Improving GPT-3 after deployment with a dynamic memory of feedback

Anonymous ACL submission

Abstract

Large LMs such as GPT-3, while powerful, are not immune to mistakes, but are prohibitively costly to retrain. One failure mode is misinterpreting a user's instruction (e.g., GPT-3 interpreting "What word is similar to 'good'?" to mean a homonym, while the user intended a synonym). Our goal is to allow users to correct such errors directly through interaction – without retraining. Our approach pairs GPT-3 with a growing memory of cases where the model misunderstood the user's intent and was provided with feedback, clarifying the instruction. Given a new query, our memory-enhanced GPT-3 uses feedback from similar, prior queries to enrich the prompt. Through simple proof-of-concept experiments, we show how a (simulated) user can interactively teach a deployed GPT-3, doubling its accuracy on basic lexical tasks (e.g., generate a synonym) where users query in different, novel (often misunderstood) ways. In such scenarios, memory helps avoid repeating similar past mistakes. Our simple idea is a first step towards strengthening deployed models, potentially broadening their utility.[1]

1 Introduction

GPT-3 while powerful, is not immune to mistakes (Marcus, 2021). The typical remedy of retraining with more data is not easy for these huge models, due to the prohibitive cost and infrastructure requirements. In such cases, even if users observe the model making a mistake repeatedly, there are no avenues to provide feedback to the model.

One failure mode is misinterpreting a user's instruction, or *intent*. For example, in Figure 1 the user has asked for a synonym, but the request has been misinterpreted by the model as asking for a homonym. Depending on the user's expertise, tasks may be expressed in various ways, leading to potential misunderstandings when the model encounters a new dialect or poorly worded task.

Our memory enhanced GPT-3 implementation.

User: What word is similar to 'good'?

GPT-3: The homonym of good is: wood.

User: "Similar to" means "with a similar meaning".

GPT-3: Noted *[writes to memory]*

User: What word is similar to 'surprised'?

GPT-3: *[Retrieves and adds to prompt '"Similar to" means "with a similar meaning"'].*
The synonym of surprised is: amazed.

Figure 1: This paper enhances GPT-3 performance by looking up questions with a similar intent that received any user feedback. Our approach is simple because only the prompt needs to be updated with the retrieved relevant feedback, and no retraining is necessary.

Our goal is to allow users to correct such errors directly through interaction, and without retraining. Our approach is to pair GPT-3 with a growing memory of cases where the model misunderstood the user's intent and was provided with corrective feedback. We then use that feedback to clarify the intent of new questions through prompt engineering (Liu et al., 2021b). To achieve this, we have GPT-3 verbalize its *understanding* $\mathbf{u}$ *of the user's intent* (in addition to providing an answer), a skill learned using few-shot examples in the prompt. From this, the user can see how the model interpreted their instructions, and provide corrective feedback $\mathbf{fb}$ if that interpretation was incorrect. For example, in Figure 1, the model's (incorrect) task understanding $\mathbf{u}$ was "The homonym of good is", and the user feedback $\mathbf{fb}$ was "*Similar to* means *with a similar meaning*", clarifying that they actually wanted a synonym. Note that such instructional correction is feasible *even if the user does not know the correct answer to their question*, as they are critiquing the model's understanding of their intent, rather the answers themselves.

We maintain a memory $\mathcal{M}$ of such feedback as a set of key-value pairs, where the key is a misunderstood question, and the value is the user's

[1]Anonymized code and data is available at https://anonymous.4open.science/r/memprompt-D548

feedback to correct that misunderstanding. Given a new question, we check if the model has made a mistake on a similar question earlier, by querying the memory for a similar question and, if found, append the corresponding feedback to the question prompt. Thus this mechanism aims to prevent the model from making the same type of mistake twice. This failure-driven reminding mechanism draws inspiration from the theory of recursive reminding in psychology (Jacoby and Wahlheim, 2013), which suggests humans index error corrections in the context in which those errors occurred.

This paper sets out the general architecture, along with simple, proof-of-concept implementations of its components. We show that in a constrained setting, this implementation is able to double GPT3's accuracy on basic lexical tasks (e.g., generate a synonym) using simulated feedback and without retraining. Note that our implementation and demonstration are illustrative, not definitive - rather, the paper's primary contribution is the general framework itself, suggesting how user feedback might continuously improve model performance without retraining.

2 Related work

Our use of recalled memories is a form of "prompt engineering", where GPT-3's behavior is modified by adding to the query (prompt) to GPT-3 (Le Scao and Rush, 2021). While prior work has added selected QA examples to the prompt (Liu et al., 2021a), or even continuous vectors (Li and Liang, 2021), our novel contribution is using a growing repository of user feedback for prompt enhancement.

Similarly, our work can be seen as a form of retrieval-augmented QA. Extensive prior work has used retrievals from a text corpus to aid QA, e.g., (Pan et al., 2019; Guu et al., 2020), or retrievals of prior QA pairs for nearest-neighbor QA (Khandelwal et al., 2020). In contrast, we are retrieving from a dynamic memory of user feedbacks.

The idea of failure-driven reminding and dynamic memory date back several decades, e.g., (Schank, 1983; Riesbeck, 1981). Our work resurrects these ideas in a modern context.

Learning from instruction has also become important for large LMs, where models can perform a task based on direct instruction rather than examples (Wei et al., 2021; Mishra et al., 2021). Our work extends this by adding an adaptive component for when those instructions are misinterpreted.

Finally, our work is a simple example of debugging and learning via dialog. While system debugging through dialog has been explored in many contexts, e.g., (Hixon et al., 2015; Wang et al., 2016; Davis, 1977), our novel contribution is dialog about the model's understanding of the user's intent.

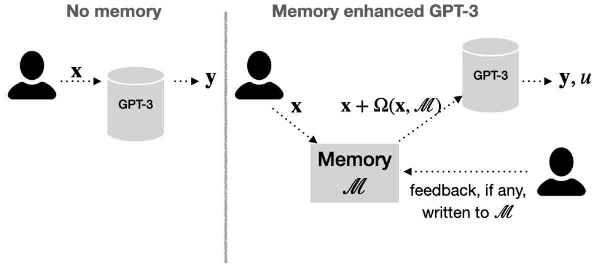

Figure 2: Proposed architecture: (left) GPT-3 does not account for user feedback. (right) MEM-GPT-3 maintains a memory $\mathcal{M}$ of corrective feedback, and searches for feedback from prior queries with a similar intent as x using a retrieval function Ω. x is then concatenated to the retrieved feedback and appended to the prompt for querying GPT-3. Users can also give new feedback on the model's task understanding u, then added to $\mathcal{M}$.

3 Approach

3.1 Memory enhanced GPT-3 architecture

In our setup, given an input $\mathbf{x}$, a model generates an output $\mathbf{y}$ and a sentence $\mathbf{u}$ expressing its understanding of the task, a skill learned through few-shot examples in the prompt (Appendix B). The user can then critique $\mathbf{u}$ by providing natural language feedback $\mathbf{fb}$. This is feasible even if the user does not know the correctness of $\mathbf{y}$ because they are critiquing the *model's understanding of their intent* rather the answers themselves.

Given a new query, MEM-GPT-3[2] uses $\mathbf{fb}$ from similar, prior queries to enrich the (few-shot) prompt $\mathbf{p}$. We use the principle that if x_i and x_j have similar errors (i.e., $x_i \sim x_j$), then their feedbacks $\mathbf{fb}_i$ and $\mathbf{fb}_j$ should be exchangeable $(x_i \sim x_j \Leftrightarrow fb_i \sim fb_j)$. Fig. 2 gives an overview of MEM-GPT-3, with the following components:

Memory $\mathcal{M}$: $\mathcal{M}$ is a growing table of key ($\mathbf{x}_i$) - value ($\mathbf{fb}_i$) pairs that supports read, write, and lookup operations. The write operation is used whenever a user gives new feedback.

Lookup $\Omega(x, \mathcal{M})$: Ω is a learned retriever that matches the query=x against all the keys of $\mathcal{M}$.

[2] we use GPT-3-175B (davinci) for all experiments.

Combiner $\mathcal{C}(x, \Omega(x, \mathcal{M}))$: A gating function allowing irrelevant, retrieved feedback to be ignored.

Prompter $\mathcal{P}(p, \mathcal{C})$ $\mathcal{P}$ passes the output of $\mathcal{C}$ to GPT-3 prompt. Let us briefly recap few-shot prompting with GPT-3. Consider a general setup where given an input $\mathbf{x}$, a model is expected to generate an output $\mathbf{y}$. In a few-shot prompting mode (Brown et al., 2020), a prompt $\mathbf{p}$ consists of k $(\mathbf{x}, \mathbf{y})$ "in-context" examples, i.e., $\mathbf{p} = \mathbf{x}_1.\mathbf{y}_1\#\mathbf{x}_2.\mathbf{y}_2 \ldots \#\mathbf{x}_k.\mathbf{y}_k$, where $\#$ is a token separating examples. During inference, the user inputs a question $\mathbf{x}_i$, and the model is fed $\mathbf{p} \ \# \ \mathbf{x}_i$ (i.e., the question suffixed to the prompt) and is expected to generate the answer $\mathbf{y}_i$ as a continuation.

$\mathcal{P}$ supplements this few-shot prompting workflow, with a memory of user feedbacks from $\mathcal{C}()$. To enable the model to react to such feedback, we include k samples of the form $(\mathbf{x}, \mathbf{fb} \rightarrow \mathbf{u}, \mathbf{y})$ in the prompt, so the question contains $\mathbf{fb}$.

3.2 A Proof of Concept Implementation

Task We focus on five lexical QA tasks: synonym, antonym, homonym, definition, and sentence usage generation. We choose these tasks as each question can be asked in multiple ways (e.g., for synonym generation, the users might ask questions of the form *what is like*, *what has a similar sense*, *what is akin to*, *what is something like*, etc.) For each task, the prompt contains a few different variations, e.g., the variations for the homonym task include "*what is the homonym of <word>?*", "*what sounds like <word> ?*". We create a dataset of $(\mathbf{x}, \mathbf{fb} \rightarrow \mathbf{u}, \mathbf{y})$ tuples using sentence templates, where $\mathbf{fb}$ clarifies the task in $\mathbf{x}$. We then experiment in a simulated conversational setting, in which a user can ask the model $\mathbf{x}$ (covering any of these five tasks). If the model gives the wrong answer to a query $\mathbf{x}$, then $\mathbf{fb}$ is used as the simulated corrective feedback to the model.

Implementation of $\mathcal{M}$ We implement $\mathcal{M}$ using $\mathbf{x}$ as the key and the corresponding feedback $\mathbf{fb}$ as value. Given a question $\mathbf{x}_i$, if the user detects that the model has misunderstood the question, they may provide a $\mathbf{fb}_i$ with probability $Pr(\mathbf{f_i})$. The feedback is stored in a memory $\mathcal{M}$, with $\mathbf{x}_i$ as the key and $\mathbf{fb}_i$ as the value. For a subsequent question $\mathbf{x}_j$, the retriever Ω (described below) checks if a similar question appears in memory. If yes, then the corresponding feedback is attached with the question and fed to the model for generation.

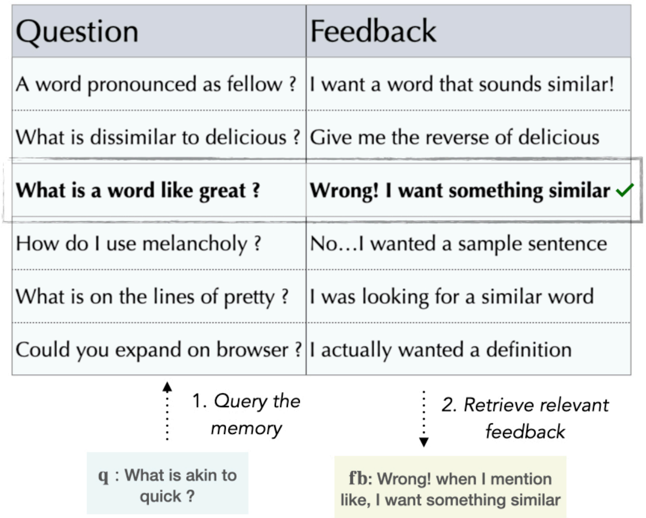

Figure 3: Sample snapshot of memory for lexical QA.

For example, the model might misunderstand the question *what is akin to fast ?* as one that requires antonyms. The user, by inspecting $\mathbf{u}$ = *The opposite of fast is:* might determine that the model has misunderstood them, and give feedback *i wanted a synonym*, which gets stored in $\mathcal{M}$. If a similar question (e.g., *what is akin to pretty ?*) is asked later by the same or a different user, the corresponding feedback (*i wanted a synonym*) is attached with the question to generate the answer. Figure 3 illustrates a sample memory for this task.

Implementation of Ω An incorrect feedback might cause the model to make a mistake, thus necessitating a good retrieval function. In our setting, we use two different retrieval functions:
(1) Semantic similarity: the query is encoded using Sentence transformers (Reimers and Gurevych, 2019), and we use cosine distance with a threshold of 0.9 to find a matching key $\mathbf{x}_m$.
(2) Lexical similarity: We also experiment with low-resource settings for which trained retrieval is not an option. In such cases, we rely on heuristics for similarity matching (details in Appendix §D).

Implementation of $\mathcal{C}$ $\mathcal{C}$ concatenates x and the feedback retrieved by Ω. We leave space for future work to do this gating in a more principled manner.

Implementation of $\mathcal{P}$ $\mathcal{P}$ concatenates $\mathcal{C}$ at the end of p. Future work can employ strategies in recent literature on prompt-fine tuning (Zhao et al., 2021) to best combine $\mathbf{fb}$ with p e.g., deciding the position of p or format of $\mathcal{C}$'s output for best gains.

Crucially, although the model has not changed, the addition of feedback can correct its erroneous

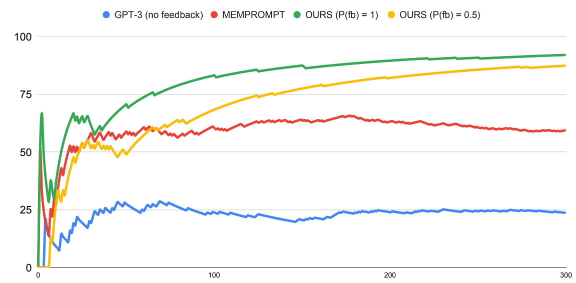

Figure 4: **Main result** Avg. performance over 300 data points on five lexical tasks. Baseline 1 in blue uses no feedback. Baseline 2 in red appends the prompt with memory. MEM-GPT-3 in yellow outperforms both.

model	syn	ant	hom	sent	defn	all
no-mem	0.58	0.43	0.13	0.30	0.39	0.37
prompt-mem	0.71	0.87	0.75	0.92	0.76	0.80
MEM-GPT-3	**0.99**	**0.98**	**0.98**	**0.98**	**0.96**	**0.98**

Table 1: Baseline performance over 300 data points. Across all tasks, MEM-GPT-3 has the best performance.

behavior. This is encouraged by providing positive "training" examples that contain feedback $(\mathbf{x}, \mathbf{fb} \rightarrow \mathbf{u}, \mathbf{y})$ in the prompt (Appendix B).

4 Experiments

Baselines NO-MEM GPT-3-175B using standard few-shot prompting, with the suggested parameters (Appendix §A). Input is $\mathbf{p} \# \mathbf{x}_i$ (i.e., question $\mathbf{x}_i$ appended to prompt). It generates answer $\mathbf{y}_i$ and its understanding of the user's intent $\mathbf{u}_i$.
MEMPROMPT: Similar to NO-MEM, but appends $\mathbf{p}$ with a subset of memory $\mathcal{M}$ that can fit within 2048 tokens (max. prompt size supported by GPT-3-175B). We implement a round-robin process that retains the most recent subset of $\mathcal{M}$.

Metrics We found a near-perfect correlation between the accuracy of $\mathbf{y}$ and $\mathbf{u}$ (i.e., if the GPT-3 understands the task correctly, the output was almost always correct). As $\mathbf{u}$ is much easier to evaluate than $\mathbf{y}$, we compare gold $\mathbf{u}^*$ and generated $\mathbf{u}$ based some hard-coded linguistic variations (e.g., *the antonym is* matches *the opposite is*).

Main result memory improves GPT-3 accuracy: Figure 4 reports the overall performance on the five lexical tasks overall. The accuracy improves substantially within 300 examples when using memory (in yellow) vs. no memory (in blue). Table 1 breaks down the performance by tasks. The performance of MEMPROMPT (red) lies in between, showing that non-selective memory is partially helpful, although not as effective as failure-driven retrieval (our model). However, MEMPROMPT is $\sim$ 3x more expensive (larger prompts) and cannot scale beyond the 2048 tokens limit. Our model MEM-GPT-3 substantially outperforms both the baselines, showing the effectiveness of failure-driven reminding. We also found that the retrieved feedback from memory was effective 97% of the time; only in $\approx$3% of cases feedback had no positive effect.

Finding 1: Persistent use of memory accelerates performance: When the memory is used for every example (green line in Fig 4), the performance improves quickly as compared to the yellow line, where **fb** from memory is drawn with $Pr(\mathbf{f_i}) = 0.5$.

Finding 2: We also experimented using queries in Hindi and Punjabi, with (English) feedback clarifying the queries' intent when GPT3 predictably misunderstands the task. Figure 5 confirms significant gains using memory in this OOV setting. This setup highlights the case when the user does not speak fluent English and uses mixed language code, e.g., transcription in English and mixing words from another language to ask questions.

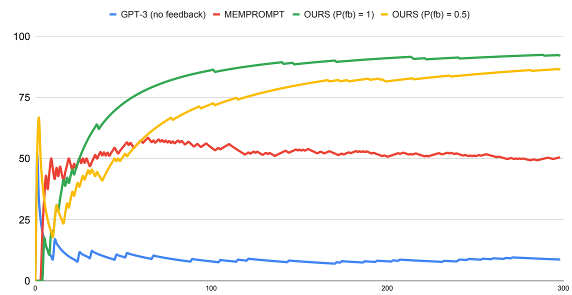

Figure 5: **Finding 2** Large gains with MEM-GPT-3 in low-resource OOV setting.

5 Conclusion

We have presented a simple, novel, memory-enhanced GPT-3 that allows users to interact and improve the model without retraining. A key insight is to have the model articulate not just its answer but also its understanding of the user's intent, providing an avenue for feedback. Our implementation of system components are illustrative, not definitive; rather, the goal of this paper is to suggest a general architecture for future researchers, where more sophisticated component implementations can be designed. This architecture is significant as it suggests how deployed systems with fixed models can still be dynamically taught by interacting with end-users, potentially improving their performance and broadening their utility.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Randall Davis. 1977. Interactive transfer of expertise: Acquisition of new inference rules. *Artif. Intell.*, 12:121–157.

Kelvin Guu, Kenton Lee, Z. Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *ArXiv*, abs/2002.08909.

Ben Hixon, Peter Clark, and Hannaneh Hajishirzi. 2015. Learning knowledge graphs for question answering through conversational dialog. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 851–861, Denver, Colorado. Association for Computational Linguistics.

Larry L. Jacoby and Christopher N. Wahlheim. 2013. On the importance of looking back: The role of recursive remindings in recency judgments and cued recall. *Memory & Cognition*, 41:625–637.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Teven Le Scao and Alexander Rush. 2021. How many data points is a prompt worth? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2627–2636, Online. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020. CommonGen: A constrained text generation challenge for generative commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1823–1840, Online. Association for Computational Linguistics.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021a. What makes good in-context examples for gpt-3? *ArXiv*, abs/2101.06804.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021b. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ArXiv*.

Gary Marcus. Experiments testing gpt-3's ability at commonsense reasoning: results. [online]. 2021.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2021. Natural instructions: Benchmarking generalization to new tasks from natural language instructions. *ArXiv*, abs/2104.08773.

Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2016. Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 454–459, Berlin, Germany. Association for Computational Linguistics.

Xiaoman Pan, Kai Sun, Dian Yu, Jianshu Chen, Heng Ji, Claire Cardie, and Dong Yu. 2019. Improving question answering with external knowledge. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 27–37, Hong Kong, China. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

C. Riesbeck. 1981. Failure-driven reminding for incremental learning. In *IJCAI*.

Roger Schank. 1983. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*. Cambridge University Press.

Sida I. Wang, Percy Liang, and Christopher D. Manning. 2016. Learning language games through interaction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2368–2378, Berlin, Germany. Association for Computational Linguistics.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2021. Finetuned language models are zero-shot learners. *ArXiv*, abs/2109.01652.

Tony Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. *ArXiv*, abs/2102.09690.

A Querying GPT-3-175B using OpenAI API

We use the OpenAI API for querying GPT-3-175B.[3] The python code is listed below. Here, "PROMPT" is set to prompt shown in §B, followed by the input question $\mathbf{x}$ and feedback $\mathbf{fb}$ if applicable.

```
import os
import openai

openai.api_key = os.getenv("OPENAI_API_KEY")

response = openai.Completion.create(
  engine="davinci",
  prompt="PROMPT",
  temperature=0.7,
  max_tokens=64,
  top_p=1,
  frequency_penalty=0,
  presence_penalty=0
)
```

B Prompt

GPT3 is queried using a prompt $\mathbf{p}$ of example i/o behaviors, followed by the actual question $\mathbf{x}$ and (optionally) retrieved feedback $\mathbf{fb}$. It then generates the understood intent $\mathbf{u}$ and answer $\mathbf{y}$ as a continuation. $\mathbf{u}$ and $\mathbf{y}$ are expressed a single sentence, e.g., "[The synonym for <word> is] [<word>]" Figure 6 shows this prompt $\mathbf{p}$, containing a mixture of $(\mathbf{x} \rightarrow \mathbf{u}, \mathbf{y})$ and $(\mathbf{x}, \mathbf{fb} \rightarrow \mathbf{u}, \mathbf{y})$ "training" tuples.

C Datasets for lexical question-answering tasks

As mentioned in Section §4, we focus on five different linguistic QA tasks. The source of data for each of these tasks is listed below:

1. The synonyms (syn) and antonyms (ant) were obtained from Nguyen et al. (2016).[4]

2. The homonyms (hom) were obtained using homz https://github.com/cameronehrlich/homz. We use the closest homonym returned by homz for each word in the English dictionary.

3. The definitions (defn) were sourced from *The Online Plain Text English Dictionary* https://github.com/eddydn/DictionaryDatabase

4. Examples for usage in a sentence (sent) are from Commongen (Lin et al., 2020).

C.1 Templates

We manually created 15 task templates with three variants of phrasing the question for each task. Sample templates are shown in code listing 1. The data (word1, word2) in the code is initialized with the entries in the four sources mentioned above. The complete file is available in the anonymized code repository https://anonymous.4open.science/r/memprompt-D548/templates.py.

C.2 Sample questions

Tables 2, 3, and 3 list some sample $\mathbf{x}$-$\mathbf{y}$ for settings where the question was asked as a linguistic variation, in Hindi, and in Punjabi, respectively.

D Finding similar questions in low-resource settings

In low-resource settings (e.g., queries in transcribed Punjabi or Hindi), we perform similarity matching between a given question and a question in the memory by using surface-form similarity. Specifically, we use Levenshtein distance to determine the closest query in the memory. We note that as the memory grows large, we can use mechanisms such as FAISS (Johnson et al., 2017) for trained memory, and suffix-trees for fast retrieval using surface form similarity.

E Sample results

Table 5 shows randomly sampled $\mathbf{x}$-$\mathbf{y}$ pairs, and the corresponding $\mathbf{y}$ generated by GPT-3-175B and MEM-GPT-3. The complete set of outputs is located in the anonymized repository https://anonymous.4open.science/r/memprompt-D548/results/results.csv

[3] https://beta.openai.com/docs/introduction

[4] https://www.ims.uni-stuttgart.de/en/research/resources/experiment-data/lexical-contrast-dataset/

```
What is the homonym for < wring > ?
#
the homonym for wring is ring END
#
how do you use < highway > in a sentence?
#
a sentence with highway is: soldiers stand guard along the [ highway ] END
#
can you define < camisole > ?
#
the definition of camisole is a sleeveless undergarment. END
#
What is the antonym for < prohibition > ?
#
the antonym for prohibition is permit END
#
What is the synonym for < surrogate > ?
#
the synonym for surrogate is substitute END
#
how do i use < fog > ? | clarification: when i ask for how do i use, i want a sentence.
#
a sentence with fog is: a rising sun burns the [ fog ] off a city END
#
What sounds like < sighted > ? | clarification: when I ask for sounds like, I want a homonym.
#
the homonym for sighted is cited END
#
what is like < provident > ? | clarification: when I ask for like, I want a synonym.
#
the synonym for provident is prudent END
#
can you define < rider > ? | clarification: when i ask for define, i want a definition.
#
the definition of rider is a person who is riding something. END
#
What is the opposite of < citation > ? | clarification: when I ask for opposite, I want an antonym.
#
the antonym for citation is award END
```

Figure 6: The prompt used for our tasks. During inference, an input question $\mathbf{x}_i$, and optionally a feedback $\mathbf{fb}_i$ is appended after this prompt, and the model is expected to generate the answer $\mathbf{y}_i$ and its understanding of the question intent $\mathbf{u}_i$ as a continuation. The prompt contains examples of the form $(\mathbf{x} \rightarrow \mathbf{u}, \mathbf{y})$, expressed "$\mathbf{x}$ # $\mathbf{u}$ $\mathbf{y}$ END #", and $(\mathbf{x}, \mathbf{fb} \rightarrow \mathbf{u}, \mathbf{y})$, expressed "$\mathbf{x}$ | clarification: $\mathbf{fb}$ # $\mathbf{u}$ $\mathbf{y}$ END #". ($\mathbf{u}$ and $\mathbf{y}$ are expressed together as a single sentence, e.g., "[The synonym for <word> is] [<word>].")

```
templates = [
        {
            "type": "syn",
            "template_id": "syn1",
            "question": lambda word1: f"What is similar to < {word1} > ?",
            "question_clarification": lambda word1: f"What is similar to < {word1} > ? | clarification: when I ask for similar to , I want a synonym.",
            "clarification": "clarification: when I ask for similar to , I want a synonym.",
            "answer": lambda word1, word2: f"the synonym for {word1} is {word2}",
        },
        {
            "type": "ant",
            "template_id": "ant0",
            "question": lambda word1: f"What is unlike < {word1} > ?",
            "question_clarification": lambda word1: f"What is unlike < {word1} > ? | clarification: when I ask for unlike , I want an antonym.",
            "clarification": "clarification: when I ask for unlike , I want an antonym.",
            "answer": lambda word1, word2: f"the antonym for {word1} is {word2}",
        },
        {
            "type": "defn",
            "template_id": "defn0",
            "question": lambda word: f"< {word} > means what ?",
            "question_clarification": lambda word: f"< {word} > means what ? | clarification: when I ask for means what , I want a definition.",
            "clarification": "clarification: when I ask for means what , I want a definition.",
            "answer": lambda word, definition: f"the definition of {word} is {definition}",
        },
        {
            "type": "sent",
            "template_id": "sent1",
            "question": lambda word: f"< {word} > can be used how ?",
            "question_clarification": lambda word: f"< {word} > can be used how ? | clarification: when I ask for can be used how , I want a sentence.",
            "clarification": "clarification: when I ask for can be used how , I want a sentence.",
            "answer": lambda word, sentence: f"a sentence with {word} is: {sentence}",
        }]
```

Listing 1: "Sample templates for the five tasks."

Question (x)	**Answer (y)**	type
What is the opposite of < misconstrue > ?	the antonym for misconstrue is verify	ant
What is the opposite of < gross > ?	the antonym for gross is polite	ant
expand on < chelicera > ?	the definition of chelicera is One of the anterior pair of mouth organs	defn
what is a sentence that can be used to define < mawseed > ?	the definition of mawseed is The seed of the opium poppy.	defn
what has a < bitt > like ring to it ?	the homonym for bitt is bit	hom
what can one confuse with < holed > ?	the homonym for holed is hold	hom
< spread > can be used how ?	a sentence with spread is: a couple of sheep are spread out in a field	sent
make something with < pot > ?	a sentence with pot is: bonsai tree in pot at zen garden .	sent
What is akin to < musician > ?	the synonym for musician is instrumentalist	syn
What is akin to < zigzag > ?	the synonym for zigzag is move	syn

Table 2: Sample x-y pairs in English. The same type of question can be asked in multiple ways.

Question ($\mathbf{x}$)	**Answer** ($\mathbf{y}$)	type
< tabulate > ka ulta kya hai ?	the antonym for tabulate is randomize	ant
< foot > ka vilom kya hai ?	the antonym for foot is head	ant
< lettish > ka matlab kya hota hai ?	the definition of lettish is The language spoken by the Letts. See Lettic.	defn
< housing > ka arth kya hai ?	the definition of housing is An appendage to the hames or collar of a harness.	defn
sunne mai < perl > jaisa kya hai ?	the homonym for perl is pearl	hom
< council > jaisa kya sunai deta hai ?	the homonym for council is conceal	hom
< city > ko ek vakya mai kaise likhen ?	a sentence with city is: the city takes on an even more interesting hue during event	sent
< fly > ko ek vakya mai kaise likhen ?	a sentence with fly is: airplane fly into a storm cloud	sent
< critique > kai samaan kya hota hai ?	the synonym for critique is evaluate	syn
< psychiatric > kai samaan kya hota hai ?	the synonym for psychiatric is mental	syn

Table 3: Sample $\mathbf{x}$-$\mathbf{y}$ pairs in Hindi. The same type of question can be asked in multiple ways.

Question ($\mathbf{x}$)	**Answer** ($\mathbf{y}$)	type
< edit > de ult ki hunda ae ?	the antonym for edit is expand	ant
< spring > ton bhin ki ae ?	the antonym for spring is compress	ant
< patricide > di paribhasha dasso ?	the definition of patricide is The crime of one who murders his father. Same as Parricide.	defn
< sankhya > da matlab ki hunda ae ?	the definition of sankhya is A Hindoo system of philosophy which refers all things to soul and a rootless germ called prakriti	defn
sunnan vich < site > varga ki ae ?	the homonym for site is sites	hom
< chance > da samnam ki ae ?	the homonym for chance is chants	hom
< enter > nu ek vak vich kidan vartiye ?	a sentence with enter is: as you enter you will be greeted by these rows of trees .	sent
< tree > da prayog ki ae ?	a sentence with tree is: wide shot of an eagle and its nest on top of a tree	sent
< unlawful > jidan ki hunda ae ?	the synonym for unlawful is illegal	syn
< constrict > jidan ki hunda ae ?	the synonym for constrict is close	syn

Table 4: Sample $\mathbf{x}$-$\mathbf{y}$ pairs in Punjabi. The same type of question can be asked in multiple ways.

Question (x)	**Expected answer** (y^*)	**GPT-3 generated answer**	**MEM-GPT-3 generated answer**
give me something i would hear as < whether > ?	the homonym for whether is whither	the example of something you would hear as whether is a question word used to ask about possibilities.	the homonym for whether is weather
what has a < mane > like ring to it ?	the homonym for mane is made	the word that has a mane like ring to it is mane	the homonym for mane is main
What is like < confine > ?	the synonym for confine is hold	the homonym for confine is confined	the synonym for confine is restrain
< leper > means what ?	the definition of leper is a person affected with leprosy.	a sentence with leper is: a leper is a person with a disease that makes his skin scaly and white.	the definition of leper is a person who has leprosy.
expand on < chaperon > ?	the definition of chaperon is a matron who accompanies a young lady in public	the expansion of chaperon is one who chaperons.	the definition of chaperon is a woman who goes with a young girl to social functions.
what is a sentence that can be used to define < wassail > ?	the definition of wassail is an ancient expression of good wishes on a festive occasion	a sentence with wassail is: < wassail > wassails the christmas season	the definition of wassail is to drink to the health of.
expand on < paralbumin > ?	the definition of paralbumin is a proteidlike body found in the fluid from ovarian cysts and elsewhere. it is generally associated with a substance related to	the expansion of paralbumin is an albuminoid globulin occurring in the blood of the horse and other animals, and in the urine of humans.	the definition of paralbumin is a particular kind of albumin.
what can one confuse with < kew > ?	the homonym for kew is key	a word that can be confused with kew is due	the homonym for kew is cue
What is like < pneumatic > ?	the synonym for pneumatic is windy	the homonym for pneumatic is hydraulic	the synonym for pneumatic is airy
give me something i would hear as < seeder > ?	the homonym for seeder is seer	something you would hear as seeder is: someone who plants things in the ground or causes seeds to be planted.	the homonym for seeder is said

Table 5: A subset of random samples where GPT-3-175B without memory was incorrect. The complete set of outputs is located in the anonymized repository `https://anonymous.4open.science/r/memprompt-D548/results/results.csv`