_{\text{LLM}}(\cdot \mid \boldsymbol{x}, \widehat{G}, \hat{\boldsymbol{y}}^{(t-1)}, \ldots)\right)$

  $\hat{\boldsymbol{y}}^{(t)} \leftarrow \hat{\boldsymbol{y}}^{(t-1)} \cdot \bar{\boldsymbol{y}}$ ▷ concatenation

  if $\hat{\boldsymbol{y}}^{(t)} \in L(\widehat{G})$ **then** ▷ try parsing with $\widehat{G}$

    return $\hat{\boldsymbol{y}}^{(t)}$ ▷ return if successful

  else ▷ if parsing fails, need to correct

    $\boldsymbol{y}_{\text{prefix}}, \Sigma[\boldsymbol{y}_{\text{prefix}}] \leftarrow \text{EarleyParse}(\hat{\boldsymbol{y}}^{(t)}, \widehat{G})$

    $\boldsymbol{w}^* \leftarrow \underset{\boldsymbol{w} \in \Sigma[\boldsymbol{y}_{\text{prefix}}]}{\arg\max} \; P_{\text{LLM}}(\boldsymbol{w} \mid \boldsymbol{y}_{\text{prefix}}, \ldots)$

    $\hat{\boldsymbol{y}}^{(t)} \leftarrow \boldsymbol{y}_{\text{prefix}} \cdot \boldsymbol{w}^*$

  end if

end while

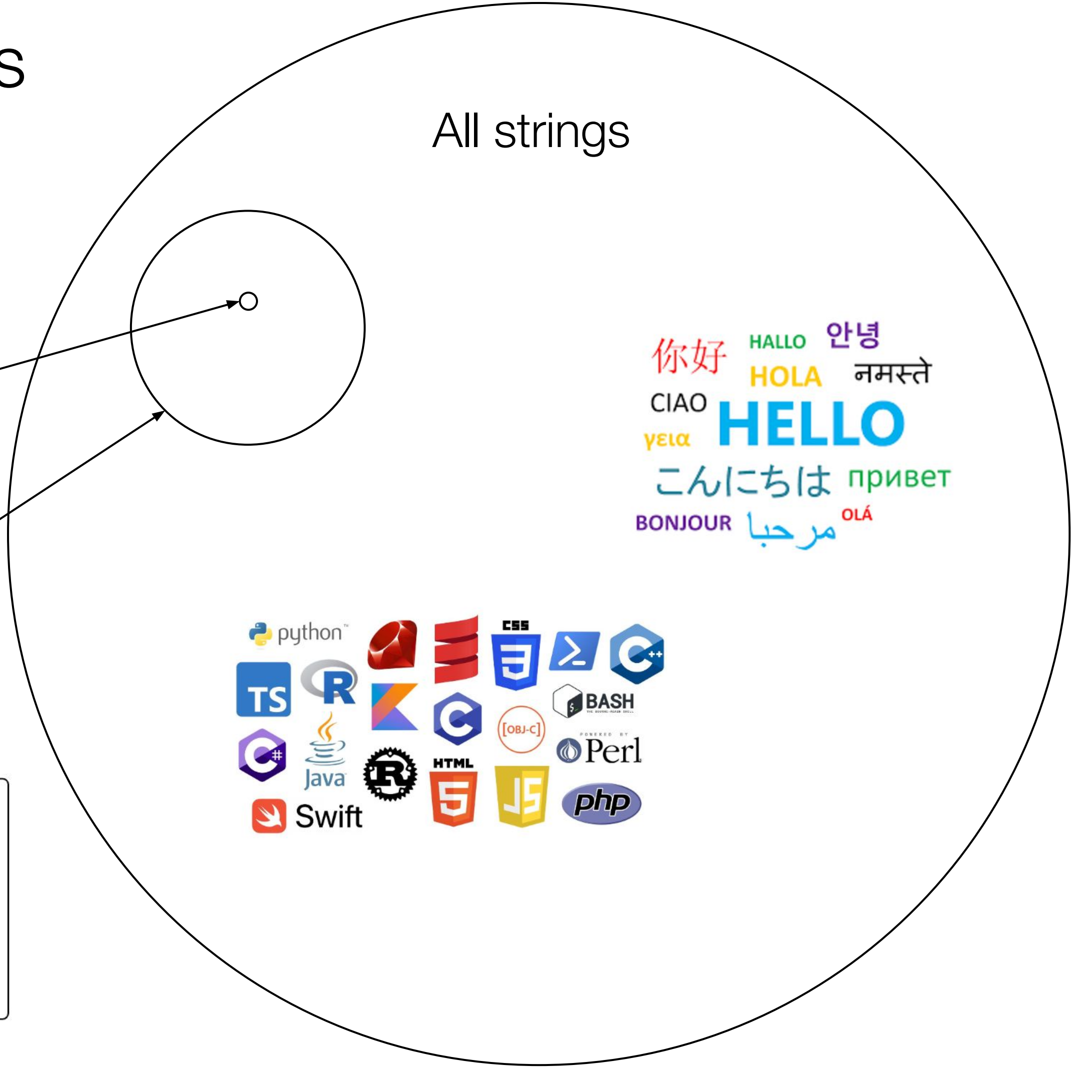
Constraining LLM Outputs

“Language” licensed by minimal DSL grammar

```
call ::= "(Yield" event ")"
event ::= "(CreateCommitEventWrapper (CreatePreflightEventWrapper"
  event_constraint "))"
event_constraint ::= "(&" event_constraint event_constraint ")" |
  "(Event.subject_? (?=" string "))" | "(Event.start_?" datetime_constraint
")" | "(Event.attendees_?"
   attendee_constraint ")"
string ::= "\"party\""
datetime_constraint ::= "(DateTime.date_?" datetime_constraint ")"
  | "(?" OP datetime ")"
OP ::= "=" datetime ::= date
                                ⋮
```

Language licensed by full DSL grammar

```
call ::= event | "(Yield"  org ")"  | "(Yield (size" org "))" | "(Yield"  event ")"  | "(Yield"  weather ")" |
             "(Yield (> (size"  event ")" number "))" | "(do" datetime call ")" | "(do" call  call ")" | "(do" org  call ")"

event ::= "(CreateCommitEventWrapper (CreatePreflightEventWrapper" event_constraint "))" |
          "(FindEventWrapperWithDefaults" event_constraint ")" |
          "(QueryEventResponse.results (FindEventWrapperWithDefaults" event_constraint "))" | "(singleton" event ")" |
              "(FindNumNextEvent" event_constraint number ")" | "(FindLastEvent" event_constraint ")" |
          "(Execute (refer (^(Dynamic) ActionIntensionConstraint)))" |
          "(Execute (refer (extensionConstraint (^(Event) EmptyStructConstraint))))" |
              "(^(Event) EmptyStructConstraint)"

event_constraint ::= "(&" event_constraint event_constraint ")" | "(Event.subject_? (?=" string "))" | "(Event.subject_? (?~=" string "))" |
                     "(Event.start_?" datetime_constraint  ")" | "(Event.end_?" datetime_constraint  ")" |
                     "(Event.location_?" location_constraint ")" | "(Event.duration_?"  duration_constraint ")" |
                     "(Event.showAs_?" status_constraint ")" | "(Event.attendees_?" attendee_constraint ")" |
                     "(EventAllDayForDateRange" event_constraint date_range ")" | "(EventAllDayOnDate" event_constraint date ")"
                                          ⋮
```



Why should this work?

LLMs may have been exposed to enough BNF grammar examples (and their derivations) during pretraining.

LLM as a metalanguage learner!

README.md

bnf

CI passing | coverage 97% | crates.io v0.5.0 | downloads 11k | license MIT

A library for parsing Backus–Naur form context-free grammars.

What does a parsable BNF grammar look like?

The following grammar from the Wikipedia page on Backus-Naur form exemplifies a compatible grammar. (*Note: parser allows for an optional ';' to indicate the end of a production)

```
 <postal-address> ::= <name-part> <street-address> <zip-part>

      <name-part> ::= <personal-part> <last-name> <opt-suffix-part> <EOL>
                    | <personal-part> <name-part>

  <personal-part> ::= <initial> "." | <first-name>

 <street-address> ::= <house-num> <street-name> <opt-apt-num> <EOL>

       <zip-part> ::= <town-name> "," <state-code> <ZIP-code> <EOL>

<opt-suffix-part> ::= "Sr." | "Jr." | <roman-numeral> | ""
    <opt-apt-num> ::= <apt-num> | ""
```

Results on Semantic Parsing

	SMCalflow	Overnight	GeoQuery
Regular Prompting	46.4%	54.7%	81.5%

GeoQuery

Which rivers run through the states bordering Texas?

```
answer(traverse(next_to(stateid('
texas'))))
```

Overnight

Which meeting has the earliest end time?

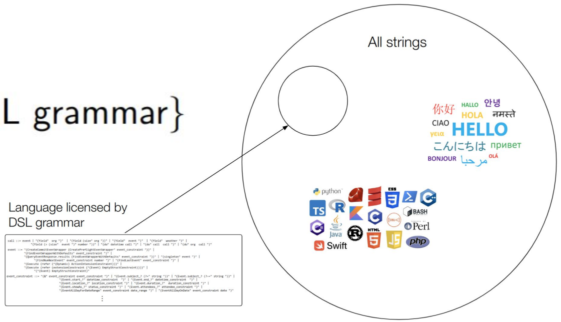
```
(call listValue (call superlative (call getProperty
(call singleton en.meeting) (string !type)) (string
min) (call ensureNumericProperty (string end_time))))
```

Results on Semantic Parsing

	SMCalflow	Overnight	GeoQuery
Regular Prompting	46.4%	54.7%	81.5%
Regular Prompting + Constrained Decoding	49.2%	54.7%	81.8%

$$P(x_t \mid \boldsymbol{x}_{<t}) \propto P_{\text{LLM}}(x_t \mid \boldsymbol{x}_{<t}) \times \mathbb{1}\{x_t \boldsymbol{x}_{<t} \text{ is licensed by unconditional DSL grammar}\}$$

(Guarantee syntactic validity)



Results on Semantic Parsing

	SMCalflow	Overnight	GeoQuery
Regular Prompting	46.4%	54.7%	81.5%
Regular Prompting + Constrained Decoding	49.2%	54.7%	81.8%
Linearized Tree Prompting	50.0%	56.4%	77.5%

Original output: `(attendee_? FindManager(Jean))`

Linearized tree output: `[constraint "(attendee_?" [attendee "FindManager(" [attendee "Jean" ")"] ")"]`

Results on Semantic Parsing

	SMCalflow	Overnight	GeoQuery
Regular Prompting	46.4%	54.7%	81.5%
Regular Prompting + Constrained Decoding	49.2%	54.7%	81.8%
Linearized Tree Prompting	50.0%	56.4%	77.5%
Grammar Prompting + Constrained Decoding	52.4%	60.9%	88.9%

Results on Semantic Parsing

	SMCalflow	Overnight	GeoQuery
Regular Prompting	46.4%	54.7%	81.5%
Regular Prompting + Constrained Decoding	49.2%	54.7%	81.8%
Linearized Tree Prompting	50.0%	56.4%	77.5%
Grammar Prompting + Constrained Decoding	52.4%	60.9%	88.9%
Grammar Prompting + Oracle Grammar	83.6%	96.5%	96.8%

More results (different LMs, retrieval-based prompting, etc.) in the paper.

Controlling LLM Outputs with New Rules

“Cancel the Wednesday 5pm meeting with the marketing team in New York and reschedule it to Thursday 2pm”

Domain expert

Controlling LLM Outputs with New Rules

```
call ::= event | "(Yield"  org ")"  | "(Yield (size" org "))" | "(Yield"  event ")"  | "(Yield"  weather ")" |
        "(Yield (> (size"  event ")" number "))" | "(do" datetime call ")" | "(do" call  call ")" | "(do" org  call ")"

event ::= "(CreateCommitEventWrapper (CreatePreflightEventWrapper" event_constraint "))" |
          "(FindEventWrapperWithDefaults" event_constraint ")" |
          "(QueryEventResponse.results (FindEventWrapperWithDefaults" event_constraint "))" | "(singleton" event ")" |
          "(FindNumNextEvent" event_constraint number ")" | "(FindLastEvent" event_constraint ")" |
          "(Execute (refer (^(Dynamic) ActionIntensionConstraint)))" |
          "(Execute (refer (extensionConstraint (^(Event) EmptyStructConstraint))))" |
          "(^(Event) EmptyStructConstraint)" | "(Execute (CancelEvent" event_constraint " event "))"

event_constraint ::= "(&" event_constraint event_constraint ")" | "(Event.subject_? (?=" string "))" | "(Event.subject_? (?~=" string "))" |
                     "(Event.start_?" datetime_constraint  ")" | "(Event.end_?" datetime_constraint  ")" |
                     "(Event.location_?" location_constraint ")" | "(Event.duration_?"  duration_constraint ")" |
                     "(Event.showAs_?" status_constraint ")" | "(Event.attendees_?" attendee_constraint ")" |
                     "(EventAllDayForDateRange" event_constraint date_range ")" | "(EventAllDayOnDate" event_constraint date ")"

                                        ⋮
```

“Cancel the Wednesday 5pm meeting with the marketing team in New York and reschedule it to Thursday 2pm”

Add function: `CancelEvent`
Add organization: `Marketing`
Add location: `New York`

Domain expert

Controlling LLM Outputs with New Rules

Full DSL with new rules

```
river ::= "shortest(" river ")" | "longest(" river ")"
city ::=  "smallest(" city ")" | "largest(" city ")" |
          "major(" state ")"
num ::= "population(" city ")" ...
```

Example 1

Input

```
what state has the largest city ?
```

Instance-specific DSL grammar

```
query ::= "answer(" answer_type ")"
answer_type ::= state
state ::= "state(" state ")" | "loc_1(" city ")"
city ::= "largest(" city ")" | ALL_CITY
ALL_CITY ::= "city(all)"
```

Output

```
answer(state(loc_1(largest(city(all)))))
```

Example 2

```
what is the capital of the state with the longest river ?
```

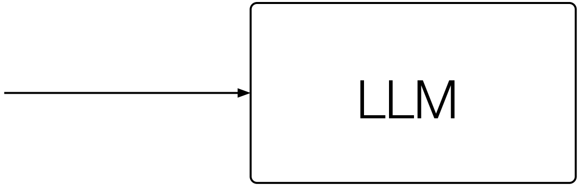
```
query ::= "answer(" answer_type ")"
answer_type ::= city
city ::= "capital(" city ")" | "loc_2(" state ")"
state ::= "state(" state ")" | "loc_1(" river ")"
river ::= "longest(" river ")" | ALL_RIVER
ALL_RIVER ::= "river(all)"
```

```
answer(capital(loc_2(state(loc_1(longest(river(all)))))))
```

⋮

User input

```
what is the smallest city in CA ?
```



Controlling LLM Outputs with New Rules

GeoQuery	New Rules	Compositional Gen.
LLM	63.3%	77.1%
LLM + Instance-level DSL grammar	90.8%	86.6%

New rules: holding out `"smallest", "shortest", "most", "highest", "sum", "population_1", "count", "major"`

Compositional generalization: Unseen combinations (e.g., test combines `"length"` and `"longest"` but training never uses them in combination).

Grammar Prompting for Molecule Generation

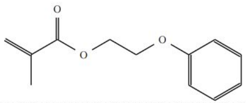
SMILES Grammar

```
smiles: atom (chain | branch)*

chain: (bond? (atom | ring_closure))+

branch: "(" bond? smiles+ ")"

atom: organic_symbol | aromatic_symbol | atom_spec
    | wildcard | group_symbol

bond: "-" | "=" | "#" | "$" | ":" | "/" | "\\" | "."

atom_spec: "[" isotope? ("se" | "as" | aromatic_symbol
| element_symbol | wildcard) chiral_class? h_count? …

organic_symbol: "Br" | "Cl" | "N" | "O" | "P" | "S"
| "F" | "I" | "B" | "C"

aromatic_symbol: "b" | "c" | "n" | "o" | "p" | "s"

group_symbol.2: "C=CC(=O)O" | "C=CC(O)=O" | "C(=C)C(=O)O"
| "C(=C)C(O)=O" |  "OC(=O)C=C" | "O=C(O)C=C" | …
                               ⋮
```

Specialized SMILES Grammar for Molecule Generation

$G[\boldsymbol{y}]$:

```
smiles          ::= atom chain branch chain chain | atom chain
atom            ::= organic_symbol
organic_symbol  ::= "C" | "N" | "O"
chain           ::= atom ring_closure bond atom | bond atom
                  | bond atom ring_closure | atom
                  | atom bond atom bond atom
                  | bond atom bond atom
ring_closure    ::= "1"
bond            ::= "="
branch          ::= "(" smiles ")"
```



$\boldsymbol{y}$: CC(= C)C(= O)OCCOC1 = CC = CC = C1

Grammar Prompting for Molecule Generation

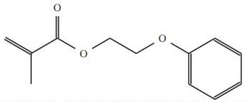
SMILES Grammar

```
smiles: atom (chain | branch)*

chain: (bond? (atom | ring_closure))+

branch: "(" bond? smiles+ ")"

atom: organic_symbol | aromatic_symbol | atom_spec
    | wildcard | group_symbol

bond: "-" | "=" | "#" | "$" | ":" | "/" | "\\" | "."

atom_spec: "[" isotope? ("se" | "as" | aromatic_symbol
| element_symbol | wildcard) chiral_class? h_count? …

organic_symbol: "Br" | "Cl" | "N" | "O" | "P" | "S"
| "F" | "I" | "B" | "C"

aromatic_symbol: "b" | "c" | "n" | "o" | "p" | "s"

group_symbol.2: "C=CC(=O)O" | "C=CC(O)=O" | "C(=C)C(=O)O"
| "C(=C)C(O)=O" |  "OC(=O)C=C" | "O=C(O)C=C" | …
⋮
```

Specialized SMILES Grammar for Molecule Generation

$G[\boldsymbol{y}]$:

```
smiles          ::= atom chain branch chain chain | atom chain
atom            ::= organic_symbol
organic_symbol  ::= "C" | "N" | "O"
chain           ::= atom ring_closure bond atom | bond atom
                  | bond atom ring_closure | atom
                  | atom bond atom bond atom
                  | bond atom bond atom
ring_closure    ::= "1"
bond            ::= "="
branch          ::= "(" smiles ")"
```

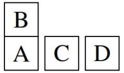


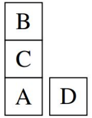
$\boldsymbol{y}$: CC(= C)C(= O)OCCOC1 = CC = CC = C1

Generating Chain Extenders	Validity	Diversity	Retrosynthesis Score
Graph Grammar [Guo et al. '22]	1.0	0.86	0.73
Standard Prompting	0.60	0.89	0.73
Grammar Prompting	0.96	0.90	0.87

Grammar Prompting for Planning

Specialized Action Grammar for PDDL Planning

$\boldsymbol{s}_0$: 

$\boldsymbol{s}_g$: 

$G[\boldsymbol{y}]$:

```
plan             ::=  ground_operator+
ground_operator  ::=  "pick_up-and-stack(" block block ")"
                   |  "unstack-and-put_down(" block block ")"
block            ::=  "A" | "B" | "C"
```

$\boldsymbol{y}$: `unstack-and-put_down(B, A); pick_up-and-stack(C, A); pick_up-and-stack(B, C)`

Use output from LM as input to a planning algorithm (greedy best-first search)

Grammar Prompting for Planning

Specialized Action Grammar for PDDL Planning

$G[y]$:

```
plan              ::=  ground_operator+
ground_operator   ::=  "pick_up-and-stack(" block block ")"
                   |   "unstack-and-put_down(" block block ")"
block             ::=  "A" | "B" | "C"
```

y: `unstack-and-put_down(B, A); pick_up-and-stack(C, A); pick_up-and-stack(B, C)`

Use output from LM as input to a planning algorithm (greedy best-first search)

"Easy" "Hard"

LLM + Planning	Success Rate			Success Rate		
No LLM-based Pruning	100%			40%		
Standard Prompting	100%			40%		
Grammar Prompting	100%			40%		

Grammar Prompting for Planning

Specialized Action Grammar for PDDL Planning

s_0: s_g:

$G[\boldsymbol{y}]$:

```
plan              ::=  ground_operator+
ground_operator   ::=  "pick_up-and-stack(" block block ")"
                   |   "unstack-and-put_down(" block block ")"
block             ::=  "A" | "B" | "C"
```

$\boldsymbol{y}$: `unstack-and-put_down(B, A); pick_up-and-stack(C, A); pick_up-and-stack(B, C)`

Use output from LM as input to a planning algorithm (greedy best-first search)

	"Easy"			"Hard"		
LLM + Planning	**Success Rate**	**Nodes Created**	**Nodes Expanded**	**Success Rate**	**Nodes Created**	**Nodes Expanded**
No LLM-based Pruning	100%	360	188	40%	18407	3870
Standard Prompting	100%	348	180	40%	17597	4039
Grammar Prompting	100%	251	124	40%	15033	3641

Summary

Including instance-specific grammars as intermediate variables improves DSL generation with LLMs.

What other type of structured representations (metalanguages) can LLMs make use of?

LLM Prompt

You are an expert programmer, and you need to write a program for the given natural language query. First, you should write a grammar that contains all the necessary BNF rules. Then, you should write programs that conform to your predicted rules.

(optional) G: [BEGIN RULES] ... [END RULES]

$\boldsymbol{x}^{(1)}$: find the meeting on Wednesday with Bob and Carol

$G[\boldsymbol{y}^{(1)}]$:

```
event       ::= "QueryEvent(" constraint ")"
constraint  ::= "(&" constraint constraint ")"
              | "(start_?" date ")"
              | "(attendee_?" attendee attendee ")"
date        ::= "Wednesday"
attendee    ::= "Bob" | "Carol"
```

$\boldsymbol{y}^{(1)}$: `QueryEvent(& (start_? Wednesday)(attendee_? Bob Carol))`

...

$\boldsymbol{x}$: Add meeting with Jean's manager on Wednesday at 3PM

LLM Output

$\widehat{G}$:

```
event       ::= "CreateEvent(" constraint ")"
constraint  ::= "(&" constraint constraint ")"
              | "(start_?" date time ")"
              | "(attendee_?" attendee ")"
date        ::= "Wednesday"
time        ::= "NumberPM(3)"
attendee    ::= "FindManager(" attendee ")" | "Jean"
```

$\widehat{\boldsymbol{y}}$: `CreateEvent(& (start_? Wednesday NumberPM(3))(attendee_? FindManager(Jean)))`

Maybe language modeling will be all you need eventually.
Meanwhile, symbolic structures can still help!

Thanks!