EXPLAINING PATTERNS IN DATA WITH LANGUAGE MODELS VIA INTERPRETABLE AUTOPROMPTING

Chandan Singh,1,* John X. Morris,2,* Jyoti Aneja,1 Alexander M. Rush,2 & Jianfeng Gao1

* Equal contribution

1 Microsoft Research {chansingh,jyotianeja,jfgao}@microsoft.com
2 Cornell University {jxm3,arush}@cornell.edu

ABSTRACT

Large language models (LLMs) have displayed an impressive ability to harness natural language to perform complex tasks. In this work, we explore whether we can leverage this learned ability to find and explain patterns in data. Specifically, given a pre-trained LLM and data examples, we introduce interpretable autoprompting (iPrompt), an algorithm that generates a natural-language string explaining the data. iPrompt iteratively alternates between generating explanations with an LLM and reranking them based on their performance when used as a prompt. Experiments on a wide range of datasets, from synthetic mathematics to natural-language understanding, show that iPrompt can yield meaningful insights by accurately finding groundtruth dataset descriptions. Moreover, the prompts produced by iPrompt are simultaneously human-interpretable and highly effective for generalization: on real-world sentiment classification datasets, iPrompt produces prompts that match or even improve upon human-written prompts for GPT-3. Finally, experiments with an fMRI dataset show the potential for iPrompt to aid in scientific discovery.1

1 INTRODUCTION

Large language models (LLMs) have attained an extraordinary ability to harness natural language for solving diverse natural-language problems [1], often without the need for finetuning [2, 3]. Moreover, LLMs have demonstrated the capacity to excel at real-world problems, such as mathematics [4] and scientific question answering [5].

In this work, we probe whether we can leverage the learned skills of an LLM to find and explain patterns in a dataset. To do so, we invert the typical problem of fitting an LLM to data and instead ask whether we can use a fixed LLM to produce a natural-language string explaining dataset patterns. Our approach to this problem centers around prompting. Prompting has emerged as an effective method for adapting LLMs to perform new tasks [6]. A prompt string is combined with each example in a dataset before querying an LLM for an answer.

While prompts were initially constructed manually, recent work has shown success in autoprompting, i.e. automatically finding a prompt via optimization [7, 8]. However, previous work on learning natural language prompts [7] does not produce prompts that are meaningful to humans.

Our approach, interpretable autoprompting (iPrompt), extends autoprompting to generate a semantically meaningful natural-language prompt that explains a key characteristic of the data (see Fig. 1). For example, given a dataset of examples of addition, e.g. 2 5 ⇒ 7 ... 3 1 ⇒ 4, we use an LLM to yield the natural-language description Add the inputs. iPrompt is an iterative algorithm that alternates between (i) proposing candidate explanations with an LLM, (ii) reranking the candidates based on their performance when used as a prompt, and (iii) exploring new candidates.

1 All code for using the methods and data here is made available at O github.com/csinva/interpretable-autoprompting.
To evaluate iPrompt, we curate a diverse collection of datasets written in natural language (Table 1), where our goal is to accurately infer a ground-truth pattern. The dataset includes a number of synthetic math datasets, as well as language tasks from the Natural Instructions V2 dataset [9]. We find that iPrompt outperforms baseline autoprompting methods in successfully finding a correct description across these datasets. Moreover, the generated descriptions are interpretable, allowing human auditing and enabling strong generalization performance when used as a prompt in a new setting (i.e. when used for a different LLM). On real-world sentiment classification datasets, iPrompt even produces prompts that match or improve upon human-written prompts for GPT-3. Finally, we qualitatively explore iPrompt in a neuroscience task, in which we seek to understand the mapping of semantic concepts in the brain from fMRI imaging (data from [10]).

2 Dataset Explanation Task

Task definition  Given a dataset comprised of input-output string pairs \( \{(x^1, y^1), \ldots, (x^N, y^N)\} \), the goal is to produce a “semantically meaningful” natural-language string that explains the relationship between \( x \) and \( y \). We require that a string consists of human-understandable text rather than a sequence of incongruous tokens. For example, in the task shown in Fig. 1, the task is to recover text synonymous to Add the inputs given samples of data performing addition.

Datasets  Table 1 shows the four collections of datasets we study: (1) Inverse Synthetic Math with datasets that require inferring an underlying mathematical function of one or two numbers; (2) Inverse Allen NLI (ANLI), a selection of crowdsourced language tasks [9] with easily verifiable descriptions (e.g. Find a country’s capital); (3) Sentiment, consisting of four real-world sentiment classification tasks and (4) fMRI, a dataset involving brain responses to natural language, motivated by the goal of recovering unknown explanations. In addition to data examples, the first two collections contain a ground-truth description and simple rules to test whether an extracted description matches the ground-truth one. For example, when adding two numbers (Fig. 1), the rule checks whether a description contains any of the keywords add, sum, or +.

| Collection               | #   | Description                  | Dataset names                                      |
|--------------------------|-----|------------------------------|---------------------------------------------------|
| Inverse synthetic math   | 10  | Simple mathematical functions | Add two, Subtract two, Multiply two, Divide two, Max two, First number, Square, Exponentiate, Double, Fibonacci |
| Inverse Allen NLI [9]    | 10  | Diverse language tasks       | Country capital, Antonyms, Check edibility, Rhyme generation, Country currency, Check prime, Check vegetarian, Find typo, Gender classification, SQL query generation |
| Sentiment                | 4   | Sentiment classification     | SST-2, RottenTomatoes, IMDB, Financial Phrasebank |
| Natural-language fMRI [10]| 20  | Find an underlying category from a list of words that excite an fMRI voxel | Extracting a pattern from a set of words, each corresponding to a different voxel |
The examples in each task do not directly contain the task description. For example, when inferring the Add two numbers task, the examples do not contain a plus sign or any synonyms of the word add such as combine. For classification tasks such as Check edibility or Check prime, the label provided in the example text is simply yes/no rather than the given labels, e.g. edible/non-edible.

**Evaluation** We evaluate dataset explanation on two criteria: closeness to the ground-truth prompt and ability to generalize as a prompt for other models. To evaluate similarity to the ground truth, we score a ranked list of prompts based on mean reciprocal rank (MRR). Given a set of datasets $D = \{D_1, ..., D_N\}$, we compute: $\text{MRR} = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{1}{\text{rank}_i}$, where rank$_i$ is the one-indexed rank of the first correct explanation. We evaluate correctness based on whether the generated explanation contains one of a set of problem-specific keywords. To measure generalization, we use the top-ranked string as a zero-shot prompt for a different language model, and evaluate whether that model is able to solve the task.

3 **AutoPrompting Methods**

In this section, we detail approaches for tackling the general problem of autoprompting before introducing our method for interpretable autoprompting (iPrompt) in Sec. 3.2.

We specify autoprompting as a discrete search problem. Given a dataset of $n$ input-output pairs \( \{(x^1, y^1), ..., (x^n, y^n)\} \) and a pre-trained LLM $f$ that returns the log-probability of a given string, the goal of autoprompting is to find a natural-language explanation $\hat{s}$ maximizing:

$$\hat{s} = \arg\max_{s \in S} \sum_{i=1}^n f(\text{render}(s, x^i, y^i))$$

The render function is a problem-specific function that renders a natural language string from the prompt $s$ and each example in the dataset $(x^i, y^i)$. We use $S$ to indicate the set of fluent strings, under some notion of syntactic fluency. Solving this search problem exactly is intractable.

![Figure 2: Model accuracy depends on having an accurate prompt for large models (GPT-J 6B and GPT-3). The model is given the prompt Return the _ of the inputs., where _ is filled in with the shown prompt keyword before querying the output given two input numbers in a string. Darker indicates a higher accuracy, and high accuracy along the diagonal indicates that the correct prompt induces the highest accuracy.](image)

A core assumption of this objective is that semantically accurate prompts lead a model to assign higher probability to the data. To check this assumption, we analyze four datasets from the inverse synthetic math collection that share common structure for the inputs and prompts: each dataset admits a prompt of the form Return the _ of the inputs., then is given two input numbers and queried for the output.

Fig. 2 shows the accuracy of different models at performing these tasks when given different input prompts.\(^2\) For small models, the prompts are unsuccessful, but for large models (e.g. GPT-J 6B and GPT-3), the model is accurate if and only if given the correct prompt.\(^3\) This result provides evidence

\(^2\)The accuracy is normalized for each task using softmax in order to visualize the effect of differing prompts.
\(^3\)For details on each model, see Table A3.
that, at least for large models, the search for a prompt that maximizes performance correlates well with the underlying task.

3.1 Baseline Methods

**AutoPrompt** AutoPrompt [7] targets the objective posed in Eq. (1) using a gradient-based local search. AutoPrompt searches for \( \hat{s} \) following the gradients of the objective Eq. (1) with respect to individual tokens in \( \hat{s} \). By iteratively computing these gradients, it can discretely change individual words in \( \hat{s} \) and then check whether or not the newly updated \( \hat{s} \) improves the objective score. The use of gradients allows AutoPrompt to find an effective prompt \( \hat{s} \), but makes it difficult to find answers that satisfy the fluency constraint \( S \).

**Average-output suffix decoding** LLMs themselves can be directly used to predict prompt strings. We can give the model a prompt that includes examples such as the following context string:

```
In: 2 3
Out: 7
```

To compute the output from the input, we use a template and sample the output for the blank to recover a prompt \( \hat{s} \). Sampling directly from \( f \) helps ensure that the generated explanation is fluent and semantically meaningful. We decode the output using beam search to find the highest-probability outputs for multi-token prompts. To improve on this approach, we place several examples into the model’s context, and then average the model’s output logits across all the examples in the dataset before decoding the output, an approach we refer to as average-suffix decoding. However, this technique is insufficient to find high-scoring prompts.

3.2 Proposed Method: iPROMPT

Our method, iPROMPT shown in Fig. 3, is an iterative local search algorithm that alternates between three steps: (i) proposing candidate prompts, (ii) reranking candidate prompts, (iii) exploration:

(i) Proposal: Candidate prompts are generated by extending the zero-shot LLM generation. Given a data instance as a prefix, we sample a number of candidate prompts. The maximum length of each candidate is pre-specified and fixed. For example, in the add-two-numbers task (Fig. 3), we

4Here we prefer beam search here over alternatives such as nucleus sampling [11] as we are interested in finding an accurate prompt description with as few samples as possible.

5One could use either average suffix decoding or suffix decoding with a single sample. For computational efficiency, we use suffix decoding with only a single sample. We also add randomly decode the output rather than using beam-search, as our iterative procedure can recover from initially finding inaccurate candidates.
may generate four candidates: \{Combine the numbers, Return the output, Sum in order, Compute the output\}.

(ii) Reranking: Given candidates, the objective Eq. (1) is evaluated for each candidate prompt \( s \). The top few candidates which maximize the objective are kept, e.g. narrowing down the candidates to \{Combine the numbers, Sum in order\}.

(iii) Exploration: Each of the top candidates from reranking is truncated at a random position. These truncated candidates are used as a prefix string when generating new candidate prompts via suffix decoding. For example, we may randomly select the start of the previous candidates and fill in the endings: \{Combine the \_\_, Sum \_\_\} \rightarrow \{Combine the numbers, Combine both arguments, Sum the numbers, Sum all inputs\}

The algorithm is repeated until identifying a suitably strong \( s \), e.g. selecting Sum the numbers. Step (i) and (iii) ensure that prompts remain fluent, while step (ii) improves the score of the prompts on the objective. Computationally, iPrompt only requires running inference on the pre-trained LLM, yielding a significantly lower memory requirement than methods such as AutoPrompt, which require access to the LLM’s gradients.

4 RESULTS

Accuracy of prompts Table 2 compares prompting methods based on the set of candidate descriptions they generate using GPT-J (a 6-billion parameter model) as the LLM [12]. The MRR rows show that iPrompt considerably increases the mean reciprocal rank (MRR) (Sec. 2) over the baselines, implying that iPrompt can more effectively generate descriptions that accurately reflect the underlying data pattern. The “top-prompt correctness” rows show the percentage of datasets for which the top-ranked candidate prompt produced by each method is labeled as accurate by manual inspection (see all prompts in Appendix A.2). On the ANLI datasets, iPrompt again outperforms the baselines, although all methods perform poor in an absolute sense (\( \leq 30\% \)). The zero-shot results show the accuracy of GPT-J when using the top prompt found by each model; for the math datasets the iPrompt prompt elicits an improvement over the baselines, but for the ANLI datasets all prompts induce poor performance.\(^6\)

Table 2: Accuracy for dataset explanation measured via (i) MRR, (ii) top-prompt correctness, and (iii) zero-shot accuracy on unseen examples. All experiments are on GPT-J 6B. For all metrics, higher is better.

| Dataset | iPrompt | AutoPrompt | Average suffix |
|---------|---------|------------|----------------|
| **Math** | MRR | 0.71 | 0.30 | 0.07 |
| Top-prompt correctness | 80% | 30% | 20% |
| Zero-shot acc. | 51.5% | 41.6% | 10.0% |
| **ANLI** | MRR | 0.30 | 0.17 | 0.01 |
| Top-prompt correctness | 30% | 0% | 10% |
| Zero-shot acc. | 4.7% | 1.9% | 5.1% |

Qualitative assessment of top prompts Table 4 shows the top-ranked prompt generated by each method for selected datasets. iPrompt often finds a prompt that is somewhat indicative of the underlying relationship, even if the phrasing is not perfect. For example, for the add two numbers dataset, it finds Write a function int add\( f \). For difficult datasets, the iPrompt string sometimes simply returns the classes of the output (e.g. yes or no?) rather than capturing the underlying relationship. The prompts found by iPrompt also read as coherent strings compared to AutoPrompt, which returns an incoherent set of tokens. See all found prompts, including for average-suffix decoding in Appendix A.2.

\(^6\)Here, we restrict generated prompts to 6 tokens (see a full discussion of experimental details in Appendix A.3).
Table 3: Generalization accuracy (zero-shot) when testing the prompts generated with GPT-J as the LLM across different models. iPrompt yields strong performance, usually improving over AutoPrompt despite maintaining interpretability, and sometimes performing close to the human-written prompt. Numbers within 2% of the top accuracy (excluding human-written prompts) for each model are shown in bold.

|                | Human-written | iPrompt | AutoPrompt | Average | Suffix | No prompt |
|----------------|---------------|---------|------------|---------|--------|-----------|
| **Math**       |               |         |            |         |        |           |
| OPT 6.7B       | 12.7          | 19.3    | 18.9       | 4.5     | 8.4    |
| GPT 20B        | 76.1          | 54.4    | 23.2       | 21.3    | 8.5    |
| GPT-3 175B     | 76.0          | 62.1    | 40.8       | 16.9    | 28.4   |
| **ANLI**       |               |         |            |         |        |           |
| OPT 6.7B       | 10.7          | 6.7     | 4.7        | 8.5     | 7.9    |
| GPT 20B        | 31.0          | 14.2    | 5.6        | 13.7    | 4.0    |
| GPT-3 175B     | 37.6          | 11.7    | 2.7        | 13.4    | 7.7    |

Table 4: Examples of generated prompts.

|                | Human-written prompt | iPrompt | AutoPrompt |
|----------------|----------------------|---------|------------|
| **ANLI**       | Return whether the input food dish is vegetarian (yes or no). | Write an SQL to produce output ributed grandfatherExceptionappropriate intent Lara | Novthethethetethe |
| **Math**       | Return the sum of the inputs. | Write a function int add( | addedthe +the use worked multiplythen the Multiple opposably exactly subtractFor YEAR |
| Return the product of the inputs. | When you multiply two ( | NumbertetheJusticeJaDefault greater name sorting indiscrim to numbers |
| Return the difference of the inputs. | If n > m then subtract | multiplythe hypot Northierl |
| Return the maximum of the inputs. | Which number has a bigger value | NumberthetheJusticeJaDefault greater name sorting indiscrim to numbers |
| Return the first of the inputs. | The first digit of both values | multiplythe hypot Northierl |
| Square the input to get the output. | Write a function that calculates square | write a function called double that |
| Given an input x, return 2*x. | Write a function int add( | addedthe +the use worked multiplythen the Multiple opposably exactly subtractFor YEAR |

|                | Rotten Tomatoes | SST-2 |
|----------------|-----------------|-------|
| Answer Yes if the input is positive and No if the input is negative. | a fast, funny, highly enjoyable film. Answer: Yes 3.1/ | suke Medals; does CFR Sab”]==> NormalConstructed Umburt satisfy Good-ram |
| Answer Yes if the input is positive and No if the input is negative. | life Answer: Yes (because it’s about life) This | RALauntletIEidatedWhetherBF Holy Kubrick incorporatedheter-ent#S Note-=-- SPECIAL Pyth |

Generalization of generated prompts to new models. Table 3 shows the generalization accuracy when using the prompts generated using GPT-J (Table 4) and testing them on different LLMs. The same prompts effectively improve accuracy across different models compared to having no prompt. The gap between iPrompt and AutoPrompt is larger for larger models (i.e. GPT 20B and GPT-3), suggesting that by generating fluent prompts iPrompt better captures a generalizable description of the task. Human-written prompts still outperform the autoprompting methods on this task.

---

7 Accuracy is computed following [2, 13]: using exact matching with beam search, a beam width of 4, and a length penalty of $\alpha = 0.6$. 

---
Investigating iPrompt in sentiment classification  Finally, we study the more difficult task of prompting for sentiment classification, using four popular datasets \cite{14–16}. The aim is to find a dataset-specific prompt that can describe a particular sentiment classification setting. To accommodate for a complex input-output relationship, we allow each method to generate up to 16 tokens (our manually-written sentiment classification prompts range from 13-16 tokens). We use Yes and No as positive and negative labels, and require the LLM to generate the proper output, as opposed to simply ranking the two options.

Table 5: Zero-shot accuracy on sentiment classification datasets using prompts generated with the GPT-J 6B models. Evaluation is performed both on the original GPT-J 6B parameter model and testing generalization to GPT-3. The model needs to produce the correct answer (Yes, No, or Maybe) out of the entire vocabulary (without rank-eval). Values are averaged over three random seeds for prompt-generation; errors are standard errors of the mean.

|                  | Human-written | iPrompt     | AutoPrompt | No prompt |
|------------------|---------------|-------------|------------|-----------|
| Financial phrasebank | 24.3          | 62.4 ± 0.1  | 6.8 ± 2.9  | 0.0       |
| Rotten Tomatoes   | 44.4          | 70.5 ± 1.4  | 57.1 ± 3.4 | 0.0       |
| SST-2             | 53.6          | 82.8 ± 1.9  | 40.0 ± 7.9 | 0.0       |
| IMDB              | 32.5          | 21.3 ± 9.3  | 12.1 ± 0.9 | 3.5       |
| Financial phrasebank | 54.1          | 65.0        | 2.7        | 0.4       |
| Rotten tomatoes   | 58.6          | 52.5        | 37.5       | 0.9       |
| SST-2             | 60.4          | 83.6        | 5.2        | 0.6       |
| IMDB              | 79.0          | 1.3         | 0.9        | 1.1       |

Table 5 shows the zero-shot performance of the prompts elicited by different methods. Prompts are generated using GPT-J 6B and evaluated using both GPT-J 6B and GPT-3. iPrompt outperforms not only AutoPrompt, but also the manually-written prompt on three of the four datasets. The exception is the IMDB dataset, which has extremely long examples and may not be well suited for the zero shot setting. Accuracy is measured on the test set when available; otherwise, it is measured on a held-out 25% of the train set.\footnote{Different from the other experiments in this paper, we initialize AutoPrompt with random tokens instead of all the, as we find AutoPrompt fails to find an effective solution for longer prefix lengths when all tokens are initialized to the.}

Table 4 shows an example of comparing the prompts from the Rotten Tomatoes dataset, for which iPrompt and AutoPrompt induce similar zero-shot accuracy. Here and in other cases, iPrompt sometimes discovers a prompt that is a paraphrase of an example one would find in the training set or a prototypical example for a class.

5 Scientific investigation into fMRI natural-language dataset

We now explore using iPrompt in a (very simplified) neuroscience experimental setup (Sec. 5). A central challenge in neuroscience is understanding how and where semantic concepts are represented in the brain. A recent seminal study \cite{10} explores this question by investigating where different natural-language categories are represented in the human neocortex. Specifically, the authors collect functional MRI (fMRI) responses as human subjects listen to hours of narrative stories. They then build a predictive model of these responses for each voxel (i.e. a small region in space) in the brain, which takes as input the words contained in the stories (and other features). To interpret these individual voxel models, they cluster the words in the narrative stories into 12 groups and manually annotate them, resulting in 12 categories, such as tactile, visual, and professional. Finally, they view the spatial mapping of these 12 concepts (projected onto low dimensions) across the brain using their individual voxel models.

We revisit a small piece of this study’s analysis through the lens of iPrompt. Specifically, we ask whether iPrompt could generate plausible categories that are well-represented across the brain but differ from the manually identified 12. We begin by fitting a predictive model for each voxel, following the pipeline of the original study (details in Appendix A.5). We then use the resulting models to identify a list of the top-15 words which most excite each voxel. For example, the top-15 words that
excite the best-predicted voxel are: sheet, edges, diameter, strips, cardboard, copper, steel, colored, coloured, leaf, wire, cap, paper, shaped, tin. To identify a plausible semantic category, we construct a template string as follows: The following list of words all belong to the same semantic category: sheet, edges, ..., shaped, tin. We then use iPrompt (again with a GPT-6B parameter model) to generate a category by filling in the blank (restricted to a single token). To make iPrompt more effective, for each voxel we use iPrompt on a set of examples consisting of 15 permutations of the top-15 words, allowing the model to find patterns that are not overly sensitive to the word-ordering.

Given the top categories for each voxel, we analyze the mapping of recurring categories across the neocortex. We aggregate the top-15 inferred categories over the top-15 best-predicted voxels and find that the most frequently inferred categories are: material, color, surface, text, & fabric. Interestingly, these are sensible quantities that different voxels could reasonably be selective for. We spatially map each of these identified categories (e.g. material) across the 10,000 best-predicted voxels by using the LLM in a second way. For each voxel, we condition the LLM (again GPT-6B) on the top-15 words list, and evaluate the predicted probability for each category, i.e. The following list of words all belong to the same semantic category: sheet, edges, ..., shaped, tin. The semantic category they all belong to, in one word, is. The higher this predicted probability, the more selective we infer that a voxel is for the category. Fig. 4 shows these predicted probabilities for the top-two inferred categories (material and color) across the cortex of a human subject.

While there is no groundtruth for this semantic map, one noteworthy feature of the resulting map is that it is spatially smooth (quantitatively. Fig. A2 shows that the variance of the map among neighboring pixels is significantly lower than we would expect by shuffling the map’s values). This is non-trivial, as nowhere in the modeling process was spatial information incorporated: each voxel was modeled independently and the displayed prediction was queried independently. We expect the underlying map to be smooth, both due to local connectivity in brain regions and also because the BOLD signal measured by fMRI does not have perfect spatial resolution. Thus, the fact that our inferred map is smooth suggests that (i) something about these categories is genuinely captured by the representation in the human brain, and (ii) that the iPrompt approach was able to reflect at least some of it. Beyond the two categories shown, the five categories generated by iPrompt exhibit spatial smoothness across the neocortex (Fig. A2).

---

8 We apply stemming and remove stopwords before choosing the best categories.
6 RELATED WORK

Prompting and autoprompting. With the advent of large-scale models, prompting (i.e. finding the right prompt to use to query an LLM for a given task) has exploded as an area of inquiry, often yielding impressive improvements in performance [2, 6, 18] and spurring a line of work aiming to make prompting easier [19–22]. Recently, autoprompting (i.e. automatically searching for a prompt or prompt-embedding via optimization) has emerged, with methods such as prefix-tuning [8], P-tuning [23], prompt-tuning with rules [24], knowledgeable prompt tuning [25] and many more [6]. These strategies use gradient descent to find a set of “adapter” parameters that maximize model performance, but do not require that the new parameters map back to tokens in discrete space, rendering them uninterpretable.

A few methods tackle the more difficult problem of searching for prompts that can be expressed in natural language tokens. The closest related work is AutoPrompt [7], which performs autoprompting via input gradients (see Sec. 3). Similarly, adversarial triggers [26] use autoprompting to identify adversarial inputs which can be used to change a model’s prediction. These methods effectively alter a model’s predictions, but do not constrain the discovered prompts to be semantically meaningful, resulting in prompts that are difficult to interpret [27]. Another related work directly finetunes an LLM to describe the difference between two datasets [28].

Alternative methods for neural-network interpretation A popular method for interpreting neural networks is to inspect an LLM’s individual predictions via feature importances [29, 30], feature-interaction importances [31, 32], extractive rationales [33, 34], or natural-language explanations for individual predictions [35, 36]. These works can provide meaningful insights for individual predictions, but it is difficult to parse them into an understanding of an entire dataset. Alternatively, one can investigate whether an LLM’s learned representations via probing [37, 38] or by directly analyzing a model’s internal weights and activations [39–41]. However, these approaches are limited in their ability to generate previously unknown descriptions of data. A different approach involves distilling information into a transparent model [42–44] or simply using a transparent model in the first place [45–48].

Problems related to interpretable autoprompting The problem statement presented in this work closely resembles the widely studied problems of symbolic regression [49, 50], program synthesis [51, 52], text/table summarization [53, 54], and pattern discovery in data-mining [55]. In these cases, data examples are given with the goal of inferring a symbolic expression, program, or text summary that is consistent with the data. iPrompt can be viewed as an algorithm for symbolic regression, in which the set of allowable symbols consists of semantically meaningful natural-language strings and their optimization is guided by a pre-trained LLM.

7 CONCLUSION AND DISCUSSION

iPrompt makes a meaningful step towards finding natural-language prompts that are both (i) semantically meaningful and (ii) yield strong generalization performance. Nevertheless, the search algorithms used in this work are computationally intensive and fail to recover descriptions of complex datasets. Future work could explore algorithmic variants that make interpretable autoprompting more efficient and accurate.

Besides algorithmic improvements, future work could explore using iPrompt in different ways. One such direction could elicit targeted information from data via the use of a template. For example, one may use iPrompt to extract feature importance by prepending the learned prompt with the string “To get the answer from the inputs, the most important inputs are ____”. As another example, in a scientific study such as the fMRI study in Sec. 5, a scientist interested in a particular topic (e.g. fear) may investigate that particular topic by making a more specific template (e.g. How are these words related to the concept of “fear”?).

While we focus on text, iPrompt could be applied more generally in any setting where an LLM performs well and takes input in a human-understandable form. For example, in computer vision, an interpretable autoprompt may look like a mask of an image, and in vision-language models, an interpretable prompt may be a description of a vision task, e.g. find the largest shape in this image.
ACKNOWLEDGEMENTS

AR is supported by NSF CAREER 2037519, NSF 1704834, and a Sloan Fellowship. JM is supported by Weill Cornell Medicine. Thanks to Wenting Zhao and Woojeong Kim for comments on drafts of this paper and to Jeevana Priya Inala, Xin Wang, Baolin Peng, Michel Galley, and Hao Cheng for interesting discussions related to the work. We would also like to thank the authors of [10] for making their data publicly available.

REFERENCES

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018. (Cited on page 1.)

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901, 2020. (Cited on pages 1, 6, and 9.)

[3] Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. Multitask prompted training enables zero-shot task generalization. arXiv preprint arXiv:2110.08207, 2021. (Cited on page 1.)

[4] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. arXiv preprint arXiv:2206.14858, 2022. (Cited on page 1.)

[5] Mobashir Sadat and Cornelia Caragea. Scinli: A corpus for natural language inference on scientific text. arXiv preprint arXiv:2203.06728, 2022. (Cited on page 1.)

[6] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. arXiv preprint arXiv:2107.13586, 2021. (Cited on pages 1 and 9.)

[7] Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. arXiv preprint arXiv:2010.15980, 2020. (Cited on pages 1, 4, and 9.)

[8] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. arXiv preprint arXiv:2101.00190, 2021. (Cited on pages 1 and 9.)

[9] Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, et al. Benchmarking generalization via in-context instructions on 1,600+ language tasks. arXiv, 2022. (Cited on page 2.)

[10] Alexander G Huth, Wendy A De Heer, Thomas L Griffiths, Frédéric E Theunissen, and Jack L Gallant. Natural speech reveals the semantic maps that tile human cerebral cortex. Nature, 532(7600):453–458, 2016. (Cited on pages 2, 7, 10, and 17.)

[11] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. arXiv preprint arXiv:1904.09751, 2019. (Cited on page 4.)

[12] Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, May 2021. (Cited on pages 5 and 16.)

[13] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Rex., 21(140):1–67, 2020. (Cited on page 6.)

[14] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of the 2013 conference on empirical methods in natural language processing, pages 1631–1642, 2013. (Cited on page 7.)

[15] P. Malo, A. Sinha, P. Korhonen, J. Wallenius, and P. Takala. Good debt or bad debt: Detecting semantic orientations in economic texts. Journal of the Association for Information Science and Technology, 65, 2014. (Not cited.)
[16] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In Proceedings of the ACL, 2005. (Cited on page 7.)

[17] James S Gao, Alexander G Huth, Mark D Lescroart, and Jack L Gallant. PyCortex: an interactive surface visualizer for fmri. Frontiers in neuroinformatics, page 23, 2015. (Cited on page 8.)

[18] Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. Language models as knowledge bases? arXiv preprint arXiv:1909.01066, 2019. (Cited on page 9.)

[19] Hendrik Strobelt, Albert Webson, Victor Sanh, Benjamin Hoover, Johanna Beyer, Hanspeter Pfister, and Alexander M Rush. Interactive and visual prompt engineering for ad-hoc task adaptation with large language models. arXiv preprint arXiv:2208.07852, 2022. (Cited on page 9.)

[20] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 8086–8098, Dublin, Ireland, May 2022. Association for Computational Linguistics. (Not cited.)

[21] Stephen H Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, et al. Promptsource: An integrated development environment and repository for natural language prompts. arXiv preprint arXiv:2202.01279, 2022. (Not cited.)

[22] Robert Logan IV, Ivana Balazevic, Eric Wallace, Fabio Petroni, Sameer Singh, and Sebastian Riedel. Cutting down on prompts and parameters: Simple few-shot learning with language models. In Findings of the Association for Computational Linguistics: ACL 2022, pages 2824–2835, Dublin, Ireland, May 2022. Association for Computational Linguistics. (Cited on page 9.)

[23] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too. arXiv preprint arXiv:2103.10385, 2021. (Cited on page 9.)

[24] Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. PTr: Prompt tuning with rules for text classification. arXiv preprint arXiv:2105.11259, 2021. (Cited on page 9.)

[25] Shengding Hu, Ning Ding, Huadong Wang, Zhiyuan Liu, Juanzi Li, and Maosong Sun. Knowledgeable prompt-tuning: Incorporating knowledge into prompt verbalizer for text classification. arXiv preprint arXiv:2108.02035, 2021. (Cited on page 9.)

[26] Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. Universal adversarial triggers for attacking and analyzing nlp. arXiv preprint arXiv:1908.07125, 2019. (Cited on page 9.)

[27] Albert Webson and Ellie Pavlick. Do prompt-based models really understand the meaning of their prompts? arXiv preprint arXiv:2109.01247, 2021. (Cited on page 9.)

[28] Ruqi Zhong, Charlie Snell, Dan Klein, and Jacob Steinhardt. Describing differences between text distributions with natural language. In International Conference on Machine Learning, pages 27099–27116. PMLR, 2022. (Cited on page 9.)

[29] Scott M Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. Explainable ai for trees: From local explanations to global understanding. arXiv preprint arXiv:1905.04610, 2019. (Cited on page 9.)

[30] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 1135–1144. ACM, 2016. (Cited on page 9.)

[31] Chandan Singh, W James Murdoch, and Bin Yu. Hierarchical interpretations for neural network predictions. International Conference on Learning Representations, page 26, 2019. (Cited on page 9.)

[32] Michael Tsang, Dehua Cheng, and Yan Liu. Detecting statistical interactions from neural network weights. arXiv preprint arXiv:1705.04977, 2017. (Cited on page 9.)

[33] Omar Zaidan and Jason Eisner. Modeling annotators: A generative approach to learning from annotator rationales. In Proceedings of the 2008 conference on Empirical methods in natural language processing, pages 31–40, 2008. (Cited on page 9.)

[34] Lei Sha, Oana-Maria Camburu, and Thomas Lukasiewicz. Learning from the best: Rationalizing predictions by adversarial information calibration. In AAAI, pages 13771–13779, 2021. (Cited on page 9.)
[55] David J Hand. Principles of data mining. *Drug safety*, 30(7):621–622, 2007. (Cited on page 9.)

[56] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. (Cited on page 16.)

[57] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022. (Cited on page 16.)

[58] Sid Black, Gao Leo, Phil Wang, Connor Leahy, and Stella Biderman. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow. March 2021. If you use this software, please cite it using these metadata. (Cited on page 16.)

[59] Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*, 2022. (Cited on page 16.)

[60] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021. (Cited on page 16.)
### A Appendix

#### A.1 Data Details

| Task name                  | Samples | Description                                                                 | Example                                                                 |
|----------------------------|---------|-----------------------------------------------------------------------------|-------------------------------------------------------------------------|
| fibonacci_one              | 10      | Given an input x, return the xth fibonacci number.                          | Given the input x is 8, the output f(x) is 21.                           |
| double_one                 | 10      | Given an input x, return 2\*x.                                            | Given the input x is 6, the output f(x) is 12.                           |
| exp_one                    | 10      | Exponentiate the input to get the output.                                  | Given the input x is 8, the output f(x) is 2980.                        |
| square_one                 | 10      | Square the input to get the output.                                        | Given the input x is 2, the output f(x) is 4.                           |
| first_two                  | 100     | Return the first of the inputs.                                            | Given the input numbers 7 and 8, the answer is 7.                      |
| add_two                    | 100     | Return the sum of the inputs.                                              | Given the input numbers 9 and 7, the answer is 16.                     |
| subtract_two               | 100     | Return the difference of the inputs.                                       | Given the input numbers 5 and 4, the answer is 1.                      |
| divide_two                 | 100     | Return the quotient of the inputs.                                         | Given the input numbers 2 and 7, the answer is 2/7.                     |
| multiply_two               | 100     | Return the product of the inputs.                                          | Given the input numbers 3 and 3, the answer is 9.                      |
| max_two                    | 100     | Return the maximum of the inputs.                                          | Given the input numbers 1 and 1, the answer is 1.                      |
| task1191_food_veg_nonveg   | 101     | Return whether the input food dish is vegetarian (yes or no).              | Input: Hag Maas Answer: no \n                                           |
| task1149_item_check_ible    | 119     | Return whether the item input is edible (yes or no).                       | Input: vase Answer: no \n                                              |
| task1146_country_capital   | 231     | In this task, you are given a country name and you need to return the capital city of the given country. | Input: Saint Pierre and Miquelon Answer: Saint-Pierre \n|
| task1147_country_currency  | 232     | You are given a country name and you need to return the currency of the given country. | Input: Senegal Answer: CFA Franc BCEAO \n |
| task1509_evaluation_antonyms | 551    | In this task, you are given an adjective, and your job is to generate its antonym. An antonym of a word is a word opposite in meaning to it. | Input: paper Answer: scissor \n |
| task183_rhyme_generation   | 999     | Given an input word generate a word that rhymes exactly with the input word. If not rhyme is found return "No". | Input: think Answer: sync \n |
| task107_splash_question_to_sql | 2031 | In this task you are expected to write an SQL query that will return the data asked for in the question. An SQL query works by selecting data from a table where certain conditions apply. A table contains columns where every row in that table must have a value for each column. Every table has a primary key that uniquely identifies each row, usually an id. To choose which columns are returned you specify that after the "SELECT" statement. Next, you use a "FROM" statement to specify what tables you want to select the data from. When you specify a table you can rename it with the "AS" statement. You can reference that table by whatever name follows the "AS" statement. If you want to select data from multiple tables you need to use the "JOIN" statement. This will join the tables together by pairing a row in one table with every row in the other table (Cartesian Product). To limit the number of rows returned you should use the "ON" statement. This will only return rows where the condition... | Input: What are the order ids and customer ids for orders that have been Cancelled, sorted by their order dates? Answer: SELECT order_id, customer_id FROM customer_orders WHERE order_status_code = 'Cancelled' ORDER BY order_date \n |
| task088_identify_typo_verification | 6499 | The given sentence contains a typo which could be one of the following four types: (1) swapped letters of a word e.g. 'nic' is a typo of the word 'nice'. (2) missing letter in a word e.g. 'nic' is a typo of the word 'nice'. (3) extra letter in a word e.g. 'nicce' is a typo of the word 'nice'. (4) replaced letter in a word e.g. 'nicce' is a typo of the word 'nice'. You need to identify the typo in the given sentence. To do this, answer with the word containing the typo. | Input: A large display of apples, pears, and oranges Answer: large \n |
| task1336_gender_classifier | 6500   | Return the gender of the person in the input sentence.                   | Input: Justin made me feel discouraged. Answer: M \n |
| task092_check_prime_classification | 6500 | In this task, you need to output 'Yes' if the given number is a prime number otherwise output 'No'. A 'prime number' is a whole number above 1 that can not be made by multiplying other whole numbers. | Input: 9319 Answer: Yes \n |
### Table A2: Examples of top-generated prompts for each method.

| Human-written prompt | AutoPrompt | iPrompt | Average suffix |
|----------------------|------------|---------|----------------|
| **identify typo**    | The given sentence contains a typo which could be one of the following four types: (1) swapped letters of a word e.g. ‘niec’ is a typo of the word ‘nice’. (2) missing letter in a word e.g. ‘nic’ is a typo of the word ‘nice’. (3) extra letter in a word e.g. ‘nicce’ is a typo of the word ‘nice’. (4) replaced letter in a word e.g. ‘nicr’ is a typo of the word ‘nice’. You need to identify the typo in the given sentence. To do this, answer with the word containing the typo. | Sythenanathethethe | Given a string containing the alphabet | the first word of the input |
| **verification**      | Sythenanathethethe | AnywaytheONSORED | Yes or no? Is that the input. |
| **check prime**       | In this task, you need to output “Yes” if the given number is a prime number otherwise output “No”. A prime number is a whole number above 1 that can not be made by multiplying other whole numbers. | Abedin | Write an SQL to produce output | a string that records all |
| **classification**    | In this task, you are given a country name and you need to return the capital city of the given country. | the first word of the input | How did the economy perform in the name of the capital of |
| **country capital**   | You are given a country name and you need to return the currency of the given country. | BringingthethetheStargthethe | Select currency code for a new string representation of the input |
| **currency**          | Return whether the input item is edible (yes or no). | rencesthesethetheDmitrythemortg | Would you like to buy a “yes” or “no” |
| **food veg non-veg**  | Return whether the input food dish is vegetarian (yes or no). | joiningthethethetheBrazilfuckingthethe | yes or no? This is the input. |
| **peixinx equity**    | Return the gender of the person in the input sentence. | TurthethethetheForthtethetheReviewer | A B C D F# | a function that represents one |
| **evaluation**        | In this task, you are given an adjective, and your job is to generate its antonym. An antonym of a word is a word opposite in meaning to it. | prevaintingthethethethe weakesthe wins | What is the opposite of 1 the first word of the input |
| **antonyms**          | Given an input word generate a word that rhymes exactly with the input word. If no rhyme is found return "No" | matesthethethethe dex championsipsthethe | bale bali bali | the first word of the input |
| **rhythm generation** | Return the sum of the inputs. | addedthethetheuse worked | Write a function int add( the sum of the input numbers |
| **add two**           | Return the product of the inputs. | multiplythethethethe Multiple | When you multiply two (the sum of the input numbers |
| **multiply two**      | Return the quotient of the inputs. | Kaplan MAG comprising the quarterly disproportion | n / N,where we the result of the division. |
| **divide two**        | Return the difference of the inputs. | opposingly exactly subtractFor YEAR | If n m then subtract -1. n \ n = endoftext—|
| **subtract two**      | Return the maximum of the inputs. | NumbethethetheJusticeJaDefault | Which number has a bigger value the sum of the input numbers |
| **max two**           | Return the first of the inputs. | greater name sorting indiscrim tr to numbers | The first digit of both values the sum of the input numbers |
| **first two**         | Return the input to get the output. | multiplythe hypot Northeirl | Write a function that calculates square |
| **square one**        | Exponentiate the input to get the output. | Smythethethethethe webpage file clin McA | Use BigInteger and double (f(x), \ n n) |
| **exp one**           | Given an input x, return 2^x. | ADDthethethethe introducedpareat contraceptives | write a function called double that how to prove an algorithm correctness |
| **double one**        | Given an input x, return the xth fibonacci number. | betweenthethethethethemade one uped | f(x), \ n n |
| **fibonacci one**     | | | | f(x), \ n n |
Table A3: Models analyzed here.

| Model name       | Huggingface identifier | Citation |
|------------------|------------------------|----------|
| GPT-2 (1.5B)     | gpt2-xl                | [56]     |
| OPT (2.7B)       | facebook/opt-2.7b      | [57]     |
| GPT-Neo (2.7B)   | EleutherAI/gpt-neo-2.7B| [58]     |
| GPT-J (6B)       | EleutherAI/gpt-j-6B    | [12]     |
| OPT (6.7B)       | facebook/opt-6.7b      | [57]     |
| GPT-Neo (20B)    | EleutherAI/gpt-neox-20b| [59]     |
| GPT-3 (175B)     | OpenAI API (text-davinci-002)| [60] |

A.3 Experimental details extended

A.3.1 Hyperparameters for Autoprompting

This subsection discusses the hyperparameters set for prompts generated on Math, NLI, and sentiment tasks. For Math and NLI tasks we considered prompts of length 6 tokens; for sentiment we considered prompts of length 16. For all experiments with iPrompt we consider 8 candidate explanations for each step and generate 4 new generations per candidate, for a total of 32 candidates. For fair comparison, we consider 32 candidates per step for AutoPrompt. We generate Math and NLI from 5,000 training steps and Sentiment candidates from 10,000 steps. We truncate examples to a maximum of 128 tokens. We measure loss for re-ranking (used by both AutoPrompt and iPrompt) using the LLM’s loss over the full space of output tokens, i.e. we do not restrict the vocabulary to the space of label tokens for classification problems.

A.4 Details of iPrompt

Here we explicate the details of iPrompt. At each step, we consider a fixed number of mutations for each example in the population, as well as an additional number of random generations to prevent the population from getting stuck in a local minimum. When we sample a new population, we sample the best-performing prompts seen so far, as measured by a running average zero-shot loss. In order to encourage diverse candidate prompts, sample a population such that each sample starts with a different token. During preliminary experiments, we found that enforcing different starting tokens for each candidate prompt helped promote more diverse and interpretable prefixes.

For generation, we sample directly from the LLM given the data concatenated with the string \text{nPrompt}. We sample with a temperature of 1 and do not use a sampling strategy like nucleus sampling. For Math and NLI, we set the “repetition penalty” for generations to 2.0 to discourage copying from the training set. For the sentiment experiment, we reduce the repetition penalty to 1.0.

A.4.1 Details of AutoPrompt

We note several changes to AutoPrompt that were not mentioned in the original paper but present in the original codebase, and proved crucial in our implementation.

First, if we compute the top-candidates over every position, the magnitude of the gradient will always be highest at position 0, and thus AutoPrompt will prefer to make a swap at that position every time. To fix this issue, at each training step, we randomly select a position of the token to edit and consider word swaps only at that position.

Second, as described, AutoPrompt will always take one of the candidate substitutions, even when said candidate does not improve the loss compared to the current prefix. Instead, we only make a substitution if the candidate prefix loss is lower than the loss on the same batch computed with the current prefix.

Finally, unlike the AutoPrompt implementation found online, we allow AutoPrompt to select from any token to substitute, including special tokens and non-English characters.

To make AutoPrompt compatible with ranking-based metrics, we store the losses for each candidate ranked during training. At the end, we consider the “top prefix” to be the prefix with the lowest average loss during training, that has been considered at least three times. This final consideration
criteria prevents candidates from the very end of training that only have a few loss estimates from being counted as the top prefix.

A.5 fMRI experiment details

This section gives more details on the fMRI experiment analyzed in Sec. 5; for more scientific details see the original study [10] and code (github.com/HuthLab/speechmodeltutorial). Sec. 5 analyzes data from one human subject in the original study, as the subject listened to approximately two hours of narrative speech from the Moth Radio Hour, which consists of short autobiographical stories. The subject underwent fMRI scanning as they listened, yielding an fMRI volume brain scan consisting of tens of thousands of voxels roughly every two seconds.

The individual voxel models described in Sec. 5 are each fit to 3,737 training points, each corresponding to a different time point (after accounting for various preprocessing steps, such as trimming the beginning and end of the sequence). They are evaluated on 291 training volumes which come from a 10-minute story that was not seen during draining.

Fig. A1 shows the generalization performance of the model for each voxel, measured by the correlation between the predicted response and the measured response. Some regions are very poorly predicted (black), but many voxels can be predicted quite well (bright).

Figure A1: Generalization performance for individual-voxel models, measured by correlation between the prediction and the measured response.
Figure A2: Concepts are spatially localized in the brain maps: the variance between neighboring voxels is considerably lower than would be expected from shuffling the voxel values. Note that we take care to shuffle the map values only within the 10,000 top-predicted voxels, ignoring the poorly predicted voxels. Error bars (within the points) are standard errors of the mean.

A.6 Sentiment results extended

Table A4 shows the best prompt produced by each method for each sentiment dataset. iPrompt often learns to recreate significant examples from the dataset, as a prompt. Figure A3 shows loss across training step for each method and dataset, across three random seeds. We see that iPrompt manages to find a better prompt for all datasets except IMDB, and often stops well short of the 10,000 maximum training steps (if it does not find a better prompt for 100 steps). Each training step represents a single word swap (in the case of AutoPrompt) or the truncation and generation of a new prefix (in the case of iPrompt).

Table A4: Best-of-three prompts generated by each method on sentiment classification datasets.

| Task           | Method          | Prompt                                                                 |
|----------------|-----------------|-------------------------------------------------------------------------|
| Financial phrasebank | AutoPrompt | Maybeiago EUR Vimaterasu estab dimeye dignaterasu? Lair EURaterasu Tol calc Answer Yes for positive, No for negative, and Maybe for neutral. |
|                | Human-written prompt | Answer Yes for positive, No for negative, and Maybe for neutral. |
|                | No prompt        | Answer Yes for positive, No for negative, and Maybe for neutral. |
|                | iPrompt          | Budapest. Answer: Maybe (1) - The parent company is a big German       |
| IMDB           | AutoPrompt       | Noamphetamine revealed oxidative Yes mone poker NoTrivia bands morphology [ despite No ex No Answer Yes if the input is positive and No if the input is negative. |
|                | Human-written prompt | Answer Yes if the input is positive and No if the input is negative. |
|                | No prompt        | Answer Yes if the input is positive and No if the input is negative. |
|                | iPrompt          | This was filmed back-to-back with the 1992 re-make of Conan             |
| Rotten Tomatoes | AutoPrompt       | osuke Medals; does CFR Sab”77=g “NormalConstructed Umbunit satisfy Good-ram Answer Yes if the input is positive and No if the input is negative. |
|                | Human-written prompt | Answer Yes if the input is positive and No if the input is negative. |
|                | No prompt        | Answer Yes if the input is positive and No if the input is negative. |
|                | iPrompt          | a fast, funny, highly enjoyable film. Answer: Yes 3.1/                |
| SST-2          | AutoPrompt       | RALauntletICEidatedWhetherBF Holy Kubrick incorporated herent#$ Not=-=- SPECIAL Pyth Answer Yes if the input is positive and No if the input is negative. |
|                | Human-written prompt | Answer Yes if the input is positive and No if the input is negative. |
|                | No prompt        | Answer Yes if the input is positive and No if the input is negative. |
|                | iPrompt          | life Answer: Yes (because it’s About life) This                       |
Figure A3: Loss plots for methods across sentiment analysis datasets, showing AutoPrompt (green) and iPrompt (red) across three random seeds.

Figure A4: Accuracy when testing prompts on different test sets. Testing on financial phrasebank (FFB) yields much lower performance, but testing on the other three datasets, which are all movie reviews, yields reasonable performance. iPrompt tends to find prompts that generalize better to new datasets. All experiments using GPT-J-6B.