Natural Language Processing with Deep Learning
CS224N/Ling284

Christopher Manning
Lecture 14: More on Contextual Word Representations and Pretraining
Thanks for your Feedback!

How's the pace of lectures so far?
458 responses

- Too slow: 79.9%
- Just right: 13.8%
- Too fast: 6.3%

How challenging was Assignment 1?
458 responses

- Easy: 72.7%
- Medium: 23.8%
- Hard: 3.5%

How challenging was Assignment 3?
458 responses

- Easy: 81.9%
- Medium: 11.6%
- Hard: 6.5%
### Thanks for your Feedback!

What do you most want to learn about in the remaining lectures?

<table>
<thead>
<tr>
<th>Feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>More cutting-edge research directions</td>
</tr>
<tr>
<td>BERT embeddings</td>
</tr>
<tr>
<td>a survey about what the different NLP techniques beyond what we've learned</td>
</tr>
<tr>
<td>I want to dive further into cutting edge NLP techniques like transformers</td>
</tr>
<tr>
<td>transformers, bert, more state-of-the-art models in nlp</td>
</tr>
<tr>
<td>BERT, GPT-2 and derivative models</td>
</tr>
<tr>
<td>How different techniques/models tackle various linguistic challenges/complexities</td>
</tr>
<tr>
<td>Image captioning</td>
</tr>
<tr>
<td>GPT-2?</td>
</tr>
<tr>
<td>Program synthesis applications from natural language</td>
</tr>
<tr>
<td>I think it would be really helpful to understand how to go about building a model from scratch and understanding what techniques to leverage in certain problems.</td>
</tr>
<tr>
<td>BERT</td>
</tr>
<tr>
<td>I am interested in learning about different contexts these models can be applied to</td>
</tr>
<tr>
<td>Guest lecture</td>
</tr>
</tbody>
</table>
Announcements

• Assignment 5 is due today

• We’re handing back project proposal feedback today

• Project milestone – due in 12 days...
Lecture Plan

Lecture 14: Contextual Word Representations and Pretraining

1. Reflections on word representations (5 mins)
2. Pre-ELMo and ELMO (20 mins)
3. ULMfit and onward (10 mins)
4. Transformer architectures (20 mins)
5. BERT (15 mins)
6. How’s the weather? (5 mins)
1. Representations for a word

• Originally, we basically had one representation of words:
  • The word vectors that we learned about at the beginning
    • Word2vec, GloVe, fastText

• These have two problems:
  • Always the same representation for a **word type** regardless of the context in which a **word token** occurs
    • We might want very fine-grained word sense disambiguation
  • We just have one representation for a word, but words have different **aspects**, including semantics, syntactic behavior, and register/connotations
Did we all along have a solution to this problem?

- In an NLM, we immediately stuck word vectors (perhaps only trained on the corpus) through LSTM layers.
- Those LSTM layers are trained to predict the next word.
- But those language models are producing context-specific word representations at each position!
# Context-free vs. contextual similarity

<table>
<thead>
<tr>
<th>Model</th>
<th>Source</th>
<th>Nearest Neighbor(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GloVe</td>
<td>play</td>
<td>playing, game, games, played, players, plays, player, Play, football, multiplayer</td>
</tr>
<tr>
<td>BiLM</td>
<td>Chico Ruiz made a spectacular play on Alusik’s grounder {...}</td>
<td>Kieffer, the only junior in the group, was commended for his ability to hit in the clutch, as well as his all-round excellent play.</td>
</tr>
<tr>
<td></td>
<td>Olivia De Havilland signed to do a Broadway play for Garson {...}</td>
<td>{...} they were actors who had been handed fat roles in a successful play, and had talent enough to fill the roles competently, with nice understatement.</td>
</tr>
</tbody>
</table>

From Peters et al. 2018 Deep contextualized word representations (ELMo paper)
2. Pre-ELMo and ELMo

• Why don’t we do semi-supervised approach where we train NLM sequence model on large unlabeled corpus, rather than just word vectors and use as pretraining for sequence model

• Idea: Want meaning of word in context, but standardly learn task RNN only on small task-labeled data (e.g., NER)

• Same general idea of transferring NLM knowledge
• Here applied to text classification
**Tag LM (Peters et al. 2017)**

**Step 1:** Pretrain word embeddings and language model.

**Step 2:** Prepare word embedding and LM embedding for each token in the input sequence.

**Step 3:** Use both word embeddings and LM embeddings in the sequence tagging model.

Two representations of the word “York”

output sequence

unlabeled data
Tag LM

\[ h_{k,1} = [\overrightarrow{h}_{k,1}; \overleftarrow{h}_{k,1}; h_{k,1}^{LM}] \]
Named Entity Recognition (NER)

- **Find** and **classify** names in text, for example:

  - The decision by the independent MP Andrew Wilkie to withdraw his support for the minority Labor government sounded dramatic but it should not further threaten its stability. When, after the 2010 election, Wilkie, Rob Oakeshott, Tony Windsor and the Greens agreed to support Labor, they gave just two guarantees: confidence and supply.
## CoNLL 2003 Named Entity Recognition (en news testb)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Year</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>TagLM Peters</td>
<td>LSTM BiLM in BiLSTM tagger</td>
<td>2017</td>
<td>91.93</td>
</tr>
<tr>
<td>Ma + Hovy</td>
<td>BiLSTM + char CNN + CRF layer</td>
<td>2016</td>
<td>91.21</td>
</tr>
<tr>
<td>Tagger Peters</td>
<td>BiLSTM + char CNN + CRF layer</td>
<td>2017</td>
<td>90.87</td>
</tr>
<tr>
<td>Ratinov + Roth</td>
<td>Categorical CRF+Wikipedia+word cls</td>
<td>2009</td>
<td>90.80</td>
</tr>
<tr>
<td>Finkel et al.</td>
<td>Categorical feature CRF</td>
<td>2005</td>
<td>86.86</td>
</tr>
<tr>
<td>IBM Florian</td>
<td>Linear/softmax/TBL/HMM ensemble, gazettes++</td>
<td>2003</td>
<td>88.76</td>
</tr>
<tr>
<td>Stanford Klein</td>
<td>MEMM softmax markov model</td>
<td>2003</td>
<td>86.07</td>
</tr>
</tbody>
</table>
Peters et al. (2017): TagLM – “Pre-ELMo”

Language model is trained on 800 million training words of “Billion word benchmark”

Language model observations
• An LM trained on supervised data does not help
• Having a bidirectional LM helps over only forward, by about 0.2
• Having a huge LM design (ppl 30) helps over a smaller model (ppl 48) by about 0.3

Task-specific BiLSTM observations
• Using just the LM embeddings to predict isn’t great: 88.17 F1
  • Well below just using an BiLSTM tagger on labeled data
Peters et al. (2018): ELMo: Embeddings from Language Models


• Initial breakout version of **word token vectors** or **contextual word vectors**

• Learn word token vectors using long contexts not context windows (here, whole sentence, could be longer)

• Learn a deep Bi-NLM and use all its layers in prediction
Peters et al. (2018): ELMo: Embeddings from Language Models

- Train a bidirectional LM
- Aim at performant but not overly large LM:
  - Use 2 biLSTM layers
  - Use character CNN to build initial word representation (only)
    - 2048 char n-gram filters and 2 highway layers, 512 dim projection
  - Use 4096 dim hidden/cell LSTM states with 512 dim projections to next input
  - Use a residual connection
  - Tie parameters of token input and output (softmax) and tie these between forward and backward LMs
Peters et al. (2018): ELMo: Embeddings from Language Models

- ELMo learns task-specific combo of biLM layer representations
- This is an innovation that improves on just using top layer of LSTM stack

\[
R_k = \{ x_k^{LM}, h_{k,j}^{LM}, \overleftarrow{h}_{k,j}^{LM} \mid j = 1, \ldots, L \} \\
= \{ h_{k,j}^{LM} \mid j = 0, \ldots, L \},
\]

\[
\text{ELMo}_k^{\text{task}} = E(R_k; \Theta^{\text{task}}) = \gamma^{\text{task}} \sum_{j=0}^{L} s_j^{\text{task}} h_{k,j}^{LM}
\]

- \( \gamma^{\text{task}} \) scales overall usefulness of ELMo to task;
- \( s^{\text{task}} \) are softmax-normalized mixture model weights
Peters et al. (2018): ELMo: Use with a task

• First run biLM to get representations for each word
• Then let (whatever) end-task model use them
  • Freeze weights of ELMo for purposes of supervised model
  • Concatenate ELMo weights into task-specific model
    • Details depend on task
      • Concatenating into intermediate layer as for TagLM is typical
      • Can provide ELMo representations again when producing outputs, as in a question answering system
ELMo used in an NER tagger

Breakout version of deep contextual word vectors

ELMo representation: A deep bidirectional neural LM

Use learned, task-weighted average of (2) hidden layers

\[ h_{k,1} = [\overrightarrow{h}_{k,1}; \overleftarrow{h}_{k,1}; h^{LM}_k] \]
*Kitaev and Klein, ACL 2018  (see also Joshi et al., ACL 2018)
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Year</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELMo</td>
<td>ELMo in BiLSTM</td>
<td>2018</td>
<td>92.22</td>
</tr>
<tr>
<td>TagLM Peters</td>
<td>LSTM BiLM in BiLSTM tagger</td>
<td>2017</td>
<td>91.93</td>
</tr>
<tr>
<td>Ma + Hovy</td>
<td>BiLSTM + char CNN + CRF layer</td>
<td>2016</td>
<td>91.21</td>
</tr>
<tr>
<td>Tagger Peters</td>
<td>BiLSTM + char CNN + CRF layer</td>
<td>2017</td>
<td>90.87</td>
</tr>
<tr>
<td>Ratinov + Roth</td>
<td>Categorical CRF+WikiPedia+word cls</td>
<td>2009</td>
<td>90.80</td>
</tr>
<tr>
<td>Finkel et al.</td>
<td>Categorical feature CRF</td>
<td>2005</td>
<td>86.86</td>
</tr>
<tr>
<td>IBM Florian</td>
<td>Linear/softmax/TBL/HMM ensemble, gazettes++</td>
<td>2003</td>
<td>88.76</td>
</tr>
<tr>
<td>Stanford</td>
<td>MEMM softmax markov model</td>
<td>2003</td>
<td>86.07</td>
</tr>
</tbody>
</table>
ELMo: Weighting of layers

- The two biLSTM NLM layers have differentiated uses/meanings
  - Lower layer is better for lower-level syntax, etc.
    - Part-of-speech tagging, syntactic dependencies, NER
  - Higher layer is better for higher-level semantics
    - Sentiment, Semantic role labeling, question answering, SNLI

- This seems interesting, but it’d seem more interesting to see how it pans out with more than two layers of network
3. Also in the air: McCann et al. 2017: CoVe


• Also has idea of using a trained sequence model to provide context to other NLP models
• Idea: Machine translation is meant to preserve meaning, so maybe that’s a good objective?
• Use a 2-layer bi-LSTM that is the encoder of seq2seq + attention NMT system as the context provider
• The resulting CoVe vectors do outperform GloVe vectors on various tasks
• But, the results aren’t as strong as the simpler NLM training described in the rest of these slides so seems abandoned
  • Maybe NMT is just harder than language modeling?
  • Maybe someday this idea will return?
Also around: ULMfit


- Same general idea of transferring NLM knowledge
- Here applied to text classification
ULMfit

Train LM on big general domain corpus (use biLM)
Tune LM on target task data
Fine-tune as classifier on target task

(a) LM pre-training
(b) LM fine-tuning
(c) Classifier fine-tuning
ULMfit emphases

Use reasonable-size “1 GPU” language model not really huge one
A lot of care in LM fine-tuning
  Different per-layer learning rates
  Slanted triangular learning rate (STLR) schedule
Gradual layer unfreezing and STLR when learning classifier
Classify using concatenation \( [h_T, \text{maxpool}(\mathbf{h}), \text{meanpool}(\mathbf{h})] \)
## ULMfit performance

- Text classifier error rates

<table>
<thead>
<tr>
<th>Model</th>
<th>Test</th>
<th>Model</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoVe (McCann et al., 2017)</td>
<td>8.2</td>
<td>CoVe (McCann et al., 2017)</td>
<td>4.2</td>
</tr>
<tr>
<td>oh-LSTM (Johnson and Zhang, 2016)</td>
<td>5.9</td>
<td>TBCNN (Mou et al., 2015)</td>
<td>4.0</td>
</tr>
<tr>
<td>Virtual (Miyato et al., 2016)</td>
<td>5.9</td>
<td>LSTM-CNN (Zhou et al., 2016)</td>
<td>3.9</td>
</tr>
<tr>
<td>ULMFiT (ours)</td>
<td>4.6</td>
<td>ULMFiT (ours)</td>
<td>3.6</td>
</tr>
</tbody>
</table>
ULMfit transfer learning

![Graph showing validation error rate vs. number of training examples for different methods: From scratch, ULMFiT, supervised, ULMFiT, semi-supervised. The graph illustrates how the error rate decreases as the number of training examples increases, with ULMFiT methods generally performing better than training from scratch.](image-url)
Let’s scale it up!

- **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](https://www.reddit.com)
In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

The scientist named the population, after their distinctive horn, Ovid’s Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

Pérez and the others then ventured further into the valley. ...
Elon Musk’s scientists have announced the creation of a terrifying artificial intelligence that’s so smart they refused to release it to the public.

OpenAI’s GPT-2 is designed to write just like a human and is an impressive leap forward capable of penning chillingly convincing text.

It was ‘trained’ by analysing eight million web pages and is capable of writing large tracts based upon a ‘prompt’ written by a real person.

But the machine mind will not be released in its fully-fledged form because of the risk of it being used for ‘malicious purposes’ such as generating fake news, impersonating people online, automating the production of spam or churning out ‘abusive or faked content to post on social media’.

OpenAI wrote: ‘Due to our concerns about malicious applications of the technology, we are not releasing the trained model.'
### 4. Transformer models

All of these models are Transformer models

<table>
<thead>
<tr>
<th>Model</th>
<th>Date</th>
<th>Training Data</th>
<th>Training Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELMo</td>
<td>Oct 2017</td>
<td>800M words</td>
<td>42 GPU days</td>
</tr>
<tr>
<td>GPT</td>
<td>June 2018</td>
<td>800M words</td>
<td>240 GPU days</td>
</tr>
<tr>
<td>BERT</td>
<td>Oct 2018</td>
<td>3.3B words</td>
<td>256 TPU days</td>
</tr>
<tr>
<td>GPT-2</td>
<td>Feb 2019</td>
<td>40B words</td>
<td>~2048 TPU v3 days according to a reddit thread</td>
</tr>
<tr>
<td>XL-Net, ERNIE, Grover, RoBERTa, T5</td>
<td>July 2019–</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Transformer

Attention is all you need. 2017. Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, Polosukhin

- Non-recurrent sequence-to-sequence encoder-decoder model
- Task: machine translation with parallel corpus
- Predict each translated word
- Final cost/error function is standard cross-entropy error on top of a softmax classifier

This and related figures from paper ⇑
Transformer Basics

• Learning about transformers on your own?
  • Key recommended resource:
    • [http://nlp.seas.harvard.edu/2018/04/03/attention.html](http://nlp.seas.harvard.edu/2018/04/03/attention.html)
    • The Annotated Transformer by Sasha Rush
  • A Jupyter Notebook using PyTorch that explains everything!
Dot-Product Attention (Extending our previous def.)

- Inputs: a query q and a set of key-value (k-v) pairs to an output
- Query, keys, values, and output are all vectors

- Output is weighted sum of values, where
- Weight of each value is computed by an inner product of query and corresponding key
- Queries and keys have same dimensionality $d_k$ value have $d_v$

\[
A(q, K, V) = \sum_i \frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j}} v_i
\]
Dot-Product Attention – Matrix notation

- When we have multiple queries $q$, we stack them in a matrix $Q$:

$$A(q, K, V) = \sum_i \frac{e^{q_k^i}}{\sum_j e^{q_k^j}} v_i$$

- Becomes:

$$A(Q, K, V) = \text{softmax}(QK^T)V$$

$$[|Q| \times d_k] \times [d_k \times |K|] \times [|K| \times d_v]$$

\[
\text{softmax} \text{ row-wise} = [|Q| \times d_v]
\]
Scaled Dot-Product Attention

- Problem: As $d_k$ gets large, the variance of $q^T k$ increases $\rightarrow$ some values inside the softmax get large $\rightarrow$ the softmax gets very peaked $\rightarrow$ hence its gradient gets smaller.

- Solution: Scale by length of query/key vectors:

$$A(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$
Self-attention in an encoder

- The input word vectors are the queries, keys and values

- In other words: the word vectors themselves select each other

- Word vector stack = Q = K = V

- They’re separated in the definition so you can different things
  - For an NMT decoder, you can do queries from the output with K/V from the encoder
Multi-head attention

- Problem with simple self-attention:
  - Only one way for words to interact with one-another
- Solution: Multi-head attention
- First map Q, K, V into h=8 many lower dimensional spaces via W matrices
- Then apply attention, then concatenate outputs and pipe through linear layer

\[
\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \\
\text{where} \quad \text{head}_i = \text{Attention}(QW^Q_i, KW^K_i, VW^V_i)
\]
Transformer (Vaswani et al. 2017)
Encoder Input

- Actual word representations are byte-pair encodings
  - As in last lecture

- Also added is a **positional encoding** so same words at different locations have different overall representations:

\[
P E_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right)
\]

\[
P E_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)
\]

or learned
Complete transformer block

Each block has two “sublayers”

1. Multihead attention
2. 2-layer feed-forward NNet (with ReLU)

Each of these two steps also has:
Residual (short-circuit) connection and LayerNorm
LayerNorm(x + Sublayer(x))

Layer norm changes input features to have mean 0 and variance 1 per layer (and adds two more parameters)

\[
\mu^l = \frac{1}{H} \sum_{i=1}^{H} a^l_i \\
\sigma^l = \sqrt{\frac{1}{H} \sum_{i=1}^{H} (a^l_i - \mu^l)^2} \\
h_i = f\left( \frac{g_i}{\sigma_i} (a_i - \mu_i) + b_i \right)
\]

Complete Encoder

- Blocks are repeated 6 or more times
  - (in vertical stack)
Transformer Decoder

- 2 sublayer changes in decoder
- Masked decoder self-attention on previously generated outputs:
  - Encoder-Decoder Attention, where queries come from previous decoder layer and keys and values come from output of encoder
- Blocks repeated 6 times also
## Experimental Results for MT

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
<th>Training Cost (FLOPs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EN-DE</td>
<td>EN-FR</td>
</tr>
<tr>
<td></td>
<td>EN-DE</td>
<td>EN-FR</td>
</tr>
<tr>
<td><strong>ByteNet [18]</strong></td>
<td>23.75</td>
<td>1.0 \cdot 10^{20}</td>
</tr>
<tr>
<td>Deep-Att + PosUnk [39]</td>
<td>39.2</td>
<td>1.4 \cdot 10^{20}</td>
</tr>
<tr>
<td><strong>GNMT + RL [38]</strong></td>
<td>24.6</td>
<td>2.3 \cdot 10^{19}</td>
</tr>
<tr>
<td><strong>ConvS2S [9]</strong></td>
<td>39.92</td>
<td>1.5 \cdot 10^{20}</td>
</tr>
<tr>
<td><strong>MoE [32]</strong></td>
<td>40.4</td>
<td>2.0 \cdot 10^{19}</td>
</tr>
<tr>
<td><strong>Deep-Att + PosUnk Ensemble [39]</strong></td>
<td>40.4</td>
<td>8.0 \cdot 10^{20}</td>
</tr>
<tr>
<td><strong>GNMT + RL Ensemble [38]</strong></td>
<td>41.16</td>
<td>1.1 \cdot 10^{21}</td>
</tr>
<tr>
<td><strong>ConvS2S Ensemble [9]</strong></td>
<td>41.29</td>
<td>1.2 \cdot 10^{21}</td>
</tr>
<tr>
<td><strong>Transformer (base model)</strong></td>
<td>27.3</td>
<td>3.3 \cdot 10^{18}</td>
</tr>
<tr>
<td><strong>Transformer (big)</strong></td>
<td>28.4</td>
<td>2.3 \cdot 10^{19}</td>
</tr>
</tbody>
</table>
Some performance numbers: LM on WikiText-103

<table>
<thead>
<tr>
<th>Model</th>
<th># Params</th>
<th>Perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grave et al. (2016) – LSTM</td>
<td></td>
<td>48.7</td>
</tr>
<tr>
<td>Grave et al. (2016) – LSTM with cache</td>
<td></td>
<td>40.8</td>
</tr>
<tr>
<td>4-layer QRNN (Merity et al. 2018)</td>
<td>151M</td>
<td>33.0</td>
</tr>
<tr>
<td>LSTM + Hebbian + Cache + MbPA (Rae et al.)</td>
<td>151M</td>
<td>29.2</td>
</tr>
<tr>
<td>Transformer-XL Large (Dai et al. 2019)</td>
<td>257M</td>
<td>18.3</td>
</tr>
<tr>
<td>GPT-2 Large* (Radford et al. 2019)</td>
<td>1.5B</td>
<td>17.5</td>
</tr>
</tbody>
</table>

(For gray haired people)
A perplexity of 18 for Wikipedia text is just stunningly low!
Size matters

- Going from 110M to 340M parameters helps a lot
- Improvements have not yet asymptoted

BERT (Bidirectional Encoder Representations from Transformers): Pre-training of Deep Bidirectional Transformers for Language Understanding, which is then fine-tuned for a task

Want: truly bidirectional information flow without leakage in a deep model

Solution: Use a cloze task formulation where 15% of words are blanked out and predicted:

```
store  gallon
↑      ↑
the man went to the [MASK] to buy a [MASK] of milk
```
BERT sentence pair encoding

<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>
</table>

<table>
<thead>
<tr>
<th>Token Embeddings</th>
<th>$E_{[CLS]}$</th>
<th>$E_{my}$</th>
<th>$E_{dog}$</th>
<th>$E_{is}$</th>
<th>$E_{cute}$</th>
<th>$E_{[SEP]}$</th>
<th>$E_{he}$</th>
<th>$E_{likes}$</th>
<th>$E_{play}$</th>
<th>$E_{#ing}$</th>
<th>$E_{[SEP]}$</th>
</tr>
</thead>
<tbody>
<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>

Token embeddings are word pieces
Learned segmented embedding represents each sentence
Positional embedding is as for other Transformer architectures
BERT model architecture and training

- Transformer encoder (as before)
- Self-attention ⇒ no locality bias
  - Long-distance context has “equal opportunity”
- Single multiplication per layer ⇒ efficiency on GPU/TPU

- Train on Wikipedia + BookCorpus
- Train 2 model sizes:
  - BERT-Base: 12-layer, 768-hidden, 12-head
  - BERT-Large: 24-layer, 1024-hidden, 16-head
- Trained on 4x4 or 8x8 TPU slice for 4 days
BERT model fine tuning

• Simply learn a classifier built on the top layer for each task that you fine tune for
BERT model fine tuning

(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Year</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flair (Zalando)</td>
<td>Character-level language model</td>
<td>2018</td>
<td>93.09</td>
</tr>
<tr>
<td>BERT Large</td>
<td>Transformer bidi LM + fine tune</td>
<td>2018</td>
<td>92.8</td>
</tr>
<tr>
<td>CVT Clark</td>
<td>Cross-view training + multitask learn</td>
<td>2018</td>
<td>92.61</td>
</tr>
<tr>
<td>BERT Base</td>
<td>Transformer bidi LM + fine tune</td>
<td>2018</td>
<td>92.4</td>
</tr>
<tr>
<td>ELMo</td>
<td>ELMo in BiLSTM</td>
<td>2018</td>
<td>92.22</td>
</tr>
<tr>
<td>TagLM Peters</td>
<td>LSTM BiLM in BiLSTM tagger</td>
<td>2017</td>
<td>91.93</td>
</tr>
<tr>
<td>Ma + Hovy</td>
<td>BiLSTM + char CNN + CRF layer</td>
<td>2016</td>
<td>91.21</td>
</tr>
<tr>
<td>Tagger Peters</td>
<td>BiLSTM + char CNN + CRF layer</td>
<td>2017</td>
<td>90.87</td>
</tr>
<tr>
<td>Ratinov + Roth</td>
<td>Categorical CRF+Wikipedia+word cls</td>
<td>2009</td>
<td>90.80</td>
</tr>
<tr>
<td>Finkel et al.</td>
<td>Categorical feature CRF</td>
<td>2005</td>
<td>86.86</td>
</tr>
<tr>
<td>IBM Florian</td>
<td>Linear/softmax/TBL/HMM ensemble, gazettes++</td>
<td>2003</td>
<td>88.76</td>
</tr>
<tr>
<td>Stanford</td>
<td>MEMM softmax markov model</td>
<td>2003</td>
<td>86.07</td>
</tr>
</tbody>
</table>
### AllenAI ARISTO: Answering Science Exam Questions

From ‘F’ to ‘A’ on the N.Y. Regents Science Exams: An Overview of the Aristo Project. Peter Clark, Oren Etzioni, Daniel Khashabi, Tushar Khot, Bhavana Dalvi Mishra, Kyle Richardson, Ashish Sabharwal, Carissa Schoenick, Oyvind Tafjord, Niket Tandon, Sumithra Bhakthavatsalam, Dirk Groeneveld, Michal Guerquin, Michael Schmitz

Which equipment will best separate a mixture of iron filings and black pepper?  
(1) magnet  
(2) filter paper  
(3) triplebeam balance  
(4) voltmeter

Which process in an apple tree primarily results from cell division?  
(1) growth  
(2) photosynthesis  
(3) gas exchange  
(4) waste removal

<table>
<thead>
<tr>
<th>Test Set</th>
<th>IR</th>
<th>TupInf</th>
<th>Multee</th>
<th>AristoBERT</th>
<th>AristoRoBERTa</th>
<th>ARISTO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regents 4th</td>
<td>64.5</td>
<td>63.5</td>
<td>69.7</td>
<td>86.2</td>
<td>88.1</td>
<td>89.9</td>
</tr>
<tr>
<td>Regents 8th</td>
<td>66.6</td>
<td>61.4</td>
<td>68.9</td>
<td>86.6</td>
<td>88.2</td>
<td>91.6</td>
</tr>
<tr>
<td>Regents 12th</td>
<td>41.2</td>
<td>35.4</td>
<td>56.0</td>
<td>75.5</td>
<td>82.3</td>
<td>83.5</td>
</tr>
<tr>
<td>ARC-Challenge</td>
<td>0.0</td>
<td>23.7</td>
<td>37.4</td>
<td>57.6</td>
<td><strong>64.6</strong></td>
<td>64.3</td>
</tr>
</tbody>
</table>
Attention visualization: Implicit anaphora resolution

Words start to pay attention to other words in sensible ways

In 5th layer. Isolated attentions from just the word ‘its’ for attention heads 5 and 6. Note that the attentions are very sharp for this word.
6. How’s the weather?
Rapid Progress from Pre-Training (GLUE benchmark)

Over 3x reduction in error in 2 years, “superhuman” performance
Yay! We now have strongly performing, deep, generic, pre-trained, neural network stacks for NLP that you can just load – in the same way vision has had for 5 years (ResNet, etc.)!
But let’s change the x-axis to compute ...

BERT-Large uses 60x more compute than ELMo
But let’s change the x-axis to compute ...

RoBERTa uses 16x more compute than BERT-Large
ALBERT uses 10x more compute than RoBERTa.
The climate cost of modern deep learning

SustaiNLP 2020
First Workshop on Simple and Efficient Natural Language Processing

Workshop at EMNLP 2020
Punta Cana, Dominican Republic
ELECTRA: “Efficiently Learning an Encoder to Classify Token Replacements Accurately”

Clark, Luong, Le, and Manning (2020)

Bidirectional model but learn from all tokens

original replaced original replaced

the painter sold the car
Generating Replacements

Plausible alternatives come from small masked language model (the “generator”) trained jointly with ELECTRA.

- **Generator** (typically a small MLM)
  - Sample
  - MLM Loss
  - The [MASK]
  - Artist
  - Sold
  - The
  - Painting [MASK]

- **Discriminator** (ELECTRA)
  - Sample
  - Original
  - Original
  - Original
  - Replaced
  - Original
  - Original
  - Original
  - Replaced
  - Car
  - The
  - Artist
  - Sold
  - The
Results: GLUE Score vs Compute

- ELMo
- GPT
- BERT-Base
- BERT-Large
- EL-Small
- EL-Base
- EL-Large
- XLNet
- RoBERTa

Pre-Train FLOPs