CSE702 Week 8: Some Principles and Presentation Ideas

I. Let's jump off with an example where writing the ECU article led me to find an illustrative example of natural bias:

When should quantity Y be an unbiased predictor for quantity Z?

My "screening test" gives results on the scale of 100 coin flips: expectation 50, standard deviation 5. This aligns with the z-scale. For instance, a screening "Raw Outlier Index" (ROI) score of 63 corresponds to $z_{ROI} = +2.60$. I don't write it that way because, as "raw" hints, the first-stage test doesn't take relevant factors of *difficulty* of a player's games into account. But this raises the question:

Is z_{ROI} an *unbiased* predictor of the weightier z-score from the full test?

I believe the theoretical---as well as practical---answer is **no**. Instead, a form of **reversion to the mean** seems to apply: The two tests are not perfectly correlated. When we pick a player with high z_{ROI} value, there is selection bias for highness of that value. The final z-value (of an innocent player) should on average be less. [When it is higher, then I take special note.]

This segues into a principal question involved in the present model revision:

II.

> In which selection settings should the model's projections be unbiased---and accurate---estimators of the actual results?

The single most burning example is:

> Over sets of positions where the model projects markedly greater (scaled) centipawn loss (or expectation loss, or markedly lower T1-match), should the actual results match the projection, or will we theoretically see "reversion to the mean" here as well?

In this case, I think the answer should be clearly **yes**---i.e., no reversion. The model is projecting itself. Instances where we see reversion can be ascribed to systematic modeling error---such as in the recent patch to projections when selecting for positions with clear standout best moves (and ones of low entropy in general). But in other cases, things may be less clear... [discuss]

III. Another principle is "modeling cheating" versus "modeling honesty"---and whether the resulting tests are two-sided or only one-sided. This one seems even clearer to talk about in the LLM context. We can:

1. Build a predictive model of outputs of an LLM---maybe even get it from the LLM's code if available. Then we detect use of the LLM as positive conformance to this model.
2. Build a predictive model of natural human word and phrase usage. Then we detect usage of an LLM as one-sided deviation from the human model. This needs no LLM access.

[Discuss]

IV. Papers mentioned in my MIT presentation are relevant to the principle of when and whether time usage is beneficial:

Carow and Witzig (2024), https://ideas.repec.org/p/jgu/wpaper/2404.html
Sunde-Zegners-Strittmatter (2022): https://ideas.repec.org/p/rco/dpaper/317.html
Various works involving Ashton Anderson and Jon Kleinberg, of which https://arxiv.org/abs/2006.01855 is representative (and cites others).

[Discuss papers---and the possibility of building on them]

One Possible Next (Classwide) Experiment: Test IPR performances by players P:

1. against players Q of rating close to that of P.
2. against players Q rated 200 or so points higher.
3. against players Q rated 200 or so points lower.

Particular question: does P takemore risks in situation 2 (as one should!---?)
General question: Is there any discernible difference in quality or nature of play among these situations? Can you detect differences in time consumed? in difficulty of positions?

LLM Scoring Functions and the Ambitious Idea

The naive-but-natural analogy is between choice of next word or phrase and choice of next move. Can a simple utility-based model, of the kind that works well in chess, be carried over? My chess model claims that success with *few* parameters---of a severely underfitted model---implies that natural laws are in play. Is this true of human language? The application of Zipf's Law to language is a hint of possibility. But several more analogies need to "click" for the idea just to be manageable:

- "Position" $\sim$ a context for the next word (or token) in a paragraph or other sequence.
  - Need not be just the previous "N-gram":
- "Player" $\sim$ a writer of a given competence level and skills profile, e.g.
  - 7th-grade student

  - newspaper reporter / editor
  - textbook writer / editor
  - literary genius.
- "Engine" $\sim$ top writing performer (like top coder).
  - Can train "referee LLM" on top writing samples for various criteria: clarity, flow, sophistication, literary beauty.
- "Engine value v_i" $\sim$ reward for selecting option i as next word/token.
  - DeepSeek uses an express reward function for reinforcement learning.
  - BertScore defines this from ground up via cosine similarity of source and target vectors. (Easy to download, with free registration, and install in PyTorch / HuggingFace.)
  - May need to distinguish between optimality of the next move/word/token on a 0-to-1 scale (as in BertScore) versus the idea of overall value.
- "Rating" $\sim$ aggregate/average reward score for output generated by a "Player".
  - Used already for coding, see e.g. CodeForces Elo Ratings.
- **Virtual Player** $Z(P)$: Model parameters $s, c, d...$ trained on output of (human) players P of a given competence level R. In the "0.000000003B"-parameter model, obtained by fitting an equation of the general form $\ell(p_i, p_1) \sim g(v_i, v_1; Z)$. In the chess model, ℓ is a ratio of logarithms of the probabilities and g is $\exp\left(-\left(\frac{\delta_i'}{s}\right)^c\right)$.

Need is to gather lots of "positions" with values for the 50-or-so most reasonable / highest scoring next words/tokens.