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:

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

  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 Promting

- 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 \( v \)
  \[ \text{Review: The movie's acting could've been better, but the visuals and directing were top-notch.} \]
  \[ \text{Out of positive, negative, or neutral, this review is} \]

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

\[(x, y) = \text{“A three-hour cinema master class.”, “It was great.”}\]

**Direct** \(P(y|x)\)

\[\text{Input} \quad \text{Output}\]

A three-hour cinema master class. \quad \text{It was great.}

**Channel** \(P(x|y) P(y) \propto P(x|y)\)

\[\text{LM} \quad \text{A three-hour cinema master class.}\]

- 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)

<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>Antonyms</td>
<td><strong>-0.41</strong></td>
<td><strong>-0.53</strong></td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>GLUE Cola</td>
<td>-0.15</td>
<td>-0.14</td>
<td>-0.04</td>
<td>-0.02</td>
</tr>
<tr>
<td>Newspaper</td>
<td><strong>-0.24</strong></td>
<td><strong>-0.26</strong></td>
<td><strong>-0.20</strong></td>
<td>-0.18</td>
</tr>
<tr>
<td>AG News</td>
<td><strong>-0.63</strong></td>
<td><strong>-0.68</strong></td>
<td><strong>-0.77</strong></td>
<td><strong>-0.81</strong></td>
</tr>
<tr>
<td>IMDB</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>DBpedia</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>Emotion</td>
<td>-0.14</td>
<td>-0.19</td>
<td><strong>-0.30</strong></td>
<td><strong>-0.32</strong></td>
</tr>
<tr>
<td>Tweet Offensive</td>
<td>-0.19</td>
<td>0.07</td>
<td>0.18</td>
<td><strong>0.23</strong></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

- 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) ... 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):

GPT-3
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: 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 $x_{test}$.
- Solution: "calibrate" the model by normalizing by that probability of null $x_{test}$.
- 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)

Rethinking Demonstrations

- 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?

Results: BIG-bench

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?

Akyürek et al. (2022)

---

• Most of these proofs (and other papers in this space) rely on Transformers being able to perform several kinds of operations.

\(\text{mov}(H; s, t, i, j, i', j')\): 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:

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

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

Akyürek et al. (2022)

---

• Several more operations as well

Akyürek et al. (2022)
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:

$$H^{(0)} = \begin{bmatrix} \ldots & 0 & y_1 & 0 & x_n & \ldots \end{bmatrix},$$

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):

- $\text{mov}(1, 0, (1, 1 + d), (1, 1 + d))$ (move $x$)
- $\text{aff}(1, 1 + d, 1, (1 + d, 2 + d), W_1 = w)$ ($w^T x$)
- $\text{aff}((1 + d, 2 + d), (0, 1), (2 + d, 3 + d), W_1 = I, W_2 = -I)$ ($w^T x - y$)
- $\text{mul}(1, d, 1, (1, 1 + d), (2 + d, 3 + d), (3 + d, 3 + 2d))$ ($x(w^T x - y)$)
- $\text{aff}((1, 1), (3 + 2d, 3 + 2d), b = w_1)$ (write $w$)
- $\text{aff}((3 + d, 3 + 2d), (3 + 2d, 3 + 3d), (3 + 3d, 3 + 4d), W_1 = I, W_2 = -\lambda) (x(w^T x - y) - \lambda w)$ ($w^T w$)
- $\text{aff}((3 + 2d, 3 + 3d), (3 + 3d, 3 + 4d), (3 + 2d, 3 + 3d), W_1 = I, W_2 = -2\alpha, 1)$ ($w^T x$)
- $\text{mov}(2, 1, (3 + 2d, 3 + 3d), (3 + 2d, 3 + 3d))$ (move $w$)
- $\text{mul}(1, d, 1, (3 + 2d, 3 + 3d), (1, 1 + d), (3 + 3d, 4 + 3d))$ ($w^T x_2$)

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).
2. Create independent examples from a shared concept. If we focus on full names, wiki bias tends to relate to nationalities.
3. Concatenate examples into a prompt and predict next word(s). Language model (LM) implicitly infers the shared concept across examples despite unnatural concatenation.

Understanding ICL: Induction Heads and Mechanistic Interpretability

Background: Transformer Circuits
- There are mechanisms in Transformers to do “fuzzy” or “nearest neighbor” versions of pattern completion, completing [A*][B*] ... [A] → [B], where A* ≈ A and B* ≈ 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?

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

Olsson et al. (2022)

- Can cluster models based on losses over time

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

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