Contextualized Word Representations

Natural Language Processing

Based on slides from Stanford cs224n (Chris Manning and Abigail See), Graham Neubig and Ashish Vaswani
We have described recurrent architectures for language modeling

But these architectures are more general purpose

Let’s first see how they can be used for general classification tasks and expand to:

- Bidirectional RNNs
- Multi-layer RNNs
- Transformers

This will teach us about “contextualized word representations”

We will then go back to language modeling and how it is used for large-scale representation learning of contextualized words
Sentence classification

• Classify sentences according to various traits
• Topic, sentiment etc.

I hate this movie

very good
good
neutral
bad
very bad

I love this movie

very good
good
neutral
bad
very bad

• It would be good to have a sentence representation
Sentence classification

• Many tasks are variants of this:
  • Textual entailment
  • Textual similarity
  • …
The movie was terribly exciting.

very good: 0.9
very bad: 0.1

• Func: sum, average, element-wise mean-pooling, element-wise max-pooling
• Objective: cross-entropy between true class and predicted distribution
• Parameters: word embeddings
Contextualized word rep.

- **Func**: sum, average, element-wise mean-pooling, element-wise max-pooling, *last hidden state*
- **Objective**: cross-entropy between true class and predicted distribution
- **Parameters**: word embeddings + RNN parameters
The movie was terribly exciting.

**Very good**

- **Func:** sum, average, element-wise mean-pooling, element-wise max-pooling, **last hidden state**
- **Objective:** cross-entropy between true class and predicted distribution
- **Parameters:** word embeddings + RNN parameters

**Contextual representation for ‘terribly’**
Contextualized word rep.

- **Func**: sum, average, element-wise mean-pooling, element-wise max-pooling, last hidden state
- **Objective**: cross-entropy between true class and predicted distribution
- **Parameters**: word embeddings + RNN parameters

This contains information on left context only!
The movie was terribly exciting!
Bidirectional RNN

• RNN here can be LSTM/GRU/whatever

• Each hidden state is now contextualized with both left and right context

• Parameters for forward and backward RNNs are separate

• These are applicable for the sentence classification task but not for generative LMs where only the history is observed.

• In general, using both sides improves performance by a lot!

  • Was the greatest hit of NLP in 2017!
Multi-layer RNN

- We can make RNNs deeper by having the input of each token in one RNN be the input to a deeper RNN.
- These are called multi-layer RNN or stacked RNNs.
The movie was terribly exciting.

Layer 1

Layer 2

Layer 3

Hidden state/output state of layer $i$ is the input to layer $i+1$
Transformers

• **Motivation:**
  
  • RNNs are hard to parallelize so it takes a lot of time to run them on large amounts of data
  
  • They also don’t explicitly model long and close range dependencies
  
  • Can we get contextualized representations in a way that overcomes these drawbacks?

  • Yes! Attention
Approximate Transformer Block

- We want to re-represent \( r_2 \) as a function of context:

\[
    r_i = W_{\text{embed}} \cdot e_i
\]
Approximate Transformer Block

- We want to re-represent $r_2$ as a function of context:

$$s_{ij} = r_i^\top r_j$$

$$r_i = W_{\text{embed}} \cdot e_i$$
Approximate Transformer Block

- We want to re-represent $r_2$ as a function of context:

\[
p_{ij} = \text{softmax}(s_i, \ldots, s_x)
\]

\[
s_{ij} = r_i^\top r_j
\]

\[
r_i = W_{\text{embed}} \cdot e_i
\]
Approximate Transformer Block

- We want to re-represent $r_2$ as a function of context:

$$o_i = \sum_j p_{ij} r_j$$

$$p_{ij} = \text{softmax}(s_{i1}, \ldots, s_{i|x|})$$

$$s_{ij} = r_i^\top r_j$$

$$r_i = W_{\text{embed}} \cdot e_i$$
Approximate Transformer Block

• We want to re-represent \( r_2 \) as a function of context:

\[
\text{FFNN}(o) = \text{relu}(W_1 o + b_1)W_2 + b_2
\]

\[
o_i = \sum_j p_{ij} r_j
\]

\[
p_{ij} = \text{softmax}(s_{i1}, \ldots, s_{i|x|})
\]

\[
s_{ij} = r_i^\top r_j
\]

\[
r_i = W_{\text{embed}} \cdot e_i
\]
Approximate Transformer Block

• We want to re-represent \( r_2 \) as a function of context:

\[
\text{FFNN}(o) = \text{relu}(W_1 o + b_1)W_2 + b_2
\]

\[
o_i = \sum_j p_{ij} r_j
\]

\[
p_{ij} = \text{softmax}(s_{i1}, \ldots, s_{i|x|})
\]

\[
s_{ij} = r_i^\top r_j
\]

\[
r_i = W_{\text{embed}} \cdot e_i
\]

(a) This is fast and parallelizable! (b) words attend to entire context!
Approximate transformer

- This architecture is just a few matrix multiplications in parallel
- All pairs of words attend to one another

\[ A(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \]
Details: Multi-head Attention

- **Problem**: Each word attends to a single location in previous layer

- **Solution**: Have $K$ attention heads
Detail: Multi-head Attention

- In our case Q, K, V are all just the representations of the inputs to the transformer block

\[
\text{MultiHeadAtt}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W_3
\]

\[
\text{head}_i = A(QW_q^i, KW_k^i, VW_v^i)
\]

- We use the linear projections \( W_q, W_k, W_v \) to reduce the dimensionality of the input by factor of \( h \), perform attention \( h \) times, and then concatenate the output to obtain a representation that has the same dimensionality as the input
Detail: Residual Connections and Normalization

• We can call both the attention layer and the feed-forward network that follows it a sublayer.

• The output of each sublayer is defined as: $\text{LayerNorm}(x + \text{sublayer}(x))$

• $x + \text{sublayer}(x)$ is a “residual connection”. It allows gradients to flow in deep networks similar to what we saw in LSTMs and GRUs

• $\text{LayerNorm}$ normalizes the inputs to the layer and improves optimization
Detail: Positional embeddings

- **Problem:**
  - Everything was permutation invariant so far!
  - You can take a training corpus, and apply a fixed permutation to every sentence and you will get the same result.
  - Order seems to matter in language.

- **Solution:** add positional embeddings
  - Input to (first) transformer block is $r_i + p_i$, where $p_i$ is a representation of the integer $i$. 
Transformer Summary

- Blocks are stacked (6, 12, 24 times)
- The encoding of a word is the output of the top-most layer
- This representation is very contextualized and good for many tasks
## Results: Machine Translation

<table>
<thead>
<tr>
<th></th>
<th>EN-DE</th>
<th>EN-FR</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNMT (orig)</td>
<td>24.6</td>
<td>39.9</td>
</tr>
<tr>
<td>ConvSeq2Seq</td>
<td>25.2</td>
<td>40.5</td>
</tr>
<tr>
<td>Transformer*</td>
<td><strong>28.4</strong></td>
<td><strong>41.8</strong></td>
</tr>
</tbody>
</table>

*Transformer models trained >3x faster than the others.*
Attention Visualization
Attention Visualization
Importance of Residuals

With residuals

Without residuals
Pre-training Contextualized Word Representations
Thought

• Pre-trained word representations are nice:
  • Can be trained on tons of data

• But:
  • They don’t take context into account

• Can we pre-train representations for words in context on tons of data and use them like word embeddings?
  • YES! Let’s just use language models!
• Train a forward LM and a backward LM (L-Layers)
  • In practice, 2-layer LSTM
• This gives for every token at index $k$, a $2L+1$ representation: $2L$ are contextualized and one is not.
ELMO

• How are these representations used?

• Learn how to combine the different layers per task

\[ R_k = \{ x_{k,j}^{LM}, h_{k,j}^{LM}, h_{k,j}^{LM} | j = 1, \ldots, L \} \]
\[ = \{ h_{k,j}^{LM} | j = 0, \ldots, L \}, \]

\[ \text{ELMo}_{k}^{\text{task}} = E(R_k; \Theta_{\text{task}}) = \gamma_{\text{task}} \sum_{j=0}^{L} s_{j}^{\text{task}} h_{k,j}^{LM}. \]

• \( s_{\text{task}} \): softmax normalized logits over the layers

• \( \gamma_{\text{task}} \): scaling factor for the entire ELMo vector

• The vector per token can be used by any task
## ELMO Results

<table>
<thead>
<tr>
<th>Task</th>
<th>Previous SOTA</th>
<th>Our baseline</th>
<th>ELMo + baseline</th>
<th>Increase (absolute/relative)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQuAD</td>
<td>Liu et al. (2017)</td>
<td>84.4</td>
<td>81.1</td>
<td>4.7 / 24.9%</td>
</tr>
<tr>
<td>SNLI</td>
<td>Chen et al. (2017)</td>
<td>88.6</td>
<td>88.0</td>
<td>0.7 / 5.8%</td>
</tr>
<tr>
<td>SRL</td>
<td>He et al. (2017)</td>
<td>81.7</td>
<td>81.4</td>
<td>3.2 / 17.2%</td>
</tr>
<tr>
<td>Coref</td>
<td>Lee et al. (2017)</td>
<td>67.2</td>
<td>67.2</td>
<td>3.2 / 9.8%</td>
</tr>
<tr>
<td>NER</td>
<td>Peters et al. (2017)</td>
<td>91.93 ± 0.19</td>
<td>90.15</td>
<td>2.06 / 21%</td>
</tr>
<tr>
<td>SST-5</td>
<td>McCann et al. (2017)</td>
<td>53.7</td>
<td>51.4</td>
<td>3.3 / 6.8%</td>
</tr>
</tbody>
</table>
GPT-2

- Replaced LSTM with large transformer
- Trained on tons of text
- Got the beautiful unicorn example we saw last time
BERT

• BERT: Bidirectional Encoder Representations from Transformers (Devlin et al., 2019)

• **Problem:**

  • LM are uni-directional, but our understanding of language can be based on both directions

  • Reason: LM are a generative model defining a probability distribution for language

  • But we don’t care about that when we just want to learn good contextualized word representations.
• **Change the objective:** predict a word based on both left and right context (**masked LM**, cross-entropy loss)

• Randomly mask $k(=15\%)$ of the words and predict them from the context

```
The man went to the [mask] to buy a [mask] of milk.
```

• Too little masking: expensive to train

• Too much masking: Not enough context
• Change the objective: guess the next sentence

Sentence A = The man went to the store.
Sentence B = He bought a gallon of milk.
Label = IsNextSentence

Sentence A = The man went to the store.
Sentence B = Penguins are flightless.
Label = NotNextSentence
- **Multi-task objective:**
  - The contextualized word representations are computed for both objectives (masked LM and sentence guessing) with the same weights.
• Change the model: replace two **unidirectional** RNNs with **bidirectional** transformer
BERT Details

- Transformer architecture
  - BERT-base: 12 layers, 768 hidden, 12-head attention
  - BERT-large: 24 layers, 1024-hidden, 16-head attention
- Trained on Wikipedia + BookCorpus
- BERT-large takes some time to train…
BERT Fine-tuning

- The entire BERT model is fine-tuned for downstream tasks with very little parameters added
## BERT Sentence Classification Results

<table>
<thead>
<tr>
<th>System</th>
<th>MNLI-(m/mm)</th>
<th>QQP</th>
<th>QNLI</th>
<th>SST-2</th>
<th>CoLA</th>
<th>STS-B</th>
<th>MRPC</th>
<th>RTE</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>392k</td>
<td>363k</td>
<td>108k</td>
<td>67k</td>
<td>8.5k</td>
<td>5.7k</td>
<td>3.5k</td>
<td>2.5k</td>
<td></td>
</tr>
<tr>
<td>Pre-OpenAI SOTA</td>
<td>80.6/80.1</td>
<td>66.1</td>
<td>82.3</td>
<td>93.2</td>
<td>35.0</td>
<td>81.0</td>
<td>86.0</td>
<td>61.7</td>
<td>74.0</td>
</tr>
<tr>
<td>BiLSTM+ELMo+Attn</td>
<td>76.4/76.1</td>
<td>64.8</td>
<td>79.9</td>
<td>90.4</td>
<td>36.0</td>
<td>73.3</td>
<td>84.9</td>
<td>56.8</td>
<td>71.0</td>
</tr>
<tr>
<td>OpenAI GPT</td>
<td>82.1/81.4</td>
<td>70.3</td>
<td>88.1</td>
<td>91.3</td>
<td>45.4</td>
<td>80.0</td>
<td>82.3</td>
<td>56.0</td>
<td>75.2</td>
</tr>
<tr>
<td>BERT$_{BASE}$</td>
<td>84.6/83.4</td>
<td>71.2</td>
<td>90.1</td>
<td>93.5</td>
<td>52.1</td>
<td>85.8</td>
<td>88.9</td>
<td>66.4</td>
<td>79.6</td>
</tr>
<tr>
<td>BERT$_{LARGE}$</td>
<td>86.7/85.9</td>
<td>72.1</td>
<td>91.1</td>
<td>94.9</td>
<td>60.5</td>
<td>86.5</td>
<td>89.3</td>
<td>70.1</td>
<td>81.9</td>
</tr>
<tr>
<td>Rank</td>
<td>Model</td>
<td>EM</td>
<td>F1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>--------------------------------------------</td>
<td>-------</td>
<td>-------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Human Performance</td>
<td>82.304</td>
<td>91.221</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>Stanford University</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>(Rajpurkar et al. '16)</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>BERT (ensemble)</td>
<td>87.433</td>
<td>93.160</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>Google AI Language</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>BERT (single model)</td>
<td>85.083</td>
<td>91.835</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>Google AI Language</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>nllnet (ensemble)</td>
<td>85.954</td>
<td>91.677</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>Microsoft Research Asia</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>nllnet (single model)</td>
<td>83.468</td>
<td>90.133</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>Microsoft Research Asia</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>QANet (ensemble)</td>
<td>84.454</td>
<td>90.490</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>Google Brain &amp; CMU</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## BERT QA Results

<table>
<thead>
<tr>
<th>Rank</th>
<th>Model Description</th>
<th>EM</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BERT + MMFT + ADA (ensemble)</td>
<td>85.082</td>
<td>87.615</td>
</tr>
<tr>
<td></td>
<td>Microsoft Research Asia</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>BERT + Synthetic Self-Training (ensemble)</td>
<td>84.292</td>
<td>86.967</td>
</tr>
<tr>
<td></td>
<td>Google AI Language</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><a href="https://github.com/google-research/bert">https://github.com/google-research/bert</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>BERT finetune baseline (ensemble)</td>
<td>83.536</td>
<td>86.096</td>
</tr>
<tr>
<td></td>
<td>Anonymous</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Lunet + Verifier + BERT (ensemble)</td>
<td>83.469</td>
<td>86.043</td>
</tr>
<tr>
<td></td>
<td>Layer 6 AI NLP Team</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>PAML+BERT (ensemble model)</td>
<td>83.457</td>
<td>86.122</td>
</tr>
<tr>
<td></td>
<td>PINGAN GammaLab</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Lunet + Verifier + BERT (single model)</td>
<td>82.995</td>
<td>86.035</td>
</tr>
<tr>
<td></td>
<td>Layer 6 AI NLP Team</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Model Size

Effect of Model Size

- MNLI (400k)
- MRPC (3.6k)

Dev Accuracy

Transformer Params (Millions)
Since BERT...

- RoBERTa: The same as BERT but trained better and on more data
  - Substantially better!
- Albert: The same as BERT but model is smaller by tying transformer layers
  - Again - substantially Better
- SenseBERT, SpanBERT, …
Disclaimer

• I glossed over a lot of the details

• **Exact details of training/pre-training/multi-task learning**

• **Sentences are represented with word-pieces and characters, not words**

• ...

• If you are interested read the paper and think of a project!

• But this covered the main ideas - which are simple!
History

- Variants of this was proposed before ELMO, but the true power was cleanly shown in the ELMO paper
  - context2vec (Melamud et al., 2016)
  - CoVe (McMann et al., 2017)
  - ULMfit (Howard and Ruder, 2018)
Scaling

ULMfit
Jan 2018
Training:
1 GPU day

GPT
June 2018
Training
240 GPU days

BERT
Oct 2018
Training
256 TPU days
~320–560 GPU days

GPT-2
Feb 2019
Training
~2048 TPU v3
days according to
a reddit thread
Summary

• Contextualized representations are replacing word representations

• Can be trained on tons of data

• Tuned for downstream tasks

• Lead to large improvements

• Based on LM in two ways:
  • Trained often from a LM-like objective
  • Use architectures from the LM world

• Recommended reading