CS388: Natural Language Processing
Lecture 8: Pre-trained Encoders

Greg Durrett

Announcements

- P2 due Tuesday
- P1 back tomorrow
- Final project released, proposals due Feb 23
- Single or pairs, combining with other courses okay
- Original research or reproduction
- Topics, deliverables, etc. given in the spec
- TACC allocation: send me usernames to be added. 4000 node-hours; usually 25% of the class uses this in a somewhat serious way, so ~10 groups

Recall: Transformers

- Vectors: \( d_{\text{model}} \)
- Queries/keys: \( d_k \), always smaller than \( d_{\text{model}} \)
- Values: separate dimension \( d_v \), output is multiplied by \( W^o \) which is \( d_v \times d_{\text{model}} \) so we can get back to \( d_{\text{model}} \) before the residual
- FFN can explode the dimension with \( W_1 \) and collapse it back with \( W_2 \)
  \[
  \text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2
  \]

Recall: Training Transformer LMs

- Loss = \(- \log P(w^* | \text{context})\)
- \( \text{loss} += \text{loss}_\text{fcn}(\text{log_probs, ex.output_tensor}) \) [seq len, num output classes]
- Batching is a little tricky with NLLLoss: need to collapse [batch, seq len, num classes] to [batch * seq len, num classes]. You do not need to batch
Today

- ELMo
- BERT
- BERT results, BERT variants
- Subword tokenization

What is pre-training?

- “Pre-train” a model on a large dataset for task X, then “fine-tune” it on a dataset for task Y
- Key idea: X is somewhat related to Y, so a model that can do X will have some good neural representations for Y as well
- ImageNet pre-training is huge in computer vision: learn generic visual features for recognizing objects
- GloVe can be seen as pre-training: learn vectors with the skip-gram objective on large data (task X), then fine-tune them as part of a neural network for sentiment/any other task (task Y)

GloVe is insufficient

- GloVe uses a lot of data but in a weak way
- GloVe gives a single embedding for each word is wrong
  
  
  \textit{they swing the bats} \quad \textit{they see the bats}

- Identifying discrete word senses is hard, doesn’t scale. Hard to identify how many senses each word has
- How can we make our word embeddings more context-dependent? Use language model pretraining!
Context-dependent Embeddings

- Train a neural language model to predict the next word given previous words in the sentence, use the hidden states (output) at each step as word embeddings
- This is the key idea behind ELMo: language models can allow us to form useful word representations in the same way word2vec did

ELMo

- CNN over each word => RNN
- Representation of visited (plus vectors from another LM running backwards)
- 4096-dim LSTMs
- 2048 CNN filters projected down to 512-dim

ELMo

- Use the embeddings as a drop-in replacement for GloVe
- Huge gains across many high-profile tasks: NER, question answering, semantic role labeling (similar to parsing), etc.
- But what if the pre-training isn’t just for the embeddings?

BERT
**BERT**

- AI2 made ELMo in spring 2018, GPT (transformer-based ELMo) was released in summer 2018, BERT came out October 2018
- Four major changes compared to ELMo:
  - Transformers instead of LSTMs
  - Bidirectional model with “Masked LM” objective instead of standard LM
  - Fine-tune instead of freeze at test time **(not just a source of word embeddings!)**
  - Operates over word pieces (byte pair encoding)

**Masked Language Modeling**

- How to learn a “deeply bidirectional” model? What happens if we just replace an LSTM with a transformer?
  
  ![ELMo Diagram](ELMo Diagram)
  ![BERT Diagram](BERT Diagram)

- BERT formula: take a chunk of text, mask out 15% of the tokens, and try to predict them
  
  \[ P(\text{Madagascar} | \text{John visited [MASK] yesterday}) \]

- You could do this with a “one-sided” transformer, but this “two-sided” model can cheat

**Devlin et al. (2019)**
**Next “Sentence” Prediction**

- Input: [CLS] Text chunk 1 [SEP] Text chunk 2
- 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
- Why is this a good idea?
- BERT objective: masked LM + next sentence prediction

NotNext

Madagascar

enjoyed

like

Transformer

... 

Transformer


**What can BERT do?**

- Artificial [CLS] token is used as the vector to do classification from
- Sentence pair tasks (entailment): feed both sentences into BERT
- BERT can also do tagging by predicting tags at each word piece

Devlin et al. (2019)

**What can BERT do?**

Entails (first sentence implies second is true)

Transformer

... 

Transformer

[CLS] A boy plays in the snow [SEP] A boy is outside

Devlin et al. (2019)

**BERT Architecture**

- BERT Base: 12 layers, 768-dim per wordpiece token, 12 heads. Total params = 110M
- BERT Large: 24 layers, 1024-dim per wordpiece token, 16 heads. Total params = 340M
- Positional embeddings and segment embeddings, 30k word pieces
- This is the model that gets pre-trained on a large corpus

**What can BERT do?**

- How does BERT model sentence pairs?
- Transformers can capture interactions between the two sentences, even though the NSP objective doesn’t really cause this to happen
Q: What was Marie Curie the first female recipient of?

Passage: One of the most famous people born in Warsaw was Marie Skłodowska-Curie, who achieved international recognition for her research on radioactivity and was the first female recipient of the Nobel Prize. Famous musicians include Władysław Szpilman and Frédéric Chopin. Though Chopin was born in the village of Żelazowa Wola, about 60 km (37 mi) from Warsaw, he moved to the city with his family when he was seven months old. Casimir Pulaski, a Polish general and hero of the American Revolutionary War, was born there in 1745.

Answer = Nobel Prize

• Assume we know a passage that contains the answer. More recent work has shown how to retrieve these effectively (will discuss when we get to QA)
Fine-tuning BERT

- Fine-tune for 1-3 epochs, batch size 2-32, learning rate 2e-5 - 5e-5
- Large changes to weights up here (particularly in last layer to route the right information to [CLS])
- Smaller changes to weights lower down in the transformer
- Small LR and short fine-tuning schedule mean weights don’t change much
- Often requires tricky learning rate schedules (‘triangular’ learning rates with warmup periods)

BERT results, BERT variants

Evaluation: GLUE

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Train</th>
<th>Test</th>
<th>Task</th>
<th>Metrics</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoLA</td>
<td>8.5k</td>
<td>1k</td>
<td>acceptability</td>
<td>Matthews corr. acc.</td>
<td>misc, movie reviews</td>
</tr>
<tr>
<td>SST-2</td>
<td>67k</td>
<td>1.8k</td>
<td>sentiment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MRPC</td>
<td>3.7k</td>
<td>1.7k</td>
<td>paraphrase</td>
<td>acc./F1</td>
<td>news</td>
</tr>
<tr>
<td>STS-B</td>
<td>7k</td>
<td>1.4k</td>
<td>sentence similarity</td>
<td>Pearson/Spearman corr.</td>
<td>misc, social QA questions</td>
</tr>
<tr>
<td>QQP</td>
<td>364k</td>
<td>391k</td>
<td>paraphrase</td>
<td>acc./F1</td>
<td></td>
</tr>
</tbody>
</table>

Inference Tasks

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Train</th>
<th>Test</th>
<th>Task</th>
<th>Metrics</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNLI</td>
<td>393k</td>
<td>20k</td>
<td>NLI</td>
<td>matched acc./mismatched acc.</td>
<td>misc, Wikipedia</td>
</tr>
<tr>
<td>QNLI</td>
<td>105k</td>
<td>5.4k</td>
<td>QA/NLI</td>
<td>acc.</td>
<td>news, Wikipedia</td>
</tr>
<tr>
<td>RTE</td>
<td>2.5k</td>
<td>3k</td>
<td>NLI</td>
<td>acc.</td>
<td></td>
</tr>
<tr>
<td>WNLI</td>
<td>634</td>
<td>146</td>
<td>coreference/NLI</td>
<td>acc.</td>
<td>fiction books</td>
</tr>
</tbody>
</table>

Results

<table>
<thead>
<tr>
<th>System</th>
<th>MNLI-(m/mm)</th>
<th>QQP</th>
<th>QNLI</th>
<th>SST-2</th>
<th>CoLA</th>
<th>STS-B</th>
<th>MRPC</th>
<th>RTE</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-OpenAI SOTA</td>
<td>80.6/80.1</td>
<td>66.1</td>
<td>82.3</td>
<td>93.2</td>
<td>35.0</td>
<td>81.0</td>
<td>86.0</td>
<td>61.7</td>
<td>74.0</td>
</tr>
<tr>
<td>BiLSTM+ELMo+Attn</td>
<td>76.4/76.1</td>
<td>64.8</td>
<td>79.9</td>
<td>90.4</td>
<td>36.0</td>
<td>73.3</td>
<td>84.9</td>
<td>56.8</td>
<td>71.0</td>
</tr>
<tr>
<td>OpenAI GPT</td>
<td>82.1/81.4</td>
<td>70.3</td>
<td>88.1</td>
<td>91.3</td>
<td>45.4</td>
<td>80.0</td>
<td>82.3</td>
<td>56.0</td>
<td>75.2</td>
</tr>
<tr>
<td>BERT_BASE</td>
<td>84.6/83.4</td>
<td>71.2</td>
<td>90.1</td>
<td>93.5</td>
<td>52.1</td>
<td>85.8</td>
<td>88.9</td>
<td>66.4</td>
<td>79.6</td>
</tr>
<tr>
<td>BERT_LARGE</td>
<td>86.7/85.9</td>
<td>72.1</td>
<td>91.1</td>
<td>94.9</td>
<td>60.5</td>
<td>86.5</td>
<td>89.3</td>
<td>70.1</td>
<td>81.9</td>
</tr>
</tbody>
</table>

- Huge improvements over prior work (even compared to ELMo)
- Effective at “sentence pair” tasks: textual entailment (does sentence A imply sentence B), paraphrase detection

Wang et al. (2019)

Devlin et al. (2018)
RoBERTa

- "Robustly optimized BERT"
- 160GB of data instead of 16 GB
- Dynamic masking: standard BERT uses the same MASK scheme for every epoch, RoBERTa recomputes them
- New training + more data = better performance

Liu et al. (2019)

<table>
<thead>
<tr>
<th>Model</th>
<th>data</th>
<th>bsz</th>
<th>steps</th>
<th>SQuAD (v1.1/2.0)</th>
<th>MNLIm</th>
<th>SST-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoBERTa</td>
<td>16GB</td>
<td>8K</td>
<td>100K</td>
<td>93.687.3</td>
<td>89.0</td>
<td>95.3</td>
</tr>
<tr>
<td>w/ extra data</td>
<td>16GB</td>
<td>8K</td>
<td>100K</td>
<td>94.087.7</td>
<td>89.3</td>
<td>95.6</td>
</tr>
<tr>
<td>+ pretrain longer</td>
<td>160GB</td>
<td>8K</td>
<td>300K</td>
<td>94.488.7</td>
<td>90.0</td>
<td>96.1</td>
</tr>
<tr>
<td>+ pretrain even longer</td>
<td>160GB</td>
<td>8K</td>
<td>500K</td>
<td>94.689.4</td>
<td>90.2</td>
<td>96.4</td>
</tr>
</tbody>
</table>

BERT Large

with Books + Wiki 13GB 25B 1M 90.981.8 86.6 93.7

ELECTRA

- Discriminator to detect replaced tokens rather than a generator to actually predict what those tokens are
- More efficient, strong performance

Clark et al. (2020)

DeBERTa

- Slightly better variant

\[ A_{i,j} = \{H_i, P_{ij}\} \times \{H_j, P_{ji}\}^T = H_i H_j^T + H_i P_{ji}^T + P_{ij} H_j^T + P_{ij} P_{ji}^T \] (2)

That is, the attention weight of a word pair can be computed as a sum of four attention scores using disentangled matrices on their contents and positions as content-to-content, content-to-position, position-to-content, and position-to-position.

He et al. (2021)

Using BERT

- HuggingFace Transformers: big open-source library with most pre-trained architectures implemented, weights available
- Lots of standard models... and “community models”

Using BERT

- HuggingFace Transformers: big open-source library with most pre-trained architectures implemented, weights available
- Lots of standard models... and “community models”

1. BERT (from Google) released with the paper “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding” by Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova.
2. GPT (from OpenAI) released with the paper “Language Models are Unicorns” by Ilya Sutskever.
3. GPT-2: from OpenAI released with the paper “Language Models are Unicorns (Extended)” by Ilya Sutskever.
4. Transformer-XL (from Google/DSU) released with the paper “Transformer-XL: Attending to Long Sequences” by Yoon Kim.
5. XLNet (from Google/DSU) released with the paper “XLNet: Generalized Autoregressive Pretraining” by Yinhan Liu, Myle Ott, Naman Joseph, Jingfei Ye, Moustapha Cisse, Thang David Tran, Quoc V. Le and Dario Amodei.
6. DeBERTa (from Facebook) released together with the paper “DeBERTa: Robustly Optimized BERT Pretraining through Task-Oriented Fine-Tuning” by Yizhe Sun, Wei Li, Mengye Ren, and Jianfeng Zhang.
7. RoBERTa (from Facebook), released together with the paper “RoBERTa: A Robustly Optimized BERT pretraining approach” by Yinhan Liu, Myle Ott, Naman Joseph, Jingfei Ye, Moustapha Cisse, Thang David Tran, Quoc V. Le and Dario Amodei.
What does BERT learn?

- Heads on transformers learn interesting and diverse things: content heads (attend based on content), positional heads (based on position), etc.

Clark et al. (2019)

---

Handling Rare Words

- Words are a difficult unit to work with. Why?

  - When you have 100,000+ words, the final matrix multiply and softmax start to dominate the computation, many params, still some words you haven’t seen, doesn’t take advantage of morphology, ...

- Character-level models were explored extensively in 2016-2018 but simply don’t work well — becomes very expensive to represent sequences
Subword Tokenization

- Subword tokenization: wide range of schemes that use tokens that are **between characters and words** in terms of granularity

- These “word pieces” may be full words or parts of words
  
  _the _eco tax _port i co _in _Po nt - de - Bu is ...

- _ indicates the word piece starting a word (can think of it as the space character).

  Sennrich et al. (2016)

---

Output: _le _port ique _écotaxe _de _Pont - de - Buis ...

- Can achieve transliteration with this, subword structure makes some translations easier to achieve

  Sennrich et al. (2016)

---

Byte Pair Encoding (BPE)

- Start with every individual byte (basically character) as its own symbol

  ```python
def main(vocab):
    for i in range(num_merges):
      pairs = get_stats(vocab)
      best = max(pairs, key=pairs.get)
      vocab = merge_vocab(best, vocab)

  Sennrich et al. (2016)
  ```

- Count bigram character cooccurrences
- Merge the most frequent pair of adjacent characters
- Doing 8k merges => vocabulary of around 8000 word pieces. Includes many whole words
- Most SOTA NMT systems use this on both source + target

  Sennrich et al. (2016)

---

<table>
<thead>
<tr>
<th>Original:</th>
<th>furiously</th>
<th>Original:</th>
<th>tricycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPE:</td>
<td>.fur iously (b)</td>
<td>BPE:</td>
<td>.tric y cles</td>
</tr>
<tr>
<td>Unigram LM:</td>
<td>.fur ious ly</td>
<td>Unigram LM:</td>
<td>.tri c y s</td>
</tr>
<tr>
<td>BPE:</td>
<td>.Complet t ely</td>
<td>BPE:</td>
<td>.suggest ions</td>
</tr>
<tr>
<td>Unigram LM:</td>
<td>.Complete ly .pre post er ou s</td>
<td>Unigram LM:</td>
<td>.suggest ions</td>
</tr>
</tbody>
</table>

- What do you see here?
- BPE produces less linguistically plausible units than another technique based on a unigram language model: rather than greedily merge, find chunks which make the sequence look likely under a unigram LM
- Unigram LM tokenizer leads to slightly better BERT

  Bostrom and Durrett (2020)
What’s in the token vocab?

I’ve just found out that several of the anomalous GPT tokens ("TheNitromeFan", "SolidGoldMagikarp", "davidji", "Smartstocks", "RandomRedditorWithNo", ...) are handles of people who are (competitively? collaboratively?) counting to infinity on a Reddit forum. I kid you not.

Tokenization Today

- All **pre-trained** models use some kind of subword tokenization with a tuned vocabulary; usually between 50k and 250k pieces (larger number of pieces for multilingual models)

- As a result, classical word embeddings like GloVe are **not used**. All subword representations are randomly initialized and learned in the Transformer models

Takeaways

- Pre-trained models and BERT are very powerful for a range of NLP tasks

- These models have enabled big advances in NLI and QA specifically

- Next time: pre-trained decoders (GPT-3) and encoder-decoder models (T5)