Reading Comprehension
Bidirectional Attention Flow

Each passage word now “knows about” the query

Seo et al. (2016)
QANet

- One of many models building on BiDAF in more complex ways

- Similar structure as BiDAF, but transformer layers (next lecture) instead of LSTMs
### SQuAD SOTA: Fall 18

<table>
<thead>
<tr>
<th>Rank</th>
<th>Model</th>
<th>EM</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BERT (ensemble)</td>
<td>87.433</td>
<td>93.160</td>
</tr>
<tr>
<td>2</td>
<td>BERT (single model)</td>
<td>85.083</td>
<td>91.835</td>
</tr>
<tr>
<td>2</td>
<td><strong>nlnet</strong> (ensemble)</td>
<td>85.356</td>
<td>91.202</td>
</tr>
<tr>
<td>2</td>
<td><strong>nlnet</strong> (ensemble)</td>
<td>85.954</td>
<td>91.677</td>
</tr>
<tr>
<td>3</td>
<td>QANet (ensemble)</td>
<td>84.454</td>
<td>90.490</td>
</tr>
<tr>
<td>4</td>
<td><strong>r-net</strong> (ensemble)</td>
<td>84.003</td>
<td>90.147</td>
</tr>
<tr>
<td>5</td>
<td>QANet (ensemble)</td>
<td>83.877</td>
<td>89.737</td>
</tr>
</tbody>
</table>

- **BiDAF**: 73 EM / 81 F1
- **nlnet, QANet, r-net** — dueling super complex systems (much more than BiDAF...)
SQuAD 2.0 SOTA: Spring 2019

- Harder variant of SQuAD
- Since spring 2019: SQuAD performance is dominated by large pre-trained models like BERT
Adversarial Examples

- Can construct adversarial examples that fool these systems: add one carefully chosen sentence and performance drops to below 50%

- Still “surface-level” matching, not complex understanding

- Other challenges: recognizing when answers aren’t present, doing multi-step reasoning

Article: Super Bowl 50

Paragraph: “Peyton Manning became the first quarterback ever to lead two different teams to multiple Super Bowls. He is also the oldest quarterback ever to play in a Super Bowl at age 39. The past record was held by John Elway, who led the Broncos to victory in Super Bowl XXXIII at age 38 and is currently Denver’s Executive Vice President of Football Operations and General Manager. Quarterback Jeff Dean had jersey number 37 in Champ Bowl XXXIV.”

Question: “What is the name of the quarterback who was 38 in Super Bowl XXXIII?”

Original Prediction: John Elway

Prediction under adversary: Jeff Dean

Jia and Liang (2017)
Pre-training / ELMo
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
  - Take a powerful language model, train it on large amounts of data, then use those representations in downstream tasks
- Having a single embedding for each word is wrong
  - they dance at balls    they hit the balls
- 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?
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.

Peters et al. (2018)
ELMo

- CNN over each word => RNN

Representation of *visited* (plus vectors from another LM running backwards)

*getting this model right took years*

Peters et al. (2018)
Training ELMo

- Data: 1B Word Benchmark (Chelba et al., 2014)

- Pre-training time: 2 weeks on 3 NVIDIA GTX 1080 GPUs
  - Much lower time cost if we used V100s / Google’s TPUs, but still hundreds of dollars in compute cost to train once
  - Larger BERT models trained on more data (next week) cost $10k+

- Pre-training is expensive, but fine-tuning is doable
How to apply ELMo?

- Take those embeddings and feed them into whatever architecture you want to use for your task

- *Frozen* embeddings (most common): update the weights of your network but keep ELMo’s parameters frozen

- *Fine-tuning*: backpropagate all the way into ELMo when training your model
### Results: Frozen ELMo

<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>4.7 / 24.9%</td>
</tr>
<tr>
<td>SNLI</td>
<td>Chen et al. (2017)</td>
<td>88.6</td>
<td>88.0</td>
<td>0.7 / 5.8%</td>
</tr>
<tr>
<td>SRL</td>
<td>He et al. (2017)</td>
<td>81.7</td>
<td>81.4</td>
<td>3.2 / 17.2%</td>
</tr>
<tr>
<td>Coref</td>
<td>Lee et al. (2017)</td>
<td>67.2</td>
<td>67.2</td>
<td>3.2 / 9.8%</td>
</tr>
<tr>
<td>NER</td>
<td>Peters et al. (2017)</td>
<td>91.93 ± 0.19</td>
<td>90.15</td>
<td>2.06 / 21%</td>
</tr>
<tr>
<td>SST-5</td>
<td>McCann et al. (2017)</td>
<td>53.7</td>
<td>51.4</td>
<td>3.3 / 6.8%</td>
</tr>
</tbody>
</table>

- Massive improvements, beating models handcrafted for each task
- These are mostly *text analysis* tasks. Other pre-training approaches needed for text generation like translation.

Peters et al. (2018)
Why is language modeling a good objective?

- “Impossible” problem but bigger models seem to do better and better at distributional modeling (no upper limit yet)

- Successfully predicting next words requires modeling lots of different effects in text

*Context:* My wife refused to allow me to come to Hong Kong when the plague was at its height and –” “Your wife, Johanne? You are married at last?” Johanne grinned. “Well, when a man gets to my age, he starts to need a few home comforts.

*Target sentence:* After my dear mother passed away ten years ago now, I became _____

*Target word:* lonely
Probing ELMo

- From each layer of the ELMo model, attempt to predict something: POS tags, word senses, etc.
- Higher accuracy => ELMo is capturing that thing more strongly

<table>
<thead>
<tr>
<th>Model</th>
<th>F$_1$</th>
<th>Model</th>
<th>Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>WordNet 1st Sense Baseline</td>
<td>65.9</td>
<td>Collobert et al. (2011)</td>
<td>97.3</td>
</tr>
<tr>
<td>Raganato et al. (2017a)</td>
<td>69.9</td>
<td>Ma and Hovy (2016)</td>
<td>97.6</td>
</tr>
<tr>
<td>Iacobacci et al. (2016)</td>
<td><strong>70.1</strong></td>
<td>Ling et al. (2015)</td>
<td><strong>97.8</strong></td>
</tr>
<tr>
<td>CoVe, First Layer</td>
<td>59.4</td>
<td>CoVe, First Layer</td>
<td>93.3</td>
</tr>
<tr>
<td>CoVe, Second Layer</td>
<td>64.7</td>
<td>CoVe, Second Layer</td>
<td>92.8</td>
</tr>
<tr>
<td>biLM, First layer</td>
<td>67.4</td>
<td>biLM, First Layer</td>
<td>97.3</td>
</tr>
<tr>
<td>biLM, Second layer</td>
<td>69.0</td>
<td>biLM, Second Layer</td>
<td>96.8</td>
</tr>
</tbody>
</table>

Table 5: All-words fine grained WSD F$_1$. For CoVe and the biLM, we report scores for both the first and second layer biLSTMs.

Table 6: Test set POS tagging accuracies for PTB. For CoVe and the biLM, we report scores for both the first and second layer biLSTMs.

Peters et al. (2018)
Analysis

Figure 2: Visualization of softmax normalized biLM layer weights across tasks and ELMo locations. Normalized weights less then 1/3 are hatched with horizontal lines and those greater then 2/3 are speckled.

Peters et al. (2018)
Takeaways

- Learning a large language model can be an effective way of generating “word embeddings” informed by their context.
- Pre-training on massive amounts of data can improve performance on tasks like QA.
- Next class: transformers and BERT.