Contextualized Representations

Instructor: Yoav Artzi
Overview

• Motivation

• Context-dependent Representations with BERT

• Tokenization for BERT (and elsewhere)

• Common usage recipes

• Examples of less common uses:
  • Cross-modality representations
  • Generation evaluation with BERT
Motivation

• Word embeddings (e.g., word2vec, GloVe):
  • Learn a vector for each word type
  • Always the same vector
  • Problem: each vector likely mixes multiple senses, regardless of specific instance use
Motivation

• Instead of a single vector: learn a different vector for each use of a word type

• Challenge: how do we define the space of uses? Isn’t it too large?

• Solution: use sentence encoders to create a custom vector for every instance of a word
Several Approaches

- Central Word Prediction Objective (context2vec) [Melamud et al. 2016]
- Machine Translation Objective (CoVe) [McMann et al. 2017]
- Bi-directional Language Modeling Objective (ELMo) [Peters et al. 2018]
- Bidirectional Encoder Representations from Transformers (BERT) [Devlin et al. 2018]
- Robustly Optimized BERT (RoBERTa) [Liu et al. 2019]
- And more and more …
• Input: a sentence or a pair of sentences with a separator and subword representation

• Why do we need positional embedding?

<table>
<thead>
<tr>
<th>Input</th>
<th>[CLS]</th>
<th>my</th>
<th>dog</th>
<th>is</th>
<th>cute</th>
<th>[SEP]</th>
<th>he</th>
<th>likes</th>
<th>play</th>
<th>#ing</th>
<th>[SEP]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Token Embeddings</td>
<td>$E_{[CLS]}$</td>
<td>$E_{my}$</td>
<td>$E_{dog}$</td>
<td>$E_{is}$</td>
<td>$E_{cute}$</td>
<td>$E_{[SEP]}$</td>
<td>$E_{he}$</td>
<td>$E_{likes}$</td>
<td>$E_{play}$</td>
<td>$E_{#ing}$</td>
<td>$E_{[SEP]}$</td>
</tr>
<tr>
<td>Segment Embeddings</td>
<td>$E_A$</td>
<td>$E_A$</td>
<td>$E_A$</td>
<td>$E_A$</td>
<td>$E_A$</td>
<td>$E_B$</td>
<td>$E_B$</td>
<td>$E_B$</td>
<td>$E_B$</td>
<td>$E_B$</td>
<td>$E_B$</td>
</tr>
<tr>
<td>Position Embeddings</td>
<td>$E_0$</td>
<td>$E_1$</td>
<td>$E_2$</td>
<td>$E_3$</td>
<td>$E_4$</td>
<td>$E_5$</td>
<td>$E_6$</td>
<td>$E_7$</td>
<td>$E_8$</td>
<td>$E_9$</td>
<td>$E_{10}$</td>
</tr>
</tbody>
</table>

[figure from Devlin et al. 2018]
BERT

- Model: multi-layer self-attention (Transformer)

[figure from Devlin et al. 2018]
BERT

- Model: multi-layer self-attention (Transformer)

- BERT Base: 12 layers, 768-dim per word-piece token, 12 heads. Total parameters = 110M

- BERT Large: 24 layers, 1024-dim per word-piece token, 16 heads. Total parameters = 340M

- RoBERTa: same model, much more data (160GB of data instead of 16GB)
Training BERT

- Key idea: self-supervised objectives with raw text
- Two objectives: masked language modeling and next sentence prediction
- Data: BookCorpus + English Wikipedia
- Later development with RoBERTa:
  - Much more data
  - Removed the next sentence prediction objective
  - Dynamic masking
Masked Language Modeling

• Similar to predicting the next word for language modeling, but adapted for non-directional self-attention

• The BERT recipe: mask and predict 15% of the tokens

  • For 80% (of 15%) replace with the input token with [MASK]

  • For 10%, replace with a random token

  • For 10%, keep the same
Next Sentence Prediction

- Input: [CLS] Text chunk 1 [SEP] Text chunk 2

- Create data: 50% of the time, take the true next chunk of text, 50% of the time take a random other chunk

- Predict whether the next chunk is the “true” next

Sub-word Tokenization

- BERT uses Word Piece tokenization

- Related models (e.g., for MT, language modeling, etc) use either Word Piece or Byte Pair Encoding tokenization

- Advantage: no unknown words problem

- Package: 
  https://github.com/huggingface/tokenizers
Byte Pair Encoding (BPE) Tokenization

1. Start with every individual byte (basically character) as its own token

2. Count bigram token cooccurrences over tokens (potentially: weight according to corpus frequencies)

3. Merge the most frequent pair of adjacent tokens to create a new token
   - Vocabulary size is controlled by the number of merges
   - With ~8000 tokens we get many whole words in English

[Sennrich et al. (2016)]
Word Piece Tokenization

1. Initialize with tokens for all characters

2. While vocabulary size is below the target size:
   1. Build a language model over the corpus (e.g., unigram language model)
   2. Merge pieces that lead to highest improvement in language model perplexity

- Need to choose a language model that will make the process tractable
- Often a unigram language model (e.g., SentencePiece library)
- Particularly suitable for machine translation

[Schuster and Nakajima (2012), Wu et al. (2016), Kudo and Richardson (2018)]
Where to get BERT?

- The Transformers library: [https://github.com/huggingface/transformers](https://github.com/huggingface/transformers)
- Provides state-of-the-art implementation of many models, including BERT and RoBERTa
- Including pre-trained models
Using BERT

• Use the pre-trained model as the first “layer” of your final model

• Train with fine-tuning using your supervised data

• Fine-tuning recipe: 1-3 epochs, batch size 2-32, learning rate 2e-5 - 5e-5

  • Large changes to weights in top layers (particularly in last layer to route the right information to [CLS])

  • Smaller changes to weights lower down in the transformer

  • Small learning rate and short fine-tuning schedule mean weights don’t change much

• More complex recipes exist, but often not necessary (see Zhang et al. 2021 for study of stability and good practices)
Sentence Classification with BERT

- CLS representation is used to provide classification decision.

- Example tasks:
  - Sentiment classification
  - Linguistic acceptability
  - Text categorization

[figure from Devlin et al. 2018]
Sentence-pair Classification with BERT

- Feed both sentences, and CLS representation used for classification

- Example tasks:
  - Textual entailment
  - Question paraphrase detection
  - Question-answering pair classification
  - Semantic textual similarity
  - Multiple choice question answering

[figure from Devlin et al. 2018]
Tagging with BERT

- Can do for a single sentence or a pair
- Tag each word piece
- Example tasks: span-based question answering, name-entity recognition, POS tagging
Results

• Fine-tuned BERT (and its variants) outperforms known methods on most NLP supervised tasks

• The larger models perform better, but even the small BERT performs better than prior methods

• Variants quickly outperformed human performance on several tasks, including span-based question answering — but what does this mean beyond the benchmarks is less clear

• Started an arms race (between industry labs) on bigger and bigger models
Hard to do with BERT

- BERT cannot generate text (at least not in an obvious way)
- Not an autoregressive model, can do weird things like stick a [MASK] at the end of a string, fill in the mask, and repeat
- Masked language models are intended to be used primarily for “analysis” tasks
What does BERT Learn?

- A lot of recent work studying this problem
- Some very interesting results
- But, it’s not completely clear how to interpret them
What does BERT Learn?

• Try to solve different linguistic tasks given each level, without fine-tuning

• Goal: see what information each new level adds

• Method: try to solve different tasks using mixing weights on levels

• Each task classifier takes a single mixed hidden representation $h_{i,\tau}$ or a pair of representations

$$i : \text{token index}$$

$$K : \text{number of levels}$$

$$\tau : \text{task}$$

$$\gamma_{\tau} : \text{task parameter}$$

$$a_{\tau} : \text{mixing parameters}$$

$$s_{\tau} = \text{softmax}(a_{\tau})$$

$$h_{i,\tau} = \gamma_{\tau} \sum_{k=0}^{K} s_{\tau}^{k} h_{i}^{k}$$

[Tenney et al. 2019]
What does BERT Learn?

- Each plot shows a task.

- Plots show $s^k_T$ weights magnitude in blue, and the number of self-attention levels.

- The performance delta when adding this layer is in purple.

- Largely: higher level semantic tasks happen in later levels.

[figure from Tenney et al. (2019)]
Vision-language Reasoning

- Goal: pre-trained representations for language and vision, where the input is a sentence and image

- Self-attention in BERT allows attending between two sentences

- How can we extend that to a sentence paired with an image?
Vision-language Reasoning

• Solution: pre-process the image to extract bounding boxes around objects

• Now the image is an unordered list of discrete objects

• Objectives: masked language model + masked region modeling + image-text matching

[Li et al. 2019; Tan and Bansal 2019; Chen et al. 2019; and several other simultaneous papers]
Vision-language Reasoning

[figure from Chen et al. 2019]
Results

- Similar trend to what we observe with BERT
- State of the art on 13 vision+language benchmarks
- Similar to BERT, there larger is better

<table>
<thead>
<tr>
<th>Tasks</th>
<th>SOTA</th>
<th>VISLANT</th>
<th>VLBERT</th>
<th>UNICODER-VL</th>
<th>VISUALBERT</th>
<th>LXMERT</th>
<th>UNITER BASE</th>
<th>UNITER LARGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>VQA</td>
<td>test-dev</td>
<td>72.27</td>
<td>73.33</td>
<td>-</td>
<td>-</td>
<td>73.42</td>
<td>74.54</td>
<td>-</td>
</tr>
<tr>
<td>VQA</td>
<td>test-std</td>
<td>72.32</td>
<td>73.82</td>
<td>74.32</td>
<td>-</td>
<td>74.62</td>
<td>75.42</td>
<td>-</td>
</tr>
<tr>
<td>VQA</td>
<td>QA→R</td>
<td>72.60</td>
<td>73.30</td>
<td>74.00</td>
<td>-</td>
<td>71.60</td>
<td>-</td>
<td>72.60</td>
</tr>
<tr>
<td>VQA</td>
<td>QA→A</td>
<td>75.70</td>
<td>76.40</td>
<td>78.40</td>
<td>-</td>
<td>73.20</td>
<td>-</td>
<td>77.20</td>
</tr>
<tr>
<td>VQA</td>
<td>QA→AR</td>
<td>55.00</td>
<td>54.80</td>
<td>55.50</td>
<td>-</td>
<td>52.40</td>
<td>-</td>
<td>58.20</td>
</tr>
<tr>
<td>NLVR²</td>
<td>dev</td>
<td>54.80</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>67.40</td>
<td>74.90</td>
<td>-</td>
</tr>
<tr>
<td>NLVR²</td>
<td>test-P</td>
<td>53.50</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>67.00</td>
<td>74.50</td>
<td>-</td>
</tr>
<tr>
<td>SNLI-VE</td>
<td>val</td>
<td>71.56</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SNLI-VE</td>
<td>test</td>
<td>71.16</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ZS IR</td>
<td>R@1</td>
<td>-</td>
<td>31.86</td>
<td>-</td>
<td>42.40</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ZS IR</td>
<td>R@5</td>
<td>-</td>
<td>61.12</td>
<td>-</td>
<td>71.80</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ZS IR</td>
<td>R@10</td>
<td>-</td>
<td>72.80</td>
<td>-</td>
<td>81.50</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IR</td>
<td>R@1</td>
<td>48.60</td>
<td>58.20</td>
<td>-</td>
<td>68.30</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IR</td>
<td>R@5</td>
<td>77.70</td>
<td>84.90</td>
<td>-</td>
<td>90.30</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IR</td>
<td>R@10</td>
<td>85.20</td>
<td>91.52</td>
<td>-</td>
<td>94.60</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IR</td>
<td>R@1</td>
<td>38.60</td>
<td>-</td>
<td>-</td>
<td>44.50</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IR</td>
<td>R@5</td>
<td>69.30</td>
<td>-</td>
<td>-</td>
<td>74.40</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IR</td>
<td>R@10</td>
<td>80.40</td>
<td>-</td>
<td>-</td>
<td>84.00</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ZS TR</td>
<td>R@1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>61.60</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ZS TR</td>
<td>R@5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>84.80</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ZS TR</td>
<td>R@10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>90.10</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TR</td>
<td>R@1</td>
<td>67.90</td>
<td>-</td>
<td>-</td>
<td>82.30</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TR</td>
<td>R@5</td>
<td>90.30</td>
<td>-</td>
<td>-</td>
<td>95.10</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TR</td>
<td>R@10</td>
<td>95.80</td>
<td>-</td>
<td>-</td>
<td>97.80</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TR</td>
<td>R@1</td>
<td>50.40</td>
<td>-</td>
<td>-</td>
<td>59.60</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TR</td>
<td>R@5</td>
<td>82.20</td>
<td>-</td>
<td>-</td>
<td>85.10</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TR</td>
<td>R@10</td>
<td>90.00</td>
<td>-</td>
<td>-</td>
<td>91.80</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ref-COCO</td>
<td>val</td>
<td>87.51</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>91.64</td>
</tr>
<tr>
<td>Ref-COCO</td>
<td>testA</td>
<td>89.02</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>92.26</td>
</tr>
<tr>
<td>Ref-COCO</td>
<td>testB</td>
<td>87.05</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>90.46</td>
</tr>
<tr>
<td>Ref-COCO</td>
<td>vald</td>
<td>77.48</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>81.24</td>
</tr>
<tr>
<td>Ref-COCO</td>
<td>testAd</td>
<td>83.37</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>86.48</td>
</tr>
<tr>
<td>Ref-COCO</td>
<td>testBd</td>
<td>70.32</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>73.94</td>
</tr>
<tr>
<td>Ref-COCO+</td>
<td>val</td>
<td>75.38</td>
<td>-</td>
<td>78.44</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>82.84</td>
</tr>
<tr>
<td>Ref-COCO+</td>
<td>testA</td>
<td>80.04</td>
<td>-</td>
<td>81.30</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>85.70</td>
</tr>
<tr>
<td>Ref-COCO+</td>
<td>testB</td>
<td>69.30</td>
<td>-</td>
<td>71.18</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>78.11</td>
</tr>
<tr>
<td>Ref-COCOg</td>
<td>val</td>
<td>68.19</td>
<td>-</td>
<td>-</td>
<td>71.34</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ref-COCOg</td>
<td>test</td>
<td>75.97</td>
<td>-</td>
<td>78.52</td>
<td>77.59</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ref-COCOg</td>
<td>vald</td>
<td>57.52</td>
<td>-</td>
<td>62.61</td>
<td>60.57</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ref-COCOg</td>
<td>testd</td>
<td>68.22</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>74.31</td>
</tr>
<tr>
<td>Ref-COCOg</td>
<td>vald</td>
<td>69.46</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>74.51</td>
</tr>
</tbody>
</table>
Text Generation Evaluation with BERT

- Can we use BERT to evaluate language generation? Such as MT, paraphrase, caption generation, etc.
- Input is two sentences: a reference and a system output
- Output: a score that tells us how similar they are
Text Generation Evaluation with BERT

- How do we do it usually? Bleu

- Bleu matches n-grams between the reference and the candidate

- When does this fail?

<table>
<thead>
<tr>
<th>Reference</th>
<th>the weather <em>is</em> cold <em>today</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>Candidate</td>
<td><em>it</em> <em>is</em> freezing <em>today</em></td>
</tr>
</tbody>
</table>
Text Generation Evaluation with BERT

• How do we do it usually? Bleu

• Bleu matches n-grams between the reference and the candidate

• When does this fail?

• Sensitive to exact phrasing and word choices

• This can bring about false negatives
BERTScore

- Instead of string matching, like in Bleu
- Use BERT embedding to compute similarity

Reference

the weather is cold today

Candidate

it is freezing today

Pairwise cosine similarity
## Matching

- Compute similarity between all possible pairs
- Build a similarity matrix

<table>
<thead>
<tr>
<th>Reference</th>
<th>Candidate</th>
<th>it</th>
<th>is</th>
<th>freezing</th>
<th>today</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>-0.713</td>
<td>0.597</td>
<td>0.428</td>
<td>0.408</td>
<td></td>
</tr>
<tr>
<td>weather</td>
<td>-0.462</td>
<td>0.393</td>
<td>0.515</td>
<td>0.326</td>
<td></td>
</tr>
<tr>
<td>is</td>
<td>-0.635</td>
<td>0.858</td>
<td>0.441</td>
<td>0.441</td>
<td></td>
</tr>
<tr>
<td>cold</td>
<td>-0.479</td>
<td>0.454</td>
<td>0.796</td>
<td>0.343</td>
<td></td>
</tr>
<tr>
<td>today</td>
<td>-0.347</td>
<td>0.361</td>
<td>0.307</td>
<td>0.913</td>
<td></td>
</tr>
</tbody>
</table>
**Greedy Matching**

<table>
<thead>
<tr>
<th>Candidate</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>0.713</td>
<td>0.713</td>
</tr>
<tr>
<td>weather</td>
<td>0.462</td>
<td>0.462</td>
</tr>
<tr>
<td>is</td>
<td>0.635</td>
<td>0.635</td>
</tr>
<tr>
<td>cold</td>
<td>0.479</td>
<td>0.479</td>
</tr>
<tr>
<td>today</td>
<td>0.347</td>
<td>0.347</td>
</tr>
</tbody>
</table>

**Match words in candidate to reference**

<table>
<thead>
<tr>
<th>Reference</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>it</td>
<td>0.597</td>
<td>0.597</td>
</tr>
<tr>
<td>is</td>
<td>0.515</td>
<td>0.515</td>
</tr>
<tr>
<td>freezing</td>
<td>0.441</td>
<td>0.441</td>
</tr>
<tr>
<td>today</td>
<td>0.796</td>
<td>0.796</td>
</tr>
</tbody>
</table>

**Match words in reference to candidate**
Greedy Matching: Aggregate

Precision

Recall

\[
F1 = 2 \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}
\]
Evaluating Evaluation

- Collect human judgements
- Measure correlations with your metric

Reference: *The weather is cold today.*
Candidate: *It is freezing today.*

Reference: *The garden is nice.*
Candidate: *The garden was pretty.*

Reference: *I like apples very much.*
Candidate: *I love apples.*

<table>
<thead>
<tr>
<th>Human</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.85</td>
<td>0.77</td>
</tr>
<tr>
<td>0.71</td>
<td>0.77</td>
</tr>
<tr>
<td>0.79</td>
<td>0.80</td>
</tr>
</tbody>
</table>
Evaluating Evaluation

Correlation Study

<table>
<thead>
<tr>
<th>Language Pair</th>
<th>BLEU</th>
<th>ITER</th>
<th>YiSi-1</th>
<th>RUSE</th>
<th>BertScore F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Czech-English</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>German-English</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>English-Czech</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>English-German</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Correlation

Language Pair

Czech-English
German-English
English-Czech
English-German