Decoding Consumer Preferences with Large Language Models: A Novel Training Framework

Finn-Ole Höner (657110)

June 4, 2024

Introduction

- Media products increase relevance of content
  - Social media, short videos TikTok, Images, Netflix, Audio / Podcasts
- What is missing from preference learning? How is this different?
  - Media products/things are different from products as attribute bundles
    - We can directly represent the product itself?
  - Brands and their role in advertising
- Market research is expensive. How can we leverage these insights better to improve its benefits?
- EU AI Act
  - Content generation is considered low risk
- Main risk of Gen AI stem from unpredictable behaviors, as generation is based on random samples from a highly non-linear model
  - lawsuits
  - damages to brands
  - missed opportunities
  - tailoring of models based on prompt engineering and RL-HF
- GDPR
  - Challenges to cookies in website morphing
- Generative AI for personalization
  - Previously: Set pieces, e.g. recommendation systems
  - True personalization: Generate solution for each individual consumer
- Consumer preferences in latent generation space
  - We can view products as attribute bundles and get utilities for these attributes
  - How do we do this for products that cannot be "discretized"?
- Novel training framework

  - Circumvent issues with the LLMs decoder
  - Input embedding represents the information contained in the sequence, as it is the starting point which makes the LLM generate the target sequence

- Comparison to prompt engineering

- Neural Text degeneration (CITE Holtzman)

- BERT and RoBERTa CITE devlin, CITE Liu, CITE Glove, CITE Mikolov 1 & 2

Methodology

- Distributional hypothesis of language CITE: Sahlgren

We reverse-engineer the input to a Large Language Model (LLM), such that it generates a pre-defined text.

For a given target sequence, we find the input-embedding that maximizes the likelihood to generate this target sequence with a unidirectional LLM (e.g. GPT, Radford et al. (2018)). We call this input embedding, the "summary embedding" of the target sequence.

We restrict the degrees of freedom in our search for these input embeddings, by making the elements of the input embedding linear combinations of a few hidden factors. This amounts to a linear factor model.

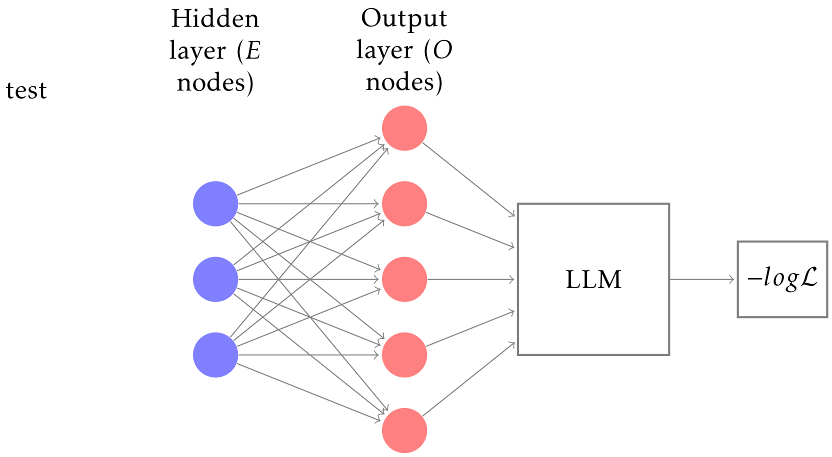

Figure 1: Factor model in Neural Network form

In turn, these hidden factors yield a generation space, where we locate our target sequences and search for new sequences.

Goodfellow, Bengio, and Courville (2016)

- Autoencoder
  - The Autoencoder $\text{AE}_{\mathcal{B}}(\cdot) : \{0,1\}^C \to \mathbb{R}^E$, with parameters $\mathcal{B}$
  - Adding covariates to the AE

- Overview
  - LLM
  - maximize likelihood of target sequence
  - autoencoder on these summary embeddings (+ covariates?)
  - also yields low-dim generation space

    - generation
    - rating of claims

Unidirectional LLMs, such as GPT or LLaMa (Radford et al. 2018; **metaMetaLlama?**), are built for text-generation. They predict the next token in a sequence of tokens autoregressively, therby, these models only work from left-to-right (or right-to-left). Bidirectional LLMs, such as BERT (Devlin et al. 2018), are built for representing text. They predict a token given the context around this token, and can hence read the text both from left-to-right and right-to-left. Both types of models share two note-worthy components, that account for a large share of their power. One, developers pre-train both types of models on large bodies of text, to learn about the structure of language itself. Two, by using a transformer architecture (Vaswani et al. 2017), which enables these language models to learn dependencies between words, such as negations (short-range dependency), or document structures, e.g. salutations in letters (long-range dependency).

- Notation
    - A large language model $\mathcal{M}$, with parameters Ω.
    - The length E summary-embedding $\boldsymbol{e}^*$.
    - The length T vector of target tokens $\boldsymbol{t}$, where the element t_i is a unique integer code representing a token, e.g. in GPT-2, the number 50267 is the End-of-Text token: <|eos|>.
    - The log-likelihood function $\mathcal{L}_{\mathcal{M}}(\boldsymbol{i},\boldsymbol{o}) : \mathbb{R}^E \rightarrow (-\infty, 0]$, that returns the log-likelihood for LLM $\mathcal{M}$ to generate the output sequence $\boldsymbol{o}$, given the input embedding $\boldsymbol{i}$.

$$\boldsymbol{e}^* = \arg\max_{\boldsymbol{e}} \mathcal{L}_{\mathcal{M}}(\boldsymbol{e},\boldsymbol{t}) = \arg\max_{\boldsymbol{e}} \sum_{i=1}^{L} \log p(t_1,\ldots,t_L|\boldsymbol{e})$$

We aim to find a summary embedding, $\boldsymbol{e}^*$, that maximizes the likelihood of generating a given target sequence, $\{t_i\}_{i=1}^L$, with a given LLM, $\mathcal{M}$. We use a gradient-based optimization algorithm to maximize this likelihood. We initialize the summary embedding as the element-wise average of the embedding of the target sequence. We then generate a sequence of length L, with the LLM and the current input embedding. For the resulting output layer, we compute the likelihood that this layer will generate the target sequence. We backpropagate the gradients and adjust the input layer accordingly. We repeat this process until the optimization converges. We want to investigate further improvements to the implementation, such as computing the likelihood contributions of the tokens in a distributed fashion. In Algorithm 1, we summarize the training process.

Algorithm 1 Training of Summary Embeddings

```
Initialize $\boldsymbol{s}$
$i \leftarrow 0$
while $Convergence == False$ do
    $i \leftarrow i + 1$
    $\boldsymbol{o}^{(i)} \leftarrow \mathcal{M}(\boldsymbol{s})$
    $l^{(i)} \leftarrow$ Likelihood($\boldsymbol{o}^{(i)}, \{t_i\}_1^L$)
    $\boldsymbol{s}^{(i+1)} \leftarrow$ GradientDecent($\boldsymbol{s}^{(i)}, l^{(i)}$)
    $Convergence \leftarrow$ CheckConvergence($\boldsymbol{s}^{(i+1)}, \boldsymbol{s}^{(i)}$)
end while
```

as the input-embedding that maximizes the likelihood of generating a given target sequence, $\{t_i\}_{i=1}^L$, where L denotes the length of the target sequence and t_i represents token i. We estimate these summary embeddings $\boldsymbol{e}^*$, by maximizing the conditional likelihood that this embedding generates the target sequence for a given LLM. We use a gradient-based optimization algorithm to maximize this likelihood.

- Figure of generation tree

  - For generation of sequences, the LLM autoregressively predicts the next token, given the previous tokens. At each step in the generation, a selection of the next token needs to take place, for which there exist various procedures (CITE). As we are optimizing with respect to the joint likelihood of the target sequence, we also need to generate with a procedure that generates this most likely sequence. Finding this sequence from the generation tree is a combinatorical problem and computationally not feasible. Hence, we use the beam-search heuristic to make our generation. Rather than only tracking the most likely token at each generation step, beam search also tracks the n-most likely tokens. The more beams we search, the more likely we are to find a sequence with a higher likelihood, then we would have found otherwise (**huggingfaceGenerateText?**).

- LLM judge to rate claims

Data

We obtain data from a market research company, that includes advertising claims for different FMCG. In our application, we focus on dairy and hair shampoo products. These data, stem from different variations of choice-based conjoint studies (**eggersChoiceBasedConjointAnalysis2022?**), which we cannot further disclose due to confidentiality agreements.

The data includes choice-experiments aggregated at the level of the advertising claims, hence we do not have responses of individuals. These measurements of consumer-preferences are on multiple dimensions, such as relevance, brand fit, or an overarching rating of the claim.

The advertising claims were designed for markets in different countries and are in english (US, UK) or translated into english (others). The date of the study is also included in the data.

We also have information on the marketers motivation behind the design of an advertising claims, such as whether she designed the claims to pronounce the health benefits of a yoghurt, or the taste yoghurt.

The data also include the brand of a product.

- Descriptives of the data
- Prepocessing of the data

Results

Implementation

- Autoencoder
- Configuration of LLM
- Connection to other measures, e.g. syntax or rating of consumers
- Different generation strategies, use beam search to get closer to max LL
- Optimization
  - (**kingma?**) (Adam)
  - Computation time, scaling
  - Issues with spikes; Figures where they come from
- Encoding space
  - example tangible, intangible

  - generations from encoding space
  - Wasteland, geometrical shape and where it comes from

- Measures of fit

  - areas in the encoding space (count out the pixels and compare to curvature of LL?)
  - analysis of the trees
  - Gap between most likely and second most likely token

- Comparison to BERT embeddings?

- Comparison to prompt engineering results?

Empirical Application

- Application to MR data

Managerial Implications

- Generative AI for personalization

- Brands

- Wording

- Interactive dashboard to explore the embedding space

Discussion

- Problems
  - Computation time
  - Benefits over existing approaches in practice?
  - Conceptual issues of using AI in decision making CITE: Cathy O'Ne; Gender biases CITE Sheng
  - How to evaluate the quality of the generated ad claims CITE: Hashimoto

- Expansion of this research
  - MAB around the fitted AE, to incorporate consumer feedback in online learning: Which areas of the space are valuable for which consumers?
  - For online learning, could wrap the decoder into an MAB and let the MAB explore the space by pulling arms
  - Could also learn preferences online with this setup
  - Better LLM
  - Image, video, website, audio data. Perhaps website a good application. Perhaps audio a rather simple way to make this multi-modal (content and sound)

References

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." https://doi.org/10.48550/ARXIV.1810.04805.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. Cambridge, Massachusetts: The MIT Press.

Radford, Alec, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. "Improving Language Understanding by Generative Pre-Training."

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. "Attention Is All You Need." *Advances in Neural Information Processing Systems* 30.