CS224C: NLP for CSS

Deep Learning Highlights

Diyi Yang
Stanford CS
Lecture Overview

✦ BERT
✦ Text Generation
✦ Prompting Based Approach
BERT for Classification

Bidirectional encoder representations from Transformers

**Context is the key**

\[ p(\text{play} \mid \text{Elmo and Cookie Monster play a game}) \neq p(\text{play} \mid \text{The Broadway play premiered yesterday}) \]
BERT demonstrated strong performances on a wide range of NLP tasks!

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
Masked Language Modeling

Mask out k% of the input words, and then predict the masked words (k=15%)

Input: The man went to the [MASK]. He bought a [MASK] of milk.
Labels: [MASK] = store; [MASK] = gallon.
Next Sentence Prediction

To learn relationship between sentences, predict whether Sentence B is actual sentence that proceeds Sentence A, or a random sentence.

Sentence A = The man went to the store.
Sentence B = He bought a gallon of milk.
Label = IsNextSentence

Sentence A = The man went to the store.
Sentence B = Penguins are flightless.
Label = NotNextSentence
## Input Representation

Each token is the sum of three embeddings

<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>
<tbody>
<tr>
<td></td>
<td>(E_{[CLS]})</td>
<td>(E_{my})</td>
<td>(E_{dog})</td>
<td>(E_{is})</td>
<td>(E_{cute})</td>
<td>(E_{[SEP]})</td>
<td>(E_{he})</td>
<td>(E_{likes})</td>
<td>(E_{play})</td>
<td>(E_{# #ing})</td>
<td>(E_{[SEP]})</td>
</tr>
<tr>
<td>Token Embeddings</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<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>
Model Architecture: Transformer

Multi-headed self attention to model context

Feed-forward layers to compute non-linear hierarchical features

Positional embeddings to allow model to learn relative positioning

Link: https://nlp.stanford.edu/seminar/details/jdevlin.pdf
1 - **Semi-supervised** training on large amounts of text (books, wikipedia..etc).

The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.

**Semi-supervised Learning Step**

**Model:** BERT

**Dataset:**

**Objective:** Predict the masked word (language modeling)

---

2 - **Supervised** training on a specific task with a labeled dataset.

**Supervised Learning Step**

**Model:**

BERT (pre-trained in step #1)

**Dataset:**

<table>
<thead>
<tr>
<th>Email message</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buy these pills</td>
<td>Spam</td>
</tr>
<tr>
<td>Win cash prizes</td>
<td>Spam</td>
</tr>
<tr>
<td>Dear Mr. Atreides, please find attached…</td>
<td>Not Spam</td>
</tr>
</tbody>
</table>

75% Spam

25% Not Spam

---

https://jalammar.github.io/illustrated-bert/
Empirical Results from BERT

### GLUE 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></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>79.9</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&lt;sub&gt;BASE&lt;/sub&gt;</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&lt;sub&gt;LARGE&lt;/sub&gt;</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>

**MultiNLI**
- **Premise:** Hills and mountains are especially sanctified in Jainism.
- **Hypothesis:** Jainism hates nature.
- **Label:** Contradiction

**CoLa**
- **Sentence:** The wagon rumbled down the road.
- **Label:** Acceptable

- **Sentence:** The car honked down the road.
- **Label:** Unacceptable
How to use BERT for Classification
(e.g., sentiment, fact-checking, rumors)
Pros and Cons of BERT for CSS

👍 Strong prediction performance

👍 Fine-tuning on top of pertained representations

👎 Prediction and representation can be hard to interpret

👎 Subject to biases in these learned representations

👎 Require computational resources
Resources to checkout

https://huggingface.co/docs/transformers/model_doc/bert
Text Generation, GPT and their Friends
Very Classical Example

“Sequence to Sequence Learning with Neural Networks” 2014
Very Classical Example

“Sequence to Sequence Learning with Neural Networks” 2014
K-dimensional vector for the context

Condition on word generated so far
The entire input is summarized in this one single vector.

In a vanilla seq2seq, the decoder state depends just on the previous state and the previous output.
Training Objective

The probability of generating $y$ given the source sentence $x$

$$\log p(y \mid x) = \sum_{j=1}^{m} \log p(y_j \mid y_{<j}, s)$$

Where $p(y_j \mid y_{<j}, s) = \text{softmax}(g(h_j))$ and $h_j = f(h_{j-1}, s)$
Training Objective

The probability of generating $y$ given the source sentence $x$

$$\log p(y | x) = \sum_{j=1}^{m} \log p(y_j | y_{<j}, s)$$

Where $p(y_j | y_{<j}, s) = \text{softmax}(g(h_j))$ and $h_j = f(h_{j-1}, s)$

As in other RNNs, we can train by minimizing the loss between what we predict at each time step and the truth
Problem with Vector Sentence Encoding

Fixed sized representation degrades as sentence length increases

Attention is all you need?

Attention significantly improves performance (in many applications)
  Allow decoder to focus on certain parts of the source

Attention solves the bottleneck problem
  Allow decoder to look directly at source; bypass bottleneck

Attention provides some interpretability
  By inspecting attention distribution, we can see what the decoder was focusing on
Highlighting GPT-2

Transformer decoder blocks only.

Utilizing GPT-2: Machine Translation

Utilizing GPT-2: Summarization

<table>
<thead>
<tr>
<th>Training Dataset</th>
<th>Article #1 tokens</th>
<th>Article #2 tokens</th>
<th>Article #3 tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;summarize&gt;</td>
<td>&lt;summarize&gt;</td>
<td>&lt;summarize&gt;</td>
</tr>
<tr>
<td>Output #1</td>
<td>Position #114</td>
<td>Time step #1</td>
<td></td>
</tr>
<tr>
<td>Output #2</td>
<td>Position #115</td>
<td>Time step #2</td>
<td></td>
</tr>
</tbody>
</table>

Grover: A State-of-the-Art Defense against Neural Fake News

As most of my readers know, I’m an optimist.

This belief applies across my life, and to various investments as well. So I am intrigued by the success of cryptocurrencies, such as Bitcoin and Ethereum. The competition they are putting up against the gold standard looks insane, as Bitcoin goes off to the races.

There’s no way to fully understand what’s going on in the crypto world — and I am not even sure anyone could if you tried to. Still, I can tell you that Bitcoin’s recent surge is really an opportunity to buy long-term real assets.

Cryptocurrencies are new and don’t even have a useful underlying technology. They will probably fail, probably sooner than later. If people forget about them quickly, it is likely to be because the underlying technology will finally mature and win out. We don’t even know whether that will happen in a generation or maybe a century, but it’s still possible it might.

Why invest in crypto now, if it’s unlikely to last?
For each article, three annotators evaluated style, content, and the overall trustworthiness; 100 articles of each category were used.

The results show that propaganda generated by Grover is rated more plausible than the original human-written propaganda.

Reflection on Neural Generation

✧ The era of neural discrimination
✧ Release of generators is critical
✧ Additional threat models
✧ Machine-generated real news?
✧ What should platforms do?

How about ChatGPT?

Examples

"Explain quantum computing in simple terms" →

"Got any creative ideas for a 10 year old's birthday?" →

"How do I make an HTTP request in Javascript?" →

Capabilities

Remembers what user said earlier in the conversation

Allows user to provide follow-up corrections

Trained to decline inappropriate requests

Limitations

May occasionally generate incorrect information

May occasionally produce harmful instructions or biased content

Limited knowledge of world and events after 2021
how should we teach students to understand chatgpt
Are you asking how to teach students about the chatbot GPT (Generative Pre-training Transformer)? If so, here are a few suggestions for teaching students about chatbots like GPT:

1. **Introduce the concept of chatbots and their capabilities.** Chatbots are computer programs designed to simulate conversation with human users through the use of artificial intelligence.

2. **Explain how chatbots work.** Chatbots use natural language processing (NLP) to understand and respond to user input. They may also use machine learning to improve their responses over time.

3. **Discuss the limitations of chatbots.** Chatbots can only respond based on the information they have been programmed with and may not be able to understand or respond to every user input.

4. **Explore the potential uses for chatbots.** Chatbots can be used in a variety of settings, including customer service, education, and entertainment.

5. **Consider the ethical implications of chatbots.** As chatbots become more advanced, it is important to consider the potential consequences of their use, including the potential loss of jobs and the potential for misuse.

6. **Encourage students to experiment with chatbots.** Students can try building their own chatbots or interacting with existing ones to get a better understanding of how they work.
Prompt-based Learning

<table>
<thead>
<tr>
<th>Paradigm</th>
<th>Engineering</th>
</tr>
</thead>
<tbody>
<tr>
<td>a. Fully Supervised Learning</td>
<td>Features (e.g. word identity, part-of-speech, sentence length)</td>
</tr>
<tr>
<td>(Non-Neural Network)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>b. Fully Supervised Learning</td>
</tr>
<tr>
<td>(Neural Network)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c. Pre-train, Fine-tune</td>
</tr>
<tr>
<td></td>
<td>d. Pre-train, Prompt, Predict</td>
</tr>
</tbody>
</table>
Prompt Paradigm

Prompt Addition:
Given input $x$, transform it into prompt $x'$ through two steps: (1) define a template with two slots, input $[x]$ and answer $[z]$; (2) fill in the input slot $[x]$

Answer Search
Search for the highest-scoring text $\hat{z}$ that maximizes the score of the LM

Answer Mapping
Go from the highest-scoring answer $\hat{z}$ to the highest-scoring output $\hat{y}$
Example: Sentiment Classification

Input: \( x = "I\ love\ this\ movie" \)

Template: \([x]\) Overall, it was a \([z]\) movie

Prompting: \( x' = "I\ love\ this\ movie.\ Overall\ it\ was\ a\ [z]\ movie."\)

Predicting: \( x' = "I\ love\ this\ movie.\ Overall\ it\ was\ a\ fantastic\ movie."\)

Mapping: fantastic \( \Rightarrow \) Positive
<table>
<thead>
<tr>
<th>Type</th>
<th>Task</th>
<th>Input ([X])</th>
<th>Template</th>
<th>Answer ([Z])</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text CLS</td>
<td>Sentiment</td>
<td>I love this movie.</td>
<td>[X] The movie is [Z].</td>
<td>great fantastic ...</td>
</tr>
<tr>
<td></td>
<td>Topics</td>
<td>He prompted the LM.</td>
<td>[X] The text is about [Z].</td>
<td>sports science ...</td>
</tr>
<tr>
<td></td>
<td>Intention</td>
<td>What is taxi fare to Denver?</td>
<td>[X] The question is about [Z].</td>
<td>quantity city ...</td>
</tr>
<tr>
<td>Text-span CLS</td>
<td>Aspect Sentiment</td>
<td>Poor service but good food.</td>
<td>[X] What about service? [Z].</td>
<td>Bad Terrible ...</td>
</tr>
<tr>
<td></td>
<td>NLI</td>
<td>[X1]: An old man with ...</td>
<td>[X1]? [Z], [X2]</td>
<td>Yes No ...</td>
</tr>
<tr>
<td></td>
<td>[X2]: A man walks ...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Text-pair CLS</td>
<td>NLI</td>
<td>NLI</td>
<td>NLI</td>
<td>Yes No ...</td>
</tr>
<tr>
<td></td>
<td>[X1]: Mike went to Paris.</td>
<td>[X1]? [Z], [X2]</td>
<td>organization location ...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[X2]: Paris</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tagging</td>
<td>NER</td>
<td>[X1]: Mike went to Paris.</td>
<td>[X1] [X2] is a [Z] entity.</td>
<td>The victim ... A woman ...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Text Generation</td>
<td>Summarization</td>
<td>Las Vegas police ...</td>
<td>[X] TL;DR: [Z]</td>
<td>The victim ... A woman ...</td>
</tr>
<tr>
<td></td>
<td>Translation</td>
<td>Je vous aime.</td>
<td>French: [X] English: [Z]</td>
<td>I love you. I fancy you. ...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Design Considerations for Prompting

Pre-trained Model Choice
Prompt Engineering
Answer Engineering
Expanding the Paradigm
Prompt-based Training Strategies