Applied Natural Language Processing

Info 256
Lecture 15: Attention/BERT (Oct. 14, 2021)

David Bamman, UC Berkeley
How do we use word embeddings for document classification?
I loved the movie!
I loved the movie!

Iyyer et al. (2015), “Deep Unordered Composition Rivals Syntactic Methods for Text Classification” (ACL)
I loved the movie!
I loved the movie!
Attention

• Let’s incorporate structure (and parameters) into a network that captures which elements in the input we should be attending to (and which we can ignore).
Define $v$ to be a vector to be learned; think of it as an “important word” vector. The dot product here measures how similar each input vector is to that “important word” vector.
I loved the movie!

\[ v \in \mathbb{R}^H \]

\[
\begin{align*}
  r_1 &= v^\top x_1 \\
  r_2 &= v^\top x_2 \\
  r_3 &= v^\top x_3 \\
  r_4 &= v^\top x_4 \\
  r_5 &= v^\top x_5 
\end{align*}
\]
Convert \( r \) into a vector of normalized weights that sum to 1.

\[
a = \text{softmax}(r)
\]

\[
\begin{array}{ccccc}
a & 0 & 0.64 & 0.02 & 0.02 & 0.32 \\
r & -3.4 & 2.4 & -0.8 & -1.2 & 1.7 \\
\end{array}
\]

\[
\begin{align*}
r_1 &= v^T x_1 \\
r_2 &= v^T x_2 \\
r_3 &= v^T x_3 \\
r_4 &= v^T x_4 \\
r_5 &= v^T x_5 
\end{align*}
\]
I loved the movie!
Attention

• Lots of variations on attention:
  • Linear transformation of $x$ into before dotting with $v$
  • Non-linearities after each operation.
  • “Multi-head attention”: multiple $v$ vectors to capture different phenomena that can be attended to in the input.
  • Hierarchical attention (sentence representation with attention over words + document representation with attention over sentences).
Yang et al. (2016), “Hierarchical Attention Networks for Document Classification”
Attention gives us a normalized weight for every token in a sequence that tells us how important that word was for the prediction.

This can be useful for visualization.
BERT

• Transformer-based model (Vaswani et al. 2017) to predict masked word using bidirectional context + next sentence prediction.

• Generates multiple layers of representations for each token sensitive to its context of use.
Each token in the input starts out represented by token and position embeddings.
The value for time step $j$ at layer $i$ is the result of attention over all time steps in the previous layer $i-1$. The values for $e_{2,1}$, $e_{1,1}$, $e_{1,2}$, and $e_{1,3}$ are shown in the diagram.
The dog barked.
The dog barked
The dog barked
The dog barked.
The dog barked.
The dog barked.
At the end of this process, we have one representation for each layer for each token.

<table>
<thead>
<tr>
<th></th>
<th>-0.2</th>
<th>0.3</th>
<th>2.1</th>
<th>1.2</th>
<th>0.6</th>
<th></th>
<th>-1.8</th>
<th>-0.2</th>
<th>-2.4</th>
<th>-0.2</th>
<th>-0.1</th>
<th></th>
<th>-0.9</th>
<th>-1.5</th>
<th>-0.7</th>
<th>0.9</th>
<th>0.2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>e_3,1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e_3,2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e_3,3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e_2,1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e_2,2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e_2,3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e_1,1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e_1,2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e_1,3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The dog barked
WordPiece

• BERT uses WordPiece tokenization, which segments some morphological structure of tokens

• Vocabulary size: 30,000
• BERT also encodes each sentence by appending a special token to the beginning ([CLS]) and end ([SEP]) of each sequence.

• This helps provides a single token that can be optimized to represent the entire sequence (e.g., for document classification)
• We can represent the entire document with this *one* [CLS] vector
• Why does this work? When we design our network so that a classification decision relies entirely on that one vector and allow all the parameters of the network to be updated, the parameters of the model are optimized to compress all the relevant information into that one vector so that it can predict well (and minimize the loss).
We can represent the entire document with this *one* [CLS] vector.

Why does this work? When we design our network so that a classification decision relies entirely on that one vector and allow all the parameters of the network to be updated, the parameters of the model are optimized to compress all the relevant information into that one vector so that it can predict well (and minimize the loss).
BERT

- Deep layers (12 for BERT base, 24 for BERT large)
- Large representation sizes (768 per layer)
- Pretrained on English Wikipedia (2.5B words) and BooksCorpus (800M words)
## BERT

<table>
<thead>
<tr>
<th>L</th>
<th>H=128</th>
<th>H=256</th>
<th>H=512</th>
<th>H=768</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2/128 (BERT-Tiny)</td>
<td>2/256</td>
<td>2/512</td>
<td>2/768</td>
</tr>
<tr>
<td>4</td>
<td>4/128</td>
<td>4/256 (BERT-Mini)</td>
<td>4/512 (BERT-Small)</td>
<td>4/768</td>
</tr>
<tr>
<td>6</td>
<td>6/128</td>
<td>6/256</td>
<td>6/512</td>
<td>6/768</td>
</tr>
<tr>
<td>8</td>
<td>8/128</td>
<td>8/256</td>
<td>8/512 (BERT-Medium)</td>
<td>8/768</td>
</tr>
<tr>
<td>10</td>
<td>10/128</td>
<td>10/256</td>
<td>10/512</td>
<td>10/768</td>
</tr>
<tr>
<td>12</td>
<td>12/128</td>
<td>12/256</td>
<td>12/512</td>
<td>12/768 (BERT-Base)</td>
</tr>
</tbody>
</table>

https://github.com/google-research/bert
Pretrained models

Here is a partial list of some of the available pretrained models together with a short presentation of each model.

For the full list, refer to https://huggingface.co/models.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Model id</th>
<th>Details of the model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>bert-base-uncased</td>
<td>12-layer, 768-hidden, 12-heads, 110M parameters. Trained on lower-cased English text.</td>
</tr>
<tr>
<td></td>
<td>bert-large-uncased</td>
<td>24-layer, 1024-hidden, 16-heads, 336M parameters. Trained on lower-cased English text.</td>
</tr>
</tbody>
</table>
Lost in (language-specific) **BERT** models? We are here to help!

We currently have indexed 31 BERT-based models, 19 Languages and 28 Tasks.

We have a total of 178 entries in this table; we also show Multilingual Bert (mBERT) results if available! (see our paper)

Curious which BERT model is the best for named entity recognition in Italian? Just type "Italian NER" in the search bar!

<table>
<thead>
<tr>
<th>Language</th>
<th>Model</th>
<th>NLP Task</th>
<th>Dataset</th>
<th>Dataset-Domain</th>
<th>Measure</th>
<th>Performance</th>
<th>mBERT</th>
<th>Difference with mBERT</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>Arabert v1</td>
<td>SA</td>
<td>AJGT</td>
<td>twitter</td>
<td>Accuracy</td>
<td>83.8</td>
<td>83.6</td>
<td>10.2</td>
<td></td>
</tr>
<tr>
<td>Arabic</td>
<td>Arabert v1</td>
<td>SA</td>
<td>HARD</td>
<td>hotel reviews</td>
<td>Accuracy</td>
<td>96.1</td>
<td>95.7</td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td>Arabic</td>
<td>Arabert v1</td>
<td>SA</td>
<td>ASTD</td>
<td>twitter</td>
<td>Accuracy</td>
<td>92.6</td>
<td>80.1</td>
<td>12.5</td>
<td></td>
</tr>
<tr>
<td>Arabic</td>
<td>Arabert v1</td>
<td>SA</td>
<td>ArSenTD-Lev</td>
<td>twitter</td>
<td>Accuracy</td>
<td>59.4</td>
<td>51.0</td>
<td>8.4</td>
<td></td>
</tr>
</tbody>
</table>
Progress — Coreference resolution

- Classic (Lee et al. 2011, Björkelund and Farkas 2012, Durrett and Klein 2013)
- Neural (Lee et al. 2017)
- ELMO (Peters et al. 2018)
- BERT (Joshi et al. 2019)

Scores:
- Classic: 56.65, 58.26, 60.6
- Neural: 67.2
- ELMO: 70.4
- BERT: 76.9
Bertology

• Hewitt et al. 2019
• Tenney et al. 2019
• McCoy et al. 2019
• Liu et al. 2019
• Clark et al. 2019
• Goldberg 2019
• Michel et al. 2019

Code

Pre-trained models for BERT, Transformer-XL, ALBERT, RoBERTa, DistilBERT, GPT-2, etc. for English, French, “Multilingual”

https://huggingface.co
Even though BERT is mainly trained on a language modeling objective, it learns a lot about the structure of language — even without direct training data for specific linguistic tasks.

Probing experiments uncover what—and where (in what layers)—pretrained BERT encodes this information.
Activity

9.neural/BERTClassification

• Explore BERT for document classification using Google Colab