CS388: Natural Language Processing

Lecture 11: Understanding In-Context Learning

Greg Durrett

The University of Texas at Austin
‣ Project 3 released Thursday

‣ Project proposals due Thursday

‣ Can be >1 page if needed

‣ Most important: have a detailed plan for models, datasets, and experiments, so we can evaluate for feasibility. Include related work!

‣ For reproduction: lots of types of papers are okay, just make sure the paper isn’t trivial. You can plan for a reproduction with minor extension beyond what was done before
Recap: Dataset Bias

- “Tough” datasets for tasks like QA may feature spurious correlations (e.g., “where” question is always a location and the model can guess a relevant location and do quite well)

- Training strong models such as BERT on these datasets leads to poor generalization

- One debiasing technique:

\[
\mathcal{L}(\theta_d) = -(1 - p_b^{(i,c)}) y(i) \cdot \log p_d
\]

- probability under a copy of the model trained for a few epochs on a small subset of data (bad model)
This Lecture

- Prompting: best practices and why it works
  - Zero-shot prompting: role of the prompt
  - Few-shot prompting (in-context learning): characterizing demonstrations
- Understanding in-context learning
  - ICL can learn linear regression
  - Induction heads and mechanistic interpretability
Zero-shot Prompting
Zero-shot Prompting

- Single unlabeled datapoint $x$, want to predict label $y$

  $x = \text{The movie’s acting could’ve been better, but the visuals and directing were top-notch.}$

- Wrap $x$ in a template we call a verbalizer $v$

  **Review:** The movie’s acting could’ve been better, but the visuals and directing were top-notch.

  Out of positive, negative, or neutral, this review is neutral.
Zero-shot Prompting

- Single unlabeled datapoint $x$, want to predict label $y$

  $$x = \text{The movie’s acting could’ve been better, but the visuals and directing were top-notch.}$$

- Wrap $x$ in a template we call a verbalizer $v$

  Review: The movie’s acting could’ve been better, but the visuals and directing were top-notch.

  On a 1 to 4 star scale, the reviewer would probably give this movie

  GPT-3

  3 stars.
Ways to do classification

- Generate from the model and read off the generation

  - What if you ask for a star rating and it doesn’t give you a number of stars but just says something else?

- Compare probs: “Out of positive, negative, or neutral, this review is _”
  Compare $P(\text{positive} \mid \text{context})$, $P(\text{neutral} \mid \text{context})$, $P(\text{negative} \mid \text{context})$

  - This constrains the model to only output a valid answer, and you can normalize these probabilities to get a distribution
Ways to do classification

\[(x, y) = ("A three-hour cinema master class.", "It was great.")\]

- Can also compute probabilities of **examples** given **labels**
  ("noisy channel" method)

Min et al. (2021)
Variability in Prompts

- Plot: large number of prompts produced by {manual writing, paraphrasing, backtranslation}
- x-axis: perplexity of the prompt. How natural is it? How much does it appear in the pre-training data?
- y-axis: task performance

Gonen et al. (2022)
### Variability in Prompts

<table>
<thead>
<tr>
<th>Task</th>
<th>Perplexity-score corr. Pearson</th>
<th>Perplexity-acc corr. Pearson</th>
<th>Avg Acc</th>
<th>Acc 50%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>-0.41</strong></td>
<td><strong>-0.53</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Antonyms</td>
<td>-0.15</td>
<td>-0.14</td>
<td>-0.04</td>
<td>-0.02</td>
</tr>
<tr>
<td>GLUE Cola</td>
<td>-0.24</td>
<td><strong>-0.26</strong></td>
<td>-0.20</td>
<td>-0.18</td>
</tr>
<tr>
<td>Newspop</td>
<td><strong>-0.63</strong></td>
<td><strong>-0.68</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AG News</td>
<td><strong>-0.35</strong></td>
<td><strong>-0.40</strong></td>
<td>0.14</td>
<td><strong>0.20</strong></td>
</tr>
<tr>
<td>IMDB</td>
<td><strong>-0.50</strong></td>
<td><strong>-0.44</strong></td>
<td><strong>-0.51</strong></td>
<td><strong>-0.42</strong></td>
</tr>
<tr>
<td>DBpedia</td>
<td>-0.14</td>
<td>-0.19</td>
<td><strong>-0.30</strong></td>
<td><strong>-0.32</strong></td>
</tr>
<tr>
<td>Emotion</td>
<td><strong>-0.19</strong></td>
<td>0.07</td>
<td>0.18</td>
<td><strong>0.23</strong></td>
</tr>
<tr>
<td>Tweet Offensive</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*OPT-175B: average of best 50% of prompts is much better than average over all prompts*

Gonen et al. (2022)
Prompt Optimization

- A number of methods exist for searching over prompts (either using gradients or black-box optimization)

- Most of these do not lead to dramatically better results than doing some manual engineering/hill-climbing (and they may be computationally intensive)

- Nevertheless, the choice of prompt is very important for zero-shot settings! We will see more next time.

- In two lectures: models that are trained to do better at prompts (RLHF)
Few-shot Prompting
Few-shot Prompting

- Form “training examples” from \((x, y)\) pairs, verbalize them (can be lighter-weight than zero-shot verbalizer)
- Input to GPT-3: \(v(x_1) v(y_1) v(x_2) v(y_2) \ldots v(x_{\text{test}})\)

Review: The cinematography was stellar; great movie!
Sentiment (positive or negative): positive

Review: The plot was boring and the visuals were subpar.
Sentiment (positive or negative): negative

Review: The movie’s acting could’ve been better, but the visuals and directing were top-notch.
Sentiment (positive or negative): positive
What can go wrong?

Review: The movie was great!
Sentiment: positive

Review: I thought the movie was alright; I would've seen it again.
Sentiment: positive

Review: The movie was pretty cool!
Sentiment: positive

Review: Pretty decent movie!
Sentiment: positive

Review: The movie had good enough acting and the visuals were nice.
Sentiment: positive

Review: There wasn't anything the movie could've done better.
Sentiment: positive

Review: Okay movie but could've been better.
Sentiment: positive

GPT-3 positive
What can go wrong?

- All one training label — model sees extremely skewed distribution

- What if we take random sets of training examples? There is quite a bit of variance on basic classification tasks

- Note: these results are with basic GPT-3 and not Instruct-tuned versions of the model. This issue has gotten a lot better

Zhao et al. (2021)
What can go wrong?

- Varies even across permutations of training examples.
- x-axis: different collections of train examples.
- y-axis: sentiment accuracy. Boxes represent results over different permutations of the data.

Zhao et al. (2021)
What can go wrong?

- Having unbalanced training sets leads to high “default” probabilities of positive; that is, if we feed in a null $x_{\text{test}}$

- Solution: “calibrate” the model by normalizing by that probability of null $x_{\text{test}}$

- Leads to higher performance; not necessarily crucial with prompt-tuned models

Zhao et al. (2021)
Rethinking Demonstrations

‣ Surprising result: how necessary even are the demonstrations?

‣ Using random labels does not substantially decrease performance??

Min et al. (2022)
Having even mislabeled demonstrations is much better than having no demonstrations, indicating that the form of the demonstrations is partially responsible for in-context learning.

Min et al. (2022)
Results: HELM

- So, how much better is few-shot compared to zero-shot?
- Each line is a different LM
- More in-context examples generally leads to better performance
- What do we see here?
Results: HELM

- What trends do these show?

Liang et al. (2022)
Results: BIG-bench

Srivastava et al. (2022)

BIG-G: internal Google model.
Understanding ICL: Regression
Linear Regression

- Input space is of the form \([y, x]\), with the "unused" components set to 0
- See if we can learn regression: given \((x, y)\) pairs, learn a linear predictor \(f(x) = w^T x\). That is, ground truth is a linear function (synthetic task)
- Equivalent to minimizing the following loss:

\[
\sum_i \mathcal{L}(w^T x_i, y_i) + \lambda \|w\|^2_2
\]

minimized by: \(w^* = (X^T X + \lambda I)^{-1} X^T y\)

Akyürek et al. (2022)
Question 1: can a Transformer learn to do linear regression?

- If we train it to do this task on many examples, does it successfully learn to do “ICL” linear regression on new instances?

- If so, there are several different “algorithms” it might correspond to!

Question 2: can we inspect what algorithm actually gets implemented?
Most of these proofs (and other papers in this space) rely on Transformers being able to perform several kinds of operations

\[ \text{mov}(H; s, t, i, j, i', j'): \text{selects the entries of the } s^{th} \text{ column of } H \text{ between rows } i \text{ and } j, \text{ and copies them into the } t^{th} \text{ column (} t \geq s \text{) of } H \text{ between rows } i' \text{ and } j', \text{ yielding the matrix:} \]

\[
\begin{bmatrix}
H_{;:t} & H_{;i-1,t} & H_{;:t+1} \\
H_{i':;s} & H_{i':;j'} & H_{;:t+1} \\
H_{j,t} & H_{j':;t} & H_{;:t+1}
\end{bmatrix}.
\]

How can this be implemented?
What does the attention need to do?

Akyürek et al. (2022)
mov\((H; s, t, i, j, i', j')\): selects the entries of the \(s\)th column of \(H\) between rows \(i\) and \(j\), and copies them into the \(t\)th column \((t \geq s)\) of \(H\) between rows \(i'\) and \(j'\), yielding the matrix:

\[
\begin{bmatrix}
H_{:,t} & H_{i-1,t} & H_{:,t+1:} \\
H_{i-1:,t} & H_{i':j',s} & H_{:,t+1:} \\
H_{j:,t} & & \\
\end{bmatrix}.
\]

mul\((H; a, b, c, (i, j), (i', j'), (i'', j''))\): in each column \(h\) of \(H\), interprets the entries between \(i\) and \(j\) as an \(a \times b\) matrix \(A_1\), and the entries between \(i'\) and \(j'\) as a \(b \times c\) matrix \(A_2\), multiplies these matrices together, and stores the result between rows \(i''\) and \(j''\), yielding a matrix in which each column has the form \([h_{:,i''-1}, A_1 A_2, h_{j'':,}]^\top\).

- Several more operations as well
Linear Regression

**Theorem 1.** A transformer can compute Eq. (11) (i.e. the prediction resulting from single step of gradient descent on an in-context example) with constant number of layers and $O(d)$ hidden space, where $d$ is the problem dimension of the input $x$. Specifically, there exist transformer parameters $\theta$ such that, given an input matrix of the form:

$$H^{(0)} = \begin{bmatrix} \cdots & 0 & y_i & 0 & \cdots \\ \cdots & x_i & 0 & x_n & \cdots \end{bmatrix},$$

(12)

the transformer’s output matrix $H^{(L)}$ contains an entry equal to $w^T x_n$ (Eq. (11)) at the column index where $x_n$ is input.

- Also another update possible based on rank-one updates (Sherman-Morrison)

Akyürek et al. (2022)
Proof of Theorem

The operations for 1-step SGD with single exemplar can be expressed as following chain (please see proofs for the Transformer implementation of these operations (Lemma 1) in Appendix C):

• \text{mov}( ; 1, 0, (1, 1 + d), (1, 1 + d)) \quad \text{(move } x) \quad \text{ (move } x)

• \text{aff}( ; (1, 1 + d), (), (1 + d, 2 + d), W_1 = w) \quad \text{(w}^\top x) \quad \text{(w}^\top x)

• \text{aff}( ; (1 + d, 2 + d), (0, 1), (2 + d, 3 + d), W_1 = I, W_2 = -I) \quad (w^\top x - y) \quad (w^\top x - y)

• \text{mul}( ; d, 1, 1, (1, 1 + d), (2 + d, 3 + d), (3 + d, 3 + 2d)) \quad (x(w^\top x - y)) \quad (x(w^\top x - y))

• \text{aff}( ; (), (), (3 + 2d, 3 + 3d), b = w,) \quad \text{(write } w) \quad \text{(write } w)

• \text{aff}( ; (3 + d, 3 + 2d), (3 + 2d, 3 + 3d), (3 + 3d, 3 + 4d), W_1 = I, W_2 = -\lambda) \quad (x(w^\top x - y) - \lambda w) \quad (x(w^\top x - y) - \lambda w)

• \text{aff}( ; (3 + 2d, 3 + 3d), (3 + 3d, 3 + 4d), (3 + 2d, 3 + 3d), W_1 = I, W_2 = -2\alpha, ) \quad (w') \quad (w')

• \text{mov}( ; 2, 1, (3 + 2d, 3 + 3d), (3 + 2d, 3 + 3d)) \quad \text{(move } w') \quad \text{(move } w')

• \text{mul}( ; 1, d, 1, (3 + 2d, 3 + 3d), (1, 1 + d), (3 + 3d, 4 + 3d)) \quad \text{(w'}^\top x_2) \quad \text{(w'}^\top x_2)

Akyürek et al. (2022)
Linear Regression

- Squared prediction difference: L2 between different predictors
- When no noise: ICL matches ordinary least square (OLS) almost exactly
Linear Regression

- Squared prediction difference: L2 between different predictors

- What gets learned changes with depth. Low-depth: more like GD. Medium-depth: more like ridge. High-depth: OLS
Bayesian Interpretation

1. Pretraining documents are conditioned on a latent concept (e.g., biographical text)

Albert Einstein was a German theoretical physicist, widely acknowledged to be one of the greatest physicists of all time. Einstein is best known for developing the theory of relativity, but he also ....

2. Create independent examples from a shared concept. If we focus on full names, wiki bios tend to relate them to nationalities.

Concept (e.g., wiki bio)

Input (x) | Output (y) | Delimiter
--- | --- | ---
Albert Einstein was | German | 
Mahatma Gandhi was | Indian | 
Marie Curie was | ? | ...

3. Concatenate examples into a prompt and predict next word(s). Language model (LM) implicitly infers the shared concept across examples despite the unnatural concatenation

Albert Einstein was German \n Mahatma Gandhi was Indian \n Marie Curie was

Xie et al. (2021)
Understanding ICL: Induction Heads and Mechanistic Interpretability
There are mechanisms in Transformers to do “fuzzy” or “nearest neighbor” versions of pattern completion, completing \([A^*][B^*] \ldots [A] \rightarrow [B]\), where \(A^* \approx A\) and \(B^* \approx B\) are similar in some space.

Olsson et al. want to establish that these mechanisms are responsible for good ICL capabilities.

We can find these heads and see that performance improves; can we causally link these?

Olsson et al. (2022)
Induction Heads

- Induction heads: a pair of attention heads in different layers that work together to copy or complete patterns.
- The first head copies information from the previous token into each token.
- Second attention head to attend to tokens based on what happened before them, rather than their own content. Likely to “look back” and copy next token from earlier.
- The two heads working together cause the sequence ...[A][B]...[A] to be more likely to be completed with [B].
Induction Heads

Step 1: Run each model / snapshot over the same set of multiple dataset examples, collecting one token’s loss per example.

Step 2: For each sample, extract the loss of a consistent token. Combine these to make a vector of losses per model / snapshot.

Step 3: The vectors are jointly reduced with principal component analysis to project them into a shared 2D space.

- Characterize performance by ICL score: loss(500th token) - loss(50th token) — average measure of how much better the model is doing later once it’s seen more of the pattern

- Can cluster models based on losses over time

Olsson et al. (2022)
Induction Heads

- Improvement in ICL (loss score) correlates with emergence of induction heads

One-layer model has no sudden improvement.
Models with more than one layer have a sudden improvement in in-c

One-layer model has no induction heads.
Models with more than one layer have induction heads form during
Induction Heads

Change architecture to promote induction heads => phase change happens earlier
If you remove induction heads, behavior changes dramatically.
Interpretability

- Lots of explanations for why ICL works — but these haven’t led to many changes in how Transformers are built or scaled

- Several avenues of inquiry: theoretical results (capability of these models), mechanistic interpretability, fully empirical (more like that next time)

- Many of these comparisons focus on GPT-3 and may not always generalize to other models
Takeaways

‣ Zero- and few-shot prompting are very powerful ways of specifying new tasks at inference time

‣ For zero-shot: form of the prompt matters, we’ll see more example next times when we look at chain-of-thought

‣ For few-shot: number and order of the examples matters, prompt matters a bit less

‣ Several analyses of why it works: it can learn to do regression and we know a bit about mechanisms that may be responsible for it