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?

\[ \downarrow \] Semantic Parser

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

\[ \downarrow \] Python Interpreter

120

Ich möchte eine Pizza bestellen

\[ \downarrow \] German

I would like to order a pizza

\[ \downarrow \] English

\[ \downarrow \] Execution Results
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?

```
Sentence
  / \                                     / \                                    / \  
Noun Phrase                  Verb Phrase                           
  / \                                / \                                / \  
Determiner    Noun Phrase          Verb                           
     / \                          / \                                / \  
    the     Noun Phrase          Verb                           
          / \                    / \                                / \  
         the     Noun Phrase      Verb                           
            / \                / \                                / \  
           the     Noun Phrase    Verb                           
              / \             / \                                / \  
             the     Noun Phrase Verb                           
```
What are Language Structures?

The cat near the children meows.

Constituency Parse Tree

```
S
  NP  VP
    the  meows
    cat  PP
    near  NP
    the  children
```

Dependency Parse Tree

```
root
The cat near the children meows
```

Truth-Conditional Semantics

\[ \lambda x.\lambda y.\text{cat}(x) \land \text{children}(y) \land \text{near}(x, y) \land \text{meow}(x) \]

SQL

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

Python

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

Caption
A cat is smiling.

Image

As Supervision

Caption
A cat is smiling.

Image

Constituency Parse Tree

S

NT

A cat is smiling

• Learning semantic parses through execution results

Natural Language Command
Count the number of characters in a string.

Program

\[
\text{def count}(s):
    \text{return} \ \text{len}(s)
\]

Informed Sampling

Program

\[
\text{def count}(s):
    \text{return} \ \text{len}(s)
\]

Execution

Output

3

• Learning to parse another language through cross-lingual grounding

L2 (German) Sentence
Ich möchte eine Pizza bestellen.

L2 (German) Parse Tree

As Supervision

Word Correspondence
I would like to order a pizza

L1 (English) Parse Tree
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.

\[ S \]

\[ S \rightarrow NT \rightarrow A \rightarrow cat \rightarrow is \rightarrow NT \rightarrow \text{standing} \rightarrow NT \rightarrow \text{on} \rightarrow NT \rightarrow the \rightarrow \text{lawn} \]

A cat is standing on the lawn.
**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. 

\( P_\Theta(v_{“a\ cat”}) \): Probability of “a cat” to be a constituent.

**V:** Semantic representation of words and word spans.

**\( \Theta \):** Structure of parse trees.

---

**Caption**  

*A cat is standing on the lawn.*

---

**Constituency Parse Tree**

```
S
  /\ NT
 /   /\ NT
A cat is NT standing NT on NT the lawn
```

---

**Text Encoder**

\( c_1 : \text{a cat} \)

\( c_2 : \text{the lawn} \)

\( c_3 : \text{on the lawn} \)

---

**Joint Embedding Space**

**Mismatched Image**

**Matched Image**

**Image Encoder**

**Reward for Parser: Estimated Text Span Concreteness**
VG-NSL: Image Encoder

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

\[ \mathbf{u}_{\text{img}} = \Phi \cdot \text{ResNet}(\mathbf{img}) \]
VG-NSL: Joint Visual-Semantic Embedding Space

**Parameters for Text Encoder:** Word representations $V$, parser parameters $\Theta$.

**Parameters for Image Encoder:** Linear projector $\Phi$.

**Joint Embedding Space:** Train $V$ and $\Phi$ – 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.

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

$$
\text{sim}(\cdot, \cdot) = \cos(\cdot, \cdot) \quad [\cdot]_+ = \max(0, \cdot) \quad \delta: \text{margin score}
$$
VG-NSL: Quantify Visual Concreteness

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

\[ \ell(c; i, i') = \text{sim}(i', c) - \text{sim}(i, c) \]

\[
\begin{align*}
\text{sim}(\text{red apple}, \text{a cat}) &= 0.2 \\
\text{sim}(\text{tail}, \text{a cat}) &= 0.9 \\
\text{sim}(\text{red apple}, \text{on the}) &= 0.4 \\
\text{sim}(\text{tail}, \text{on the}) &= 0.4
\end{align*}
\]

Value of \( \ell \):

\( \ell = -0.7 \) for \( \text{red apple} \) and \( \text{a cat} \)

\( \ell = 0 \) for \( \text{tail} \) and \( \text{on the} \)

**Key Idea:** Smaller \( \ell(c) \) \( \implies \) \( c \) 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 $\Theta$:

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

$\eta$: learning rate
VG-NSL: Results on the MSCOCO (Lin et al., 2014) Dataset

F₁ Score (↑)

<table>
<thead>
<tr>
<th>Model</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRPN</td>
<td>52.5</td>
</tr>
<tr>
<td>ON-LSTM</td>
<td>45.5</td>
</tr>
<tr>
<td>VG-NSL (ours)</td>
<td>54.4</td>
</tr>
</tbody>
</table>

Self-F₁ Score (↑)

<table>
<thead>
<tr>
<th>Model</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRPN</td>
<td>60.3</td>
</tr>
<tr>
<td>ON-LSTM</td>
<td>69.3</td>
</tr>
<tr>
<td>VG-NSL (ours)</td>
<td>89.8</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>F&lt;sub&gt;1&lt;/sub&gt; Score, English (↑)</th>
<th>F&lt;sub&gt;1&lt;/sub&gt; Score, French (↑)</th>
<th>F&lt;sub&gt;1&lt;/sub&gt; Score, German (↑)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRPN</td>
<td>30.8</td>
<td>27.5</td>
<td>31.5</td>
</tr>
<tr>
<td>ON-LSTM</td>
<td></td>
<td>27.7</td>
<td>34.9</td>
</tr>
<tr>
<td>VG-NSL</td>
<td>38.7</td>
<td>38.1</td>
<td>38.3</td>
</tr>
<tr>
<td>(ours)</td>
<td></td>
<td>(ours)</td>
<td>(ours)</td>
</tr>
</tbody>
</table>

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.

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

**Execution**
count("cat")
3

**Output**

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

**Answer**: Yes, on semantic parsing without program supervision [MSWLT, 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

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

import collections
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.

\[
\max_{\Theta} \prod_x P_{\Theta}(x) = \prod_{i=1}^{\left| x \right|} P_{\Theta}(x_i \mid x_1, \ldots, x_{i-1})
\]

\( \Theta \): Model parameters.
\( x \): Training example.
\( x_i \): \( i^{th} \) token of \( x \).

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

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

\( x_1, \ldots, x_c \): 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:**

\[
\begin{align*}
\text{text} & : \text{Execution results of } x_i. \\
\text{s}_1 & : P_\Theta(s_1 | \ldots) = 0.4 \\
\text{s}_2 & : P_\Theta(s_2 | \ldots) = P_\Theta(s_3 | \ldots) = 0.3 + 0.3 = 0.6
\end{align*}
\]

**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 $P_{\Theta}(s \mid \ldots)$.

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

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

```python
def count(s):
    return len(s)
def count(string):
    cnt = 0
    for ch in string:
        cnt += ch.islower()
    return cnt
def count(s):
    return len([c for c in s if c.islower()])
```

**Output**:

<table>
<thead>
<tr>
<th>Program</th>
<th>Input 1:</th>
<th>Input 2:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>Output 1: 5</td>
<td>Output 2: 5</td>
</tr>
<tr>
<td>$x_2$</td>
<td>Output 1: 5</td>
<td>Output 2: 4</td>
</tr>
<tr>
<td>$x_3$</td>
<td>Output 1: 5</td>
<td>Output 2: 4</td>
</tr>
</tbody>
</table>

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

**Step 2**: Synthesize input cases

- Input 1: "hello"
- Input 2: "Hello"

**Step 3**: Execute

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

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

Question: Can execution results, as grounding signals, help learn semantic structures?
Answer: Yes, on semantic parsing without program supervision [MSWLT, NeurIPS 2021].
Answer: Yes, on natural language to code translation [SFGZW, 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.

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

---

[I would like to order a pizza](ROOT)

[Ich möchte eine Pizza bestellen](ROOT)

Target Language Parser Training after Arc Distribution Projection

**Input:** Sentences and dependency parse trees in source language; Translated sentences in target language; Pretrained Multilingual Models

**Output:** Dependency parse trees in target language.

\[ P_{i,j}: P(\text{head} = w_j \mid \text{word} = w_i) \]

\[ A_{i,j}: 1 \text{ if source word } w_i \text{ and target word } u_j \text{ are aligned, otherwise } 0. \]

\[ \hat{P} = A^T PA \]

Sentences (Source Language) -> Source Language Parser -> Source Arc Distributions \( P \) -> Target Language Parser (with Supervision) -> Target Arc Distributions \( \hat{P} \)
Zero-Shot Cross-Lingual Dependency Parsing: Results

**Metric:** Unlabeled attachment score (UAS, ↑).

**Source language:** English (95.8 UAS on English).

**Nearby Languages**

<table>
<thead>
<tr>
<th>Language</th>
<th>Prior SotA</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>French</td>
<td>80.6</td>
<td>84.8</td>
</tr>
<tr>
<td>German</td>
<td>74.1</td>
<td>82.8</td>
</tr>
<tr>
<td>Italian</td>
<td>83.7</td>
<td>88.2</td>
</tr>
<tr>
<td>Spanish</td>
<td>78.3</td>
<td>83.9</td>
</tr>
</tbody>
</table>

**Distant Languages**

<table>
<thead>
<tr>
<th>Language</th>
<th>Prior SotA</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>55.4</td>
<td>63.8</td>
</tr>
<tr>
<td>Hindi</td>
<td>52.4</td>
<td>58.3</td>
</tr>
<tr>
<td>Korean</td>
<td>37.1</td>
<td>54.3</td>
</tr>
<tr>
<td>Turkish</td>
<td>38.4</td>
<td>56.9</td>
</tr>
</tbody>
</table>

**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 [SGL, ACL 2022].
Overview

Part I: Grammar from Vision

Part II: Semantics through Execution

Part III - Structure with X-Lingual Grounding

Thanks!

Kevin Gimpel
Karen Livescu
Daniel Fried
Marjan Ghazvininejad
Roger Levy
Jiayuan Mao

Mirac Suzgun
Josh Tenenbaum
Sida Wang
Jiajun Wu
Luke Zettlemoyer
Denny Zhou

- Armen Aghajanyan
- Xinyun Chen
- Hyung Won Chung
- David Dohan
- Dipanjan Das
- Markus Freitag
- Lingyu Gao
- Vikram Gupta
- Yuning Jiang
- Mike Lewis
- Lei Li
- Jessy Lin
- Kanishka Misra
- Sebastian Ruder
- Mrinmaya Sachan
- Nathan Scales
- Nathanael Schärli
- Bowen Shi
- Suraj Srivats
- Jian Sun
- Yi Tay
- Shubham Toshniwal
- Soroush Vosoughi
- Eric Wallace
- Xuezhi Wang
- Jason Wei
- Tete Xiao
- Scott Yih
- Ruiqi Zhong
- Hao Zhou