Learning Language Structures through Grounding

Freda Shi
Toyota Technological Institute at Chicago
freda@ttic.edu

September 21, 2023
@Peking University

Conversations between Freda and ChatGPT

Here is a sentence: **The cat near the children meows.** How many nouns, noun phrases and verbs are there in the given sentence? What are they?

There are three nouns in the sentence: "cat," "children," and "meows."
There is one noun phrase: "the cat near the children."
There is one verb: "meows."

Large language models do not understand syntactic structures well.

Large language models do not understand discourse structures, and can be easily distracted by irrelevant context [**Shi** et al., ICML 2023].

Can you show me a picture of a cat?

I'm sorry, as an AI language model, I don't have the ability to display images...

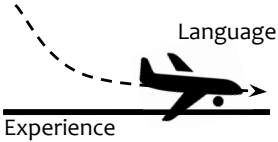
Large language models need

- A better understanding of language structures;
- A better connection with the real world.

Learning **Language Structures** through **Grounding**

What is Grounding?

Grounding: Landing the *plane of language* onto the *ground of experience*.



A cat is standing on the lawn.

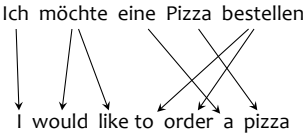
What is the factorial of 5?

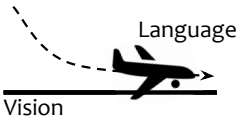
⇓ Semantic Parser

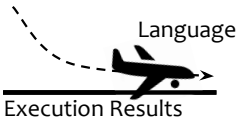
```
from utils import factorial
print(factorial(5))
```

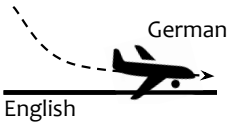
⇓ Python Interpreter

```
120
```





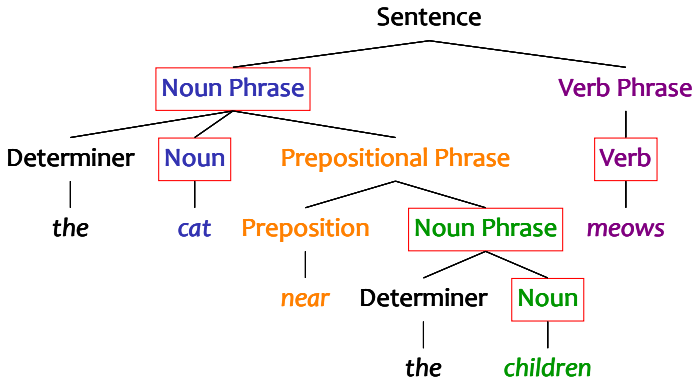




What are Language Structures?

The cat near the children meows.

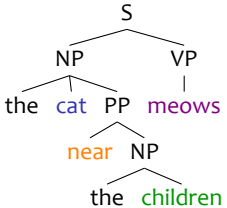
How many nouns, noun phrases and verbs are there in the sentence? What are they?



What are Language Structures?

The cat near the children meows.

Constituency Parse Tree



Dependency Parse Tree

Syntactic Structures

Truth-Conditional Semantics

$$\lambda x.\lambda y.cat(x) \wedge children(y) \wedge near(x, y) \wedge meow(x)$$

SQL

```
SELECT * FROM catsNearChildren
    WHERE meows = true;
```

Python

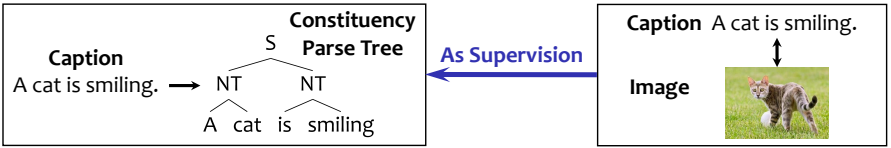
```
def find(cats):
  for cat in cats:
    if cat.near(children) and cat.meows:
      return cat
```

Semantic Structures

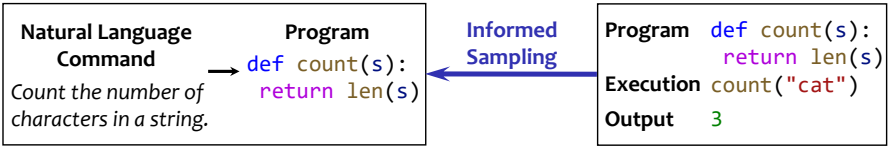
Learning Language Structures through Grounding

- Why do we care about learning language structures?
  Language structures can
  - Model human language processing;
  - Test or even inform linguistic theories;
  - Enable better interaction between humans and machines;
  - Improve machine learning models.
- Language structures are useful, but expensive to annotate.
  Many grounding signals exist naturally.
- Byproduct: Derived methods and analysis can benefit broader NLP community.

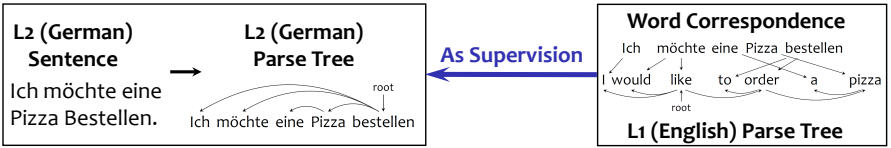
- **Learning to parse sentences through visual grounding**



- **Learning semantic parses through execution results**



- **Learning to parse another language through cross-lingual grounding**



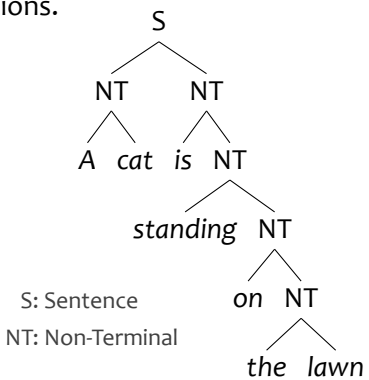
Part I: Learning to Parse Sentences through Visual Grounding

Question: Can visual grounding help induce linguistic structures?

Problem Formulation

Task: Visually grounded grammar induction.
Input: Captioned images.
Output: Linguistically plausible structure for captions.

A cat is standing on the lawn.

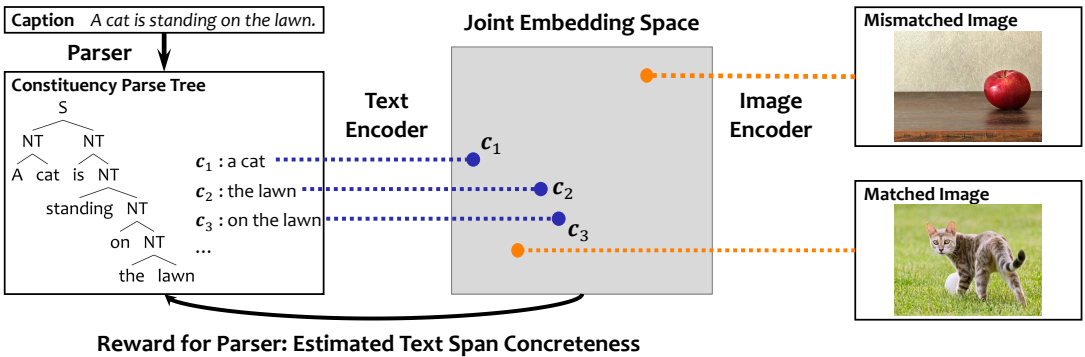


[**Shi***, Mao*, Gimpel, Livescu. Visually Grounded Neural Syntax Acquisition. ACL 2019]

The Visually Grounded Neural Syntax Learner (VG-NSL)

Hypothesis: More visually concrete word spans are more likely to be constituents.

Joint Embedding Space: Higher similarity for matched image-constituent pairs;
Lower similarity for mismatched pairs.



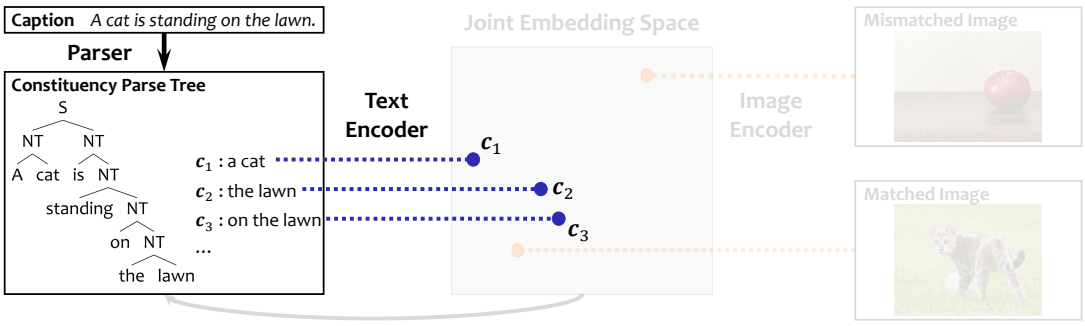
VG-NSL: Text Parser and Encoder

Parser Output: Predicted phrases and their vector representations.

$P_\Theta(\mathbf{v}_{\text{“a cat”}})$: Probability of *“a cat”* to be a constituent.

$\mathbf{V}$: Semantic representation of words and word spans.

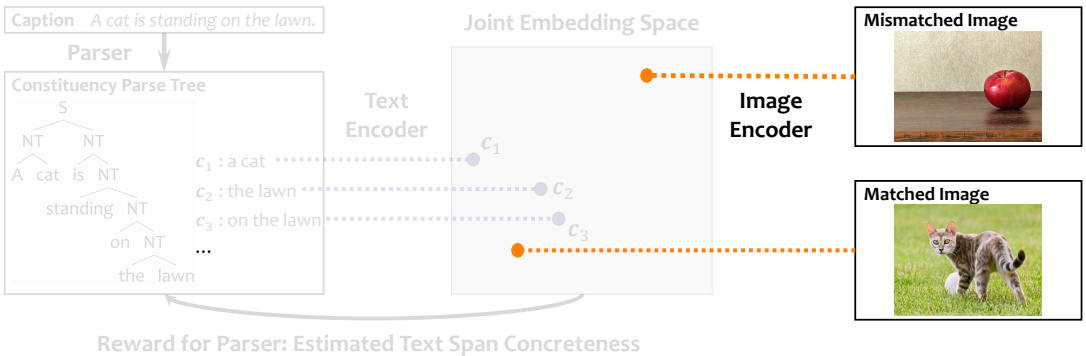
$\mathbf{\Theta}$: Structure of parse trees.



VG-NSL: Image Encoder

Image Encoder: Frozen ResNet (He et al., 2015) + Linear Projection.

$$\mathbf{u}_{img} = \mathbf{\Phi} \cdot ResNet(img)$$

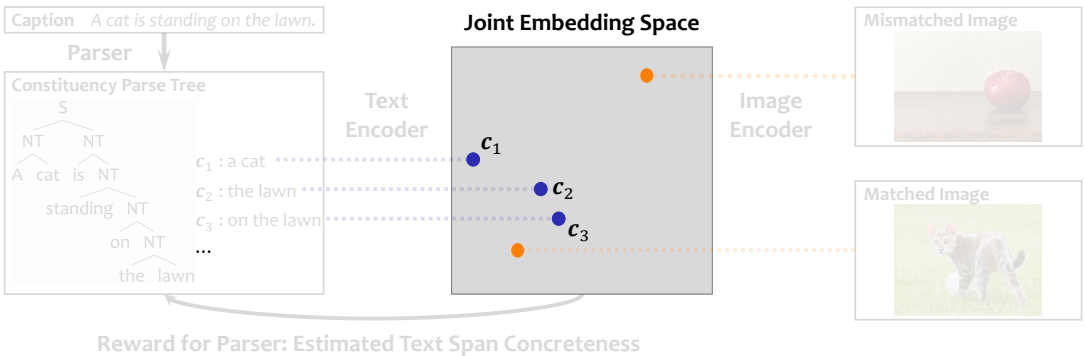


VG-NSL: Joint Visual-Semantic Embedding Space

Parameters for Text Encoder: Word representations $\mathbf{V}$, parser parameters $\mathbf{\Theta}$.

Parameters for Image Encoder: Linear projector $\mathbf{\Phi}$.

Joint Embedding Space: Train $\mathbf{V}$ and $\mathbf{\Phi}$ – align meanings of word spans and images.



VG-NSL: Joint Embedding Space

Key Idea: Higher similarity for matched image-constituent pairs,
Lower similarity for mismatched pairs.

Approach: Minimize hinge-based triplet loss (Kiros et al., 2015)
between images and constituents.

$$\mathcal{L}(i, c; \mathbf{V}, \mathbf{\Phi}) = \sum_{(i', c') \neq (i, c)} \left[\text{sim}(i', c) - \text{sim}(i, c) + \delta\right]_+ + \left[\text{sim}(i, c') - \text{sim}(i, c) + \delta\right]_+$$

a cat

a cat

Similarity ↓

a cat

Similarity ↑

an apple

Similarity ↓

a cat

Similarity ↑

$\text{sim}(\cdot, \cdot) = \cos(\cdot, \cdot)$ $[\cdot]_+ = \max(0, \cdot)$ δ : margin score

VG-NSL: Quantify Visual Concreteness

Joint Embedding Space: Higher similarity for matched image-constituent pairs; Lower similarity for mismatched pairs.

Image i

Candidate Constituent c

a cat on the

Another Image i'

$$\ell(c; i, i') = \mathrm{sim}(i', c) - \mathrm{sim}(i, c)$$

Value of ℓ

$\mathrm{sim}($  $, \textit{a cat}) = 0.2$ $\quad \mathrm{sim}($  $, \textit{a cat}) = 0.9$ $\quad \ell = -0.7$

$\mathrm{sim}($  $, \textit{on the}) = 0.4$ $\quad \mathrm{sim}($  $, \textit{on the}) = 0.4$ $\quad \ell = 0$

Key Idea: Smaller $\ell(c)$ $\leftrightsquigarrow$ c is more visually concrete.

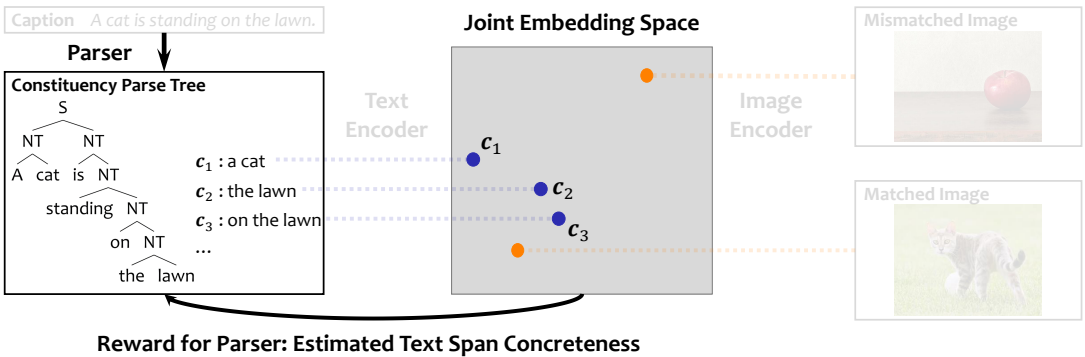
Quantify *visual concreteness* of word spans using loss values.

VG-NSL: Concreteness as Rewards for Text Parser

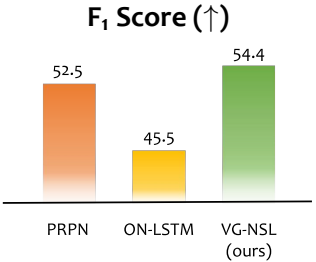
REINFORCE (Williams, 1992) as the gradient estimator for parsing parameter Θ:

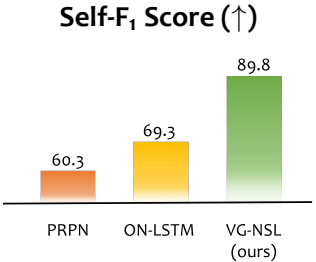
$$\Theta \leftarrow \Theta + \eta \cdot \nabla_\Theta \sum_{(i,c)} p_\Theta(c) \textit{concreteness}(c; i)$$

η : *learning rate*



VG-NSL: Results on the MSCOCO (Lin et al., 2014) Dataset

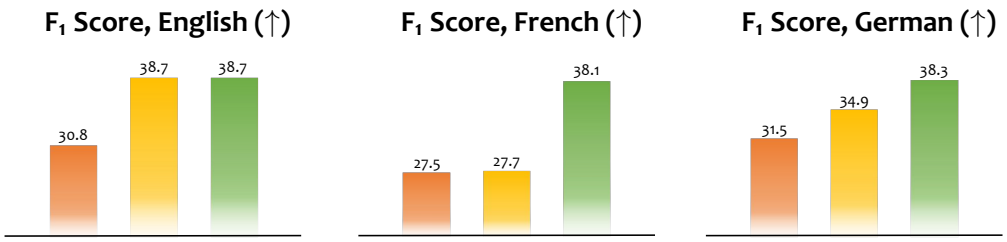




Text-Only Models : { PRPN: Shen et al., 2018 ; ON-LSTM: Shen et al., 2019

5 Runs: { Same hyperparameters ; Different random seeds

VG-NSL: Results on the Multi30K (Elliott et al., 2016) Dataset

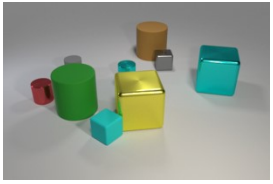


Text-Only Models : { PRPN: Shen et al., 2018 ; ON-LSTM: Shen et al., 2019

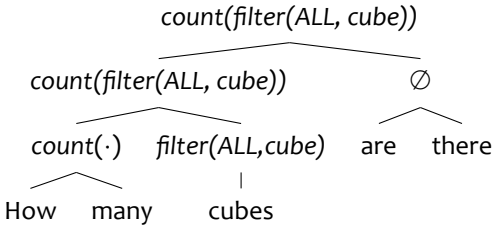
Question: Can visual grounding help induce linguistic structures?
Answer: Yes, on syntactic (constituency) parsing.

Joint Syntax and Semantics Induction through Visual Grounding

Dataset: CLEVR (Johnson et al., 2017).



Question: How many cubes are there?
Answer: 4.

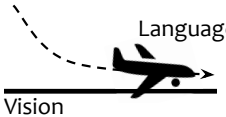


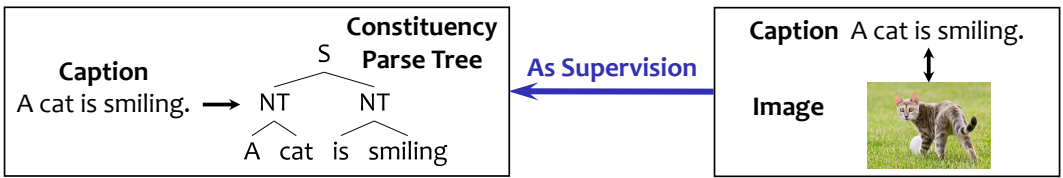
Question answering accuracy (↑) on program-depth generalization:

81.6 (prior SotA) → **98.5**

Question: Can visual grounding help induce linguistic structures?
Answer: Yes, on semantic parsing.

Part I: Learning Syntactic/Semantic Parses through Visual Grounding





Question: Can visual grounding help induce linguistic structures?
Approach: Propose the task of visually grounded grammar induction.
Answer: Yes, on syntactic (constituency) parsing [**S**MGL, ACL 2019].
Answer: Yes, on semantic parsing [M**S**WLT, NeurIPS 2021].

Part II: Learning Semantic Structures through Execution

Natural Language Command

Count the number of characters in a string.

Program

```
def count(s):
    return len(s)
```

Informed Sampling

Program

```
def count(s):
    return len(s)
```

Execution `count("cat")`

Output `3`

Question: Can execution results, as grounding signals, help learn semantic structures?
Answer: Yes, on semantic parsing without program supervision [M**S**WLT, NeurIPS 2021].

Problem Formulation

Task: Convert natural language to code, leveraging execution (grounding) of programs.
Input: Command in natural language.
Output: Corresponding program.

Example Input: *Write a Python function that counts lowercase letters in a string.*
Example Output:

```python
def count(string):
    cnt = 0
    for ch in string:
        if ch.islower():
            cnt += 1
    return cnt
```

```python
import collections
def count(s):
    cnt = collections.Counter(s)
    return sum(
        cnt[c] for c in cnt
        if c.islower()
    )
```

```python
def count(s):
    return len([
        c for c in s
        if c.islower()
    ])
```

Background: Codex

Transformer-based generative model for code (Chen et al., 2021).

- **Training**: Model probability of natural language and GitHub code snippets.

$$\max_{\Theta} \prod_{\mathbf{x}} P_\Theta(\mathbf{x})$$

$$P_\Theta(\mathbf{x}) = \prod_{i=1}^{|x|} P_\Theta(x_i \mid x_1, \ldots, x_{i-1})$$

Θ : Model parameters.

$\mathbf{x}$: Training example.

x_i : i^{th} token of $\mathbf{x}$.

- **Inference**: Generate code conditioned on natural language description.

$$P_\Theta(\mathbf{x} \mid x_1, \ldots, x_c) = \prod_{i=c+1}^{c+L} P_\Theta(x_i \mid x_1, \ldots, x_{i-1})$$

$x_1, \ldots, x_c$: Natural language description.

L :Maximum decoding step.

Natural Language to Code: Decoding Method

Task: Generate code conditioned on natural language description.
Example Input: *Write a Python function that counts lowercase letters in a string.*
Example Output:

$\mathbf{x}_1$

```
def count(s):
  return len(s)
```

$P_\Theta(\mathbf{x}_1 \mid \ldots) = 0.4$

$\mathbf{x}_2$

```
def count(string):
  cnt = 0
  for ch in string:
    cnt += ch.islower()
  return cnt
```

$P_\Theta(\mathbf{x}_2 \mid \ldots) = 0.3$

$\mathbf{x}_3$

```
def count(s):
  return len([
    c for c in s
    if c.islower()
  ])
```

$P_\Theta(\mathbf{x}_3 \mid \ldots) = 0.3$

$\mathbf{s}_i$: Execution results of $\mathbf{x}_i$.

$\mathbf{x}_2 \overset{\text{execution}}{\Longleftrightarrow} \mathbf{x}_3$

$P_\Theta(\mathbf{s}_1 \mid \ldots) = 0.4$ $\quad P_\Theta(\mathbf{s}_2 \mid \ldots) = P_\Theta(\mathbf{s}_3 \mid \ldots) = 0.3 + 0.3 = 0.6$

Key Idea: Consider program semantics (i.e., execution results)–based equivalent classes.

Empirical Solution

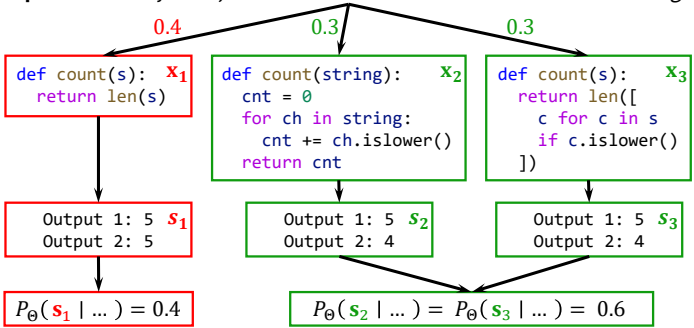
Hypothesis: Codex assigns higher probability to correct execution results.

Approach: Rank programs with $P_\Theta(\mathbf{s} \mid \ldots)$.

$$P_\Theta(\mathbf{s} \mid \ldots) = \sum_{\mathbf{x}} P_\Theta(\mathbf{x}, \mathbf{s} \mid \ldots) = \sum_{exec(\mathbf{x})=\mathbf{s}} P_\Theta(\mathbf{x} \mid \ldots)$$

x: Program.
s: Execution results.

Input: *Write a Python function that counts lowercase letters in a string.*



Step 1: Sample $\sim P_\Theta(\mathbf{x} \mid \ldots)$

Step 2: Synthesize input cases

Input 1: "hello"

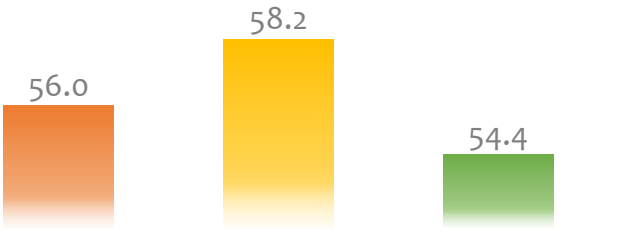
Input 2: "Hello"

Step 3: Execute

Step 4: Estimate $P_\Theta(\mathbf{s} \mid \ldots)$

Step 5: Select a program $\mathbf{x}^*$ with semantics $\mathbf{s}^* = \arg\max_{\mathbf{s}} P_\Theta(\mathbf{s})$ for test

Results: Natural Language to Python Translation (Austin et al., 2021)



$P(\mathbf{s})$: Probability of execution results.

$P(\mathbf{x})$: Probability of program.

Question: Can execution results, as grounding signals, help learn semantic structures?

Answer: Yes, execution result–based method improves natural language to Python translation.

Part II: Learning Semantic Structures through Execution

Natural Language Command

Count the number of characters in a string.

Program

```
def count(s):
    return len(s)
```

Informed Sampling

Program

```
def count(s):
    return len(s)
count("cat")
```

Execution

Output 3

Question: Can execution results, as grounding signals, help learn semantic structures?
Answer: Yes, on semantic parsing without program supervision [M**S**WLT, NeurIPS 2021].
Answer: Yes, on natural language to code translation [**S**FGZW, EMNLP 2022].

Part III: Towards Language-Universal NLP through Cross-Lingual Grounding

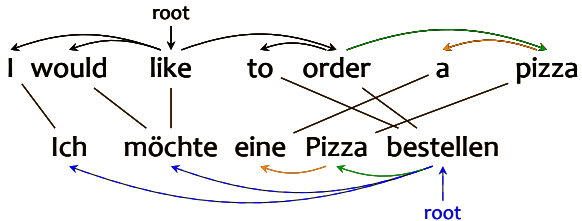
Question: Can we transfer NLP models to another language through cross-lingual grounding?

Problem Formulation

Task: Zero-shot cross-lingual dependency parsing.

Input: Sentences and dependency parse trees in source language;
Translated sentences in target language;
Word correspondence between parallel sentences.

Output: Dependency parse trees in target language.



⇢: Missing arcs with simple projection

Key Idea: Leverage the nature of trained source dependency parser that it can capture "*unannotated*" dependency relations.

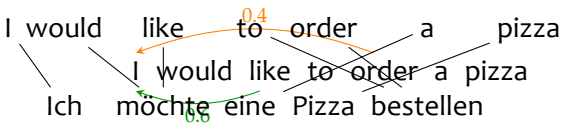
Target Language Parser Training after Arc Distribution Projection

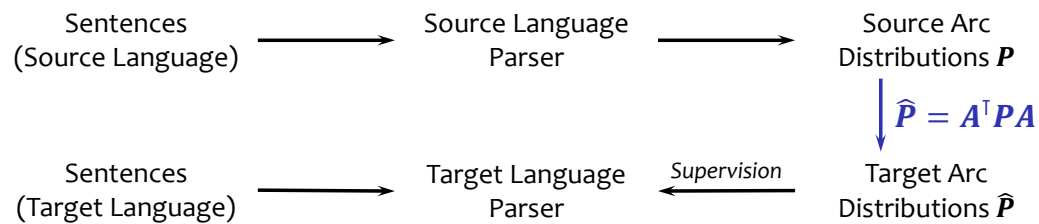
Input: Sentences and dependency parse trees in source language;
Translated sentences in target language;
Word correspondence between parallel sentences. } Pretrained Multilingual Models

Output: Dependency parse trees in target language.

$\boldsymbol{P}_{i,j}$: $P(\text{head} = \boldsymbol{w}_j \mid \text{word} = \boldsymbol{w}_i)$

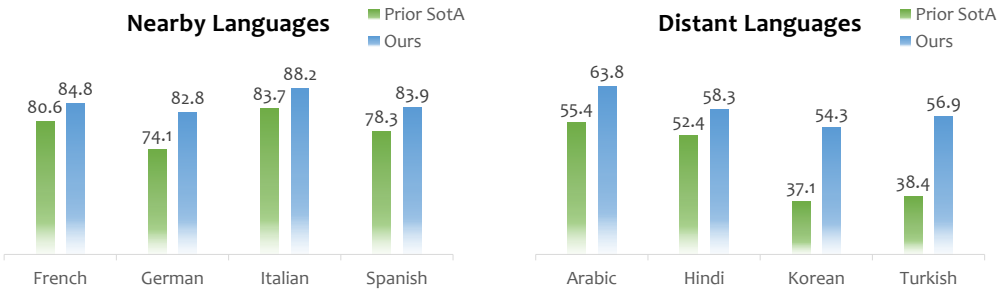
$\boldsymbol{A}_{i,j}$: 1 if source word $\boldsymbol{w}_i$ and target word $\boldsymbol{u}_j$ are aligned, otherwise 0.





Zero-Shot Cross-Lingual Dependency Parsing: Results

Metric: Unlabeled attachment score (UAS, ↑).
Source language: English (95.8 UAS on English).



Question: Can we transfer NLP models to another language through cross-lingual grounding?
Answer: Yes, through substructure (arc) distribution projection.

Part III: Towards Language-Universal NLP through Cross-Lingual Grounding

Question: Can we transfer NLP models to another language through cross-lingual grounding?

Approach: (1) Train a source language parser;
(2) Project the source parser prediction to the target language;
(3) Train a target language parser to fit projected distribution.

Answer: Yes, through substructure (arc) distribution projection [**S**GL, ACL 2022].

Thanks!

Kevin Gimpel

Karen Livescu

Daniel Fried

Marjan Ghazvininejad

Roger Levy

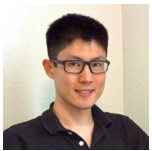
Jiayuan Mao

Mirac Suzgun

- Armen Aghajanyan
- Xinyun Chen
- Hyung Won Chung
- David Dohan
- Dipanjan Das

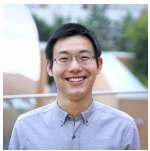
Josh Tenenbaum

- Markus Freitag
- Lingyu Gao
- Vikram Gupta
- Yuning Jiang
- Mike Lewis



Sida Wang

- Lei Li
- Jessy Lin
- Kanishka Misra
- Sebastian Ruder
- Mrinmaya Sachan

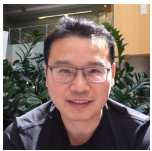


Jiajun Wu

- Nathan Scales
- Nathanael Schärli
- Bowen Shi
- Suraj Srivats
- Jian Sun

Luke Zettlemoyer

- Yi Tay
- Shubham Toshniwal
- Soroush Vosoughi
- Eric Wallace
- Xuezhi Wang



Denny Zhou

- Jason Wei
- Tete Xiao
- Scott Yih
- Ruiqi Zhong
- Hao Zhou