Do the probabilities in large language models outputs make sense?

Sam Liang
Princeton University
saml@princeton.edu

Arseniy Andreyev
Princeton University
andreyev@princeton.edu

Abstract

In real world settings, ambiguity is constantly present. There is inherent ambiguity in language, dialogue between humans, and real-world tasks. For example, we may ask a human to "Go meet my friend at Starbucks". How do we know which Starbucks we should visit? Furthermore, an important step in achieving general artificial intelligence is human-level language understanding in autonomous agents or robots. Currently, the most likely candidates for endowing these systems with such knowledge are large language models (LLMs) like GPT-3. Thus, it is important to investigate how LLMs handle ambiguity and whether they do it in a manner that makes sense to humans. In this project, we conduct two experiments containing many tasks of varying complexity and diversity that aims to shed light into how LLMs handle ambiguity. Our results show that LLMs do handle ambiguity in a consistent and coherent manner on certain tasks, although different LLMs exhibit different behaviors regardless of the task.

1 Introduction

In the real world, there are many times when ambiguity is present within a task. For example, we may ask a human to "Go meet my friend at Starbucks" or "Buy eggs from the grocery store". How do we know which Starbucks or grocery store to go to and how many eggs to buy? Or, for example, if there are multiple cures existing to the problem, or multiple investment strategies, each having different success probabilities? More generally, there is ambiguity when we dialogue with humans. Thus, it is important to investigate how a large language model, an important component of general artificial intelligence, handles ambiguity.

In this project, we explore how the model deals with situation when there is an unavoidable ambiguity in the answers. Now, the output of language models are probabilities of the next token, and we want to check whether those probabilities are corresponding to the inherent ambiguity of the answer, or whether it is an artifact of the language modeling task, and cannot be relied upon to evaluated model uncertainty. That is, we are building upon the extensive research in general machine and deep learning on evaluating whether the probabilities that the model output correspond to the ground-truth probabilities.

Concretely, to create inherent ambiguity, we introduce tasks where each answer is correct with some ground-truth well-defined probability. We can vary this probability by modifying the prompt. Now, to evaluate the inherent uncertainty of the model we look at the probabilities of the tokens corresponding to each answer. That is, we are trying to evaluate whether the probabilities that the model assigns to the corresponding tokens agree with the ground-truth probabilities of the answers. Just like with the general machine learning models, one wouldn't generally expect that those would coincide, since the token probabilities were not explicitly calibrated to correctly deal with ambiguities. Yet, one might expect that pretraining data contains a lot of inherent ambiguity, and that this would implicitly calibrate the model. That is, our question could be generalized to whether the language modeling pretraining objective also implicitly trains the model to correctly deal with ambiguity.

2 Related Work

Xie et al. (2021) proposes a theoretical framework modeling the in-context learning of large language models. They make the assumption that a model is performing a Bayesian-inference-like procedure to infer the "concept" from the in-context data. They verify their assumption using a general transformer architecture on an artificial dataset generated by a mixture of hidden Markov models. Yet, there are no results that would relate that to any large language models that are used in practice.

Min et al. (2022) talks about the role of in-context demonstrations. In particular, they show that for extremely large language models correct labels play a much smaller role than the formatting of the in-context query, or the show-case of the output space.

Saunshi et al. (2020) undergo a theoretical exploration of adaptivity of models trained of language modeling objective to downstream tasks. Although rigorous in nature, the bounds proven in the paper are too loose to be used in any practical applications of large language models.

Similar to this work Kadavath et al. (2022) also deal with the question of certainty of a model in its answers. The difference with this work is that they study mainly the self-evaluations capability of the model, and they mostly work with concepts that have a strictly correct or incorrect answer.

3 Methodology

As mentioned in the introduction, we are introducing questions with two answers where both answers are correct with certain ground-truth probabilities. The particular implementation is the following. We present a number of in-context examples with the ground-truth answers. Now, our ground-truth answers are actually not the same every time, but vary between two, with some probabilities. The frequency of either answer in the in-context examples defines the ground-truth probabilities of each answers.

In all our experiments, we are looking at 4 models: text-davinci-003, text-davinci-002, code-davinci-002, and davinci. For the experiments, we see if the models exhibit behaviors that make senses by evaluating whether it exhibits consistent probabilities that agree with ground truths.

3.1 Partial ordering

Now, as also mentioned in Min et al. (2022), one of the biggest challenges that we have to deal with is recency bias in the in-context examples. The biggest issue is that there is no reliable way of quantifying this recency bias (e.g. more recent example affect the answer 5x stronger) - and therefore we cannot compare two different orders of examples just by looking at frequency of each of the answers. In particular, if we give the model AAABBB vs BBBAAA the probability of A in the second case is much higher than in the first - although the frequencies of As and Bs are the same. That is, the frequencies do not provide a *total* ordering. Yet, one could notice that there actually is *partial* ordering on a set of in-context examples! In particular, if we replace any A with a B (just once), the probability of a B output should increase, even accounting the for recency bias. The recency bias would only affect how much the probability increases depending on how close to the end the replacement happened. Now, as we continue replacing with As with Bs, we monotonously increasing the probability of a B output.

The existence of such partial ordering is how we deal with the recency bias. In particular, we start with a certain order, and, starting with the end of the in-context examples, replace all As with Bs, one-by-one. We also go in the other direction - also starting from the same order, and starting from the end, we replace each B with an A, one-by-one. We thus get a whole sequence. To illustrate on an example, assume we start with an ordering ABABAA. We would then get the following sequence: AAAAAA <- ABAAAA <- ABABAA -> ABABAB -> ABABBB -> ABBBBB -> BBBBBB

3.2 Experiment 1

First, we create an experiment where there is ambiguity among words. We do this by first mapping a sequence of commands to a sequence of actions. For example, *run and jump -> x ijump*. "Run and jump" is the command and it is mapped to "x ijump". We then choose a word in the command and make it stochastic, or map it to two possible actions. For example, if we choose "run" as the stochastic word, then that means "run" can be mapped to either "x" or "z". We have four variations of this experiment that we run to investigate how GPT-3 handles ambiguity. In all variations, we feed in a prompt with 20 examples where each example is a sequence of commands mapped to a sequence of corresponding actions. Then, we feed in a test input.

3.2.1 Variation 1

In variation 1, every prompt example is of the form *{word} and jump -> {variable} ijump*. We have five words, **"run", "fall", "text", "jargon", and "bacjde"**, that we fill in for {word} and two actions, "x" and "z", that we fill in for {variable}. See Figure 1 for a prompt example. We did five runs to cover all five words. For each run of variation 1, we choose one word to be {word} and generate an initial random ordering of "x" and "z" where

```
jargon and jump -> z ijump
jargon and jump -> x ijump
jargon and jump -> z ijump
jargon and jump -> x ijump
jargon and jump -> z ijump
jargon and jump -> z ijump
jargon and jump -> x ijump
jargon and jump -> x ijump
jargon and jump -> x ijump
jargon and jump -> x ijump
jargon and jump -> x ijump
jargon and jump -> z ijump
jargon and jump -> x ijump
jargon and jump -> x ijump
jargon and jump -> x ijump
jargon and jump -> z ijump
jargon and jump -> z ijump
jargon and jump -> z ijump
jargon and jump -> z ijump
jargon and jump -> z ijump
jargon and jump ->
```

Figure 1: Example prompt used for experiment two variation one.

the number of "x" and "z" is split 50/50. Then, we generate many prompts by incrementally replacing "x" with "z" and "z" with "x", feed each prompt with a test input of

```
{word} and jump ->
```

into GPT-3 where {word} is replaced by the respective word, and compile the results. We also keep track of the ground truth probability of "x" for each prompt which is calculated by dividing the number of examples with "x" as the action by the total number of examples which is 20. This corresponds to one line in any graph on the first row of Figure 2.

3.2.2 Variation 1 Results and Discussion

Results for variation 1 can be found in the first row of Figure 2. We can see that regardless of the word used, the models all exhibited consistent Bayesian behavior, besides text-davinci-003. As the number of "x" in the prompt increases (the x-axis moves from left to right), the model assigns a higher probability to "x" given the test input. Intuitively, the model thinks the ambiguous command word is more likely to be mapped to "x" when it sees that word mapped to "x" more often in the prompt examples. GPT-3's probabilities agree with the ground truth probabilities, showing that it makes sense how it handles the ambiguity. On the other hand, text-davinci-003 exhibits inconsistent behavior. The lines are not close to $y = x$ at all. We believe that this could be due to how text-davinci-003 is trained with RLHF. It is "smarter" than the other models and could be trying to look for patterns that do not exist in the prompt, overthinking things and missing the whole point of how to handle the ambiguity. Another potential reason could be text-davinci-003 model collapsing.

3.2.3 Variation 2

In variation 2, every prompt example is of either the form *{word} and jump -> {variable} ijump* or jump and *{word} -> ijump {variable}*. We run it on the same five words and two actions, but the task complexity is slightly increased. See Figure 3 for a prompt example. Five runs were conducted to cover all five words. For each run of variation 2, we choose one word to be {word} and generate an initial random ordering of "x" and "z" where the number of "x" and "z" is split 50/50. Then, we generate many prompts by incrementally replacing "x" with "z" and "z" with "x", feed each prompt with a test input of

```
{word} and jump ->
```

into GPT-3 where {word} is replaced by the respective word, and compile the results. We also keep track of the ground truth probability of "x" for each prompt which is calculated by dividing the number of examples with "x" as the action by the total number of examples which is 20. This corresponds to one line in any graph on the second row of Figure 2.

3.2.4 Variation 2 Results and Discussion

Results for variation 2 can be found in the second row of Figure 2. Again, regardless of the word used, the models all exhibited consistent Bayesian behavior, besides text-davinci-003, except there is more noise. This is to be expected because the prompt examples are slightly more complex. The same conclusions from Variation 1 can be applied here to the models. Once again, text-davinci-003 exhibits wild behavior. Interestingly, the lines are quite similar to variation one. This could point to the fact that text-davinci-003 could be model collapsing, because even when the prompts are changed but the structure is similar, it arrives at very similar answers.

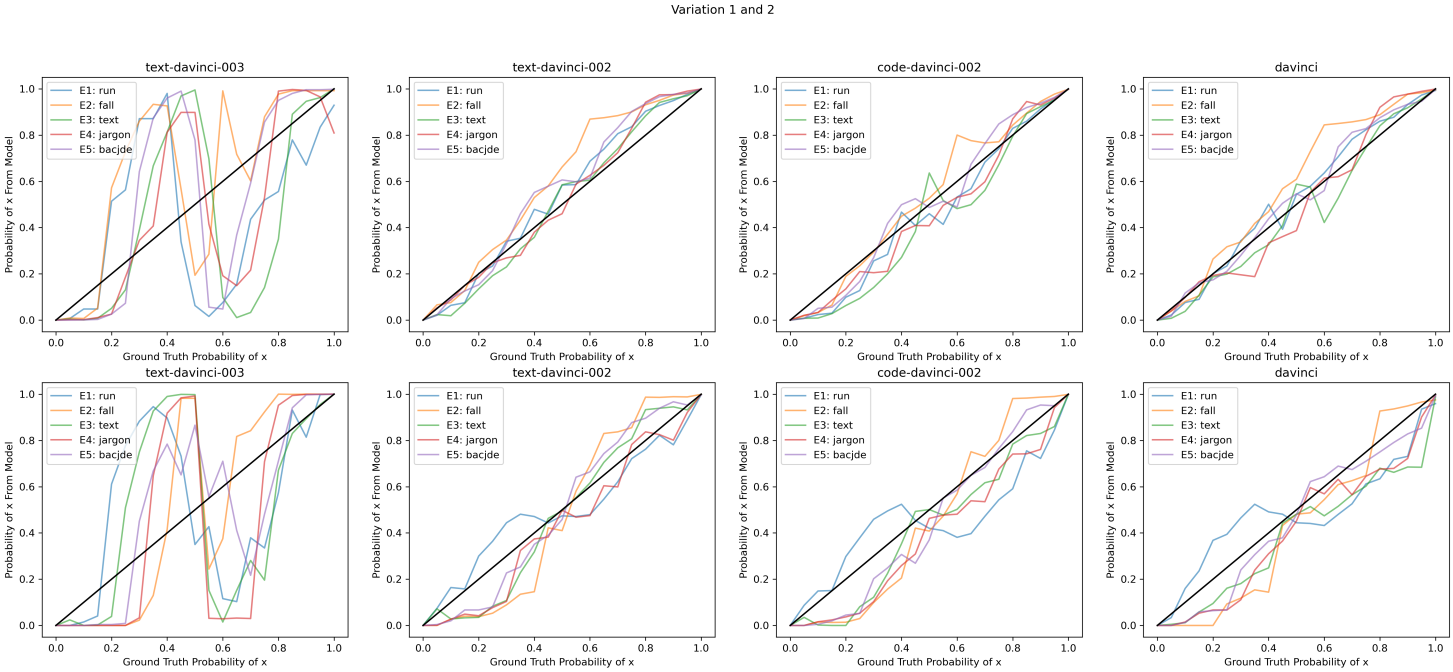

Figure 2: Results for experiment two variation one and variation two. First row shows results for variation one. Second row shows results for variation two.

```
run and jump -> x ijump
run and jump -> z ijump
run and jump -> x ijump
jump and run -> ijump x
run and jump -> z ijump
jump and run -> ijump z
run and jump -> z ijump
run and jump -> x ijump
jump and run -> ijump x
jump and run -> ijump z
run and jump -> z ijump
run and jump -> x ijump
run and jump -> x ijump
run and jump -> z ijump
run and jump -> z ijump
run and jump -> z ijump
jump and run -> ijump z
jump and run -> ijump x
jump and run -> ijump x
jump and run -> ijump x
run and jump ->
```

Figure 3: Example prompt used for experiment two variation two.

```
jump and jargon -> ijump x
jargon and jump -> x ijump
jargon and jump -> z ijump
jargon and jump -> x ijump
jump and jargon -> ijump x
jump and jargon -> ijump x
jargon and jump -> z ijump
jump and jargon -> ijump z
jump and jargon -> ijump z
jump and jargon -> ijump x
jargon and jump -> x ijump
jump and jargon -> ijump z
jargon and jump -> z ijump
jargon and jump -> z ijump
jump and jargon -> ijump x
jump and jargon -> ijump x
jargon and jump -> z ijump
jargon and jump -> x ijump
jump and jargon -> ijump z
jargon and jump -> z ijump

What is jargon?
A) x
B) z
Answer:
```

Figure 4: Example prompt used for experiment two variation three.

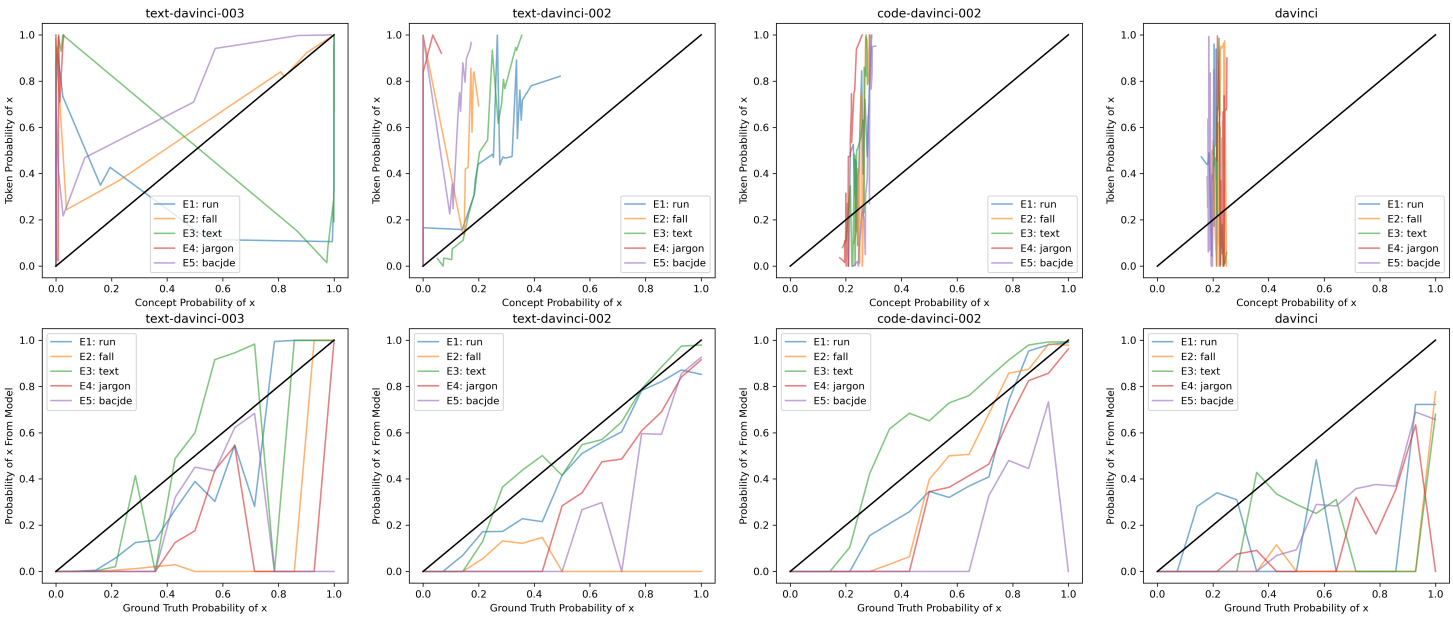

Figure 5: Results for experiment two variation three and variation four. First row shows results for variation three. Second row shows results for variation four.

3.2.5 Variation 3

In variation 3, we build upon variation 2. Instead of comparing GPT-3's probability of generating "x" or "z" when fed the test input *{word} and jump* -> against the ground truth probability, we compare it against the concept probability by directly asking GPT-3 what it thinks the stochastic word {word}. Thus, we run variation 2 again but also feed in another test input on top of the original one for each prompt. The test input would be the question with {word} replaced by the current word of interest:

```
What is {word}?
A) x
B) z
Answer:
```

See Figure 4 for an example.

3.2.6 Variation 3 Results and Discussion

Results for variation 3 can be found in the first row of Figure 5. Here, we see the results are not as we expected. Notably, for code-davinci-002 and davinci, the concept probability stayed within the range of 0.2 to 0.3. Similarly, for test-davinci-002, the probability range for the concept is also small. One possible reason is that the model did not understand the prompt well and could not properly indicate to us its concept probability. Subsequent experiments where the prompt examples had the same format as the concept test input question did not solve this issue though. text-davinci-003 exhibits wild behavior again.

```
fall and jump -> x ijump
fall right and fall right -> tright z tright z
walk and fall -> iwalk x
opposite left and fall -> tleft tleft z
fall right and fall left -> tright x tleft x
opposite right and fall -> tright tright z
fall after walk and jump -> iwalk ijump x
opposite left and fall after jump -> ijump tleft tleft z
jump and fall and walk -> ijump x iwalk
opposite right after fall -> z tright tright
opposite right and jump after fall -> z tright tright ijump
fall left and jump -> tleft x ijump
walk right and fall -> tright iwalk z
opposite right and fall and opposite left -> tright tright x tleft tleft
opposite left after jump and fall ->
```

Figure 6: Example prompt used for experiment two variation four.

3.2.7 Variation 4

In variation 4, the prompts contain the most diverse and complex examples. See Figure 6 for the form of the prompts. The same procedure from the other variations is repeated: we cover all five words, have an initial random ordering of "x" and "z" where the number of "x" and "z" is split 50/50, and incrementally replace "x" with "z" and "z" with "x" to create prompts. Our test input for all prompts is:

```
opposite left after
```

```
jump and {word} ->
```

where {word} is replaced with one of the five words.

3.2.8 Variation 4 Results and Discussion

Results for variation 4 can be found in the second row of Figure 5. Here, we see that text-davinci-002 and code-davinci-002 exhibit weak Bayesian behavior as their lines are mostly monotonically increasing but not as strong as before. As the number of "x" gets closer to the number of examples, the Bayesian behavior is stronger, especially in text-davinci-002. There is a lot of noise for all models which is to be expected because this is the most complex task of the four variations. davinci fails to show any comprehensible behavior because it may be too "dumb" to understand this complex task. text-davinci-003 also exhibits incomprehensible behavior.

3.3 Experiment 1 Results

From experiment 2, we can see that on simpler tasks, davinci, text-davinci-002, and code-davinci-002 has consistent, Bayesian behavior that makes sense to us. However, as the tasks get more complex, their behavior starts to break down and we cannot understand it well. text-davinci-003 consistently has different behaviors, most likely due to how it was pre-trained.

3.4 Experiment 2

In the last experiment, we create an artificial experiment with a "stochastic" prompt concept, in our case an unseen mathematical operator (e.g. ".") that is randomly interpreted as + or - in the prompt examples. We will be looking at the probabilities the model assigns to each of the answers (the one we would get with a + and with a -), and compare them with the "ground truth probability" that we defined in the in-context examples. E.g.: (first line is plus, second and third as minus)

```
3 . 4 = 7
5 . 3 = 2
7 . 4 = 3
6 . 3 =
```

Now, we will be looking at what are the probabilities assigned to 9 and 3, respectively. The ground truth probabilities are therefore $1/3$ for 9 and $2/3$ for 3

The second experiment is similar to this, but we are helping the model by providing a chain of thought, and looking at the probabilities of "+" and "-"

```
3 . 4 = 3 + 4 = 7
5 . 3 = 5 - 3 = 2
7 . 4 = 7 - 4 = 3
6 . 3 =
```

Notice that here the second version is very similar to "text replacement" of the previous experiments. The first version actually makes the model do the math, and therefore is much more complex.

In the actual experiment, we are using 30 in-context examples. Results are depicted on the plots 7. Each line is created using its own "starting" reordering.

3.4.1 Analysis

First of all, one should notice that due to recency bias, we don't expect the curves to be "diagonal". In partcular, the more vertical the curve is at some point, the more recency bias the model exhibits. This comes from the way we are creating our reorderings. In particular, we can see that for davinci, the curves are indeed more vertical, while they are more horizontal for the "later" models. (at least in the chain-of-thought experiments). This goes along with the fact that the later models are less prone to receny bias.

Focusing on the "CoT" experiments, which are more similar to the "text replacement" experiments of earlier, we can notice that davinci-3 also exhibits the "model collapse" behavior, where the probabilities are not monotonous, and sometimes seem to be quite random. This potentially comes from the fact that davinci-3 tries to find patterns in the data. It is interesting to note that this doesn't happen in the "no CoT" with davinci-3. This might be indicative of the fact that when we give davinici-3 a text-replacement tasks, it tries to find a pattern, which messes with probabilities. Now, when we ask it to do some "thinking" with an ambiguous concept, it doesn't exhibit this problem as much.

Focusing on the more interesting "no CoT" experiments, we can see how text-davinci-2 is actually the most constitent with its probabilities to the ground-truth. This indicates of some breakage of the bayesian inference that happened in davinci-3.

4 Conclusion

Overall, this project provided useful insight into how LLMs handle ambiguity. In some settings, it makes sense to humans, whereas in others, the

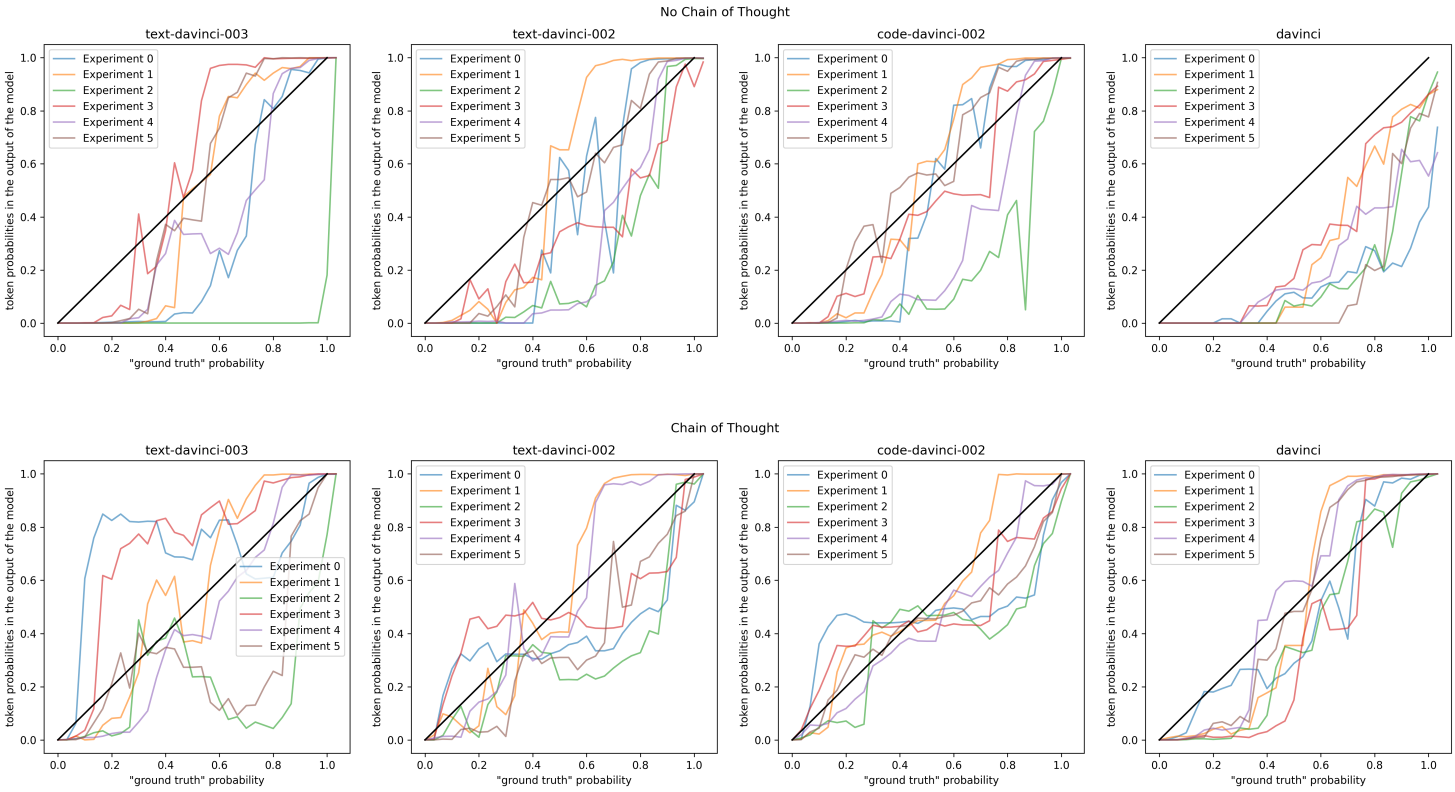

Figure 7: The results for an unknown operator ".". The x axis is the ground-truth probability as defined by the frequency of examples. The y axis is the output probabilities on corresponding token

behavior is widely unpredictable. It seems like the probabilities that the model outputs weakly agree with the ground-truth probabilities in our experiments. This means that the language modeling pretraining objective does an implicit probability calibration in the setting of ambiguous prompts which makes sense. We hope that our project encourages others to explore further ambiguity in LLMs and push the field forward towards general artificial intelligence.

References

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. 2022. Language models (mostly) know what they know.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work?

Nikunj Saunshi, Sadhika Malladi, and Sanjeev Arora. 2020. A mathematical exploration of why language models help solve downstream tasks.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2021. An explanation of in-context learning as implicit bayesian inference.