Compositional Pre-Training for Semantic Parsing with BERT
Arnaud Autef, Simon Hagege
Stanford University

Overview
We study Transformer-based Encoder Decoder architectures on a semantic parsing task: Geoquery. We investigate the effects of a BERT Encoder and data recombination methods to augment the dataset. Since the report, our latest result with BERT achieves 0.70 strict accuracy without copying mechanism!

Introduction
Semantic parsing conversion of natural language utterances to logical forms Transformers neural architecture based on Multi-head Attention and FeedForward sublayers, that we use in an Encoder Decoder framework as in [3]

BERT Transformer pre-trained to learn a language model through two tasks [1]:
• Masked Language Model
• Next sequence prediction

Data Recombination Data augmentation technique introduced in [2], generates new data using synchronous context-free grammars (SCFG):
• Entity: abstracting entities with their types, based on predicates in the logical form (e.g. stateid)
• Nesting: abstracting both entities and whole phrases with their types
• Concat-k: combining k sentences into a single

Approach
GeoQuery dataset
what states border Texas?
what is the highest mountain in Colorado?

SCFG
Generate rules

Recombiant Examples
what states border Texas?
what is the highest mountain in Colorado?

Train model

TSP Transformer semantic parser: Encoder-Decoder with N stacked transformer layers
BSP BERT semantic parser: TSP with BERT as the Encoder

Evaluation metric
Strict Match between output query strings \((\hat{y}_i)_{1 \leq i \leq n}\) and the target strings \((y_i)_{1 \leq i \leq n}\)

Jaccard evaluation Match between the sets of characters of a model output string and the corresponding target query string

\[
\text{Strict} = \frac{1}{n} \sum_{i=1}^{n} (y_i = \hat{y}_i)
\]

Jaccard \(\text{Jacc}(y_i, \hat{y}_i)\)

Knowledge-based evaluation (KB) Interpreting the outputs of our model \(\hat{y}_i\) as SQL queries and compute the share of model outputs that both
• Correspond to a valid query to the database
• Yield to an identical output to the target query \(y_i\)

Large models

Data Recombination methods
Recombination Strict KB Jaccard \(\text{Jacc}_{\text{strict}}\)
No recombination 0.136 0.169 0.891 0.225
Entity 0.189 0.279 0.902 0.282
Nesting 0.189 0.241 0.900 0.282
Concat-2 0.157 0.192 0.892 0.204

• Size \(k_{\text{model}}\) = 512, \(N = 6\) transformer sublayers, Adam optimizer with fixed learning rate
• Two-steps approach: training - fine tuning with / without data recombination examples
• Entity best single-shot recombination method

BSP - TSP comparison

<table>
<thead>
<tr>
<th>Models</th>
<th>Strict KB Jaccard (\text{Jacc}_{\text{strict}})</th>
<th>TSP fixed 0.189 0.259 0.902 0.282</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSP adaptive 0.086 0.144 0.859 0.118</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSP adaptive 0.293 0.425 0.944 0.457</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relative Impropr.</th>
<th>55.4%</th>
<th>64.1%</th>
<th>4.7%</th>
<th>62.1%</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Absolute Impropr.</th>
<th>0.014</th>
<th>0.106</th>
<th>0.012</th>
<th>0.175</th>
</tr>
</thead>
</table>

Tests with adaptive (increasing then decreasing) learning rate
Recombination method used: Entity
Poor performances, better results with BSP

Multi-Head Attention

Figure: Visualization of self-attention activation on Encoder and Decoder first layer

Conclusion
• Strong performances of Transformers, improvements with BERT encoder, even with shallow architectures
• Best results obtained with a shallow architecture due to the limited number of training examples

Next steps to improve the results:
• Push data recombination further
• Implement a copying mechanism for the Decoder, as in [2] with RNNs
• Models architecture engineering: Decoder dimensions, freezing fewer BSP layers

References