Notations

Conventions for Notations

- Lower case letters for variables
- Upper letters for mappings (functions, distributions, etc.)
- Calligraphic letters for sets or spaces, except for conventional notations such as $\mathbb{R}$ or $[0, 1]$.
- Greek letters for random variables

List of Notations

- $\mathcal{V}$: the vocabulary, or the set of tokens
- $\mathcal{S} = \bigcup_{n=1}^{\infty} \mathcal{V}^n$: the set of token sequences
- $\mathcal{P}_{\mathcal{S}}$: the space of all probability distributions on $\mathcal{S}$ (language models)
- $\Pi_{\mathcal{S}} = \mathcal{P} \circ \mathcal{P}_{\mathcal{S}}$: the space of distributions over language models

The Shorter Version

A language model is optimized to produce a distribution that matches the data distribution. Suppose we have a prior distribution over language models, denoted by $\mu \in \Pi_{\mathcal{S}}$. We can interpret the learning process as:

$$M = \arg\max_{M} P(\mathcal{D}|M)P(M|\mu).$$

I would like to make a strong argument: We should learn μ instead of any single constant language model M. I interpret μ as a "filter" that discards all models H where $P(\mathcal{D}|H) = 0$. For example, a model that believes both *"Tom likes playing sports"* and *"Tom doesn't like playing sports"* is not feasible. For any specific query q, we should sample from μ to generate responses:

$$r \sim P(r|q, \mu) = \int_M P(r|q, M)P(M|\mu)\mathrm{d}M,$$

and alternatively, we can alter μ without changing its support set for some "unconventional" responses (still filtering out impossible models):

$$r \sim P(r|q, \hat{\mu}) = \int_M P(r|q, M)P(M|\hat{\mu})\mathrm{d}M.$$

Here $\hat{\mu} \propto \mu \times g$, and $g : \mathcal{P}_{\mathcal{S}} \to \mathbb{R}^+$.

A stochastic language model should, at least, consist of

- a learned prior from data over language models μ
- a modifying function $g : \mathcal{P}_{\mathcal{S}} \to \mathbb{R}^+$ that steers behaviors of μ towards a more desired direction

μ characterizes all possible hypothesis potentially by assuming rules or facts absent in the training data. In comparison, a constant language model is just one hypothesis within μ that is considered most "likely" given the observed data, which could be biased or inaccurate.

Thoughts on Implementation

In a discrete formulation of component language models (similar to mixture-of-experts), let's assume we have K components $M_{1:K}$, each of them assumes a unique set of facts $\mathcal{A}_{1:K}$. We can model each component as a base model B plus additional assumptions:

$$P(\cdot|M_k) = P(\cdot|B, \mathcal{A}_k).$$

To implement this modeling, I am thinking of two potential approaches.

- ***In-context Components***: Formulate each $\mathcal{A}_k$ as a set of explicitly or implicitly context prepended to the model. These contexts should be sampled and optimized during the pre-training stage.
- ***Layer Selection*** Consider each set of assumptions as a unique combination of model layers. The model dynamically sample a combination of the layers to reflect different components.

The Longer Version with Rationales

Constant Language Models

A language model $M \in \mathcal{P}_{\mathcal{S}}$ is a distribution over token sequences: It gives a probability $P(s|M)$ for any sequence $s \in \mathcal{S}$. Moreover, given any partial sequence $x_{1:n} \in \mathcal{V}^n$ from a longer sequence $x_{1:m} \in \mathcal{V}^m$, the language model naturally computes a conditional distribution $P(x_{n+1:m}|x_{1:n}, M) = \frac{P(x_{1:m}|M)}{P(x_{1:n}|M)}$. This is used by "Chat" models, which response to a user query q by sampling a response $r \sim P(\cdot|q, M)$.

We usually train such a model over three stages:

- Pre-train on a text corpus

- Supervised fine-tuning (SFT) on query-response pairs
- Preference alignment on human preference data of (query, preferred response, disliked response) triples

In all three stages, a language model is optimized to produce a distribution that matches the data distribution. Suppose we have a prior distribution over language models, denoted by $\mu \in \Pi_{\mathcal{S}}$. The prior comes from the architectural design and the modeling of the language (e.g., autoregressive transformers). We can interpret the learning process as the Bayesian inference:

$$M = \arg\max_{M} P(\mathcal{D}|M)P(M|\mu).$$

It might seem tricky to interpret the preference alignment in a Bayesian fashion. However, since RLHF and its variants are essentially optimizing a text distribution where responses' observed frequencies are rescaled according to the rewards assigned, the preference alignment also fits the framework.

Despite the success of language modeling demonstrated by models like GPT-4, I think this paradigm is fundamentally undesirable because ***Language Modeling Only Explains Data In a Most Probable Way***.

First consider a uniform prior assignment μ, where the optimization becomes

$$M = \arg\max_{M} P(\mathcal{D}|M).$$

This means that the optimal language model under uniform prior gives zero probability to unseen or novel sentences, indicating zero generalization. No matter how much data we have, this is not desired since:

- ***No generalization means no new knowledge can be created.*** For example, the learned model cannot help researchers on innovative ideas, or solving problems that are not yet solved by humans.
- ***Mimicking the real world is not always the best strategy***. For example, if the training data contains toxic or biased content, the learned model will be exactly toxic or biased to the same extent. Another toy example would be predicting results of tossing an imbalanced coin (say 0.6 head and 0.4 tail). A desired model that maximize accuracy should always predict "head", but a model that matches observations only predicts head with a probability of 0.6.

Hence, if we stick to the optimization objective above, the prior μ must be non-trivial to play a role in achieving generalization. Fortunately, current LLMs do have a non-trivial prior, since they have been remarkable in generalizing to unseen data. However, what are these priors?

- It is hard to figure out the architectural prior of autoregressive transformers. There is an intuition-based qualitative characterization of this prior though: sentences that look similar have correlated probabilities. This prior makes sense sometimes. For example, *"Tom likes playing sports"* is somewhat correlated to *"Tom enjoys playing sports"*. However, we also observe weird correlations: *"Tom likes playing sports"* could also correlate positively to *"Tom doesn't like playing sports"*, meaning that increasing one probability leads to an increase in the other.
- In the stage-wise training paradigm, a checkpoint from an earlier stage naturally carries a prior from the previous data. For example, SFT is essentially optimizing a model with a prior dependent on the pre-training data. Moreover, we don't always pre-train models until convergence. This means that earlier samples may form a prior to later samples, which indicates that the order of samples during training could be important.

I suppose the existing prior for LLMs is not perfect. In fact, recent work on knowledge editing shows that generalizing an edited knowledge is very hard. Moreover, the well-known reversal curse also indicates that the current prior from the autoregressive language modeling does not generalize well enough. I think a perfect prior μ should potentially encompass all the rules for generalization such as logical deduction rules. Moreover, these rules should be learned from data, either through explicit mentions of rules or implicit induction from observations, rather than prepared by human in advance, since it is hard to exhaust all such rules. μ ***should come from the data.***

I interpret μ as a "filter" that discards all models H where $P(\mathcal{D}|H) = 0$ (or smaller than a threshold), reflecting rules of deduction. For example, a model that believes both *"Tom likes playing sports"* and *"Tom doesn't like playing sports"* is not a feasible model and should be discarded. In other words, μ characterizes all possible hypothesis that can explain the real world, potentially by assuming unstated facts absent in the training data.

Stochastic Language Models

I would like to make a strong argument about learning language models: We should learn μ instead of any single constant language models. For any specific query q, we should sample from μ to generate responses:

$$r \sim P(r|q, \mu) = \int_M P(r|q, M) P(M|\mu) \mathrm{d}M,$$

and alternatively, we can alter μ without changing its support set for some "unconventional" responses (still filtering out impossible models):

$$r \sim P(r|q, \hat{\mu}) = \int_M P(r|q, M) P(M|\hat{\mu}) \mathrm{d}M.$$

Here $\hat{\mu} \propto \mu \times g$, and $g : \mathcal{P}_{\mathcal{S}} \rightarrow \mathbb{R}^+$.

A stochastic language model should, at least, consist of

- a learned prior from data over language models μ
- a modifying function $g : \mathcal{P}_{\mathcal{S}} \rightarrow \mathbb{R}^+$ that steers behaviors of μ towards a more desired direction

This formulation mimics the current paradigm in that: (1) Learning μ corresponds to pre-training. (2) Learning g corresponds to preference alignment, except that we pick preferred language models instead of altering language model distribution.

I also want to further compare the learning of μ with the learning of a constant language model.

- A constant language model is the best probabilistic explanation of the data given some prior. However, it is not guaranteed to be correct, and in fact, there is no guarantee that we can find the correct model of the world from observations we have: Otherwise there is no more need for scientific research to understand the world. However, the "best" model refuses other plausible hypothesis. It should be questioned whether it is really optimal, since data could be of insufficient accuracy: Thinking of Newton's Laws versus Theory of Relativity, the former could be optimal or at least equally optimal when we only observe data in low-speed scenarios.
- Fine-tuning a constant language model (or RLHF) sometimes involves a KL-regularization term. We can interpret this procedure as treating a constant language model as a Gaussian stochastic language model centered around its pre-trained state. However, this interpretation implicitly implies that we should not deviate too much from a pre-trained language model, or all modifications can be "local". However, in real world, there are cases where we have two contradicting hypotheses, and upon subsequent observations we choose one of them. We may need drastic changes to correct errors. This is not a problem for alignment of μ.

Thoughts on Implementation

Firstly, different components in μ are based on different assumptions absent in the training data. In a discrete formulation of component language models (similar to mixture-of-experts), let's assume we have K components $M_{1:K}$, each of them assumes a unique set of facts $\mathcal{A}_{1:K}$. We can model each component as a base model B plus additional assumptions:

$$P(\cdot|M_k) = P(\cdot|B, \mathcal{A}_k).$$

To implement this modeling, I am thinking of two potential approaches.

In-context Components

We may formulate each $\mathcal{A}_k$ as a set of explicitly or implicitly context prepended to the model. These contexts should be sampled and optimized during the pre-training stage. I don't have a clear idea of how to make this efficient yet. Moreover, this paradigm inevitably increase the inference cost.

Layer Selection

We may consider each set of assumptions as a unique combination of model layers. During forward passes, the model dynamically sample a combination of the layers to reflect different components. This should be efficient, and can be optimized via variational inference.