Neural Language Modeling
Cont. and Pretrained LMs

CSCI 1460: Computational Linguistics
Lecture 12

Ellie Pavlick
Fall 2022
Consider a Transformer language model that is initialized randomly and then trained on a language modeling task. Both the word embeddings and the positional encodings are randomly initialized and then trained via backprop. Which of the following pairs would be the most similar according to the final trained model?

38 / 79 correct responses

- Word 1 = the word "cat" in the phrase "the cat in the hat"; word 2 = the word "cat" in the phrase... (19%)
- Word 1 = the word "cat" in the phrase "the cat in the hat"; word 2 = the word "dog" in the phrase... (32.9%)
- ✓ it is impossible to say without inspecting the model; it would depend on the training data... (48.1%)
Figure 9.20  A simple way to model position: simply adding an embedding representation of the absolute position to the input word embedding.
Topics

- Contextualized Word Representations:
  - ELMo
  - BERT
  - GPT
- Finetuning and Other Transfer Methods
Topics

• Contextualized Word Representations:
  • ELMo
  • BERT
  • GPT
• Finetuning and Other Transfer Methods
(Regular/“Static”) Word Embeddings
(Regular/“Static”) Word Embeddings

embedding matrix
(randomly initialized)
(Regular/“Static”) Word Embeddings

embedding matrix (randomly initialized)
(Regular/“Static”) Word Embeddings

the cat sat

cat \rightarrow \text{embedding matrix (randomly initialized)} \rightarrow \text{sat}
(Regular/“Static”) Word Embeddings

the cat sat

cat ➔ embedding matrix (randomly initialized) ➔ softmax ➔ sat

projection/activation function
distribution over context words
(Regular/“Static”) Word Embeddings

the cat sat

cat → embedding matrix (randomly initialized) → softmax → sat

updated via backprop

projection/activation function distribution over context words
(Regular/"Static") Word Embeddings

the cat sat

cat \rightarrow \text{softmax} \rightarrow \text{sat}

one embedding for each word ("cat"), updated each time the word is seen in training
Contextualized Word Representations

Basic Idea

Neural Language Model (e.g., an RNN)

`t_0 = the`, `t_1 = cat`, `t_2 = sat`
Contextualized Word Representations

Basic Idea

\[ t_0 = \text{the} \quad t_1 = \text{cat} \quad t_2 = \text{sat} \]

word embedding for “cat” (e.g., word2vec, or trained from scratch by this model)
Contextualized Word Representations

Basic Idea

"contextual" word embedding for "cat"
Contextualized Word Representations

Basic Idea

function of static word embedding, but also of prior hidden state
Contextualized Word Representations

Basic Idea

“cat”  “sat”  “on”

at least a little different for each occurrence of cat (can be a lot different)
Topics

- Transformers Cont. (+ any questions you have!)
- What is pretraining?
- Contextualized Word Representations:
  - ELMo
  - BERT
  - GPT
- Finetuning and Other Transfer Methods
Deep contextualized word representations

Matthew E. Peters†, Mark Neumann†, Mohit Iyyer†, Matt Gardner†,
{matthewp,markn,mohiti,mattg}@allenai.org

Christopher Clark*, Kenton Lee*, Luke Zettlemoyer†*
{csquared,kentonl,lsz}@cs.washington.edu

†Allen Institute for Artificial Intelligence
*Paul G. Allen School of Computer Science & Engineering, University of Washington

https://allenai.org/allennlp/software/elmo
ELMo

- Architecture: Two-layer BiLSTM
  - “bi” = bidirectional, i.e., trained left-to-right and also right-to-left
- Layer 0 = pretrained static embeddings (Glove)
- Trained on vanilla language modeling task
- “Finetuned” by learning a simple linear combination of the learned embeddings
ELMo
Architecture

```
<table>
<thead>
<tr>
<th></th>
<th>“cat”</th>
<th>“sat”</th>
<th>“on”</th>
</tr>
</thead>
<tbody>
<tr>
<td>O₀</td>
<td>h₀₀₀</td>
<td>h₀₁₀</td>
<td>h₀₂₀</td>
</tr>
<tr>
<td>O₁</td>
<td>h₀₀₁</td>
<td>h₀₁₁</td>
<td>h₀₂₁</td>
</tr>
<tr>
<td>O₂</td>
<td>h₀₀₂</td>
<td>h₀₁₂</td>
<td>h₀₂₂</td>
</tr>
<tr>
<td>e₀</td>
<td>t₀ = the</td>
<td>t₁ = cat</td>
<td>t₂ = sat</td>
</tr>
<tr>
<td>e₁</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e₂</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
ELMo Architecture

https://paperswithcode.com/method/bilstm
ELMo Transfer

```
<table>
<thead>
<tr>
<th></th>
<th>h10</th>
<th>h11</th>
<th>h12</th>
</tr>
</thead>
<tbody>
<tr>
<td>h00</td>
<td>o0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>h01</td>
<td>o1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>h02</td>
<td>o2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

- t0 = the
- t1 = cat
- t2 = sat

Task-Specific Classifier

Linear Combination
## ELMo Evaluation

<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>
ELMo
Evaluation

"Our Baseline"

Each task has a custom neural network model as a starting point

E.g., sentiment analysis model might be an RNN

t_0 = the
t_1 = food
t_2 = sucks
ELMo

Evaluation

negative

add

O₀

O₁

O₂

h₁₀

h₁₁

h₁₂

h₀₀

h₀₁

h₀₂

e₀

e₁

e₂

t₀ = the
t₁ = food
t₂ = sucks

“Baseline+ELMo”

Add linear combo of ELMo embeddings as starting point

Baseline+ELMo

Add linear combo of ELMo embeddings as starting point
ELMo Evaluation

Add linear combo of ELMo embeddings as starting point

Evaluation negative

"Baseline+ELMo"
ELMo

Evaluation

O₀

h₀₀

e₀

t₀ = the

h₀₁

e₁

t₁ = food

h₀₂

e₂

t₂ = sucks

O₁

h₁₁

O₂

h₁₂

"Baseline+ELMo"

Add linear combo of ELMo embeddings as starting point
<table>
<thead>
<tr>
<th>Source</th>
<th>Nearest Neighbors</th>
</tr>
</thead>
<tbody>
<tr>
<td>GloVe</td>
<td>play, 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 {…} 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 {…} {…} 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>

Table 4: Nearest neighbors to “play” using GloVe and the context embeddings from a biLM.
Topics

- Contextualized Word Representations:
  - ELMo
  - BERT
  - GPT
- Finetuning and Other Transfer Methods
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin  Ming-Wei Chang  Kenton Lee  Kristina Toutanova
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com
BERT

- Architecture: Deep Transformer (small = 12, large = 24)
- Layer 0 = wordpiece embeddings
- Trained on masked language modeling + next-sentence prediction
- Typically finetuned by updating all parameters (though there are other strategies)
Big (12 or 24) stack of transformer blocks, just like we discussed last lecture.
BERT

- Architecture: Deep Transformer (small = 12, large = 24)
- Layer 0 = wordpiece embeddings
- Trained on masked language modeling + next-sentence prediction
- Typically finetuned by updating all parameters (though there are other strategies)
BERT
Masked Language Modeling

• Traditional language modeling:
  • The cat sat on ???
  • The cat sat on the ???
  • etc.
• Masked Language Modeling (MLM)
  • The cat sat on the [MASK]
  • The [MASK] sat on the mat
  • etc.
The cat sat on the mat.
He stretched and yawned and went to sleep.

BERT
Masked Language Modeling

The cat sat on the mat.
He stretched and yawned and went to sleep.


Special tokens
BERT
Masked Language Modeling

The cat sat on the mat.
He stretched and yawned and went to sleep.

[MASK] hides words so that they have to be predicted based on context
[SEP] = tells the model where the sentence boundary is
[CLS] = Used for the “next sentence prediction” training objective (more soon)
BERT
Masked Language Modeling

[CLS] the [MASK] sat [SEP] he was [MASK]
**BERT**

Masked Language Modeling

- Choose 15% of words to me “masked”. If a word is “masked”
  - 80% of the time, replace with the token `[MASK]`
  - 10% of the time, replace with a random token
  - 10% of the time, do nothing
- At training time, model has to predict the masked words
- The above weirdness is so that the model learns to encode all words, since `[MASK]` won’t occur at test time
BERT
Next Sentence Prediction

• In addition to MLM, model has to predict whether or not the two sentences are consecutive
• Like SkipGram but for sentences
• Intended to encourage discourse structure
• 50% of time, sentence B actually follows sentence A in the corpus, 50% of the time, sentence B is a random sentence
BERT
Masked Language Modeling

True

[CLS] the [MASK] sat [SEP] he was [MASK]
Bert
Next Sentence Prediction

<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>

**Token Embeddings**

<table>
<thead>
<tr>
<th>[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>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

**Segment Embeddings**

<table>
<thead>
<tr>
<th>E_A</th>
<th>E_A</th>
<th>E_A</th>
<th>E_A</th>
<th>E_A</th>
<th>E_A</th>
<th>E_B</th>
<th>E_B</th>
<th>E_B</th>
<th>E_B</th>
<th>E_B</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

**Position Embeddings**

<table>
<thead>
<tr>
<th>E_0</th>
<th>E_1</th>
<th>E_2</th>
<th>E_3</th>
<th>E_4</th>
<th>E_5</th>
<th>E_6</th>
<th>E_7</th>
<th>E_8</th>
<th>E_9</th>
<th>E_{10}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
BERT
Next Sentence Prediction

CLS token used for binary prediction in NSP task.
BERT

- Architecture: Deep Transformer (small = 12, large = 24)
- Layer 0 = wordpiece embeddings
- Trained on masked language modeling + next-sentence prediction
- Typically finetuned by updating all parameters (though there are other strategies)
BERT
Masked Language Modeling

True
o
h

CLS
the

[MASK]
cat

h

sat

[SEP]

h
he

was

[MASK]
BERT
Finetuning on Sentiment Analysis

[CLS] the food was absolutely disgusting! :-0
In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called “showers”.

What causes precipitation to fall? **gravity**

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail? **graupe**

Where do water droplets collide with ice crystals to form precipitation? **within a cloud**
BERT
Finetuning on SQUAS

[CLS] who died? [SEP] John died on Friday
# BERT Evaluation

<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></td>
<td>392k</td>
<td>363k</td>
<td>108k</td>
<td>67k</td>
<td>8.5k</td>
<td>5.7k</td>
<td>3.5k</td>
<td>2.5k</td>
<td></td>
</tr>
<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>79.8</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>87.4</td>
<td>91.3</td>
<td>45.4</td>
<td>80.0</td>
<td>82.3</td>
<td>56.0</td>
<td>75.1</td>
</tr>
<tr>
<td>BERT&lt;sub&gt;BASE&lt;/sub&gt;</td>
<td>84.6/83.4</td>
<td>71.2</td>
<td>90.5</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&lt;sub&gt;LARGE&lt;/sub&gt;</td>
<td>86.7/85.9</td>
<td>72.1</td>
<td>92.7</td>
<td>94.9</td>
<td>60.5</td>
<td>86.5</td>
<td>89.3</td>
<td>70.1</td>
<td>82.1</td>
</tr>
</tbody>
</table>
Topics

• Transformers Cont. (+ any questions you have!)
• What is pretraining?
• **Contextualized Word Representations:**
  • ELMo
  • BERT
  • GPT
• Finetuning and Other Transfer Methods
Improving Language Understanding by Generative Pre-Training

Alec Radford
OpenAI
alec@openai.com

Karthik Narasimhan
OpenAI

Tim Salimans
OpenAI

Ilya Sutskever
OpenAI

Language Models are Unsupervised Multitask Learners

Alec Radford *1  Jeffrey Wu *1  Rewon Child 1  David Luan 1  Dario Amodei **1  Ilya Sutskever **1

Language Models are Few-Shot Learners

Tom B. Brown*  Benjamin Mann*  Nick Ryder*  Melanie Subbiah*
Jared Kaplan†  Prafulla Dhariwal  Arvind Neelakantan  Pranav Shyam  Girish Sastry
Amanda Askell  Sandhini Agarwal  Ariel Herbert-Voss  Gretchen Krueger  Tom Henighan
GPT

- Architecture: Deep Transformer
- Layer 0 = BPE
- Trained on vanilla language modeling
- Notable because the recent versions (GPT-3) are HUGE, and very impressive
- Typically finetuned by updating all parameters (for the small models) or (for the large models) “prompting” (more soon)
Figure 1: (left) Transformer architecture and training objectives used in this work. (right) Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.
Topics

- Transformers Cont. (+ any questions you have!)
- What is pretraining?
- **Contextualized Word Representations:**
  - ELMo
  - BERT
  - GPT
- **Finetuning and Other Transfer Methods**
Pretraining

Transferring pretrained representations

• Frozen: Just pool representations and train a new classifier on top
• Finetuning: Just treat pretraining as a good initialization. On new task, continue to update all parameters
• Parameter-Efficient or “Prompt” Tuning: Just update a small number of parameters at the bottom of the network
• Adapters: Just update a small number of parameters throughout the network
• Zero-Shot or “In-Context” Learning: Cast all tasks as an instance of the task the model was trained on (e.g., language modeling)
Pretraining

Transferring pretrained representations

- Frozen: Just pool representations and train a new classifier on top
- Finetuning: Just treat pretraining as a good initialization. On new task, continue to update all parameters
- Parameter-Efficient or “Prompt” Tuning: Just update a small number of parameters at the bottom of the network
- Adapters: Just update a small number of parameters throughout the network
- Zero-Shot or “In-Context” Learning: Cast all tasks as an instance of the task the model was trained on (e.g., language modeling)
Pretraining
Transferring frozen pretrained representations

Task: Sentiment Analysis
Input: “the food sucks”
Expected: Negative
Pretraining
Transferring frozen pretrained representations

Freeze the parameters (no more updating).

Task: Sentiment Analysis
Input: “the food sucks”
Expected: Negative
Pretraining

Transferring frozen pretrained representations

Pool some representations (e.g., final layer or CLS)

Task: Sentiment Analysis
Input: “the food sucks”
Expected: Negative
Pretraining
Transferring frozen pretrained representations

Train a new network on top of the pooled representation (optionally, backprop through the pooled representation).

Task: Sentiment Analysis
Input: “the food sucks”
Expected: Negative
Pretraining

Transferring pretrained representations

• Frozen: Just pool representations and train a new classifier on top
• Finetuning: Just treat pretraining as a good initialization. On new task, continue to update all parameters
Pretraining

Finetuning

Task: Sentiment Analysis
Input: “the food sucks”
Expected: Negative
Pretraining
Finetuning

Adjust output layers to reflect target task
(for BERT, can use CLS for this)

Task: Sentiment Analysis
Input: “the food sucks”
Expected: Negative

Diagram:
- 0 or 1
- \( h_{00} \)
- \( h_{01} \)
- \( h_{02} \)
- \( e_0 \) (the)
- \( e_1 \) (food)
- \( e_2 \) (sucks)
Pretraining

Finetuning

Keep training network, backproping through everything as needed.

Task: Sentiment Analysis
Input: “the food sucks”
Expected: Negative
Pretraining
Transferring pretrained representations

• Frozen: Just pool representations and train a new classifier on top
• Finetuning: Just treat pretraining as a good initialization. On new task, continue to update all parameters
• Parameter-Efficient or “Prompt” Tuning: Just update a small number of parameters at the bottom of the network
• Zero-Shot or “In-Context” Learning: Cast all tasks as an instance of the task the model was trained on (e.g., language modeling)
Task: Sentiment Analysis
Input: “the food sucks”
Expected: Negative
Pretraining
Parameter-Efficient ("Prompt") Tuning

Adjust output layers to reflect target task

Task: Sentiment Analysis
Input: “the food sucks”
Expected: Negative
Pretraining

Parameter-Efficient ("Prompt") Tuning

Freeze everything except a small number of parameters (e.g., the embeddings)

Task: Sentiment Analysis
Input: “the food sucks”
Expected: Negative
Pretraining

Transferring pretrained representations

- Frozen: Just pool representations and train a new classifier on top
- Finetuning: Just treat pretraining as a good initialization. On new task, continue to update all parameters
- Parameter-Efficient or “Prompt” Tuning: Just update a small number of parameters at the bottom of the network
- Adapters: Just update a small number of parameters throughout the network
- Zero-Shot or “In-Context” Learning: Cast all tasks as an instance of the task the model was trained on (e.g., language modeling)
Pretraining
Adapters

Task: Sentiment Analysis
Input: “the food sucks”
Expected: Negative
Pretraining

Parameter-Efficient (“Prompt”) Tuning

Add a few parameters at each layer to tune, freeze the rest.

Task: Sentiment Analysis
Input: “the food sucks”
Expected: Negative
Pretraining
Transferring pretrained representations

- Frozen: Just pool representations and train a new classifier on top
- Finetuning: Just treat pretraining as a good initialization. On new task, continue to update all parameters
- Parameter-Efficient or “Prompt” Tuning: Just update a small number of parameters at the bottom of the network
- Adapters: Just update a small number of parameters throughout the network
- Zero-Shot or “In-Context” Learning: Cast all tasks as an instance of the task the model was trained on (e.g., language modeling)
Pretraining
Zero-Shot Transfer

Task: Sentiment Analysis
Input: “the food sucks”
Expected: Negative
Pretraining
Zero-Shot Transfer

Task: Sentiment Analysis
Input: “the food sucks”
Expected: Negative

Recast target task to look like language modeling...
Pretraining
Zero-Shot Transfer

Task: Sentiment Analysis
Input: “the food sucks”
Expected: Negative

Frozen, Pretrained Model

P(“positive”) vs. P(“negative”)

Is the following review positive or negative?
This food sucks.

Recast target task to look like language modeling...
Pretraining
In-Context Learning

Task: Sentiment Analysis
Input: “the food sucks”
Expected: Negative

P(“positive”) vs. P(“negative”)

Frozen, Pretrained Model

Is the following review positive or negative?
Awesome place -> positive
Would not recommend -> negative
My absolute fav -> positive
This food sucks. ->

Optionally, offering some examples to guide the model on task and format
Pretraining
How well do these methods work?

- In general, very well.
- Most evaluations focus on task performance, models do very well on that front.
- **Finetuning**: Models perform better and require less data to learn the target task. Still require 100s or 1000s of examples, typically. And can result in “catastrophic forgetting”.
- **Parameter-Efficient/Adapters**: Newer, so still being explored. But current results suggest very large models can be adapted in ~15 minutes with O(100) examples.
- **Zero-Shot/In-Context Learning**: More controversial. Results are high variance, depend on exact wording of the prompts. Unclear what was seen during training so hard to know how much they are truly learning and generalizing vs. parroting.