Contextual word representations

Christopher Potts

Stanford Linguistics

CS 224U: Natural language understanding
May 20
Overview

1. Overview: Resources and guiding insights
2. ELMo: Embeddings from Language Models
3. Transformers
4. BERT: Bidirectional Encoder Representations from Transformers
5. contextualreps.ipynb: Easy ways to bring ELMo and BERT into your project
Associated materials

1. Notebook: contextualreps.ipynb
2. Smith 2019
3. CS224n lecture: slides and YouTube version
4. ELMo:
   ▶ Peters et al. 2018
   ▶ Project site: https://allennlp.org/elmo
5. Transformer
   ▶ Vaswani et al. 2017
   ▶ Alexander Rush: The Annotated Transformer [link]
6. BERT
   ▶ Devlin et al. 2019
   ▶ Project site: https://github.com/google-research/bert
   ▶ bert-as-service [link]
Word representations and context
Word representations and context

1. a. The vase broke.
   b. Dawn broke.
   c. The news broke.
   d. Sandy broke the world record.
   e. Sandy broke the law.
   f. The burgler broke into the house.
   g. The newscaster broke into the movie broadcast.
   h. We broke even.
Word representations and context

1. a. The vase broke.
   b. Dawn broke.
   c. The news broke.
   d. Sandy broke the world record.
   e. Sandy broke the law.
   f. The burgler broke into the house.
   g. The newscaster broke into the movie broadcast.
   h. We broke even.

2. a. flat tire/beer/note/surface
   b. throw a party/fight/ball/fit
Word representations and context

1. a. The vase broke.
   b. Dawn broke.
   c. The news broke.
   d. Sandy broke the world record.
   e. Sandy broke the law.
   f. The burgler broke into the house.
   g. The newscaster broke into the movie broadcast.
   h. We broke even.

2. a. flat tire/beer/note/surface
   b. throw a party/fight/ball/fit

3. a. A crane caught a fish.
   b. A crane picked up the steel beam.
   c. I saw a crane.
Word representations and context

1. a. The vase broke.
   b. Dawn broke.
   c. The news broke.
   d. Sandy broke the world record.
   e. Sandy broke the law.
   f. The burgler broke into the house.
   g. The newscaster broke into the movie broadcast.
   h. We broke even.

2. a. flat tire/beer/note/surface
    b. throw a party/fight/ball/fit

3. a. A crane caught a fish.
    b. A crane picked up the steel beam.
    c. I saw a crane.

4. a. Are there typos? I didn’t see any.
    b. Are there bookstores downtown? I didn’t see any.
Model structure and linguistic structure

The rock rules x 47 x 30 x 34
h₁ h₂ h₃

The Rock
rules

x₁
x₂
x₃

attention
Guiding idea: Attention (from the NLI slides)

\[
\begin{align*}
&h_1 \quad h_2 \quad h_3 \\
&x_3 \quad x_2 \quad x_1 \\
&\text{every} \quad \text{dog} \quad \text{danced} \\
&x_{27} \quad x_{21} \quad x_{11} \\
&\text{some} \quad \text{poodle} \quad \text{danced}
\end{align*}
\]
Guiding idea: Attention (from the NLI slides)

Scores

\[ \tilde{\alpha} = \begin{bmatrix} h_C^T h_1 & h_C^T h_2 & h_C^T h_3 \end{bmatrix} \]

Diagram:

- \( h_1 \)
- \( h_2 \)
- \( h_3 \)
- \( h_A \)
- \( h_B \)
- \( h_C \)

Words:

- every
- dog
- danced
- some
- poodle
- danced
Guiding idea: Attention (from the NLI slides)

attention weights \[ \alpha = \text{softmax}(\tilde{\alpha}) \]

scores \[ \tilde{\alpha} = \begin{bmatrix} h_C^T h_1 & h_C^T h_2 & h_C^T h_3 \end{bmatrix} \]
Guiding idea: Attention (from the NLI slides)

context \( \kappa = \text{mean}(\alpha_1 h_1, \alpha_2 h_2, \alpha_3 h_3) \)

attention weights \( \alpha = \text{softmax}(\tilde{\alpha}) \)

scores \( \tilde{\alpha} = \begin{bmatrix} h_C^\top h_1 & h_C^\top h_2 & h_C^\top h_3 \end{bmatrix} \)
Guiding idea: Attention (from the NLI slides)

Attention combo: \( \tilde{h} = \tanh([\kappa; h_C]W_\kappa) \)

Context: \( \kappa = \text{mean}(\alpha_1 h_1, \alpha_2 h_2, \alpha_3 h_3) \)

Attention weights: \( \alpha = \text{softmax}(\tilde{\alpha}) \)

Scores: \( \tilde{\alpha} = \begin{bmatrix} h_C^\top h_1 & h_C^\top h_2 & h_C^\top h_3 \end{bmatrix} \)
Guiding idea: Attention (from the NLI slides)

attention combo \( \tilde{h} = \tanh([\kappa; h_C]W_k) \) or \( \tilde{h} = \tanh(\kappa W_k + h_C W_h) \)

context \( \kappa = \text{mean}(\alpha_1 h_1, \alpha_2 h_2, \alpha_3 h_3) \)

attention weights \( \alpha = \text{softmax}(\tilde{\alpha}) \)

scores
\[
\tilde{\alpha} = \begin{bmatrix}
h_C^T h_1 & h_C^T h_2 & h_C^T h_3
\end{bmatrix}
\]
Guiding idea: Attention (from the NLI slides)

classifier \[ y = \text{softmax}(\tilde{h}W + b) \]

attention combo \[ \tilde{h} = \tanh([\kappa; h_C]W_\kappa) \]

context \[ \kappa = \text{mean}(\alpha_1 h_1, \alpha_2 h_2, \alpha_3 h_3) \]

attention weights \[ \alpha = \text{softmax}(\tilde{\alpha}) \]

scores \[ \tilde{\alpha} = \begin{bmatrix} h_1^T h_1 & h_2^T h_2 & h_3^T h_3 \end{bmatrix} \]
Guiding idea: Subword modeling

rules
Guiding idea: Subword modeling
Guiding idea: Subword modeling
Guiding idea: Subword modeling

Filters of different length, obtained via dense layers processing the input character embeddings and combined via max-pooling:
Guiding idea: Subword modeling

Filters of different length, obtained via dense layers processing the input character embeddings and combined via max-pooling:
Guiding idea: Subword modeling

Max-pooling layers concatenated to form the word representation.

Filters of different length, obtained via dense layers processing the input character embeddings and combined via max-pooling:
Guiding idea: Positional encoding

From 'The Annotated Transformer'
Guiding idea: Positional encoding

From ‘The Annotated Transformer’
ELMo

1. Overview: Resources and guiding insights

2. ELMo: Embeddings from Language Models

3. Transformers

4. BERT: Bidirectional Encoder Representations from Transformers

5. contextualreps.ipynb: Easy ways to bring ELMo and BERT into your project
Core model structure
Core model structure
Core model structure

The Rock rules

\[
\begin{align*}
\text{rules} & \quad x_{34} \\
\text{The} & \quad \text{Rock} \\
\text{rules}_{4,1} & \quad \text{rules}_{4,1} \\
\text{rules}_{4,2} & \quad \text{rules}_{4,2}
\end{align*}
\]
Core model structure

\[
\text{rules} = s_0^{\text{task}} \cdot x_{34} + s_1^{\text{task}} \cdot \text{rules}_{4,1} + s_2^{\text{task}} \cdot \text{rules}_{4,2}
\]
Word embeddings
Word embeddings
Word embeddings

A series of convolutional filters with max pooling, concatenated to form the initial representation.
Word embeddings

A series of convolutional filters with max pooling, concatenated to form the initial representation
Word embeddings

A series of convolutional filters with max pooling, concatenated to form the initial representation
Word embeddings

A series of convolutional filters with max pooling, concatenated to form the initial representation
Word embeddings

Highway layer

Highway layers introduce gating information between layers:

\[(cW + b)(cW_T + b_T) + c(1 - (cW_T + b_T))\]

A series of convolutional filters with max pooling, concatenated to form the initial representation
Word embeddings

Linear projection to the embedding dimensionality, which must be double the RNN hidden dimensionality

Highway layers introduce gating information between layers:

\[(cW + b)(cW_T + b_T) + c(1 - (cW_T + b_T))\]

A series of convolutional filters with max pooling, concatenated to form the initial representation
Word embeddings

Highway layers introduce gating information between layers:

\[(cW + b)(cW + b^T) + c(1 - (cW + b^T))\]

Linear projection to the embedding dimensionality, which must be double the RNN hidden dimensionality

A series of convolutional filters with max pooling, concatenated to form the initial representation
## ELMo model releases

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameters</th>
<th>Hidden size</th>
<th>Output size</th>
<th>Highway layers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>13.6M</td>
<td>1024</td>
<td>128</td>
<td>1</td>
</tr>
<tr>
<td>Medium</td>
<td>28.0M</td>
<td>2048</td>
<td>256</td>
<td>1</td>
</tr>
<tr>
<td>Original</td>
<td>93.6M</td>
<td>4096</td>
<td>512</td>
<td>2</td>
</tr>
<tr>
<td>Original (5.5B)</td>
<td>93.6M</td>
<td>4096</td>
<td>512</td>
<td>2</td>
</tr>
</tbody>
</table>

Additional details at [https://allennlp.org/elmo](https://allennlp.org/elmo); the options files reveal additional information about the subword convolutional filters, activation functions, thresholds, and layer dimensions.
Transformers

1. Overview: Resources and guiding insights
2. ELMo: Embeddings from Language Models
3. Transformers
4. BERT: Bidirectional Encoder Representations from Transformers
5. contextualreps.ipynb: Easy ways to bring ELMo and BERT into your project
Core model structure
Core model structure

\[ a_{\text{input}} + x_{47} + p_1 = \text{The} + 1 \]

\[ b_{\text{input}} + x_{30} + p_2 = \text{Rock} + 2 \]

\[ c_{\text{input}} + x_{34} + p_3 = \text{rules} + 3 \]

\[ c_{\text{input}} = x_{34} + p_3 \]
Core model structure

\[ c_{\text{attn}} = \text{sum}( [\alpha_1 a_{\text{input}}, \alpha_2 b_{\text{input}}] ) \]

\[ \alpha = \text{softmax}(\tilde{\alpha}) \]

\[ \tilde{\alpha} = \begin{bmatrix} \frac{c_{\text{input}} a_{\text{input}}}{\sqrt{d_k}}, & \frac{c_{\text{input}} b_{\text{input}}}{\sqrt{d_k}} \end{bmatrix} \]

\[ c_{\text{input}} = x_{34} + p_3 \]
Core model structure

\[ y = \text{softmax}(\tilde{h}W + b) \]

Attention combo
\[ \tilde{h} = \tanh([\kappa; h_c]W_k) \]

Context
\[ \kappa = \text{mean}(\alpha_1 h_1, \alpha_2 h_2, \alpha_3 h_3) \]

Attention weights
\[ \alpha = \text{softmax}(\tilde{\alpha}) \]

Scores
\[ \tilde{\alpha} = \begin{bmatrix} h_c^T h_1 & h_c^T h_2 & h_c^T h_3 \end{bmatrix} \]

\[ c_{\text{attn}} = \text{sum}\left([\alpha_1 a_{\text{input}}, \alpha_2 b_{\text{input}}]\right) \]

\[ \alpha = \text{softmax}(\tilde{\alpha}) \]

\[ \tilde{\alpha} = \begin{bmatrix} \frac{c_{\text{input}}^T a_{\text{input}}}{\sqrt{d_k}} & \frac{c_{\text{input}}^T b_{\text{input}}}{\sqrt{d_k}} \end{bmatrix} \]

\[ c_{\text{input}} = x_{34} + p_3 \]
Core model structure

\[ c_{\text{attn}} = \text{sum} ([\alpha_1 a_{\text{input}}, \alpha_2 b_{\text{input}}]) \]

\[ \alpha = \text{softmax}(\tilde{\alpha}) \]

\[ \tilde{\alpha} = \left[ \frac{c_{\text{input}}^\top a_{\text{input}}}{\sqrt{d_k}}, \frac{c_{\text{input}}^\top b_{\text{input}}}{\sqrt{d_k}} \right] \]

\[ c_{\text{input}} = x_3 + p_3 \]
Core model structure

\[ c_{\text{alayer}} = c_{\text{attn}} + \text{Dropout}(c_{\text{input}}) \]

\[ c_{\text{attn}} = \text{sum}([\alpha_1 a_{\text{input}}, \alpha_2 b_{\text{input}}]) \]

\[ \alpha = \text{softmax}(\tilde{\alpha}) \]

\[ \tilde{\alpha} = \begin{bmatrix} c_{\text{input}}^\top a_{\text{input}} / \sqrt{d_k} \\ c_{\text{input}}^\top b_{\text{input}} / \sqrt{d_k} \end{bmatrix} \]

\[ c_{\text{input}} = x_{34} + p_3 \]
Core model structure

\[
\begin{align*}
    c_{\text{anorm}} &= \frac{c_{\text{alayer}} - \text{mean}(c_{\text{alayer}})}{\text{std}(c_{\text{alayer}}) + \varepsilon} \\
    c_{\text{alayer}} &= c_{\text{attn}} + \text{Dropout}(c_{\text{input}}) \\
    c_{\text{attn}} &= \text{sum}([\alpha_1 a_{\text{input}}, \alpha_2 b_{\text{input}}]) \\
    \alpha &= \text{softmax}(\tilde{\alpha}) \\
    \tilde{\alpha} &= \left[ \frac{c_{\text{input}}^\top a_{\text{input}}}{\sqrt{d_k}}, \frac{c_{\text{input}}^\top b_{\text{input}}}{\sqrt{d_k}} \right] \\
    c_{\text{input}} &= x_{34} + p_3
\end{align*}
\]
Core model structure

The Rock rules

\[ x_{47} x_{30} x_{34} \]

1 2 3

\[ \alpha = \text{softmax}(\tilde{\alpha}) \]

\[ \tilde{\alpha} = \frac{c_{\text{input}} \mathbf{a}_{\text{input}}}{\sqrt{d_k}} + \frac{c_{\text{input}} \mathbf{b}_{\text{input}}}{\sqrt{d_k}} \]

\[ c_{\text{input}} = x_{34} + p_3 \]

\[ c_{\text{ff}} = \text{ReLU}(c_{\text{anorm}} W_1 + b_1) W_2 + b_2 \]

\[ c_{\text{anorm}} = \frac{c_{\text{alayer}} - \text{mean}(c_{\text{alayer}})}{\text{std}(c_{\text{alayer}}) + \varepsilon} \]

\[ c_{\text{alayer}} = c_{\text{attn}} + \text{Dropout}(c_{\text{input}}) \]
Core model structure

\[ c_{\text{flayer}} = c_{\text{anorm}} + \text{Dropout}(c_{\text{ff}}) \]

\[ c_{\text{ff}} = \text{ReLU}(c_{\text{anorm}}W_1 + b_1)W_2 + b_2 \]

\[ c_{\text{anorm}} = \frac{c_{\text{alayer}} - \text{mean}(c_{\text{alayer}})}{\text{std}(c_{\text{alayer}}) + \varepsilon} \]

\[ c_{\text{alayer}} = c_{\text{attn}} + \text{Dropout}(c_{\text{input}}) \]

\[ c_{\text{attn}} = \text{sum} ([\alpha_1 a_{\text{input}}, \alpha_2 b_{\text{input}}]) \]

\[ \alpha = \text{softmax}(\tilde{\alpha}) \]

\[ \tilde{\alpha} = \left[ \frac{c_{\text{input}}^\top a_{\text{input}}}{\sqrt{d_k}}, \frac{c_{\text{input}}^\top b_{\text{input}}}{\sqrt{d_k}} \right] \]

\[ c_{\text{input}} = x_{34} + p_3 \]
Core model structure

\[ c_{\text{out}} = \frac{c_{\text{fflayer}} - \text{mean}(c_{\text{fflayer}})}{\text{std}(c_{\text{fflayer}}) + \varepsilon} \]

\[ c_{\text{fflayer}} = c_{\text{anorm}} + \text{Dropout}(c_{\text{ff}}) \]

\[ c_{\text{ff}} = \text{ReLU}(c_{\text{anorm}}W_1 + b_1)W_2 + b_2 \]

\[ c_{\text{anorm}} = \frac{c_{\text{alayer}} - \text{mean}(c_{\text{alayer}})}{\text{std}(c_{\text{alayer}}) + \varepsilon} \]

\[ c_{\text{alayer}} = c_{\text{attn}} + \text{Dropout}(c_{\text{input}}) \]

\[ c_{\text{attn}} = \text{sum}([\alpha_1 a_{\text{input}}, \alpha_2 b_{\text{input}}]) \]

\[ \alpha = \text{softmax}(\bar{\alpha}) \]

\[ \bar{\alpha} = \begin{bmatrix} \frac{c_{\text{input}}^\top a_{\text{input}}}{\sqrt{d_k}} & \frac{c_{\text{input}}^\top b_{\text{input}}}{\sqrt{d_k}} \end{bmatrix} \]

\[ c_{\text{input}} = x_{34} + p_3 \]
Multi-headed attention

\[ \text{The} \times_{47} a_{\text{input}} + \text{Rock} \times_{30} b_{\text{input}} + \text{rules} \times_{34} c_{\text{input}} \]
Multi-headed attention

The Rock rules
x_{47} x_{30} x_{34}
\begin{align*}
1 \\
p_1 \\
2 \\
p_2 \\
3 \\
p_3
\end{align*}
Multi-headed attention

\[ c_{\text{attn1}} = \text{sum} \left( [\alpha_1(a_{\text{input}} W_1^V), \alpha_2(b_{\text{input}} W_1^V)] \right) \]

\[ \alpha = \text{softmax}(\tilde{\alpha}) \]

\[ \tilde{\alpha} = \left[ \left( \frac{c_{\text{input}} W_1^Q}{\sqrt{d_k}} \right)^\top a_{\text{input}} W_1^K, \left( \frac{c_{\text{input}} W_1^Q}{\sqrt{d_k}} \right)^\top b_{\text{input}} W_1^K \right] \]
Multi-headed attention

\[ c_{\text{attn1}} = \text{sum} \left( [\alpha_1(a_{\text{input}} W_1^V), \alpha_2(b_{\text{input}} W_1^V)] \right) \]

\[ \alpha = \text{softmax}(\tilde{\alpha}) \]

\[ \tilde{\alpha} = \left[ \frac{(c_{\text{input}} W_1^Q)^\top (a_{\text{input}} W_1^K)}{\sqrt{d_k}}, \frac{(c_{\text{input}} W_1^Q)^\top (b_{\text{input}} W_1^K)}{\sqrt{d_k}} \right] \]
Multi-headed attention

The Rock rules

xa_{attn1} + xb_{attn1} + xc_{attn1} 

h1

xa_{attn2} + xb_{attn2} + xc_{attn2} 

h2

xa_{input} + xb_{input} + xc_{input} 

x_{47} + p_1 + x_{30} + p_2 + x_{34} + p_3 

1 + 2 + 3

The Rock rules
Multi-headed attention

\[ c_{\text{attn2}} = \text{sum} \left( [\alpha_1(a_{\text{input}} W_2^V), \alpha_2(b_{\text{input}} W_2^V)] \right) \]

\[ \alpha = \text{softmax}(\tilde{\alpha}) \]

\[ \tilde{\alpha} = \left[ \frac{(c_{\text{input}} W_2^Q)^\top (a_{\text{input}} W_2^K)}{\sqrt{d_k}}, \frac{(c_{\text{input}} W_2^Q)^\top (b_{\text{input}} W_2^K)}{\sqrt{d_k}} \right] \]
Multi-headed attention

\[ c_{\text{attn2}} = \text{sum} \left( [\alpha_1(a_{\text{input}}W_2^V), \alpha_2(b_{\text{input}}W_2^V)] \right) \]

\[ \alpha = \text{softmax}(\tilde{\alpha}) \]

\[ \tilde{\alpha} = \left[ \frac{(c_{\text{input}}W_2^Q)^\top(a_{\text{input}}W_2^K)}{\sqrt{d_k}}, \frac{(c_{\text{input}}W_2^Q)^\top(b_{\text{input}}W_2^K)}{\sqrt{d_k}} \right] \]
Multi-headed attention
Multi-headed attention

\[ c_{\text{attn3}} = \text{sum} \left( \left[ \alpha_1 (a_{\text{input}} W^V_3), \alpha_2 (b_{\text{input}} W^V_3) \right] \right) \]

\[ \alpha = \text{softmax}(\tilde{\alpha}) \]

\[ \tilde{\alpha} = \frac{(c_{\text{input}} W^K_3)^\top (a_{\text{input}} W^Q_3)}{\sqrt{d_k}}, \frac{(c_{\text{input}} W^K_3)^\top (b_{\text{input}} W^Q_3)}{\sqrt{d_k}} \]

The Rock rules
\[
\begin{align*}
\text{input} & 1 \\
\text{input} & 2 \\
\text{input} & 3
\end{align*}
\]
Multi-headed attention

\[ c_{\text{attn3}} = \text{sum}\left(\left[\alpha_1(a_{\text{input}}W_3^V), \alpha_2(b_{\text{input}}W_3^V)\right]\right) \]

\[ \alpha = \text{softmax}(\tilde{\alpha}) \]

\[ \tilde{\alpha} = \left[ \frac{(c_{\text{input}}W_3^Q)^T(a_{\text{input}}W_3^K)}{\sqrt{d_k}}, \frac{(c_{\text{input}}W_3^Q)^T(b_{\text{input}}W_3^K)}{\sqrt{d_k}} \right] \]
Multi-headed attention

\[
c_{\text{attn}3} = \text{sum}\left( \left[ \alpha_1(a_{\text{input}}W^V_3), \alpha_2(b_{\text{input}}W^V_3) \right] \right)
\]

\[
\alpha = \text{softmax}(\tilde{\alpha})
\]

\[
\tilde{\alpha} = \left[ \frac{(c_{\text{input}}W^Q_3)^\top(a_{\text{input}}W^K_3)}{\sqrt{d_k}}, \frac{(c_{\text{input}}W^Q_3)^\top(b_{\text{input}}W^K_3)}{\sqrt{d_k}} \right]
\]
Repeated transformer blocks

Repeated 6 times, with $c_{\text{out}}$ serving as $c_{\text{input}}$ to each successive layer

Reminder that we also do multi-headed attention in each layer
The architecture diagram

Figure 1: The Transformer - model architecture.
The architecture diagram

The left side is repeated for every state in the encoder.

Figure 1: The Transformer - model architecture.
The architecture diagram

Each decoder state self-attends with all of its fellow decoder states and with all the encoder states.

The left side is repeated for every state in the encoder.

Figure 1: The Transformer - model architecture.
The architecture diagram

Each decoder state self-attends with all of its fellow decoder states and with all the encoder states.

The left side is repeated for every state in the encoder.

The right side is repeated for every decoder state, with outputs for each state that has them (all of them for dialogue and machine translation, only the final one for NLI).

Figure 1: The Transformer - model architecture.
The architecture diagram

Each decoder state self-attends with all of its fellow decoder states and with all the encoder states.

The left side is repeated for every state in the encoder.

The right side is repeated for every decoder state, with outputs for each state that has them (all of them for dialogue and machine translation, only the final one for NLI).

In the decoder, self-attention is limited to preceding words.

Figure 1: The Transformer - model architecture.
1. Overview: Resources and guiding insights

2. ELMo: Embeddings from Language Models

3. Transformers

4. BERT: Bidirectional Encoder Representations from Transformers

5. contextualreps.ipynb: Easy ways to bring ELMo and BERT into your project
Core model structure
Core model structure
Masked Language Modeling (MLM)
Transfer learning and fine-tuning

![Diagram of transfer learning and fine-tuning](image-url)
Binary sentence prediction pretraining

Positive: Actual sentence sequences

- [CLS] the man went to [MASK] store [SEP]
- he bought a gallon [MASK] milk [SEP]
- Label: IsNext

Negative: Randomly chosen second sentence

- [CLS] the man went to [MASK] store [SEP]
- penguin [MASK] are flight # # less birds [SEP]
- Label: NotNext
Tokenization and the BERT embedding space

```
In [1]: import random
   # In the code from https://github.com/google-research/bert
   from tokenization import FullTokenizer

In [2]: vocab_filename = "uncased_L-12_H-768_A-12/vocab.txt"

In [3]: with open(vocab_filename) as f:
   vocab = f.read().splitlines()

In [4]: len(vocab)
Out[4]: 30522

In [5]: random.sample(vocab, 5)
Out[5]: ['folder', '##gged', 'principles', 'moving', '##ceae']

In [6]: tokenizer = FullTokenizer(vocab_file=vocab_filename, do_lower_case=True)

In [7]: tokenizer.tokenize("This isn't too surprising!")
Out[7]: ['this', 'isn', '', 't', 'too', 'surprising', '']

In [8]: tokenizer.tokenize("Does BERT know Snuffleupagus?")
Out[8]: ['does', 'bert', 'know', 's', '##nu', '##ffle', '##up', '##ag', '##us', '?']
```
BERT model releases

Base

- Transformer layers: 12
- Hidden representations: 768 dimensions
- Attention heads: 12
- Total parameters: 110M

Large

- Transformer layers: 24
- Hidden representations: 1024 dimensions
- Attention heads: 16
- Total parameters: 340M

Limited to sequences of 512 tokens due to dimensionality of the positional embeddings.
1. Overview: Resources and guiding insights
2. ELMo: Embeddings from Language Models
3. Transformers
4. BERT: Bidirectional Encoder Representations from Transformers
5. contextualreps.ipynb: Easy ways to bring ELMo and BERT into your project
Guiding idea

- Your existing architecture can benefit from contextual representations.

- `contextualreps.ipynb` shows you how to bring in ELMo and BERT representations.

- You don’t get the benefits of fine-tuning (for that, you need to integrate more fully with ELMo and BERT code), but you still get a reliable boost!
Standard RNN dataset preparation

<table>
<thead>
<tr>
<th>Examples</th>
<th>[a, b, a]</th>
<th>[b, c]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indices</td>
<td>[1, 2, 1]</td>
<td>[2, 3]</td>
</tr>
<tr>
<td>Vectors</td>
<td>[−0.42 0.10 0.12], [−0.16 −0.21 0.29], [−0.42 0.10 0.12]</td>
<td>[−0.16 −0.21 0.29], [−0.26 0.31 0.37]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>
RNN contextual representation inputs

Examples

[a, b, a]
[b, c]

Vectors

\[
\begin{bmatrix}
-0.41 & -0.08 & 0.27 \\
0.17 & -0.22 & 0.78 \\
-0.46 & 0.24 & 0.12 \\
-0.02 & -0.56 & 0.11 \\
-0.45 & 0.43 & 0.32
\end{bmatrix}
\]
Code snippet: ELMo RNN inputs

In [1]: from allennlp.commands.elmo import ElmoEmbedder
   import os
   import sst
   from torch_rnn_classifier import TorchRNNClassifier

In [2]: SST_HOME = os.path.join("data", "trees")

In [3]: elmo = ElmoEmbedder()

In [4]: def elmo_phi(tree):
   vecs = elmo.embed_sentence(tree.leaves())
   return vecs.mean(axis=0)

In [5]: def fit_rnn(X, y):
   mod = TorchRNNClassifier(vocab=[], max_iter=50, use_embedding=False)
   mod.fit(X, y)
   return mod
Code snippet: ELMo RNN inputs

```
In [6]: elmo_experiment = sst.experiment(
    SST_HOME,
    elmo_phi,
    fit_rnn,
    train_reader=sst.train_reader,
    assess_reader=sst.dev_reader,
    vectorize=False
)
```

Finished epoch 50 of 50; error is 0.07357715629041195

<table>
<thead>
<tr>
<th></th>
<th>precision</th>
<th>recall</th>
<th>f1-score</th>
<th>support</th>
</tr>
</thead>
<tbody>
<tr>
<td>negative</td>
<td>0.700</td>
<td>0.687</td>
<td>0.693</td>
<td>428</td>
</tr>
<tr>
<td>neutral</td>
<td>0.353</td>
<td>0.284</td>
<td>0.315</td>
<td>229</td>
</tr>
<tr>
<td>positive</td>
<td>0.710</td>
<td>0.795</td>
<td>0.750</td>
<td>444</td>
</tr>
<tr>
<td>micro avg</td>
<td>0.647</td>
<td>0.647</td>
<td>0.647</td>
<td>1101</td>
</tr>
<tr>
<td>macro avg</td>
<td>0.588</td>
<td>0.589</td>
<td>0.586</td>
<td>1101</td>
</tr>
<tr>
<td>weighted avg</td>
<td>0.632</td>
<td>0.647</td>
<td>0.638</td>
<td>1101</td>
</tr>
</tbody>
</table>
Code snippet: BERT RNN inputs

```python
In [1]: # bert-serving-start -model_dir data/bert/uncased_L-12_H-768_A-12/ 
   # -pooling_strategy NONE -max_seq_len NONE -show_tokens_to_client
   from bert_serving.client import BertClient
   import os
   import sst
   from torch_rnn_classifier import TorchRNNClassifier

In [2]: SST_HOME = os.path.join("data", "trees")

In [3]: # Load the train and dev sets as strings, to let BERT tokenize:
   sst_train = [(" ".join(t.leaves()), label) for t, label in sst.train_reader(SST_HOME)]
   sst_dev = [(" ".join(t.leaves()), label) for t, label in sst.dev_reader(SST_HOME)]

In [4]: X_str_train, y_train = zip(*sst_train)
   X_str_dev, y_dev = zip(*sst_dev)

In [5]: X_str_dev, y_dev = zip(*sst_dev)
```
**Code snippet: BERT RNN inputs**

In [6]: bc = BertClient(check_length=False)

In [7]: # Prefetch all the BERT representations:
   X_bert_train = bc.encode(list(X_str_train), show_tokens=False)
   X_bert_dev = bc.encode(list(X_str_dev), show_tokens=False)

In [8]: # Create a look-up for fast featurization:
   BERT_LOOKUP = {}
   for sents, reps in ((X_str_train, X_bert_train), (X_str_dev, X_bert_dev)):
      assert len(sents) == len(reps)
      for s, rep in zip(sents, reps):
         BERT_LOOKUP[s] = rep
Code snippet: BERT RNN inputs

```
In [9]: def bert_phi(tree):
    s = " ".join(tree.leaves())
    return BERT_LOOKUP[s]

In [10]: def fit_rnn(X, y):
    mod = TorchRNNClassifier(vocab=[], max_iter=50, use_embedding=False)
    mod.fit(X, y)
    return mod

In [11]: bert_rnn_experiment = sst.experiment(
    SST_HOME,
    bert_phi,
    fit_rnn,
    train_reader=sst.train_reader,
    assess_reader=sst.dev_reader,
    vectorize=False)

Finished epoch 50 of 50; error is 2.6541710644960403

<table>
<thead>
<tr>
<th>precision</th>
<th>recall</th>
<th>f1-score</th>
<th>support</th>
</tr>
</thead>
<tbody>
<tr>
<td>negative</td>
<td>0.767</td>
<td>0.668</td>
<td>0.714</td>
</tr>
<tr>
<td>neutral</td>
<td>0.322</td>
<td>0.323</td>
<td>0.322</td>
</tr>
<tr>
<td>positive</td>
<td>0.737</td>
<td>0.827</td>
<td>0.779</td>
</tr>
<tr>
<td>micro avg</td>
<td>0.660</td>
<td>0.660</td>
<td>0.660</td>
</tr>
<tr>
<td>macro avg</td>
<td>0.608</td>
<td>0.606</td>
<td>0.605</td>
</tr>
<tr>
<td>weighted avg</td>
<td>0.662</td>
<td>0.660</td>
<td>0.659</td>
</tr>
</tbody>
</table>