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

Finn-Ole Höner (657110)

May 14, 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
- 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

Methodology

- Overview
  - LLM
  - maximize likelihood of target sequence
  - autoencoder on these summary embeddings (+ covariates?)
  - also yields low-dim generation space
  - generation
  - rating of claims
- LLM
  - Attention
  - Transformer
  - GPT
  - 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, 50267 is the End-of-Text token <|eos|>.
- Maximize the likelihood to generate the data
  - The log-likelihood function $\mathcal{L}(\cdot) : \mathbb{R}^E \to (-\infty, 0]$

To express that we generate tokens with the LLM, $\mathcal{M}$, based on length E input-embedding $\boldsymbol{e}$, we use the notation $\mathcal{M}(\boldsymbol{e})$. To express the number of tokens, k, that we generate, we use the subscript $\mathcal{M}_k(\boldsymbol{e})$. This function call returns an output vector of length V, where V denotes the vocabulary size. We denote the output vector as $\boldsymbol{o}$. Each element o_v of this output vector represents the probability of generating token v as the next token. We predict the next token, by selecting the largest element of $\boldsymbol{o}$. When $k = 1$, we use the previous tokens to predict the next token. When $k > 1$, we append the predicted tokens autoregressively to the input embedding, such that we can predict more than one token. While there are many ways to design this procedure and select tokens along the way, we perform a standard greedy procedure, where we select the token with the highest probability at each step. We denote the probability that $\mathcal{M}$ generates the target sequence, given the input embedding, as $p(t_1, \ldots, t_L|\boldsymbol{e})$. For a given Large Language Model, we define $\boldsymbol{e}^*$,

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

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.

- Autoencoder
  - The Autoencoder $\text{AE}_{\mathcal{B}}(\cdot) : \{0,1\}^C \to \mathbb{R}^E$, with parameters $\mathcal{B}$
  - Adding covariates to the AE
- Figure of generation tree
  - beam-search for generation
- LLM judge to rate claims

Implementation

- Autoencoder
- Configuration of LLM
- Optimization
- Connection to other measures, e.g. syntax or rating of consumers
- Different generation strategies, use beam search to get closer to max LL

Data

- Descriptives of the data
- Prepocessing of the data

Results

- Optimization
  - Computation time, scaling
- 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
- Application to MR data
- Comparison to BERT embeddings

Managerial Implications

- Generative AI for personalization
- Brands
- Wording
- Interactive dashboard to explore the embedding space

Discussion

- Problems
  - Computation time
- 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)