Language Models with Transformers

Chenguang Wang, Mu Li, Alexander J. Smola
Amazon Web Services
Background
Language Model (LM)

• Predict what word comes next

Start to learn English
Language Model (LM)

• Predict what word comes next
• Useful in many NLP applications

Start to learn English
Language Model (LM)

• Predict what word comes next
• Useful in many NLP applications

Start to learn English
Learn to start business

• Many NLP problems share similar definition

Word order matters!
Language Model with RNNs

• RNN uses one-hot encoding
Language Model with RNNs

- RNN models the word order in hidden state
Language Model with RNNs

• RNN models the word order in hidden state
Language Model with RNNs

- RNN models the word order in hidden state
SOTA NLP with Transformers

Positional encoding

Other components are omitted for simplicity [Devlin, Jacob, et al 2018]
SOTA NLP with Transformers

Other components are omitted for simplicity [Devlin, Jacob, et al 2018]

- Parallelizable
- Efficient
SOTA NLP with Transformers

Transformer

Self-attention

Positional encoding

Other components are omitted for simplicity [Devlin, Jacob, et al 2018]

- With less word order
- Parallelizable
- Efficient

- With word order
- Sequential
- Less efficient
SOTA NLP with Transformers

• BERT: a stack of 12 (or 24) Transformer blocks
SOTA NLP with Transformers

• BERT: a stack of 12 (or 24) Transformer blocks
• Trained on large language model datasets
  • Full training cost in excess of $10,000 (16 TPU, 4 days)
• Achieved SOTA results on 11 NLP applications
  • Sentence level tasks: care less about word order
Approach:
Make Best Use of BERT for Language Model
LM: Adapted BERT

- Linear
- Transformer 11
- Transformer 0
- Embedding

BERT with Linear Layer

- Fixed weights
- Tunable weights
LM 1: Adapted BERT with Fixed Weights

<table>
<thead>
<tr>
<th>Model</th>
<th>Test PPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>69.32</td>
</tr>
<tr>
<td>RNN</td>
<td>42.25</td>
</tr>
</tbody>
</table>

Only moderate results (the Lower, the Better)

Fixed weights

Tunable weights
LM 2: Adapted BERT with All Weights

<table>
<thead>
<tr>
<th>Model</th>
<th>Test PPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>69.32</td>
</tr>
<tr>
<td>BERT-All</td>
<td>67.43</td>
</tr>
<tr>
<td>RNN</td>
<td>42.25</td>
</tr>
</tbody>
</table>

Fixed weights

Tunable weights

Overfitting
LM 3: Adapted BERT with Partial Weights

<table>
<thead>
<tr>
<th>Model</th>
<th>Test PPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>69.32</td>
</tr>
<tr>
<td>BERT-All</td>
<td>67.43</td>
</tr>
<tr>
<td>BERT-Subset</td>
<td>40.56</td>
</tr>
<tr>
<td>RNN</td>
<td>42.25</td>
</tr>
</tbody>
</table>

Fix a subset of weights is promising

However, enumerating is not feasible
LM 4: Adapted BERT with RNN

Add RNN to capture word order is promising

<table>
<thead>
<tr>
<th>Model</th>
<th>Test PPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>69.32</td>
</tr>
<tr>
<td>BERT-RNN</td>
<td>41.64</td>
</tr>
<tr>
<td>RNN</td>
<td>42.25</td>
</tr>
</tbody>
</table>

However, enumerating is not feasible
- Where
- How many
Where to add the RNN layers?
Where to add the RNN layers?

Which layer’s pre-trained weights should be fixed?
Coordinate Architecture Search (CAS)

- Step 1: Choose a layer’s weights to fix

![Diagram of CAS process with fixed and tunable weights for Transformers and embeddings.]

- Greedy strategy: fine-tune the resulting BERT and keep the best embedding
Coordinate Architecture Search (CAS)

• Step 1: Choose a layer’s weights to fix
• Step 2: Choose a position to add a RNN layer
Coordinate Architecture Search (CAS)

- Step 1: Choose a layer’s weights to fix
- Step 2: Choose a position to add a RNN layer
- Step 3: Go to Step 1 or **Terminate**

Add a linear layer
Coordinate Architecture Search (CAS)

- Step 1: Choose a layer’s weights to fix
- Step 2: Choose a position to add a RNN layer
- Step 3: Go to Step 1 or Terminate

• Greedy strategy: fine-tune the resulting BERT and keep the best
Best LM: Adapted BERT with CAS
Best LM: Adapted BERT with CAS

- AWD-LSTM-MoS-BERTVocab
- BERT-CAS-Subset
- BERT-CAS
- BERT-Large-CAS

BERT-Large+CAS is best
Best LM: Adapted BERT with CAS

BERT-Large+CAS is best

Capture word order

Test Perplexity

0 20 40 60 80 100 120

PTB  WT-103

AWD-LSTM-MoS-BERTVocab  BERT  BERT-CAS-Subset  BERT  BERT-CAS-LSTM  BERT-Large-CAS

BERT-Dense+LAS is best

Capture word order
Best LM: Adapted BERT with CAS

Achieve SOTA: 31.34 PPL with 0.5 GPU days

BERT-Large+CAS is best

Capture word order
Best LM: Adapted BERT with CAS

BERT-Large+CAS is best

Capture word order

Achieve SOTA: 31.34 PPL with 0.5 GPU days

Achieve 20.42 PPL with 1B tokens
Take-aways

• BERT needs to be adapted for language model
• Add RNN layers with neural architecture search works
• Fix pre-trained weights with neural architecture search works