Contextualized embeddings

• Models for learning static embeddings like word2vec and Glove learn a single representation for a word type.
Types and tokens

- Type: bears

- Tokens:
  - The bears ate the honey
  - We spotted the bears from the highway
  - Yosemite has brown bears
  - The chicago bears didn’t make the playoffs

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>1.4</td>
<td>-2.7</td>
<td>0.3</td>
</tr>
</tbody>
</table>
Contextualized word representations

- Big idea: transform the representation of a token in a sentence (e.g., from a static word embedding) to be sensitive to its local context in a sentence and trainable to be optimized for a specific NLP task.
Language Model

• Language models provide us with a way to quantify the likelihood of a sequence — i.e., plausible sentences.
To see great Pompey passe the streets of Rome:
And when you saw his Chariot but appeare,
Haue you not made an Universall shout,
That Tyber trembled vnderneath her bankes
To heare the replication of your sounds,
Made in her Concaue Shores?

• to see great Pompey passe the streets of Rome:
Machine translation

- Fidelity (to source text)
- Fluency (of the translation)
natural lan

natural language processing
natural language understanding
natural language processing with python
natural language generation
Speech Recognition

- 'Scuse me while I kiss the sky.
- 'Scuse me while I kiss this guy
- 'Scuse me while I kiss this fly.
- 'Scuse me while my biscuits fry
Classical (causal) language model

Consider only the left context to predict the next word (i.e., the final word in a sequence is masked)

$$P (w_t \mid w_1, \ldots, w_{t-1})$$
Masked language model

Use any context (left or right) to predict a masked word

\[ P(w_t \mid w_{-t}) \]
**ELMo**

Stacked BiRNN trained to predict next word in language modeling task

Peters et al. 2018

---

**BERT**

Transformer-based model to predict masked word using bidirectional context + next sentence prediction.

Devlin et al. 2019
ELMo


• Big idea: transform the representation of a word (e.g., from a static word embedding) to be sensitive to its local context in a sentence and optimized for a specific NLP task.

• Output = word representations that can be plugged into just about any architecture a word embedding can be used.
Recurrent neural network language model

<table>
<thead>
<tr>
<th>the</th>
<th>a</th>
<th>like</th>
<th>love</th>
<th>go</th>
<th>home</th>
<th>movie</th>
<th>dinner</th>
<th>.</th>
<th>&lt;STOP&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.15</td>
<td>0.13</td>
<td>0.04</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.001</td>
<td>0.0001</td>
</tr>
</tbody>
</table>

The table above shows the attention weights for each word in the sentence "I love going home for dinner."

The circular diagram represents the recurrent neural network, with weights indicated at the connections.
I loved Recurrent neural network language model.
I loved the Recurrent neural network language model.
Recurrence neural network language model

I loved the movie.

<table>
<thead>
<tr>
<th>the</th>
<th>a</th>
<th>like</th>
<th>love</th>
<th>go</th>
<th>home</th>
<th>movie</th>
<th>dinner</th>
<th>.</th>
<th>&lt;STOP&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.98</td>
<td>0.0001</td>
</tr>
</tbody>
</table>
I loved the movie.

Recurrent neural network language model

<table>
<thead>
<tr>
<th>the</th>
<th>a</th>
<th>like</th>
<th>love</th>
<th>go</th>
<th>home</th>
<th>movie</th>
<th>dinner</th>
<th>.</th>
<th>&lt;STOP&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0001</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.001</td>
<td>0.98</td>
</tr>
</tbody>
</table>

2.7  3.1  -1.4  -2.3  0.7  -0.7  -0.8  -1.3  -0.2  -0.9  2.3  1.5  1.1  1.4  1.3  -0.9  -1.5  -0.7  0.9  0.2  -0.1  -0.7  -1.6  0.2  0.6
Recurrent neural network LM

\[ h \in \mathbb{R}^d \]
\[ W \in \mathbb{R}^{d \times V} \]
\[ o = \text{softmax} (h^T W) \]
I loved the movie!
I loved the movie!

Bidirectional RNN

backward RNN
I loved the movie!

Bidirectional RNN
• Train a bidirectional RNN language model with L layers on a bunch of text.

• Learn parameters to combine the RNN output across all layers for each word in a sentence for a specific task (NER, semantic role labeling, question answering etc.). Large improvements over SOTA for lots of NLP problems.
<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>85.8</td>
</tr>
<tr>
<td>SNLI</td>
<td>Chen et al. (2017)</td>
<td>88.6</td>
<td>88.0</td>
<td>88.7 ± 0.17</td>
</tr>
<tr>
<td>SRL</td>
<td>He et al. (2017)</td>
<td>81.7</td>
<td>81.4</td>
<td>84.6</td>
</tr>
<tr>
<td>Coref</td>
<td>Lee et al. (2017)</td>
<td>67.2</td>
<td>67.2</td>
<td>70.4</td>
</tr>
<tr>
<td>NER</td>
<td>Peters et al. (2017)</td>
<td>91.93 ± 0.19</td>
<td>90.15</td>
<td>92.22 ± 0.10</td>
</tr>
<tr>
<td>SST-5</td>
<td>McCann et al. (2017)</td>
<td>53.7</td>
<td>51.4</td>
<td>54.7 ± 0.5</td>
</tr>
</tbody>
</table>

Table 1: Test set comparison of ELMo enhanced neural models with state-of-the-art single model baselines across six benchmark NLP tasks. The performance metric varies across tasks – accuracy for SNLI and SST-5; F₁ for SQuAD, SRL and NER; average F₁ for Coref. Due to the small test sizes for NER and SST-5, we report the mean and standard deviation across five runs with different random seeds. The “increase” column lists both the absolute and relative improvements over our baseline.
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 dog barked

- $e_{2,1}$
- $e_{1,1}$
- $e_{1,2}$
- $e_{1,3}$
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.
• 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).
• Learn the parameters of this model with two objectives:
  • Masked language modeling
  • Next sentence prediction
Masked LM

• Mask one word from the input and try to predict that word as the output.

• More powerful than an RNN LM (or even a BiRNN LM) since it can reason about context on both sides of the word being predicted.

• A BiRNN models context on both sides, but each RNN only has access to information from one direction.
The [MASKED] bark #ed [SEP]
The [MASKED] bark #ed
The dog [MASKED] bark #ed
Next sentence prediction

• For a pair of sentences, predict from [CLS] representation whether they appeared sequentially in the training data:

  ➕ [CLS] The dog bark #ed [SEP] He was hungry
  ➖ [CLS] The dog bark #ed [SEP] Paris is in France
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)
Yosemite has brown bears.

We saw a moose in Alaska.

Da bears lost again!

Go pack go!
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
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
Probing

- 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.

Tenney et al. (2019), "BERT RedisCOVERS the Classical NLP Pipeline"
Activity

BERT.ipynb

• Explore BERT through the huggingface transformers library and use it to find contextual nearest neighbors.

• What else could we use contextual embeddings for?