

Playing Dialogue-Based Games with Large Language Models

Master's Thesis of

Tobias Polly

Artificial Intelligence for Language Technologies (AI4LT) Lab
Institut für Anthropomatik und Robotik (IAR)
KIT Department of Informatics

Reviewer: Prof. Dr. Jan Niehues
Second reviewer: TT-Prof. Dr. Barbara Bruno
Advisor: M.Sc. Supriti Sinhamahapatra
Second advisor: M.Sc. Lukas Hilgert

15.06.2024 – 16.12.2024

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Karlsruhe, 16.12.2024

.....
(Tobias Polly)

Abstract

Large Language Models (LLMs) have proven capable of performing not only tasks like summarization, sentiment analysis or text completion, but also question answering, machine translation and, to some extent, game playing. This work explores whether LLMs are able to play language-based guessing games, mainly the Twenty Questions game, what parameters influence performance, and how performance can be improved by different prompt tuning and training modalities. “Twenty Questions” is a game where one player (the answerer) thinks of a concept and the other player (the questioner) has to guess it by asking yes-no questions. We measure two different scores: the game win rate, which measures both performance of the questioner and answerer, and the response F1 score of the answerer individually, which measures how accurately it can answer questions. Initial evaluation of different foundation models reveals modest success, with Llama-3-70B-Instruct achieving a win rate of 23%. Smaller models like Llama-3-70B-Instruct and Phi-3-mini-4k-Instruct only reach a win rate of 10%.

In more detailed experiments, we find that the untrained model’s performance is mainly influenced by prompt design and domain restriction of the possible concepts in the game. Our prompt tuning approaches reveal that the simplest prompt, a zero-shot prompt with binary choice questions, performs best. We investigate different training modalities to improve performance of Phi-3-mini-4k-Instruct and Llama-3-8B-Instruct. For the answerer, we find that supervised fine-tuning using a suitable dataset significantly improves F1 score of both models. We also find that models trained on multiple choice answers perform better on the binary choice task. The other way around, this is only true for Phi-3-mini-4k-Instruct, while Llama-3-8B-Instruct performs better on the binary choice task when trained on binary choice answers. For the questioner, for the lack of ground-truth data, we employ three different sources of generating training data. We find that this is a challenging task and the best source of training data is found to be dialogues generated by GPT-4o [20]. Still, in evaluation of the questioner with an untrained answerer, we only find slight increases in win rate. However, in joint evaluation of both a trained questioner and a trained answerer using single question prompt, we find that compared to our initial evaluation, Phi-3-mini-4k-Instruct improves its result by a factor of 3.6, reaching an average win rate of 24.3%, while Llama-3-8B-Instruct improves by a factor of around 2.1, reaching an average win rate of 27.3%.

Zusammenfassung

Die Fähigkeiten großer Sprachmodelle (Large Language Models, LLMs) übersteigen klassische Aufgaben wie Zusammenfassung, Sentiment-Analysen oder Textvervollständigung. LLMs können eingesetzt werden zur Beantwortung von Fragen in natürlicher Sprache, maschineller Übersetzung und, bis zu einem gewissen Grad, um Spiele zu spielen. Diese Arbeit untersucht, ob LLMs in der Lage sind, sprachbasierte Ratespiele, insbesondere das Spiel "Twenty Questions" zu spielen, welche Parameter das Spielergebnis beeinflussen und wie das Ergebnis durch verschiedene Ansätze wie Prompt-Tuning und Trainingsmodalitäten verbessert werden kann. Das Spiel "Twenty Questions" wird von zwei Personen gespielt. Der Antwortgeber denkt sich ein Konzept aus und der Fragesteller muss es durch Ja-Nein-Fragen erraten. Wir messen zwei verschiedene Metriken: die Gewinnrate im Spiel, die sowohl die Fähigkeit des Fragestellers als auch des Antwortgebers bewertet, und den F1-Score des Antwortgebers, die misst, wie genau er auf Fragen antwortet. Erste Evaluierungen verschiedener Sprachmodelle zeigen mäßigen Erfolg, wobei Llama-3-70B-Instruct eine Gewinnrate von 23% erreicht. Kleinere Modelle wie Llama-3-70B-Instruct und Phi-3-mini-4k-Instruct erreichen lediglich eine Gewinnrate von 10%.

In detaillierteren Experimenten stellen wir fest, dass die Gewinnrate des untrainierten Modells hauptsächlich durch die Gestaltung der Prompts und die Eingrenzung der Konzepte im Spiel beeinflusst wird. Unsere Untersuchungen zu Prompt-Tuning zeigen, dass der einfachste Prompt, ein Zero-Shot-Prompt mit binären Entscheidungsfragen, am besten abschneidet. Wir untersuchen verschiedene Trainingsmodalitäten, um die Gewinnrate von Phi-3-mini-4k-Instruct und Llama-3-8B-Instruct zu verbessern. Für den Antwortgeber zeigt sich, dass Supervised Fine-Tuning mit einem geeigneten Datensatz den F1-Score von beiden Modellen signifikant verbessert. Außerdem stellen wir fest, dass Modelle, die auf Multiple-Choice-Antworten trainiert wurden, bei binären Entscheidungen besser abschneiden. Umgekehrt gilt dies nur für Phi-3-mini-4k-Instruct, während Llama-3-8B-Instruct bei binären Entscheidungen besser abschneidet, wenn es speziell darauf trainiert wurde. Für den Fragesteller verwenden wir aufgrund fehlender Ground-Truth-Daten drei verschiedene Quellen zur Generierung von Trainingsdaten. Die besten Trainingsdaten stammen aus Dialogen, die von GPT-4o generiert wurden. Dennoch finden wir bei der Evaluierung des Fragestellers mit einem untrainierten Antwortgeber nur geringe Verbesserungen in der Gewinnrate. Allerdings zeigt die gemeinsame Evaluierung eines trainierten Fragestellers und eines trainierten Antwortgebers, der als Prompt einzelne Fragen erhält, dass im Vergleich zu unserer ersten Evaluierung das Ergebnis von Phi-3-mini-4k-Instruct um einen Faktor von 3,6 verbessert ist und eine durchschnittliche Gewinnrate von 24,3% erreicht wird, während Llama-3-8B-Instruct sich um einen Faktor von 2,1 verbessert und eine durchschnittliche Gewinnrate von 27,3% erzielt.

Acknowledgments

This work was performed on the computational resource bwUniCluster 2.0 funded by the Ministry of Science, Research and the Arts Baden-Württemberg and the Universities of the State of Baden-Württemberg, Germany, within the framework program bwHPC. Additionally, computational resources of the Institute for Anthropomatics and Robotics at the Karlsruhe Institute of Technology were used.

Contents

Abstract	i
Zusammenfassung	iii
Acknowledgments	v
1. Introduction	1
1.1. Motivation	1
1.2. Research Questions	1
1.3. Outline	2
2. Background and Related Work	3
2.1. Evolution of Large Language Models	3
2.1.1. Language modeling	3
2.1.2. Statistical language models	4
2.1.3. Neural language models	4
2.1.4. Pre-trained language models	4
2.1.5. Large Language Models	5
2.1.6. Optimizing model training and inference efficiency	7
2.1.7. Model families used in this work	8
2.2. Language-based guessing games	11
2.2.1. Game description	11
2.2.2. Non-LLM approaches to playing guessing games	12
2.2.3. LLMs playing guessing games	13
2.2.4. Guessing games in other applications	14
2.3. Datasets	15
2.3.1. Twenty Questions datasets	15
2.3.2. WordNet	18
2.3.3. Visual Genome	18
3. Methodology	19
3.1. Metrics	19
3.1.1. Game scores	19
3.1.2. Answerer accuracy	19
3.2. Evaluation setup	20
3.2.1. Evaluation using BIG-bench task	20
3.2.2. Evaluation using custom task	20
3.2.3. Evaluation of answerer accuracy	21

3.3.	Prompt tuning	22
3.4.	Model fine-tuning	22
3.4.1.	Supervised fine-tuning	23
3.4.2.	Reinforcement learning	24
3.4.3.	Joint evaluation	24
4.	Experiments and results	25
4.1.	Implementation	25
4.2.	Dataset preprocessing	25
4.3.	Hyperparameters	27
4.4.	Measuring task performance	27
4.4.1.	BIG-bench task	27
4.4.2.	Custom game task	28
4.4.3.	Common failure modes	29
4.4.4.	Analyzing domain restriction	30
4.4.5.	Measuring answerer performance	31
4.4.6.	Model selection	33
4.5.	Prompt tuning	33
4.5.1.	Multiple choice answer options	34
4.5.2.	Few-shot prompts	35
4.5.3.	Single question answerer prompt	36
4.5.4.	Adding visual input	37
4.6.	Fine-tuning the answerer	38
4.6.1.	Supervised fine-tuning of multiple choice task	38
4.6.2.	Supervised fine-tuning of binary choice task	39
4.6.3.	Evaluation on game	40
4.7.	Fine-tuning the questioner	43
4.7.1.	Supervised fine-tuning with taxonomy from WordNet	43
4.7.2.	Supervised fine-tuning with data from 20Q dialogues	46
4.7.3.	Supervised fine-tuning with model distillation	48
4.7.4.	Reinforcement learning	51
4.8.	Joint evaluation of trained questioner and answerer	51
5.	Conclusion	55
5.1.	Answering Research Questions	55
5.2.	Limitations and Future Work	56
	Bibliography	57
A.	Appendix	65
A.1.	Datasets	65
A.2.	Evaluation	66
A.3.	Experiments	70

List of Figures

2.1.	Transformer architecture	6
2.2.	Handheld 20Q device for playing Twenty Questions	11
2.3.	Similarity of user intent disambiguation, conversational search and Twenty Questions game	15
2.4.	Example data point from the AllenAI TQ dataset	16
2.5.	Overlap of the word lists for the Twenty Questions game	18
4.1.	Distribution of answer options in filtered AllenAI TQ dataset	26
4.2.	Question length distribution of filtered AllenAI TQ dataset	26
4.3.	Results of the answerer’s performance	32
4.4.	Usage of answer options in multiple choice task	35
4.5.	WordNet taxonomy graph	44
4.6.	Data collection setup for 20Q	46
4.7.	Reinforcement learning training run	51
A.1.	Distribution of answer options in unfiltered AllenAI TQ dataset	65
A.2.	Question length distribution of unfiltered AllenAI TQ dataset	65
A.3.	Answerer’s perspective for TQ game in original BIG-bench task	66
A.4.	Questioner’s perspective for TQ game in original BIG-bench task	67
A.5.	List of concepts in original TQ task	67
A.6.	Questioner’s perspective for TQ game with corrected prompt format	68
A.7.	Usage of yes/no/maybe answer options in multiple choice task	70
A.8.	System prompt from questioner’s perspective for few-shot approach	71
A.9.	Single question answerer prompt	72
A.10.	Results of the answerer’s performance	73
A.11.	Confusion matrices for binary answerer evaluation	74
A.12.	Confusion matrices for multi-choice answerer evaluation	75
A.13.	Training and evaluation loss for multi-choice answerer	75
A.14.	Training and evaluation loss for multi-choice answerer	76
A.15.	Prompt for filtering WordNet concepts for supervised fine-tuning	76

List of Tables

2.1.	Word lists for the Twenty Questions game	17
3.1.	Different sets of answer options	22
4.1.	Results of Twenty Questions task with original task format	28
4.2.	Results of Twenty Questions task with customized task	29
4.3.	Results of Twenty Questions task with domain-specific word lists	31
4.4.	Comparison of different multiple choice answer options	34
4.5.	Results of using few-shot prompt	36
4.6.	Results of single question answerer prompt	36
4.7.	Results of visual guessing game	37
4.8.	Evaluation of answerer at different stages of training	38
4.9.	Results of multiple choice fine-tuning of answerer	39
4.10.	Results of multiple choice fine-tuning of answerer	40
4.11.	Results of answerer evaluation in game	41
4.12.	Results of answerer evaluation in game	42
4.13.	Results of training with WordNet taxonomy	45
4.14.	Results of training with collected 20Q dialogues	47
4.15.	Results of fine-tuning Phi-3-mini-4k-Instruct using model distillation	49
4.16.	Results of fine-tuning Llama-3-8B-Instruct using model distillation	50
4.17.	Joint evaluation of questioner and answerer	52

1. Introduction

This chapter introduces the motivation for this thesis, the derived research questions, and its outline.

1.1. Motivation

In recent years, it has become apparent that Large Language Models (LLMs) are capable of solving an increasingly wide range of tasks such as translation, summarization, and question answering. LLM-based chat applications have become more sophisticated, holding conversations with humans and following instructions, yet some tasks, like playing language-based games, remain challenging for LLMs. One such game is Twenty Questions or, as a similar multimodal variant, I Spy. Such games have also been identified as possible benchmarks for evaluating LLMs [5, 46]. Guessing games are an interesting evaluation, as they require some form of knowledge, reasoning and strategy to effectively narrow down the possible answers.

These games are also interesting in the search for new applications of LLMs in less explored environments like nursery schools. Language-based games can teach children new words and concepts in a playful way. For usage in this setting, LLMs need to be able to run locally to be compliant with data protection regulations. Ideally, models should be small enough to run on a mobile device. We are interested in combining these two aspects: Can we use small, open-weight LLMs to play language-based guessing games, and can we use them in a way that could be beneficial for preschool children? If successful, this may enable new uses for LLMs in the field of language education.

1.2. Research Questions

From the motivation, we derive the following research questions to structure our work.

RQ1: Can language-based guessing games be played by foundation language models?

Foundation language models are not fine-tuned for a specific task, except for chat models that are fine-tuned to hold conversations with humans, follow instructions, and answer questions. We are interested in whether these models can play language-based guessing games. For this, we need to formalize the games in a way that is understandable for the models, find suitable prompts, and evaluate the models' performance.

RQ2: What factors are relevant for game performance? How can they be measured?

We are interested in the factors that influence the models' game performance in playing these games. For example, how does the size of the model influence performance? Do different prompt formats work better than others, like few-shot approaches or different sets of possible answer options? How does the multimodal version of the game compare to the text-only version? What is the influence of limiting the domain of the occurring words in the game?

RQ3: Can the model's playing performance be improved by fine-tuning?

Finally, we investigate whether the models' performance can be improved by fine-tuning. What are possible training setups for the questioner and answerer? What data sources can be used for training, including the generation of game dialogues? Does supervised fine-tuning improve the models' performance? Can we also train the model without training data using reinforcement learning?

1.3. Outline

The rest of this thesis is structured as follows: In chapter 2, we provide the necessary background information on LLMs, guessing games, and the datasets used in this work. Also, we present relevant work related to our research questions. In chapter 3, we describe the methodology used to answer the research questions, including the models, training setups, and evaluation metrics. In chapter 4, we present the experiments conducted to answer the research questions, as well as the results. Finally, in chapter 5, we conclude the thesis by answering the research questions, discussing limitations and suggestions for future work.

2. Background and Related Work

In this chapter, we provide the necessary background information to understand the rest of this thesis. First, we give an overview of the development of LLMs and the Transformer architecture. Two Transformer-based models used in this work are Llama and Phi, which we introduce in more detail. We present different training modalities for LLMs, including supervised fine-tuning and reinforcement learning. We also present techniques to improve computation efficiency for training and inference of models. Second, we describe the guessing games that we are working with, Twenty Questions and I Spy, and approaches to how they have been played computationally. Finally, we introduce the datasets that we use in this work.

2.1. Evolution of Large Language Models

Large Language Models have come to dominate the field of natural language processing (NLP) in recent years [56]. In the following, we will give a brief overview of what language modeling is, what LLMs are, how they have evolved and how they can be fine-tuned efficiently.

2.1.1. Language modeling

Language modeling can be described as the task of predicting the probability of a token, given a sequence of tokens [56]. A token can be a word or a subword, like a character or a part of a word [44]. The predicted probability can be calculated for any token in a sequence, like a missing token or the next token in a sequence. The probability of a sequence of tokens t_1, t_2, \dots, t_n can be decomposed as the product of the conditional probabilities of each token given the previous tokens [3]:

$$P(t_1, t_2, \dots, t_n) = P(t_1) \cdot P(t_2|t_1) \cdot \dots \cdot P(t_n|t_1, \dots, t_{n-1}) = \prod_{i=1}^n P(t_i|t_1, \dots, t_{i-1}) \quad (2.1)$$

The probability distribution of the next token in a sequence can then be sampled, in the case of language generation. While language generation modeled as next token prediction seems like a simple approach, it is general enough to be applied to any task that can be specified via text, with surprising results even for complex tasks [46]. The development of language models is described in four major stages by Zhao et al. [56]. In the next sections, we will describe these stages and the major developments of each stage.

2.1.2. Statistical language models

The first stage is based on *statistical learning methods* built on the Markov assumption, which states that the probability of a word only depends on its close previous context [56]. At this stage, models are developed for specific tasks like information retrieval or sentence classification. Simple statistical language models use a fixed context length n of tokens or words to predict the next, which is known as n -gram models [27]. A bigram (2-gram) model can be described as a table of probabilities of pairs of words. However, as this context length n increases, the number of possible sequences grows exponentially and for large n , no such table can be feasibly constructed. Furthermore, such a model cannot be efficiently trained, as most sequences cannot be observed in the training data.

2.1.3. Neural language models

The second stage is marked by the advent of *neural language models* [56]. These models use sequence-to-sequence neural networks to learn the probability distribution of sequences of tokens, like Recurrent Neural Networks (RNNs). The use of recurrence allows for the modeling of longer context lengths. Neural networks can be layered to create deeper networks, which makes models able to learn more complex patterns in the data. A major contribution is the introduction of word embeddings as continuous representations of words, like word2vec [30]. This approach allows to move from one-hot vectors (indices in a vocabulary), which are sparse, high-dimensional and do not capture semantic information, to dense, lower-dimensional vectors. Word embeddings are learned from large text corpora. In the case of word2vec, this is done by training a neural network to either predict the context words from a target word (skip-gram) or to predict the target word from the context words (continuous bag of words). Taking the learned input vectors of the neural network as the word embedding, this leads to a representation of words in a continuous space where words that are semantically related are closer to each other [30].

2.1.4. Pre-trained language models

The third stage has been characterized by the development of *pre-trained language models* [56]. Pre-trained language models are trained on large amounts of text data and can be fine-tuned for specific tasks. One example is the BERT model [11], which is trained on predicting randomly masked words in a sentence. The model is then fine-tuned with one additional output layer for specific tasks. At the time of its introduction, BERT outperformed previous models on eleven NLP tasks [11]. The method of pre-training has been shown to be a very effective way of applying language models to a wide range of tasks, as pre-training is the most expensive part of the training process. BERT is also an example of a model employing the Transformer architecture, which is another major development in this stage.

Transformer architecture The Transformer architecture [50] is a neural network architecture for sequence-to-sequence tasks that originated from the field of machine translation. Previous sequence-to-sequence models used Recurrent Neural Networks (RNNs) and their

gated variants like long short term memory (LSTM) [16] cells or gated recurrent units (GRU) [7]. For independent length of input and output, these models feature an encoder-decoder architecture. Between these two parts, an attention mechanism is used to improve the information flow between the encoder and decoder and allowing the decoder to focus on different parts of the input sequence. The attention mechanism is a way to weigh the importance of different parts of the input sequence for each part of the output sequence. This is done by calculating a learned key, query, and value vector for each token in the input sequence (this includes the previously generated tokens for an autoregressive model). In the original work introducing the Transformer, the attention matrix is calculated for each mapping of query and key-value pairs as a scaled dot product [50]:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (2.2)$$

where Q , K , and V are the query, key, and value vectors, respectively, and d_k is the dimension of the key vectors. This attention calculation is done multiple times in parallel, with different learned weights for the query, key, and value vectors, which are then concatenated and linearly transformed. This is called multi-head attention and is shown in Figure 2.1 on the right. In contrast to previous recurrent networks, the Transformer not only uses attention between the encoder and decoder but also within the encoder and decoder in the form of self-attention and in turn does not use recurrent connections. Without recurrence, there is no information of the order of the tokens in the sequence, which is why positional encoding is added to the input embeddings. The full architecture is shown in Figure 2.1 on the left. This architecture is not only simpler, more parallelizable and faster to train in comparison to recurrent networks, but also outperforms previous models on many tasks. Furthermore, the Transformer architecture alleviates the vanishing/exploding gradient problem that occurs with the training of recurrent networks [35], as the attention mechanism allows for direct connections between any two tokens in the sequence. The LLMs that we use are decoder-only Transformers, which means that they only use the decoder part of the Transformer architecture and thus, the attention mechanism can only access the input and the words that have been generated so far.

2.1.5. Large Language Models

The fourth stage according to Zhao et al. [56], *Large Language Models*, is the result of scaling the model size and training data. This scaling has produced models that are exceeding the performance of smaller models by a large margin and are displaying new abilities. One of the first examples of such a scaled Transformer model is GPT-3 [4], a 175 billion parameter model that has been trained on 300 billion tokens of text data. The model reaches strong performance in tasks that it was not directly fine-tuned for in a few-shot setting. This means that for many tasks, the model can be used without any additional training data, only by providing a few examples of the task. Multiple approaches have been developed to move from few-shot to zero-shot performance, where the model is able to perform a task without any examples, only by providing a description of the task. Furthermore, methods have been proposed to align models to human preferences, which we present in the following.

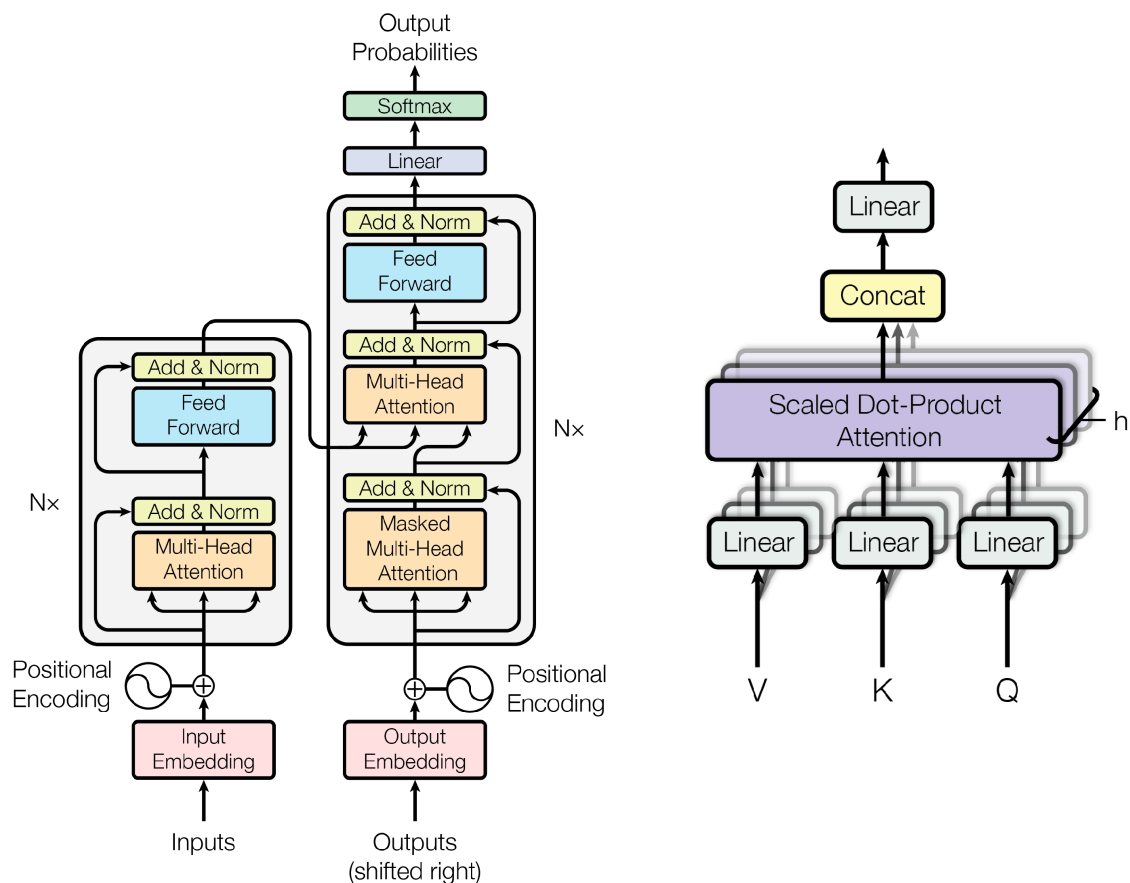


Figure 2.1.: Left: The Transformer architecture [50] featuring an encoder-decoder architecture with self-attention and cross-attention mechanism. Positional encoding is concatenated to the input embeddings. Right: Multi-head attention mechanism in the Transformer consisting of concatenated scaled dot product matrices. Image source for both images: Vaswani et al. [50].

Instruction tuning Instruction tuning has been introduced by Wei et al. [51]. They train a 137 billion parameter pre-trained model on a collection of tasks phrased as instructions from 62 publicly available datasets. The collection includes both language understanding tasks, like reading comprehension, and language generation tasks, like summarization and translation. For each dataset, ten different instruction templates are manually created. The fine-tuned model, which they named FLAN, reaches much higher performance in unseen tasks than larger non-tuned models like GPT-3 [4], even when comparing FLAN zero-shot to GPT-3 few-shot.

Visual instruction tuning for multimodal models Visual instruction is the extension of instruction tuning to multimodal tasks. Liu et al. [26] train a model called LLaVA (Large Language and Vision Alignment), which is a combination of the visual encoder CLIP [36], that is developed by jointly training an image and text encoder, and the Llama model. This model is trained on custom language-image instruction-following data generated by GPT-4[32]. They find that the model reaches up to 90% accuracy on the science QA dataset (a multimodal multiple choice question answering dataset).

Alignment using reinforcement learning Another approach to fine-tune models is targeted to reduce model outputs that are untruthful, unhelpful or in other ways undesirable. Ouyang et al. [33] summarize this as the problem of aligning the model to human preferences. They first collect human demonstrations of desired behavior to perform supervised fine-tuning (equivalent to instruction tuning). Second, they collect comparison data to train a reward model (which is a fine-tuned GPT-3 [4] model) that can predict which of two model outputs is preferred by a human. Finally, they use reinforcement learning, namely proximal policy optimization [43], using the reward model. This second and third stage is called reinforcement learning with human feedback (RLHF). They find that the aligned model is strongly preferred by humans over the unaligned model, is more likely to follow instructions and is less likely to hallucinate. This is true even if the aligned model is much smaller than the unaligned model.

2.1.6. Optimizing model training and inference efficiency

Handling multi-billion parameter models would be unfeasible for a thesis like this without performance enhancement techniques. In this section, we mention performance enhancement techniques for training and inference of LLMs. We present FlashAttention, which speeds up attention calculation in both training and inference. For training, we use Low-Rank Adaptation, a technique that allows training a model with fewer parameters than the original model but still reaching comparable performance.

FlashAttention FlashAttention is an implementation of the attention mechanism used Transformer models by Dao et al. [10] that uses the GPU memory hierarchy more efficiently. This is achieved by restructuring the attention computation into blocks, processed sequentially (a process known as tiling), reducing the number of reads and writes between operations, as multiple operations are calculated per block without the need to store and

load intermediate attention matrices. Also, for the backward pass, instead of storing and loading intermediate attention matrices, only the softmax normalization factor is stored and attention recomputed, which reduces memory usage from quadratic to linear usage in size of the input sequence. They find that this can lead to a speedup of about 15% for training BERT-large [11]. Further optimization in FlashAttention 2 increase the speedup to up to another 2x [9]. FlashAttention requires the use of Nvidia Ampere, Ada, or Hopper GPUs (e.g. A100, H100 or RTX 6000 Ada).

Low-Rank Adaptation As full retraining of all parameters of a large models for fine-tuning is expensive, Low-Rank Adaptation (LoRA) has been proposed as a technique to fine-tune a model with much fewer parameters than the original model but still reaching comparable performance [18]. This is done by adding adapters to linear layers of the model in the form of rank decomposition matrices. Given a pre-trained weight matrix $W_0 \in \mathbb{R}^{d \times k}$, a weight matrix $\Delta W = BA$ is added, where $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$. The rank r is a hyperparameter that determines the number of trained parameters and is usually chosen to be much smaller than the original dimensions d and k , e.g. $r = 16$. Additionally, ΔW is scaled by $\frac{\alpha}{r}$ using another hyperparameter α , which has a similar effect as the learning rate. Usually, α is chosen to be equal to r and only used as a scaling factor when varying r , so that other hyperparameters don't need to be re-tuned [18]. For example, Llama 3-8B Instruct contains 8,072,204,288 parameters. When adding adapters to all linear layers except for the language modeling head, with a rank of $r = 16$ and $\alpha = 16$, the model contains 41,943,040 parameters, which is only 0.52% of the original model size.

2.1.7. Model families used in this work

In this section, we introduce two model families that we are working with more closely in this thesis, the Llama model family and the Phi model family. Both are models of the fourth stage of language model development.

2.1.7.1. Llama model family

First, we describe the Llama model family, which is a series of open-weight models developed by Meta¹.

Original LLaMA model The Llama model family's original LLaMA model by Touvron et al. [49] is a foundation model developed to reach higher model performance by training with more tokens than what has usually been performed due to proposed scaling laws [17], which are formulated for finding the best model given a particular training budget. However, in search for the best performance given an inference budget, which is a more relevant target for serving LLMs at scale, Touvron et al. [49] find that model performance of a 7B parameter model continues to improve even after 1T tokens. They train different sizes of models, from 1B to 70B parameters, on 1.0 to 1.4T tokens. Only publicly available data was used in training. This means that, together with open source code for inference, the model

¹<https://llama.com/> (last accessed on 12/09/2024)

architecture, weights and training data are available to the public. LLaMA uses a modified Transformer architecture, in particular employing pre-normalization (normalizing the input of Transformer sub-layers instead of the output for stability), SwiGLU activation function [45] and rotary positional embeddings (RoPE) [47]. For tokenization, byte pair encoding (BPE) is used (first proposed for tokenization by Sennrich et al. [44]). LLaMA-13B has been found to outperform GPT-3 [4] on most benchmarks. Fine-tuning LLaMA on specific tasks has been shown to further improve performance, for example on the MMLU (Massive Multitask Language Understanding) benchmark [15, 49].

Llama 2 The second iteration of Llama models, Llama 2, is again a publicly available model, additionally released as fine-tuned variant for conversational tasks [48]. The model is very similar to the original LLaMA model, the key architectural differences being the increase of the context length to 4k tokens, which is double the size of the original model (2k tokens) and the use of grouped-query attention (GQA) [2] (for the 34B and 70B model only). Training data size for the model is 40% larger than for the original LLaMA model, a mix of publicly available data and proprietary annotations. This data has been filtered to remove personal information. After pre-training, the model is trained using supervised fine-tuning (again, using public data), as well as further fine-tuned using reinforcement learning with human feedback (RLHF), in particular with proximal policy optimization (PPO) [43] and rejection sampling. At the time of release, Llama 2 is leading in human evaluation of helpfulness in comparison to Vicuna models (which are LLaMA models fine-tuned on ChatGPT dialogues [57]) of similar number of parameters.

Llama 3 The third iteration of the Llama model family, Llama 3 [13], is a further development of the Llama 2 model that was released parallel to the work on this thesis. The model is again publicly available and comes in different sizes, for the first time also in a 405B parameter version. Still, like the original LLaMA model, Llama 3 uses SwiGLU activation functions and RoPE (with a different base frequency hyperparameter for better performance with long context). Again, the size and quality of training data has been drastically improved and scaled, with a corpus of 15T tokens (which is an eight-fold increase) and more preprocessing and filtering steps, that include using LLaMA 2 as quality classifier, than the previous models. Llama 3 again delivers (at the time of release) leading performance on many benchmarks and tasks. While this work was ongoing, Llama 3.1² and Llama 3.2³ have been released, introducing additional functionality, in particular multilinguality, multimodality, longer context and, for the instruct-tuned models, tool use.

2.1.7.2. Phi model family

Phi is another open-weight model family of smaller models developed by researchers at Microsoft [14].

²<https://ai.meta.com/blog/meta-llama-3-1/> (last accessed on 12/09/2024)

³<https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/> (last accessed on 12/09/2024)

Phi-1 The first Phi-1 model is a 1.3B Transformer model (using RoPE and FlashAttention) for Python coding, trained on less than 7B tokens. The idea is to deliberately use less training data, but to ensure higher quality of the data. Their training data comes from three sources: Code-language data from The Stack [23] and StackOverflow, filtered by a random forest model trained on annotations by GPT-4 [32] (around 6B tokens), synthetic textbook data generated by GPT-3.5 (less than 1B tokens) and synthetic exercises, also generated by GPT-3.5 (around 180M tokens). They find that phi-1, despite its size, outperforms almost all open-source models on HumanEval and other coding benchmarks in Python coding.

Phi-1.5 As a follow-up, they published Phi-1.5, a model with the same architecture and size of phi-1 on 30B tokens [25]. In addition to phi-1’s training data, they created a new curated dataset of 20B tokens of synthetic textbook data geared towards common sense reasoning in natural language. To test the effect of filtering and curated data, they also trained phi-1.5-web-only for comparison on 95B of filtered web data (no synthetic data) and phi-1.5-web model on a mix of curated and web data. They find that the web-only model performs the worst of the three models, but still outperforms models of similar size in common sense reasoning benchmarks like WinoGrande, ARC-Easy and others. The best performance is achieved by phi-1.5-web, which is only slightly better than phi-1.5.

Phi-2 The next model in the family is Phi-2, a 2.7B parameter model trained on 1.4T tokens of similarly filtered and generated data (no technical report available, only mentioned in the Phi-3 technical report [1] and Microsoft’s blog⁴).

Phi-3 Finally, Phi-3-mini is a 3.8B parameter language model trained on 3.3T tokens [1]. It uses the same block structure and tokenizer as Llama 2. There are also 7B small and 14B medium versions that use a modified architecture (tiktoken tokenizer⁵, GEGLU activation function [45], GQA [2], and a different block structure employing blocksparse attention) and are trained on more data (4.8T tokens). Again, training data is a combination of heavily filtered publicly available data and synthetic LLM-generated data. Their goal is again, to optimize training data for the size of model. They use a two-phase training approach where the first phase teaches general knowledge and language understanding and the second phase teaching logical reasoning and niche skills through more strongly filtered web data. Post-training is done to turn the model into a chat model using another two stages: first, supervised fine-tuning using a curated dataset and second, direct preference optimization (DPO) [37] to align the model to chat format, elicit reasoning and reduce unwanted behavior. Phi-3-mini is found to perform similarly or better in comparison to Mistral-7B [21], Gemma-7B [29] and Llama3-8B-Instruct [13] on a range of benchmarks like MMLU [15], HellaSwag [53], and others. Another model in the third version of the family is Phi-3-vision, a 4.2B parameter model based on Phi-3-mini. Phi-3-vision uses the CLIP [36] ViT-L/14 image encoder and Phi-3-mini as the text encoder. The model is trained on 0.5T tokens of a mix of image-text documents, image-text pairs and synthetic

⁴<https://www.microsoft.com/en-us/research/blog/phi-2-the-surprising-power-of-small-language-models/> (last accessed on 12/09/2024)

⁵<https://github.com/openai/tiktoken> (last accessed on 12/09/2024)

data. Similar two-stage post-training as Phi-3-mini is done with supervised fine-tuning using custom multimodal instruct-tuning datasets (15B tokens), fine-tuned using another custom dataset to align the model to act as a chat model. Phi-3-vision is found to perform better than LLaVA-1.6 [26] in MMMU [52], ScienceQA [28] and other benchmarks.

2.2. Language-based guessing games

This work is centered around language-based guessing games, i.e. games that require no physical components and can be played with speech or text only, where the task is to guess a concept. Two of the most popular games in this category that we are referencing are Twenty Questions and I Spy, which we will describe in the following sections.

2.2.1. Game description

Guessing games are games where one player has to guess a concept that the other player has picked. In this work, we mainly focus on the Twenty Questions game, but also mention I Spy. Both games are described in the following.

Twenty Questions Twenty Questions is a spoken guessing game between two players. One of the players, which we call the answerer, (or “Alice” in case of the Twenty Questions task inside of the BIG-bench benchmarking suite, see section 2.2.3), picks a concept. The other player, which we call the questioner (or “Bob”), asks questions that can only be answered with yes or no. Only based on the answers to these questions, the questioner has to guess the concept within 20 questions at most, which is the origin of the game’s name. It is not entirely clear where or when the game originated from, but it has been played in various forms since the 19th century [19]. The game has become popular through a TV show with the same name in the 1950s⁶ and has been put into consumer toy form as seen in Figure 2.2. Variants of the game mainly differ in the number of possible answer options from the answerer. In human play, this may not be restricted. The task from the BIG-bench benchmarking suite only allows binary yes/no answers [46] while other instances allow an “unknown” option [19], or multiple answers (e.g. in some of the datasets, see subsection 2.3.1).



Figure 2.2.: Handheld 20Q device for playing Twenty Questions by Radica (public domain⁷).

⁶<https://www.imdb.com/title/tt0320997/> (last accessed on 12/09/2024)

⁷[https://commons.wikimedia.org/wiki/File:20Q_red_\(Radica\)_front.png](https://commons.wikimedia.org/wiki/File:20Q_red_(Radica)_front.png) (last accessed on 12/09/2024)

I Spy I Spy is a similar guessing game that is popular among children. The key difference is that the game is played with a visual component. Again, there are two participants, the first being the answerer to pick an object or scenery item that both participants can see in their field of vision. In the most widespread version, the answerer announces the first letter of the object (“I spy with my little eye something beginning with...”). The questioner then has to guess what the answerer has picked. Guesses can be simply listing possible visible items or be categorical questions that would be used in a Twenty Questions game or include questions about shape or spatial relations (“is it next to a tree?”). Usually, the number of guesses is not limited. In other versions of the game, the color of the object is specified instead of the first letter. The game is one of the suggested activities by educational research organization HighScope for preschool children to improve critical thinking skills as well as the identification of colors and shapes⁸.

2.2.2. Non-LLM approaches to playing guessing games

Guessing games are simple in their rules, but require a certain amount of reasoning and deduction to be played effectively. This is why they are interesting to implement computationally and why they have occurred repeatedly in AI research. In this section, we will present non-LLM approaches to playing these games computationally. These approaches are ways to play the game from the role of the questioner. Playing the questioner does not necessarily require language understanding or generation, but can be done by selecting the best possible next question from a set of possible questions or by using a tree-based approach. In fact, for just playing the game itself, these approaches can be more successful in game performance than using Large Language Models. We present some of these approaches in the following.

Burgener’s 20Q algorithm An early popular approach for playing the questioner role in Twenty Questions was built by Burgener in the form of a neural network decision algorithm in 2005⁹. His algorithm is also what powers the 20Q series of consumer toys. The online variant at <http://www.20q.net/> (last accessed on 12/09/2024) actively learns from user input and has been trained on millions of games. If the algorithm cannot guess the object, it asks the user for the object and for a question that distinguishes the object from the guessed object. Burgener’s algorithm uses a matrix of weights to represent relevance between potential objects and questions. When the player responds, the algorithm adjusts the weights between nodes, ranking possible objects and questions. The complete algorithm is not public (target rank calculation is not known), but there are public reimplementations¹⁰. The web version has different themed games for categories like TV series, movies, people or sports. There are also unconstrained versions that only differ in the language of the game.

⁸https://highscope.org/wp-content/uploads/2024/04/Preschool_Lets-Play-and-Learn_I-Spy.pdf (last accessed on 12/09/2024)

⁹U.S. Patent Application US20060230008A1, <https://patents.google.com/patent/US20060230008A1> (last accessed on 12/09/2024)

¹⁰<https://github.com/chrispen/asklet> (last accessed on 12/09/2024)

Reinforcement learning approach In 2018, Hu et al. [19] presented a policy-based reinforcement learning approach to play the game. They collected a dataset of 1,000 well-known people and 500 questions answered for each person. In the dataset, noise is added to the answers to simulate human error. They formulate the game as a finite Markov decision process and use a simple neural network that estimates reward for each time step as reward signal. The policy network is trained using REINFORCE. Within their constrained domain of 500 questions about well-known people, they achieve a win rate of over 90%.

Binary search in dictionary Parallel to this work, a Kaggle competition was hosted centered around playing Twenty Questions with small LLMs (1 Tesla T4 GPU was given as compute resource)¹¹. Kaggle is a platform for data science competitions and has been used for various AI competitions in the past. The way the contest was set up, entrants found out that it was possible to play the game without any LLM at all, but instead by using large dictionaries of nouns and searching through them using binary search¹². If the answerer and questioner both have a dictionary that is big enough to cover all possible words, they can simply ask whether a guessed word appears lexicographically before or after a certain word in the dictionary. The competition is a good example of how the game can be played computationally without language understanding.

2.2.3. LLMs playing guessing games

In the following, we present work related to playing guessing games with LLMs. As LLMs are able to directly consume and generate text, they can directly be used to play both the role of the answerer and the questioner in guessing games.

LLMs in the role of the answerer Bruyn et al. [5] used the Twenty Questions game to analyze world knowledge of LLMs. They collected a dataset of 2,832 questions from the Akinator game¹³, which is a web-based guessing game that is similar to Twenty Questions [5]. In their setup, the Akinator game is supplying questions, humans are in the role of the answerers. The dataset contains questions about objects only (Akinator only supports characters, objects or animals), which are labeled with a binary answer (yes/no). In total, the questions are answered for 126 different objects. The dataset does not contain full dialogue, only single questions and answers. At the time of this writing, the dataset is not publicly available although stated in the paper (paper links to empty Huggingface dataset¹⁴, personal communication with the authors ongoing). They evaluated the performance of different LLMs (T5 11B [38], T0pp 11B [41], GPT-3 175B [4], among others) in the role of the answerer on their dataset and found that most LLMs are not able to play the game effectively, with the exception of GPT-3, which reached an F1 score of 83% in a few-shot evaluation [5]. Augmenting the prompt with additional information from Wikipedia articles improves the performance of T0pp by 10%. Adding content from

¹¹<https://www.kaggle.com/competitions/llm-20-questions/overview> (last accessed on 12/09/2024)

¹²<https://www.kaggle.com/competitions/llm-20-questions/discussion/511343#2866948> (last accessed on 12/09/2024)

¹³<https://akinator.com/> (last accessed on 12/09/2024)

¹⁴<https://huggingface.co/datasets/maximedb/twenty> (last accessed on 12/09/2024)

Wikipedia to the prompt is also successfully tested in a course project by Parikh et al. [34]. They additionally tested training a multilayer perceptron classifier and an LLM, in this case DistilBERT [40], on a dataset of questions and answers, namely BoolQ [8] with inconclusive results for the lack of ground truth data.

LLMs in the role of the questioner LLMs have also been used to play the role of the questioner in Twenty Questions as well, mostly to evaluate their capabilities. For example, BIG-bench [46] contains a Twenty Questions task¹⁵. BIG-bench (Beyond the Imitation Game benchmark) is a benchmark suite for evaluating Large Language Models on a wide range of tasks, aimed at better understanding and characterizing the capabilities of these models [46]. The suite consists of 204 tasks at its publication date, with topics ranging from linguistics, math, sciences, social bias, coding, and more. Tasks are deliberately chosen to be outside of the current (at the time of publication) capabilities of Large Language Models. The general finding of the benchmark is that while performance generally scales with model size, there seem to be tasks that improve gradually with scale, usually tasks that require world knowledge, and tasks that improve sharply with scale, usually tasks that require reasoning. Guessing games require both of these capabilities. Tasks are benchmarked using GPT-3 [4] and BIG-G [46], in sizes ranging from 2M parameters to 137B parameters. Both benchmarked models attain pretty much the lowest possible score, with only incidentally won games.

When trying to improve performance of the questioner in the Twenty Questions task, there are two main approaches: supervised fine-tuning and reinforcement learning. In the previously mentioned Kaggle competition, many of the best scoring LLM-based entries were fine-tuned on model-generated dialogue. There are also hybrid approaches that begin the game with a previously optimized fixed decision tree and then switch to a language model after a certain number of questions¹⁶. Training on model generated dialogue is also shown to be effective in playing Twenty Questions as the questioner by Zhang et al. [55], who showed that Vicuna-7B [57] shows more than 70% improvement when trained on dialogues generated by GPT-4 [32], reaching a success rate of 23% (when trained on only successful dialogues). In their evaluations, they used GPT-3.5-turbo as answerer. Reinforcement learning from game play is also performed by Zhang et al. [55], who use proximal policy optimization to train a questioner model with a reward function derived from the game’s score. They find that if the previously fine-tuned Vicuna-7B with 23% success rate is trained in this way, it can reach success rates of 26%. Additionally, they collected a 145 human game play sessions for a subset of their “Things”-dataset, finding a win rate of 24%.

2.2.4. Guessing games in other applications

Playing guessing games computationally is not only interesting for the sake of playing the game itself, but also for other applications. Other than benchmarking, we briefly mention

¹⁵https://github.com/google/BIG-bench/tree/main/bigbench/benchmark_tasks/twenty_questions (last accessed on 12/09/2024)

¹⁶<https://www.kaggle.com/competitions/llm-20-questions/discussion/532018> (last accessed on 12/09/2024)

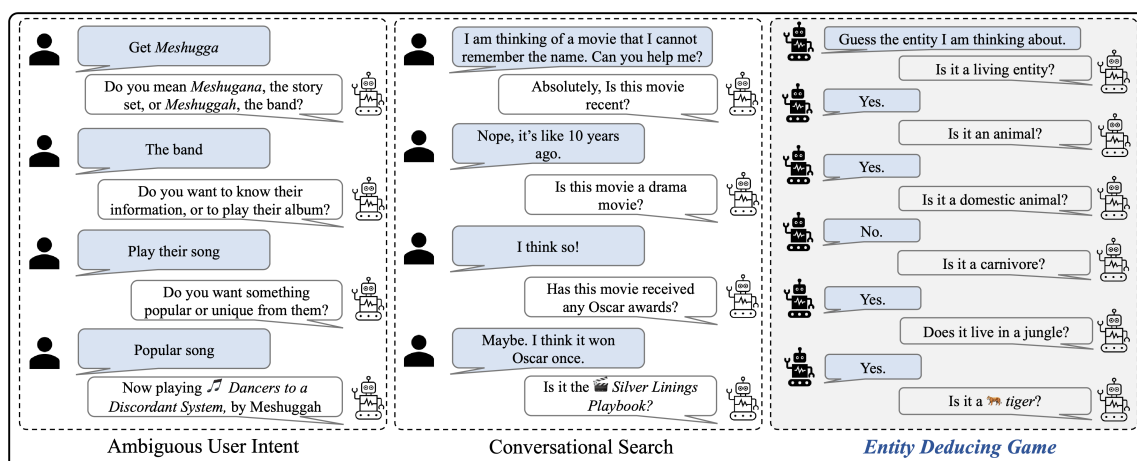


Figure 2.3.: Similarity of three different tasks: user intent disambiguation, conversational search and Twenty Questions game by Zhang et al. [55].

examples of how guessing games have been used in other applications: for knowledge acquisition and disambiguation of user intent in dialogue systems.

Guessing games for knowledge acquisition In an effort to build a robust and accurate knowledge base to power applications like question answering systems, Twenty Questions has been used to gather knowledge by Chen et al. [6]. Their approach uses a reinforcement learning agent to play Twenty Questions with a human player to gather knowledge, similar to the game by Burgener (see subsection 2.2.2). It is essentially a two-step process, of which the first is the identification of the target entity. After the target entity is identified, the agent uses the remaining questions to gather new knowledge about the target entity.

Disambiguation of user intent in dialogue systems Disambiguation of user intent in dialogue systems is a similar task to playing a guessing game where the system has to ask questions to narrow down the user's intent as efficiently as possible. Example dialogue by Zhang et al. [55] displaying the parallels is shown in Figure 2.3.

2.3. Datasets

In this section, we introduce the datasets that we use in this work. We introduce several datasets for the Twenty Questions game, WordNet and Visual Genome.

2.3.1. Twenty Questions datasets

In this section, we introduce the datasets that we use for the Twenty Questions game.

Collected questions and answers from human-played games Researchers from AllenAI collected a dataset¹⁷ using a web application for playing Twenty Questions on Mechanical Turk, a service for crowdsourcing tasks like labeling data or answering questions by human annotators¹⁸. The dataset was published in 2020¹⁹. It contains 78,890 individual questions and answers (not full game dialogue) split into train, test and development sets. Each data point has a subject, a question, an answer from the person who played the game. The dataset allows for six different possible answers to a question: always, usually, sometimes, rarely, never or irrelevant. Some questions that were obtained outside of Twenty Questions games were also included, which don't include an answer. However, each data point was additionally labelled by three different annotators that each added an answer and a quality label. For the three labelers, the dataset contains a majority vote of the labelers' answers as a simple binary yes/no answer. There is also an aggregated quality score as the sum of all labelers that assessed the question as high quality. An example data point is shown in Figure 2.4. We refer to this dataset in this work as the AllenAI TQ dataset.

```
{
  "subject": "pretzel",
  "question": "Is this food cooked?",
  "answer": "always",
  "quality_labels": ["good", "good", "good"],
  "score": 3,
  "high_quality": true,
  "labels": ["always", "always", "always"],
  "is_bad": false,
  "true_votes": 3,
  "majority": true,
  "subject_split_index": 1,
  "question_split_index": 1
}
```

Figure 2.4.: Example data point from the AllenAI TQ dataset where all labelers agree on label and quality

Word lists for Guessing games We collect three word lists from different sources to use in the Twenty Questions game for evaluation and training.

As first list, we take the concepts from the AllenAI TQ dataset's test set. There are multiple questions per concepts, so we sort by the frequency of occurrence and sort by the most frequent concepts. This list is referred to as "AllenAI TQ test concepts" in the following. We created the analogous list for the train set of the AllenAI TQ dataset, which

¹⁷<https://github.com/allenai/twentyquestions> (last accessed on 12/09/2024)

¹⁸<https://www.mturk.com/> (last accessed on 12/09/2024)

¹⁹<https://github.com/allenai/twentyquestions/commit/48f0de26d2dca963cdae1263245d3b267ae7a771> (last accessed on 12/09/2024)

is referred to as “AllenAI TQ train concepts” in the following. The AllenAI TQ concepts are unconstrained, meaning that they can be any concept, not just things.

The second list is what Zhang et al. [55] used in their work for evaluation of LLMs in playing Twenty Questions. Their list²⁰ contains 500 entities of commonly found things such as common objects, animals, foods, plants, vehicles etc., split into 300 for training, 100 for testing and 100 for validation. The test list of this set is referred to as “Things concepts” in the following.

As we are interested in playing the game with kids, we are also interested in words that are within the vocabulary of a preschool child. For this, we use nouns from the Dolch word list, which is a list of the most frequently used words in children’s books [12]. We apply minor modifications to the list of 95 nouns: We remove words that are simply the plural form of another word, as the game is about guessing a concept, not a word. To create a list of 100 words, we add five nouns from the Fry word list (which follows a similar approach²¹). Five exemplary concepts from the list are “apple”, “box”, “Christmas”, “eye” and “garden”. This list is referred to as “Dolch concepts” in the following.

The three lists are summarized in Table 2.1. We also analyze the three lists for overlapping concepts and find that between Things and AllenAI TQ concepts, there are 5 overlapping concepts, between Dolch and AllenAI TQ concepts, there are 11 overlapping concepts and between Things and Dolch, there are 4 overlapping concepts. This is shown in Figure 2.5 and a full list of overlapping words is shown in A.2. We considered whether to remove overlapping concepts from the lists, but decided against it, as the overlap is small and it would change the semantics of the lists.

Name	Size	Description	Five example entries
AllenAI TQ test	100	Most frequent concepts from human-played games	“comma”, “widow”, “cucumber”, “librarian”, “interview”
Things	100	Common objects, animals, foods, plants, vehicles etc.	“egg”, “guitar”, “sloth”, “umbrella”, “marble”
Dolch	100	Most frequently used words in children’s books	“apple”, “box”, “Christmas”, “eye”, “garden”

Table 2.1.: Word lists for the Twenty Questions game

Model-generated dialogues from GPT-4o In the previously mentioned Kaggle competition, entrants Madha et al. used GPT-4o [20] to generate dialogues for the Twenty Questions game which they published²². It contains 9502 dialogues in total, generated from different word lists. One of the lists is the previously mentioned Things concepts. Another list is the words from the Kaggle competition dataset as well as replays from the competition.

²⁰<https://github.com/apple/ml-entity-deduction-arena/tree/main/data/things> (last accessed on 12/09/2024)

²¹<https://www.readstern.com/wp-content/uploads/2013/03/ComparingDolchAndFryLists.pdf> (last accessed on 12/09/2024)

²²<https://www.kaggle.com/datasets/sambitmukherjee/gpt-4o-game-play-data-llm-20-questions> (last accessed on 12/09/2024)

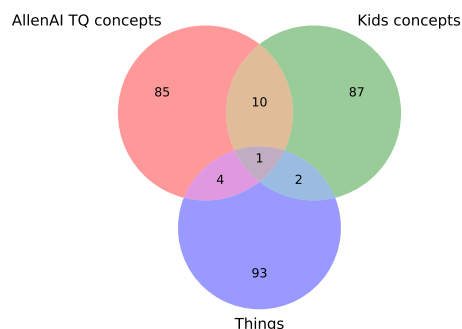


Figure 2.5.: Overlap of the word lists for the Twenty Questions game

They also tasked GPT-4o to predict new similar words given words from the competition dataset. We refer to this dataset in this work as the GPT-4o TQ dataset.

2.3.2. WordNet

WordNet is a large lexical database of the English language that was developed at Princeton University [31]. It is freely available and can be used for various natural language processing tasks. Words are organized as synonym sets, so-called synsets, which are connected by different types of relations. The dataset contains over 117,000 synsets. Relations between words include synonymy (same meaning), antonymy (opposite meaning), hyponymy (super-subordinate relation), meronymy (part-whole relation) and troponymy (relation between verbs that expresses different manner of action). Because the hyponymy relation is transitive, it essentially forms a taxonomy of all synsets.

2.3.3. Visual Genome

Visual Genome is a large dataset containing over 108,000 images [24]. For each image, the dataset contains object and attribute labels, region descriptions as noun phrases, answers to visual questions, relationships between objects as well as region and scene graphs. Objects, attributes, relationships and region descriptions are mapped to synsets from WordNet. The dataset is publicly available²³ and can be used for various tasks in computer vision and natural language processing. Main motivating factor behind the dataset is to provide a basis for learning cognitive scene understanding. Containing images are collected to be general purpose and not to be biased towards a specific task [24].

²³<https://homes.cs.washington.edu/~ranjay/visualgenome/index.html> (last accessed on 12/09/2024)

3. Methodology

In this chapter, we describe the approach we take to answer the research questions we have posed in section 1.2. We start by describing our metrics and the evaluation setup for the initial evaluation as well as our adapted version of the Twenty Questions task for subsequent evaluations. We then detail the prompt tuning and fine-tuning approaches for answerer and questioner that we explore to improve the models' performance.

3.1. Metrics

There are two metrics that we can calculate to evaluate the task performance of the models, which we describe in the following: Game score and answerer accuracy.

3.1.1. Game scores

The first is the measurement of the game scores, which requires playing the game with a questioner and an answerer for a list of words and then evaluating the generated game dialogues. This score measures both the ability of the questioner to ask good questions and the ability of the answerer to answer them correctly. If either of them fails, the game is lost. As scores, we use similar approaches as used in the BIG-bench [46] task and by Zhang et al. [55] in their evaluation. Our first score (identical to Zhang et al.) is the success rate, which is the percentage of games that the model wins, which we denote as "Win". The second score is the average number of questions asked in games that the model wins, which we note as "Turns". As the maximum number of questions is 20, that is the worst possible score. To get a better understanding of the actual dialogue length for games that are won, we also measure the average number of questions separately for only the games that are won. We denote this score as "Ts. won". A final score, denoted "Yes", that is replicated from Zhang et al. [55] is the average number of questions that have been answered with a "Yes". In their evaluation, they found that this metric is somewhat correlated with the model's performance. It can also be seen as a diagnostic metric for the answerer. For the "Turns", "Ts. won" and "Yes" scores, we also calculate the standard deviation (printed in parentheses in the evaluation tables).

3.1.2. Answerer accuracy

The second score only concerns the answerer. Using the AllenAI TQ dataset, we can measure the answerer's accuracy in answering questions individually without playing a game. For a set of binary yes/no questions and answers, we can calculate the F1 score,

which is the harmonic mean of precision and recall:

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (3.1)$$

For a set of multiple choice questions and answers, we can calculate the accuracy of the answerer:

$$\text{accuracy} = \frac{\text{correct answers}}{\text{all answers}} \quad (3.2)$$

Unfortunately, there is no analogous way to measure the questioner’s ability individually to ask the right questions without playing the game in conjunction with an answerer which is why we have to rely on the game score for the questioner.

3.2. Evaluation setup

Given these metrics, we can evaluate task performance in the Twenty Questions task for foundation language models to answer research question RQ1. We begin with the existing Twenty Questions task from BIG-bench for comparability and continue to develop a custom evaluation task that seeks to minimize the shortcomings of the original task to evaluate the models in a more precise way. These evaluations also provide a baseline comparison point for later evaluations.

3.2.1. Evaluation using BIG-bench task

To ensure comparability with other work, we use the Twenty Questions task as it is implemented in BIG-bench [46] (see section 2.2.3). At the time of this research, there are a lot of newer, more powerful models of different sizes that have no published scores in this Twenty Questions task. We test a large number of open-weight models with different numbers of parameters. Model size is one of the factors that we want to investigate in research question RQ2, but is also a leading question of the BIG-bench project [46]. Our expectation is that the score will be better for larger models, but that the task is still challenging for all models. Furthermore, the results of this evaluation are used to decide which models to choose for fine-tuning in the next step.

3.2.2. Evaluation using custom task

The initial implementation from BIG-Bench is problematic in three dimensions: The win condition is not congruent with the game goal, the number of sample points is very low and the prompt format is incorrect for fine-tuned models. In the following, we will detail each shortcoming and how we address it.

More precise win condition First, the win condition causes some games to be won that shouldn’t be. In one preliminary example game, the concept is “car” and the questioner is asking whether the concept is a carnivore, thereby winning the game, as “car” is a substring of “carnivore”. We improve this by only counting games as won if the concept is

mentioned by the questioner as full word (not substring). On the other hand, some games aren't won that should be. In another game dialogue in preliminary tests, the questioner is asking for "plane" and "airliner", but not for the actual concept "airplane". This causes the answerer to confirm each of the questions, but the game is not actually count as won. For the Things concepts list which contains synonyms, we also count games as won if the concept is a synonym of the answer. For the other lists, we considered checking for synonyms of the concept, but decided against it as it increased the chance of false positives. We also considered adding an option for the answerer to decide a game as won or to employ another LLM as judge. However, we decided against this as it would introduce more uncertainty to the results and increase the complexity of the evaluation. We are aware of this shortcoming in our evaluation that in few cases may cause scores to be lower than they could be.

Bigger word lists The original task only performs the game for 41 words, of which only 13 are so-called "coarse concepts", the rest being harder to guess. This makes statements from an already brittle metric even more unreliable. This is why we use the AllenAI TQ test concepts for testing (see subsection 2.3.1). We chose 100 words from the test dataset as a compromise between a higher number of concepts and the time it takes to evaluate the models. We consider these to be suitable concepts for the task, as they are taken from real Twenty Questions games played by humans. As the list is sorted by frequency, the words at the top are easier to guess than the words at the bottom (the first five examples from the list are "food", "air", "wall", "pen", "ocean", the last five examples from the list are "employee", "lung", "model", "sweathouse", "dumbwaiter"). As we are also interested in the effect of constraining possible concepts, we run the evaluation on the Things concepts list from Zhang et al. [55] and the Dolch concepts list.

Correct prompt format for fine-tuned models We ran the original task for comparability reasons, but the lack of correct prompts will likely lead to worse performance for fine-tuned models. We adapt the task for instruction-tuned models by using the correct prompt format that each model was fine-tuned for.

3.2.3. Evaluation of answerer accuracy

To measure the answerer's performance, we use the filtered AllenAI TQ dataset using a question answering prompt while allowing either a binary yes/no answer or a multiple choice answer (the dataset contains both labels). This dataset is much bigger than what Bruyn et al. [5] used, which allows us to get a more reliable score. Furthermore, the dataset contains both binary yes/no answers and multiple choice answers, which we both use for evaluation. We expect the answerer to perform well on this task, as question answering is a more common task that is likely to appear in LLMs' training data. We also expect larger models to perform better, as they have more parameters and thus more capacity to learn the required world knowledge to answer the questions.

3.3. Prompt tuning

As we are asking in RQ2 about what impacts the game performance, our first and simplest approach to improve the models’ performance is to tune the prompt. There are several ways that we test, which we describe in the following.

Multiple choice answers First, we test the effect of allowing multiple choice answers instead of only a binary yes/no answer. There are three different sets of multiple choice answers we are testing: The first is simply adding a “maybe” option as done by Zhang et al. [55]. The second set is the set of six answer options from the AllenAI TQ dataset. The third is the set of twelve answer options from Burgener’s 20Q game. The three sets of answer options are shown in Table 3.1. These were chosen because they differ in granularity and

Set	Count	Options
Yes/No	2	Yes, No
Yes/No/Maybe	3	Yes, No, Maybe
AllenAI TQ options	6	Always, Usually, Sometimes, Rarely, Never, Irrelevant
20Q options	12	Yes, No, Unknown, Irrelevant, Sometimes, Maybe, Probably, Doubtful, Usually, Depends, Rarely, Partly

Table 3.1.: Different sets of answer options used in evaluation.

number of options. As this allows the answerer to give more information, we expect the models to be able to guess the concept faster.

Few-shot prompts As second prompt-tuning test, we are employing a few-shot approach where the model is given two examples of dialogues from dialogues with Burgener’s 20Q game to learn from. Again, we expect this to improve the models’ performance.

Single question answerer prompts As a third approach, we test using a single question prompt for the answerer. Because each answer does not depend on previous answers, we expect this to work well and because of the simpler task, it may improve the models’ performance.

Visual information Finally, we test the effect of adding visual information to the prompt, effectively turning the game into I Spy. This reduces the number of possible concepts drastically, so we anticipate much better performance from the models.

3.4. Model fine-tuning

In RQ3, we are asking about improving the models’ performance by fine-tuning. We test both supervised fine-tuning and reinforcement learning to train our models.

3.4.1. Supervised fine-tuning

In supervised fine-tuning, we train the two partners of the game separately because their tasks differ greatly. The answerer’s task can be seen as a form of sequence classification for truth value, which is a task that mainly requires world knowledge. On the other hand, the questioner’s task is to predict questions that optimally reduce uncertainty, which additionally requires some form of reasoning.

For the answerer, similar to the evaluation of answerer accuracy, we can work with single questions as training dataset and don’t need full dialogues, because the answer of each question does not depend on previous questions and answers. There are two possible datasets that contain questions and labeled answers that we could use for training: The AllenAI TQ dataset from AllenAI and the dataset from Bruyn et al. [5]. We prefer the AllenAI dataset because the data is taken from real Twenty Questions games played by humans. Furthermore, the dataset is much bigger (78,890 data points when unfiltered) than the dataset from Bruyn et al. (2,832 data points). Moreover, the dataset contains multiple labels for each data point, which further improves data quality. The dataset is also more flexible, as it contains five different answer options as well as a binary yes/no answer. We train and test both with the five answer options as well as with the binary answer.

For the questioner, we need full dialogue, as the best possible next question in a game is highly dependent on previous questions and answers. As there is no dataset with human annotations publicly available, we will generate datasets using different sources. In the following, we describe the three different ways of generating dialogue data.

20Q game For our first source for training dialogues, we are using a Burgener’s 20Q algorithm online¹ to play Twenty Questions. We let an LLM play as the answerer against 20Q and generate a dialogue for each game. This is analogous to Bruyn et al. [5], who used the Akinator game to generate dialogues. We decided to collect dialogues using 20Q instead, as Akinator is only available for characters, animals and objects, not abstract concepts like colors or emotions.

WordNet taxonomy Secondly, we use WordNet’s hypernym labels as a taxonomy of concepts. For all concepts from the AllenAI TQ train dataset, we look at their hypernym path to the root synset, (“entity.n.01”). For more plausible dialogues, we filter nodes from the paths that are not known to children using an LLM as judge. Taking all paths, we can generate a tree that contains the shortest paths to enumerate each concept. We use this tree to generate a game dialogue for each concept.

Model-generated dialogues Finally, we use a bigger model to generate dialogue to train a smaller model, a process called distillation or behavior cloning. We are generating dialogues with Mixtral-8x7B-Instruct-v0.1 [22] and use third-party dialogues generated with GPT-4o [32].

¹<http://www.20q.net/> (last accessed on 12/09/2024)

3.4.2. Reinforcement learning

In RQ3, we also ask whether reinforcement learning can be used to train the models, in particular PPO [43]. Instead of training data, reinforcement learning needs a reward signal, for which we use a scoring mechanism based on the game score. PPO is usually used for RLHF to align language models, with a second model trained on human feedback to generate rewards. In our case, we are using a hand-crafted reward function and therefore don't need a second model or dataset. We are interested to see whether the score as training signal can improve performance for both untrained foundation models as well as fine-tuned models.

3.4.3. Joint evaluation

Finally, we are interested in the effect of evaluating both a trained answerer and trained questioner together in game. So far, we only evaluate a trained questioner with an untrained answerer and a trained answerer with an untrained questioner for comparability reasons. We are interested to see whether the performance increases when both partners are trained.

4. Experiments and results

In this chapter, we present the experiments conducted to answer the research questions, following the methodology described in chapter 3. We also present and discuss our results.

4.1. Implementation

All experiments are implemented using the Huggingface Transformers library¹ and the PyTorch library². For faster attention calculation that more efficiently uses GPU compute and memory, we are using FlashAttention 2 [9]. Our setup is running inside Nvidia’s PyTorch docker container³ to allow for simpler deployment of dependencies on the compute nodes, especially the requirements for installing FlashAttention 2. We run most of our experiments on the High Performance Computing cluster bwUniCluster2.0⁴. The cluster offers three different GPU types: V100 (32GB), A100 (80GB) and H100 (94GB), with different compute capabilities⁵. The cluster uses the Slurm workload manager⁶ for job scheduling. In our experience, scheduling time increases exponentially with the number of requested GPUs, from days to multiple weeks, so we try to use as little resources as possible. Some tasks are run on a node of the cluster of the Institute for Anthropomatics and Robotics at the Karlsruhe Institute of Technology featuring Nvidia RTX 6000 Ada GPUs. This cluster does not use scheduling, but is shared with other researchers. Depending on memory requirements and GPU availability, we use different GPU types for different tasks.

4.2. Dataset preprocessing

In this section, we describe preprocessing steps for the datasets we use.

Filtering AllenAI TQ dataset To improve the data quality and to reduce noise for evaluation and for training, we introduce a filtering step to the dataset to only include questions that all three labelers consider high quality and where all three labelers agree on the answer. This was done because we expect data points with mismatching labels to be questions that do not have a clear answer and thus, do not help the model to learn. This process reduces the dataset to 37,122 data points which is around 47% of the original dataset. Around 12,000

¹<https://huggingface.co/docs/transformers/index> (last accessed on 12/09/2024)

²<https://pytorch.org/> (last accessed on 12/09/2024)

³<https://ngc.nvidia.com/catalog/containers/nvidia:pytorch> (last accessed on 12/09/2024)

⁴https://www.scc.kit.edu/dienste/bwUniCluster_2.0.php (last accessed on 12/09/2024)

⁵https://wiki.bwhpc.de/e/BwUniCluster2.0/Hardware_and_Architecture#Components_of_bwUniCluster (last accessed on 12/09/2024)

⁶<https://slurm.schedmd.com/> (last accessed on 12/09/2024)

4. Experiments and results

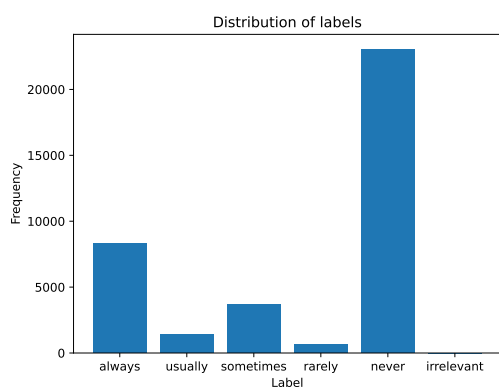


Figure 4.1.: Distribution of answer options in filtered AllenAI TQ dataset (37,122 entries)

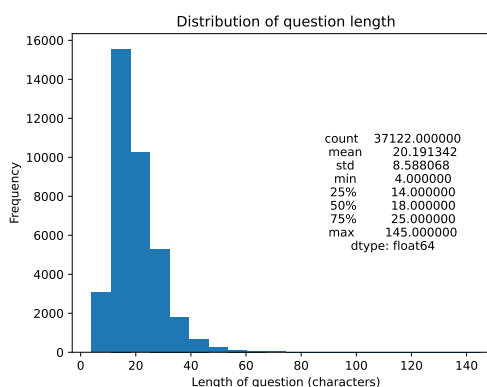


Figure 4.2.: Question length distribution of filtered AllenAI TQ dataset

questions have been removed because of their quality score, the rest has been removed because of non-agreeing labels. We are using the existing train/test/validation split of the dataset. After applying the filter, the train set contains 21,832 data points, the test set contains 8,035 data points and the validation set contains 7,255 data points, which is approximately a 60/20/20 split.

In the dataset, the five possible answer options occur with highly varying frequency. The distribution can be seen in Figure 4.1. There are much more questions labeled as “never” than the other labels. This is likely due to the nature of the game, where the answerer tries to find a concept that is hard to guess. As this is human annotated data, we accept this as a natural bias of the dataset that is inherent to the game. In Figure 4.2, we show the distribution of question length in characters. Most questions are rather short with the upper quartile being 25 characters. We show the distributions for the unfiltered dataset in Figure A.1 and Figure A.2. In comparison, the filtered dataset contains less questions that are labeled as “usually” and “rarely”. This is possibly due to the filtering of questions with non-agreeing labels, as we are left with questions that have a clear answer. The distribution of question length is very similar between the filtered and unfiltered dataset.

Filtering GPT-4o generated dialogues For later training of questioner models, we filter the generated dialogues from the GPT-4o model to filter out game dialogues whose answer occur in one of our three word lists that we use for testing. The original dataset contains 9502 dialogues, of which 1791 are won games. After removing dialogues with answers from the word lists, we are left with 9191 dialogues, of which 1634 are won games. This is a reduction of around 3% of the total dialogues and around 9% of the won dialogues.

4.3. Hyperparameters

As we are limited on computational resources, we are not able to perform hyperparameter optimization. Therefore, we use default values for most hyperparameters.

Inference For inference of the questioner, we use the models’ default generation configurations from their authors for temperature and sampling settings. E.g. for Llama-3-8B-Instruct, we use a temperature of 0.6 and top- p sampling with a p of 0.9. We decode a maximum of 100 tokens for the questioner and stop generation at a question mark or end of the sequence. For the answerer model, we choose a different approach, as the answerer model should only answer one of the possible answer options. The answerer model is not decoded freely, but only probed for which of the tokens of the given answer options is more likely given the input. This way, the answerer can only ever produce one of the answer options.

LoRA parameters For fine-tuning with LoRA, we use the following hyperparameters, if not stated differently: We choose conventional default values for our LoRA parameters, with a rank of 16, alpha of 16, and dropout of 0.05. We add adapters to all linear layers of the model except for the language modeling head. As training parameters, we train with a learning rate of 5.0×10^{-6} for 2 epochs and cosine learning rate scheduling with a warmup ratio of 0.2. The batch size is usually set to 32, depending on sequence length and memory requirements of the model. The maximum sequence length differs between tasks, whether we train on single questions or whole dialogues.

4.4. Measuring task performance

In this section, we describe the evaluation of the models in the Twenty Questions task. First, we run the original Twenty Questions task from BIG-Bench. Second, we run our customized Twenty Questions task that we developed to address the shortcomings of the original task. We also use this task for further evaluations. Third, we run measurement on the answerer individually using the AllenAI TQ dataset. Finally, we discuss our results and select models for further testing.

4.4.1. BIG-bench task

We implemented the necessary functionality to play the Twenty Questions task from BIG-Bench [46] with Huggingface Transformers models. In the following, we describe the original task format. The task uses prompts that are strings without any special begin, end or role tokens, as the models that it was originally designed for are not fine-tuned. The dialogue is prepended with either “Alice” for the answerer or “Bob” for the questioner. To initialize, the dialogue starts with a shared prompt explaining the game using a zero-shot approach, which means there is no example dialogue in the prompt. Only in the answerer’s perspective of the dialogue, the concept that they are thinking of is mentioned. Next, the game starts with the questioner being prompted to ask a question. In turn, the answerer is

Model	Size	Win (\uparrow)	Turns (\downarrow)	Ts. won (\downarrow)	Yes
Llama-3-70B-instruct	70.6B	0.38	15.8 (5.5)	3.5 (4.6)	10.4 (6.4)
Mixtral-8x7B-v0.1	46.7B	0.0	20.0 (0.0)	0.0 (0.0)	4.2 (4.4)
Mixtral-8x7B-instruct-v0.1	46.7B	0.0	20.0 (0.0)	0.0 (0.0)	0.0 (0.0)
Gemma-7B	8.5B	0.0	20.0 (0.0)	0.0 (0.0)	19.8 (0.5)
Gemma-7B-it	8.5B	0.0	20.0 (0.0)	0.0 (0.0)	14.1 (0.3)
Llama-3-8B-instruct	7.5B	0.15	18.0 (4.7)	1.1 (2.6)	14.6 (4.4)
Llama-3-8B	7.5B	0.0	20.0 (0.0)	0.0 (0.0)	20.0 (0.0)
Mistral-7B-instruct-v0.2	7.2B	0.0	20.0 (0.0)	0.0 (0.0)	4.6 (5.5)
Mistral-7B-v0.1	7.2B	0.0	20.0 (0.0)	0.0 (0.0)	18.0 (0.0)
Phi-3-mini-4k-instruct	3.8B	0.0	20.0 (0.0)	0.0 (0.0)	11.2 (3.6)
Gemma-2B	2.5B	0.0	20.0 (0.0)	0.0 (0.0)	19.2 (0.4)
Gemma-2B-it	2.5B	0.0	20.0 (0.0)	0.0 (0.0)	11.0 (0.0)
Tinyllama-1.1B-chat-v1.0	1.1B	0.0	20.0 (0.0)	0.0 (0.0)	20.0 (0.0)

Table 4.1.: Results of Twenty Questions task with original task format. Note that this task is only run with 13 words. See subsection 3.1.1 for explanation of columns.

prompted with the question. The dialogue then continues with answerer and questioner taking turns on each other’s output. Each time, the full dialogue up until the current moment is given as prompt to both of the models. The game is considered won and ends immediately if a question contains the concept. The game is considered failed if the questioner has asked a maximum number questions without mentioning the correct concept. The prompts are shown in full in the appendix in Figure A.3 for the answerer and Figure A.4 for the questioner. Our list of evaluated models contains models from the Llama [49] family (TinyLlama [54] and Llama 3 [13]), the Gemma [29] family, the Phi [1] family as well as Mistral 7B [21] and Mixtral 8x7B [22]. We evaluated instruction-tuned and non-instruction-tuned models. As largest model, we evaluate Llama-3-70B-Instruct (we only ran the instruction-tuned model because of the high computational cost). This original task only tests 13 concepts, however as we expected only very low scores and are only looking for a qualitative measure of whether the models are able to play at all, this is tolerated.

Our results are shown in Table 4.1. We can see that only Llama 3-70B-Instruct and Llama 3-8B-Instruct are able to win any games. The low outcome is similar to the original published results for the BIG-bench task. However, in BIG-bench, the only wins seem accidental, in our case, it seems reasonable that the biggest models are able to win some games. To determine whether the low performance is really due to the models or bad task design (especially wrong prompt format), we run the evaluation using our improved task in the following.

4.4.2. Custom game task

Our custom task is an updated version according to subsection 3.2.2 that uses the correct chat template, stricter checking for the win condition and our AllenAI TQ test concepts

dataset containing 100 words for the evaluation. Concerning chat templates, models differ on whether their training includes an initial message from a system role or not (e.g. Llama and Phi3 models are trained with a system role, Mistral and Gemma are not). We use the system role for Phi-3-mini-4k-Instruct only, as otherwise model performance is significantly lower. For all other models, the system role is unused, as it is either not available or worsens performance. Everything else is unchanged from the BIG-bench task. The updated task is re-run for all of the instruct-tuned models and results are displayed in Table 4.2. The results show some success for the larger models, especially Llama-3-70B-

Model	Size	Win (\uparrow)	Turns (\downarrow)	Ts. won (\downarrow)	Yes
Llama-3-70B-Instruct	70.55 B	0.23	17.5 (5.0)	9.0 (3.8)	7.8 (4.3)
Mixtral-8x7B-Instruct-v0.1	46.70 B	0.23	18.3 (3.7)	12.6 (4.2)	5.4 (2.4)
Gemma-7B-Instruct	8.53 B	0.00	20.0 (0.0)	N/A	12.0 (6.2)
Llama-3-8B-Instruct	7.50 B	0.10	18.8 (3.6)	8.2 (2.7)	13.4 (3.9)
Mistral-7B-Instruct-v0.2	7.24 B	0.08	19.4 (2.5)	12.2 (4.8)	5.0 (2.3)
Phi-3-mini-4k-Instruct	3.82 B	0.10	18.9 (3.3)	9.3 (2.6)	10.1 (3.8)
Gemma-2B-Instruct	2.51 B	0.00	20.0 (0.0)	N/A	19.0 (0.0)
TinyLlama-1.1B-Chat-v1.0	1.10 B	0.01	19.8 (1.8)	2.0 (0.0)	19.8 (1.9)

Table 4.2.: Results of Twenty Questions task with customized task. Models are sorted by size. See subsection 3.1.1 for explanation of columns.

Instruct and Mixtral-8x7B-Instruct-v0.1. Some smaller models like Phi-3-mini-4k-Instruct and Llama-3-8B-Instruct are also able to win some games. Model performance seems to scale with model size, but there are exceptions. The Gemma models are still not able to win any games.

Results for smaller models are much better than the so-far published BIG-bench results for GPT-3 [4] and BIG-G [46], which barely showed any success. This is likely due to enhanced model architectures, datasets and training procedures of the newer models we tested. In comparison to Zhang et al. [55], our results show similar performance (they used a different word list for evaluation, which we use in the next section). Their best model, GPT-4 [32] reaches a success rate of 0.31 on the Things dataset, which is still a lot better than our best model. Regarding the number of questions answered as “Yes”, our best models are also comparable to Zhang et al.’s results, with our best model Llama-3-70B-Instruct getting 7.8% “Yes”-answers and GPT-4 getting 5.9% (however, they used yes/no/maybe-options [55]).

4.4.3. Common failure modes

In the following, we want to describe common failure modes of models in both roles of the Twenty Questions task.

Nonsensical answers Many of the smaller and non-instruction-tuned models display monotonous behaviour independent of the prompt and always output the same answer. This is the case for TinyLlama, Gemma-2B and 7B non-instruction-tuned.

Wrong answers This problem occurs with the larger models as well that are able to answer many questions correctly, but still go wrong in some places. Unfortunately, Twenty Questions dialogue is quite sensitive to wrong answers; one wrong answer can mislead the questioner completely and cause the game to be lost. In our evaluation, only the largest Llama3-70B-Instruct model was able to answer most questions correctly in the role of the answerer. Another form of this issue is that the answerer is seemingly forgetting the original concept and then answering positively, as if the concept has been correctly guessed even if it is not the correct concept. This occurs with many models, e.g. with Llama3-8B.

Nonsensical questions The most common misbehavior from models in questioner role is to ask for numbers, e.g. asking “Is the concept 10?”, then repeating the question with rising numbers. This behavior was displayed by all models from the Gemma family and all Mistral and Mixtral models. This misbehavior may be caused by the prompt asking for a question about a “concept”, but it remains unclear. Some other forms of nonsensical question generation occurs with smaller models like TinyLlama that always ask the same question independent of the context.

Unhelpful questions For models that are able to generate questions, the questions are often unhelpful in the sense that they do not reduce the uncertainty about the concept. This is the issue we most want to address in fine-tuning.

Questioner failing to constrain to game dialogue Some models in the questioner role do not generate questions after a certain point of the dialogue. One example is the Llama model family, which at some point generate dialogue about frustration and ask Alice for the concept.

To summarize, we find from this initial evaluation that very small models fail entirely, to the point of not generating any meaningful output. We find that non-instruction-tuned models are unsuitable for the task, which is expected because it relies on following the initial instruction. Small instruction-tuned models do generate meaningful output, but still struggle to play the game effectively. The largest instruction-tuned models are able to play the game with some success.

4.4.4. Analyzing domain restriction

To analyze the effect of restricting the domain, we use our same evaluation task, but with the other words lists, namely the Things test concepts list and the Dolch concepts (see Table 2.1). We perform this evaluation for Llama-3-8B-Instruct, Mistral-7B-Instruct-v0.2

and Phi-3-mini-4k-Instruct. Results for evaluation with the domain-specific word lists are shown in Table 4.3.

Model	Dataset	Win (\uparrow)	Turns (\downarrow)	Ts. won (\downarrow)	Yes
Llama-3-8B-Instruct					
	AllenAI TQ	0.10	18.8 (3.6)	8.2 (2.7)	13.4 (3.9)
	Things	0.10	19.0 (3.0)	10.3 (2.3)	13.4 (3.4)
	Dolch	0.19	17.8 (4.7)	8.6 (3.4)	13.2 (4.9)
Mistral-7B-Instruct-v0.2					
	AllenAI TQ	0.08	19.4 (2.5)	12.2 (4.8)	5.0 (2.3)
	Things	0.08	19.6 (1.6)	14.6 (2.5)	4.5 (2.2)
	Dolch	0.18	18.3 (4.0)	10.7 (4.2)	4.8 (2.3)
Phi-3-mini-4k-Instruct					
	AllenAI TQ	0.10	18.9 (3.3)	9.3 (2.6)	10.1 (3.8)
	Things	0.02	19.9 (1.1)	14.0 (5.0)	10.2 (3.4)
	Dolch	0.08	19.3 (2.9)	11.4 (6.2)	11.2 (4.0)

Table 4.3.: Results of Twenty Questions task with domain-specific word lists.

We find that the Dolch concepts list is the most successful for all models, with the Things dataset showing similar performance to the AllenAI TQ concepts. This is likely due to the Dolch concepts being more common and easier to guess. The things list shows identical performance for Llama-3-8B-Instruct and Mistral-7B-Instruct-v0.2, but worse performance for Phi-3-mini-4k-Instruct. In comparison of all three lists, the 10% score of Phi-3-mini-4k-Instruct on the AllenAI TQ concepts seems like an outlier, as performance in the other lists is lower, although it seems like the most difficult list of words. This is possibly due to noise in the evaluation given a sample size of only 100 words per list.

4.4.5. Measuring answerer performance

We measure the performance of the answerer individually as described in subsection 3.2.3. We use a prompt consisting of concept and question (full prompt shown in the appendix in Figure A.9). Similar to the game dialogue generation, the answerer is probed for the probability of the possible answer options to determine the answer. Because the cost of this evaluation is low (only the calculation of one forward pass per model per question), we run this evaluation on a bigger set of models. We run this evaluation using the multiple-choice labels as well as a binary choice of “Yes” and “No”. For this, we use the majority vote of the three labelers as the binary choice. In the dataset, labels “always”, “usually”, and “sometimes” are mapped to “Yes” and the others are mapped to “No”. F1 and accuracy of the answerer evaluation are shown in Figure 4.3. Additionally to the F1 scores and accuracies, confusion matrices for both evaluations can be found in the appendix in Figure A.11 and Figure A.12.

4. Experiments and results

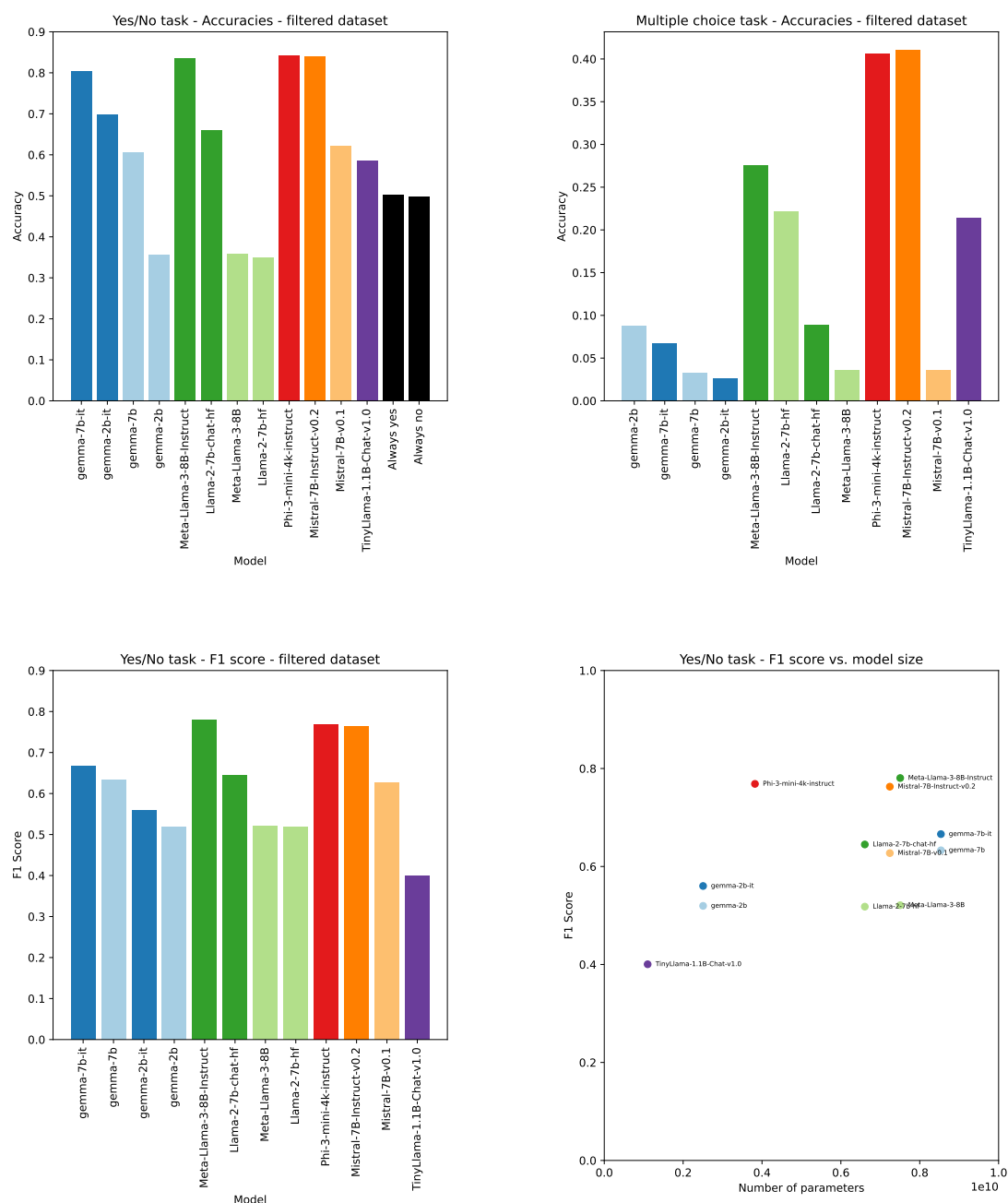


Figure 4.3.: Results of the answerer's performance. Top left: Accuracies of yes/no task. Top right: Accuracies of multiple choice task. Bottom left: F1 score of yes/no task. Bottom right: F1 score of yes/no task over model size. Model families are in a common color with solid shades for instruction-tuned models and lighter shades for non-instruction-tuned models. All evaluations are using the filtered dataset, evaluations on the unfiltered dataset can be found in Figure A.10

In the binary evaluation, the top scoring models are very close in scores to each other. Llama3-8B-Instruct performs best, reaching an F1 score of 0.781 and an accuracy of 0.836, followed closely by Phi-3-mini-4k-Instruct (F1 score: 0.769, accuracy: 0.842) and Mistral-7B-Instruct (F1 score: 0.763, accuracy: 0.840). Except for the Gemma family, the instruction-tuned models perform better than the non-instruction-tuned models, which makes sense as we used an instruction in the prompt. In the multiple choice evaluation, Mistral-7B-Instruct performs best with an accuracy of 0.411, followed by Phi-3-mini-4k-Instruct (accuracy: 0.407) and Llama3-8B-Instruct (accuracy: 0.276). In the confusion matrices, we can see that Phi-3-mini-4k-Instruct and Mistral-7B-Instruct predict more diverse labels, with most errors being the wrong prediction of “Sometimes”. Llama3-8B-Instruct almost exclusively predicts “Usually” and “Never”. The multiple choice evaluation shows lower scores, which could be explained with higher noise because of the higher number of choices. In the unfiltered evaluation of the binary task, the F1 scores are slightly lower for all models, but the ranking stays the same. In contrast, the accuracies in the unfiltered binary evaluation are minimally higher for the top models. The multiple choice evaluation shows a bigger increase in performance in using the filtered data over the unfiltered (Phi-3-mini-4k-Instruct: 0.407 vs. 0.336, Mistral-7B-Instruct: 0.411 vs. 0.322, Llama3-8B-Instruct: 0.276 vs. 0.262). Again, this shows the impact of filtering to reduce noise especially in the multiple choice evaluation.

When regarding the relationship between model size and F1 score in the binary evaluation, we see that the performance of the models increases with model size, however the correlation is moderate (Pearson correlation of 0.5012). One outlier is the Phi-3-mini-4k-Instruct model, which performs unusually well for its size.

Although our evaluation is not completely comparable to the evaluation by Bruyn et al. [5] (different prompt, different dataset and different models), we can see that the models perform better in our evaluation. Their best F1 score was reached by T0pp [42] (an 11B parameter model pre-trained on reading comprehension datasets) at 0.685 and an accuracy of 0.819 [5], which is lower than our best performing models.

4.4.6. Model selection

Based on the initial evaluation in this section, we decide to focus on Phi-3-mini-4k (a 3.8B parameter model) and Llama3-8B-Instruct in the following. We choose these two models because they are small enough to be trained with a relatively low amount of compute resources, but still large enough to be able to generate meaningful output. They are also chosen because they are different in size and training data. Both were able to generate at least some meaningful questions and correct answers, so we test them whether they are trainable to play the game more effectively.

4.5. Prompt tuning

As a first step to improve the game performance, we investigate different prompts for the Twenty Questions task. These evaluations are performed for the models Phi-3-mini-4k-Instruct (in case of visual input, its counterpart Phi-3-vision-128k-instruct) and Llama3-8B-

Instruct. There are several aspects that we are investigating: First, we are allowing more answer options for the answerer to choose from instead of just “Yes” or “No”. Second, we are using few-shot prompts by providing example dialogues from the 20Q dataset. Third, we are changing the answerer prompt to only include a single question instead of the full dialogue. Finally, we are adding visual input to the prompt, which turns the game into a version of the I Spy game. We are using the AllenAI TQ questions for evaluation in this section, except for the final task which uses Visual Genome as evaluation dataset.

4.5.1. Multiple choice answer options

This prompt tuning approach is to allow the answerer to choose from more than two answer options. Other works have used different sets of answer options of which we chose three for our evaluation. The first is simply adding a “maybe” option as done by Zhang et al. [55]. The second set is the set of six answer options from the AllenAI TQ dataset. The third is the set of twelve answer options from Burgener’s 20Q game. The three sets of answer options are shown in Table 3.1.

Other than that, we are using the same prompt for both sets of answer options. We ran this evaluation with Phi-3-mini-4k-Instruct and Llama3-8B-Instruct. The game results are shown in Table 4.4.

Model	Answer options	Win (\uparrow)	Turns (\downarrow)	Ts. won (\downarrow)
Phi-3-mini-4k-Instruct				
	Yes/No	0.10	18.9 (3.3)	9.3 (2.6)
	Yes/No/Maybe	0.02	19.8 (1.8)	7.5 (1.5)
	AllenAI TQ options	0.04	19.5 (2.6)	8.5 (6.2)
	20Q options	0.04	19.5 (2.6)	7.0 (0.7)
Llama3-8B-Instruct				
	Yes/No	0.10	18.8 (3.6)	8.2 (2.7)
	Yes/No/Maybe	0.09	19.0 (3.4)	8.3 (2.2)
	AllenAI TQ options	0.10	19.1 (3.0)3	10.6 (3.6)
	20Q options	0.06	19.4 (2.6)	10.5 (5.2)

Table 4.4.: Results of Twenty Questions task with different sets of answer options.

We find that the two models show different results for the different sets of answer options. Phi-3-mini-4k-Instruct performs a lot worse with more answer options, while the performance of Llama3-8B-Instruct stays mostly the same except for the 20Q options, where it performs worse. We also analyze the frequency of answer options in the games in Figure 4.4 and Figure A.7. Adding the “Maybe” option, we find very similar results for the two models with the “Maybe” option being the least used by both models. In the other two sets of answer options, the models differ significantly. We find that Llama3-8B-Instruct uses “Usually” most of the time in both cases. Concerning the AllenAI TQ options, Llama3-8B-Instruct practically only uses “Usually” and “Never”, very similar to a binary choice.

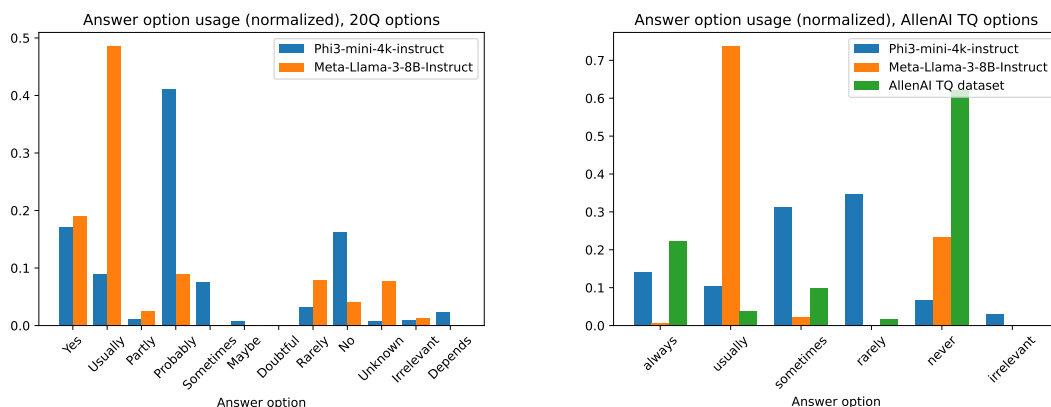


Figure 4.4.: Usage of answer options in Twenty Questions task for 20Q options (left) and AllenAI TQ options (right). For yes/no/maybe options, see Figure A.7

In the 20Q case, Llama3-8B-Instruct uses some of the other options as well. On the other hand, Phi-3-mini-4k-Instruct uses “Probably” most of the time in the 20Q options case and “Sometimes” or “Rarely” in the AllenAI TQ options case. For the 20Q options, we have no data to compare to human baseline, but for the AllenAI TQ options, we can compare to the frequency of responses from the dataset from real Twenty Questions games. The comparison shows that human players are more concrete in their answers, with “Never” and “Always” being the most common answers.

4.5.2. Few-shot prompts

In the next step, we are using few-shot prompts by including two example game dialogues in the beginning prompt. They are two samples from the collected dialogues from 20Q⁷ (see subsection 4.7.2 for more detailed description of how they were collected). The first example is a game where the concept is “elephant” and the second example is a game where the concept is “the sun”. The full prompt system prompt from questioner perspective is shown in Figure A.8. As the examples from 20Q use the 20Q answer options (see subsection 4.5.1), we are testing both the original yes/no evaluation as well as the 20Q answer options for this evaluation. Results for Phi-3-mini-4k-Instruct and Llama3-8B-Instruct are shown in Table 4.5.

⁷<http://20q.net> (last accessed on 12/09/2024)

4. Experiments and results

Model	Prompt	Win (\uparrow)	Turns (\downarrow)	Ts. won (\downarrow)	Yes
Phi-3-mini-4k-Instruct					
	Zero shot	0.10	18.9 (3.3)	9.3 (2.6)	10.1 (3.8)
	Few shot Yes/No	0.03	19.7 (1.8)	9.7 (1.2)	11.1 (4.4)
	Few shot 20Q	0.03	19.6 (2.2)	8.0 (4.5)	8.2 (3.5)
Llama3-8B-Instruct					
	Zero shot	0.10	18.8 (3.6)	8.2 (2.7)	13.4 (3.9)
	Few shot Yes/No	0.06	19.4 (2.7)	10.0 (5.3)	16.8 (3.4)
	Few shot 20Q	0.03	19.6 (2.1)	8.3 (3.9)	5.2 (2.2)

Table 4.5.: Results of Twenty Questions task with inclusion of few-shot prompt.

We find that again, performance decreases for both models. This is surprising, as we expected the models to perform better with more examples. Performance does not improve with the 20Q answer options either. Looking at the dialogues, we find that models consistently copy the first question from the example dialogues, which is “Is it a living thing?”. Unfortunately, in the yes/no task, the answerer often wrongly answers “Yes” to this question for concepts where it is unclear, which causes the game to go off track. In the 20Q answer options case, the answer option answers more often with “Unknown”. Still, the models are not able to extract a strategy from the examples. Giving the two example dialogues strongly steers the models in the direction of the concepts of the examples. As it is only two examples, that is not representative for the whole space of possible concepts.

4.5.3. Single question answerer prompt

Another variation of the game is to change the answerer prompt to not include the full dialogue, but instead only a single question. The prompt is shown in Figure A.9. This is a simplification of the task, as the answerer does not have to keep track of the full dialogue. The results are shown in Table 4.6.

Model	Answer options	Win (\uparrow)	Turns (\downarrow)	Ts. won (\downarrow)	Yes
Phi-3-mini-4k-Instruct					
	Full dialogue	0.10	18.9 (3.3)	9.3 (2.6)	10.1 (3.8)
	Single question	0.14	18.8 (3.6)	11.4 (5.2)	2.2 (1.4)
Llama3-8B-Instruct					
	Full dialogue	0.10	18.8 (3.6)	8.2 (2.7)	13.4 (3.9)
	Single question	0.07	19.4 (2.4)	10.7 (2.1)	8.5 (2.5)

Table 4.6.: Results of Twenty Questions task with single question answerer prompt.

We find that the single question answerer prompt delivers different results for both models. For Phi-3-mini-4k, when supplying the whole dialogue to the answerer, it would often start to answer wrongly after a few turns, seemingly forgetting the original concept. In many cases, the answerer would start to agree with the questioner on a concept that is not the original concept. This is effectively prevented by only asking a single question each time. Llama-3-8B-Instruct does not show this behavior as much, but may not be optimally adapted to the single question prompt.

4.5.4. Adding visual input

Our final approach to evaluating the impact of the prompt on the task is the addition of visual input to the prompt. This requires a model that is fine tuned on multimodal data, which is why we only use Phi-3-vision-128k-instruct for this evaluation. As dataset, we are using Visual Genome [24] (see also subsection 2.3.3). For each image from the dataset, we are using the first object and its first label to play a guessing game. Our prompt is the same to the previous tasks except for two changes: We are prepending the image to be beginning of the prompt and include the first letter of the object in the announcement of the answerer (as is common in I Spy games). Adding the information about the first letter and the image introduces two new pieces of information. This is why we are evaluating model performance with and without showing the image (with otherwise identical prompts). The results are shown in Table 4.7.

Model	Win (↑)	Turns (↓)	Ts. won (↓)	Yes
Phi-3-vision-128k-instruct				
First Letter / No image	0.14	17.2 (6.5)	2.2 (1.1)	2.3 (5.6)
First Letter / With image	0.26	15.0 (8.2)	1.5 (1.0)	0.7 (3.0)

Table 4.7.: Results of visual guessing game task with or without image for Phi-3-vision-128k-instruct.

We find that just the mention of the first letter significantly increases winning chances and changes the model’s strategy to simple enumeration. For the won games, the model only needed to guess less than 3 times on average. For the lost games, for many letters the model seems to not be able to guess any concept or just repeatedly guess the same word. For the letter “a”, it guesses “a skateboard”. This may be due to tokenization, as the letter “a” is a different token than the tokens of words beginning with “a”.

The addition of image data increases the winning chances, with a bit more than a quarter of the games being won. Similarly, the average round count of won games is reduced by almost a whole round with many games being won with just one guess. We find that this game is a very different task to the text-only Twenty Questions game and that this task that is a lot easier. Both the addition of announcing the first letter and providing an image as additional source of information are strong reductions in the space of possible concepts. Furthermore, limiting the possible concepts to visible entities in

general is a strong reduction in the space of possible concepts. Models do not display a guessing strategy to the same extent as in the text-only game, but instead simply list possible solutions. Because of the small solution space, this is an effective strategy. The same strategy could be achieved using an object recognition model like YOLO [39] and asking for the recognized objects in descending order of confidence. As we are interested in strategies that are beyond simple enumeration, we are not further investigating this task in this work.

4.6. Fine-tuning the answerer

In this section, we describe our approach to fine-tuning the answerer. We are using the AllenAI TQ dataset for training and evaluation (see subsection 2.3.1). We evaluate both the individual answerer performance in the question answering task as well as the performance in the game together with a questioner model.

4.6.1. Supervised fine-tuning of multiple choice task

First, we run supervised fine-tuning with the train set of the dataset on Phi-3-mini-4k-Instruct and Llama3-8B-Instruct. We first train using the multiple choice labels of the dataset. We are training single questions, so we use the single question answerer prompt. All sequences were padded to a maximum length of 128 tokens (the questions in the dataset are very short). We used a batch size to 32, which caused training to require close to the limit of our GPU’s memory. This allowed us to train the models to 2 epochs in 1.5 hours. Evaluation loss very quickly converged to a minimum (see appendix Figure A.13). To test whether the model still learns after one epoch, we evaluated Phi-3-mini-4k-Instruct on the test set after four different points of training (0.5, 1, 1.5 and 2 epochs). The results are shown in Table 4.8. We can see that training has not fully saturated after two













Epoch	Train loss	Eval loss	Accuracy		Accuracy filtered	
			1.0		1.0	
untrained	N/A	N/A	0.336		0.394	
0.5	0.307	0.310	0.592		0.706	
1.0	0.302	0.305	0.604		0.715	
1.5	0.301	0.303	0.615		0.728	
2.0	0.298	0.302	0.619		0.730	

Table 4.8.: Evaluation of Phi-3-mini-4k-Instruct as answerer at different stages of training. Accuracy is measured on the unfiltered dataset and the filtered test dataset.

epochs, but that it is close to saturation. One epoch of training already improves the model significantly. The filtered evaluation shows a higher accuracy than the unfiltered evaluation, which is expected as the filtered dataset contains less noise (all labelers agree on the answer and consider the question to be of high quality). The evaluation loss is very close to the training loss, which indicates that the model is not overfitting.

After training both models, we evaluated each model’s performance on the multiple choice task and the binary choice task. The results are shown in Table 4.9. For the multiple

Model	Test Task	Trained	Acc. (↑)	Acc. F. (↑)	F1 (↑)	F1 F. (↑)
Phi-3-mini-4k-Instruct						
	Multiple choice task					
		Untrained	0.336	0.394		
		Trained	0.619	0.730		
	Binary choice task					
		Untrained	0.813	0.842	0.799	0.769
		Trained	0.821	0.869	0.797	0.797
Llama3-8B-Instruct						
	Multiple choice task					
		Untrained	0.262	0.302		
		Trained	0.631	0.745		
	Binary choice task					
		Untrained	0.834	0.836	0.836	0.781
		Trained	0.860	0.848	0.873	0.814

Table 4.9.: Results of supervised fine-tuning of **multiple choice task**. The test dataset has 16921 entries, the filtered test dataset has 8037 entries.

choice task, we find that the accuracy on the filtered dataset for Phi-3-mini-4k-Instruct increases by a factor of around 1.9 and for Llama for a factor of around 2.5. Again, the filtered evaluation yields higher results, which is expected as the filtered dataset contains questions with clearer answers. Applying these models to the binary task that they have not been trained for, we find that the trained models perform slightly better overall than the untrained models, which means that training on multi-choice questions also improves the binary choice task. Overall, Llama3-8B performs slightly better than Phi-3-mini-4k which may be due to the larger model size. In the following, we test training the binary task directly.

4.6.2. Supervised fine-tuning of binary choice task

Analogous to the previous section, we perform fine-tuning on the binary labels from the dataset. All other settings are the same. Training and evaluation losses show a similar pattern as the multiple-choice task. Charts are shown in the appendix in Figure A.14. Our results are shown in Table 4.10.

4. Experiments and results

Model	Test Task	Trained	Acc. (↑)	Acc. F. (↑)	F1 (↑)	F1 F. (↑)
Phi-3-mini-4k-Instruct						
	Multiple choice task					
		Untrained	0.336	0.394		
		Trained	0.475	0.584		
	Binary choice task					
		Untrained	0.813	0.842	0.799	0.769
		Trained	0.856	0.888	0.850	0.841
Llama3-8B-Instruct						
	Multiple choice task					
		Untrained	0.262	0.302		
		Trained	0.165	0.158		
	Binary choice task					
		Untrained	0.834	0.836	0.836	0.781
		Trained	0.828	0.821	0.852	0.794

Table 4.10.: Results of supervised fine-tuning of **binary choice task**. Accuracy is measured on the unfiltered dataset and the filtered test dataset (noted with F.).

We find that training on the binary choice only increases the F1 score on the filtered dataset of Phi-3-mini-4k-Instruct by a factor of 1.1 and for Llama3-8B-Instruct by a factor of 1.02. Accuracy decreases slightly for Llama3-8B-Instruct, but increases for Phi-3-mini-4k-Instruct. On the multiple choice task, where these models have not been fine-tuned for, Phi-3-mini-4k-Instruct gains performance while Llama3-8B-Instruct loses performance. This indicates that generalizing from binary choice to multiple choice is not as easy as the other way around. Intuitively, this makes sense, as going from a more diverse set of answers to a more restricted set is a reduction of answer resolution. In the opposite case, when training on only binary choice answers, the model may lose some of the nuance needed to answer multiple choice questions.

4.6.3. Evaluation on game

With the trained models, we evaluate their performance as answerer in the game. As questioner, we use the untrained models. We use the same evaluation setup as in the original evaluation, however we ran both a multiple choice and a binary choice evaluation. For the multiple choice evaluation, we used the models trained on multiple choice answers and used the multiple choice prompt. We used the single question answerer prompt that does not include full dialogue, as that is what the answerer models were trained on. We perform this evaluation on all three word lists for more robust results. The results are shown in Table 4.11 and Table 4.12.

Model	Train set	Test set	Win (\uparrow)	Turns (\downarrow)	Ts. won (\downarrow)	Yes
Phi-3-mini-4k-Instruct						
Untrained						
		AllenAI TQ	0.14	18.8 (3.6)	11.4 (5.2)	2.2 (1.4)
		Things	0.12	19.1 (2.8)	12.6 (3.9)	3.1 (2.1)
		Dolch	0.16	18.3 (4.5)	9.3 (5.5)	2.6 (1.7)
Multiple choice task						
		AllenAI TQ	0.15	18.6 (3.8)	10.7 (4.6)	1.9 (1.2)
		Things	0.13	19.0 (3.0)	12.2 (4.0)	2.9 (2.0)
		Dolch	0.17	18.2 (4.5)	9.5 (5.3)	2.2 (1.6)
Binary choice task						
		AllenAI TQ	0.15	18.6 (3.8)	10.8 (4.7)	1.9 (1.2)
		Things	0.13	19.0 (3.0)	12.2 (4.0)	3.0 (2.1)
		Dolch	0.16	18.2 (4.5)	9.0 (5.0)	2.3 (1.7)
Llama3-8B-Instruct						
Untrained						
		AllenAI TQ	0.07	19.4 (2.4)	10.7 (2.1)	8.5 (2.5)
		Things	0.03	19.8 (1.4)	12.3 (2.6)	9.2 (2.2)
		Dolch	0.16	18.7 (3.4)	11.9 (4.4)	7.9 (2.6)
Multiple choice task						
		AllenAI TQ	0.21	18.0 (4.4)	10.4 (4.2)	6.8 (2.5)
		Things	0.18	18.7 (3.4)	12.7 (4.5)	7.5 (2.6)
		Dolch	0.29	17.3 (4.6)	10.7 (3.5)	6.3 (2.6)
Binary choice task						
		AllenAI TQ	0.02	19.8 (1.6)	9.0 (2.0)	11.0 (3.1)
		Things	0.01	19.9 (1.1)	9.0 (0.0)	10.7 (2.9)
		Dolch	0.02	19.9 (1.0)	13.0 (2.0)	11.7 (3.5)

Table 4.11.: Results of answerer evaluation in game, using single question answerer prompt and **binary choice yes/no answers in evaluation game**. Shown for model trained on the binary choice task and the multiple choice task.

Model	Train set	Test set	Win (\uparrow)	Turns (\downarrow)	Ts. won (\downarrow)
Phi-3-mini-4k-Instruct					
	Untrained				
		AllenAI TQ	0.10	19.0 (3.3)	9.9 (4.0)
		Things	0.03	19.7 (1.8)	10.3 (3.4)
		Dolch	0.12	18.6 (4.1)	8.6 (4.9)
	Multiple choice task				
		AllenAI TQ	0.10	18.8 (3.6)	8.3 (3.0)
		Things	0.01	19.9 (1.3)	7.0 (0.0)
		Dolch	0.10	18.7 (4.0)	7.2 (3.7)
	Binary choice task				
		AllenAI TQ	0.09	19.0 (3.4)	8.7 (3.2)
		Things	0.09	19.4 (2.3)	12.9 (3.6)
		Dolch	0.11	18.8 (3.9)	8.9 (5.2)
Llama3-8B-Instruct					
	Untrained				
		AllenAI TQ	0.02	19.8 (1.9)	7.5 (5.5)
		Things	0.0	20.0 (0.0)	N/A
		Dolch	0.03	19.7 (2.2)	9.0 (6.2)
	Multiple choice task				
		AllenAI TQ	0.18	18.6 (3.4)	12.3 (4.2)
		Things	0.16	19.3 (1.9)	15.5 (2.6)
		Dolch	0.28	18.0 (4.2)	12.7 (4.9)
	Binary choice task				
		AllenAI TQ	0.02	19.7 (2.2)	4.5 (2.5)
		Things	0.01	19.9 (1.2)	8.0 (0.0)
		Dolch	0.02	19.8 (1.9)	8.0 (6.0)

Table 4.12.: Results of answerer evaluation in game, using single question answerer prompt and **multiple choice (TQ options) answers in evaluation game**. Shown for model trained on the binary choice task and the multiple choice task. (This table does not contain the count of Yes-answers as the TQ options do not contain a Yes-answer.)

Phi-3-mini-4k-Instruct seems to be only slightly affected by training. Its scores increase in the binary choice evaluation but decrease in the multiple choice evaluation. Both changes are only minor. Llama3-8B-Instruct, on the other hand, is affected much more by training. The performance of the model trained on the multiple choice task strongly outperforms the untrained model, especially in the multiple choice evaluation. This suggests that the model has learned to use the six answer options better. In the binary evaluations, Llama3-8B-Instruct performs worse after training in the case of the binary choice task and similarly bad in the case of the multiple choice task. The poor performance can be explained by looking at the dialogues: Llama3-8B-Instruct opens the game by asking for a living thing, which the answerer too often wrongly answers with “Yes”, which derails the game by not finding objects or concepts. Another change in the successful Llama3-8B-Instruct model is that the length of the won games increases, which may also be due to more complex concepts being found after longer dialogues. It is important to note that the result of this evaluation is also limited by the quality of the questioner, which in this case is the untrained model. We perform joint evaluation later in section 4.8.

4.7. Fine-tuning the questioner

We are testing two different training modalities for the questioner: supervised fine-tuning and reinforcement learning. For supervised fine-tuning, we first need to collect a dataset of meaningful game dialogues. We don’t just want to train the questioner on single questions, but on full dialogues, to give examples on good questions in context that optimally reduce uncertainty. Otherwise, the model may just ask good questions, but not in the right order or in a way that is helpful. We have test three different approaches to dialogue generation: taxonomy-based, sourced from a different algorithm and sourced from a bigger model to train a smaller one. In the following, we describe the data sources, training setups and evaluations for each approach.

4.7.1. Supervised fine-tuning with taxonomy from WordNet

WordNet, as described in subsection 2.3.2, is a large lexical database of English. We are interested in the hypernym relations, which form a taxonomy of concepts. This taxonomy can be used to generate synthetic game dialogue. Taking all concepts from the train set of the AllenAI TQ dataset, we first filter for words that are tagged as nouns in WordNet. For each concept, we follow the hypernym relations to the root concept, which is “entity”. We collect all hypernyms of all concepts and filter them for concepts that would be known to a five-year-old by using an LLM as a judge. This filtering is important, as otherwise the taxonomy contains very abstract classes like “causal agent” that would not occur in a game of Twenty Questions. The prompt for the judge can be seen in Figure A.15. Similarly to the decoding of the answerer, we are only probing the probability of the tokens for “Yes” and “No” to determine the answer. We ran the prompt on Phi-3-mini-4k-Instruct and Llama3-8B-Instruct and selected the concepts that were estimated simple by both models. From the list of remaining hypernyms, we build a graph by starting at the root node and then recursively splitting the child nodes in two sets. Given a set of concepts, we check

4. Experiments and results

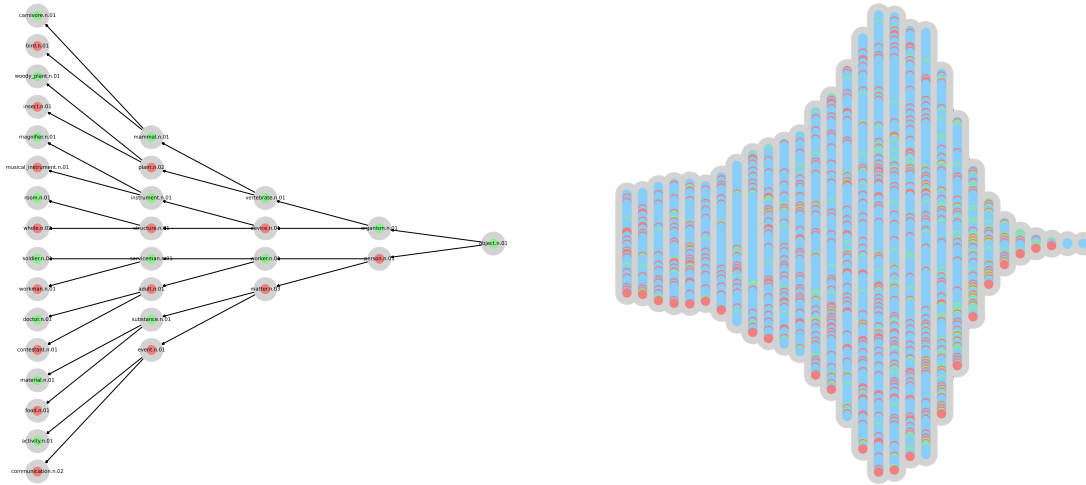


Figure 4.5.: WordNet taxonomy graph. Left: First 5 layers, green and red color indicates whether the node is in the category of the parent node or not. Right: First 30 layers, blue color is for concepts, other colors are the same as on the left.

for all its child nodes, which of the nodes is parent to half the number of concepts. This way, we split the tree as evenly as possible. This process is repeated until we reach a given depth. If we limit the depth to 20, the constructed graph contains 2956 of the 6476 concepts (45.6%) from the AllenAI TQ train dataset. This means that a guesser based on this filtered taxonomy could only find the correct concept in 45.6% of the cases. The concepts from the test set are not in the graph, so the model has to generalize to unseen concepts. Although the game usually ends after 20 questions, we set the depth to 30 to contain more concepts. With a depth of 30, the constructed graph contains 3836 of the 6476 concepts (59.2%) from the AllenAI TQ train dataset.

The first 5 layers of the graph are shown in the left of Figure 4.5. Colors of nodes indicate whether the node is in the category of the parent node or not. For example, the root node asks whether a concept is an object or not. If yes, the next question is whether it is an organism. If not, the next question is whether it is a person. We can see from this example that a hard taxonomy like WordNet may contain unintuitive relations, like considering an organism an object. When looking at 30 layers, the graph has a different form, shown in Figure 4.5 on the right. The first six layers of the graph show a binary structure where every node has two child nodes. After the sixth layer, nodes have less than two child nodes on average and the graph resembles an urn. We also color the concepts from the AllenAI TQ train dataset in blue, all other concepts are added to build the tree structure. We see that most concepts in the tree, also the intermediate concepts, are part of the AllenAI TQ train dataset.

The concepts from the graph are used to generate dialogues. All dialogues are considered won, the average round count is 16.1 (standard deviation of 5.8) and the average number of “Yes” tokens is 3.3 (standard deviation of 1.3). We run supervised fine-tuning on the generated dialogues with Phi-3-mini-4k-Instruct and Llama3-8B-Instruct for 2 and 4 epochs.

Afterwards, we run our usual evaluation on the test set of the AllenAI TQ dataset. The results are shown in Table 4.13.

Model	Train time	Test set	Win (\uparrow)	Turns (\downarrow)	Ts. won (\downarrow)	Yes
Phi-3-mini-4k-instruct						
Untrained						
		AllenAI TQ	0.09	19.2 (2.8)	11.0 (3.7)	3.4 (2.2)
		Things	0.09	19.4 (2.3)	13.2 (3.8)	3.2 (2.1)
		Dolch	0.10	18.8 (4.0)	7.6 (4.3)	4.0 (2.3)
2 epochs						
		AllenAI TQ	0.13	19.0 (3.1)	12.5 (5.2)	3.3 (1.8)
		Things	0.07	19.4 (2.4)	11.3 (2.9)	3.0 (1.0)
		Dolch	0.16	18.0 (4.9)	7.7 (5.0)	3.3 (1.9)
4 epochs						
		AllenAI TQ	0.10	19.3 (2.4)	13.3 (4.0)	3.6 (1.9)
		Things	0.09	19.3 (2.4)	12.4 (3.2)	3.5 (1.7)
		Dolch	0.09	18.8 (4.0)	7.1 (4.9)	3.4 (2.0)
Llama-3-8B-Instruct						
Untrained						
		AllenAI TQ	0.10	18.9 (3.4)	9.0 (2.9)	13.4 (3.7)
		Things	0.09	19.2 (2.6)	11.4 (2.6)	13.0 (3.1)
		Dolch	0.17	18.1 (4.4)	9.0 (3.6)	13.3 (4.3)
2 epochs						
		AllenAI TQ	0.08	19.3 (2.6)	11.5 (4.3)	12.0 (3.6)
		Things	0.09	19.0 (3.4)	8.6 (2.5)	11.2 (3.9)
		Dolch	0.17	18.0 (4.6)	8.1 (2.3)	11.1 (4.7)
4 epochs						
		AllenAI TQ	0.10	18.9 (3.7)	8.7 (4.7)	11.5 (4.0)
		Things	0.11	18.8 (3.6)	9.3 (3.6)	11.5 (3.9)
		Dolch	0.23	17.3 (5.2)	8.4 (3.7)	10.4 (4.8)

Table 4.13.: Results of training with dialogues generated using WordNet taxonomy.

We see inconclusive results. While Phi-3-mini-4k-instruct gains slight performance after 2 epochs, original performance is reached after 4 epochs. Interestingly, the first question that both untrained models ask (“Is it a living thing?”) is unchanged from the untrained model, even though the first question in the dataset is “Is it an object?”. In the case of Llama-3-8B-Instruct, performance is slightly worsened from the untrained model after 2 epochs and slightly improved in comparison to the untrained model after 4 epochs. At the same time, the number of “Yes” answers decreases to a value that is closer to the trained dataset. This may indicate that the model is learning to ask better questions, but the performance is not improving, which may be due to the untrained answerer. As the

training has not strongly reduced performance, the generated dialogues are not of zero quality, but our models may have had the same performance without training to begin with.

4.7.2. Supervised fine-tuning with data from 20Q dialogues

The second approach to generating dialogue data is to use the “20Q” game by Burgener (see subsection 2.2.2) to collect dialogues. As word list, we used AllenAI TQ train concepts, sorted by frequency in the AllenAI TQ dataset. We collected 1073 dialogues from the website⁸ with a simple script that allows Phi-3-mini-4k-Instruct playing as answerer to answer questions generated by “20Q”. The setup allows the model to run inside the HPC cluster using a middle server that runs a web browser via Selenium⁹ and the 20Q website, as seen in Figure 4.6. We made sure to employ rate limiting to not overload the website and to conform to terms of use.

We let the game run for a maximum of 35 rounds instead of the usual 20, to be able to collect more won rounds. We were surprised to find that for our word list, the algorithm only wins 11 of the 1073 games in 20 rounds (1%), but 114 in 35 rounds (10.6%). For the won games, the average number of rounds is 27.5 rounds. We consider these dialogues to not be of high quality, but we still test their use for training. Results are shown in Table 4.14.

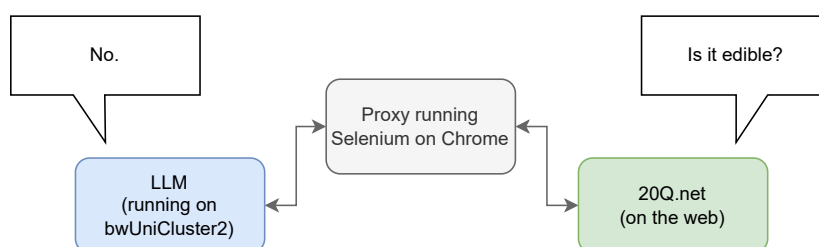


Figure 4.6.: Setup for allowing LLM to play as answerer with 20Q as questioner to collect dialogues.

⁸<http://20q.net> (last accessed on 12/09/2024)

⁹<https://www.selenium.dev/> (last accessed on 12/09/2024)

Model	Train time	Test set	Win (\uparrow)	Turns (\downarrow)	Ts. won (\downarrow)	Yes
Phi-3-mini-4k-instruct						
Untrained						
		AllenAI TQ	0.09	19.2 (2.8)	11.0 (3.7)	3.4 (2.2)
		Things	0.09	19.4 (2.3)	13.2 (3.8)	3.2 (2.1)
		Dolch	0.10	18.8 (4.0)	7.6 (4.3)	4.0 (2.3)
2 epochs						
		AllenAI TQ	0.11	19.0 (3.1)	11.3 (4.7)	3.1 (1.9)
		Things	0.02	19.8 (1.4)	10.5 (2.5)	4.0 (2.8)
		Dolch	0.13	18.5 (4.2)	8.5 (4.3)	3.0 (2.0)
4 epochs						
		AllenAI TQ	0.08	19.3 (2.8)	10.9 (4.6)	4.2 (1.4)
		Things	0.03	19.7 (1.9)	9.0 (0.8)	4.5 (2.3)
		Dolch	0.12	18.6 (4.2)	8.1 (4.9)	4.0 (1.4)
Llama-3-8B-Instruct						
Untrained						
		AllenAI TQ	0.10	18.9 (3.4)	9.0 (2.9)	13.4 (3.7)
		Things	0.09	19.2 (2.6)	11.4 (2.6)	13.0 (3.1)
		Dolch	0.17	18.1 (4.4)	9.0 (3.6)	13.3 (4.3)
2 epochs						
		AllenAI TQ	0.04	19.7 (1.9)	12.0 (4.9)	13.1 (2.9)
		Things	0.04	19.9 (0.8)	16.8 (2.3)	13.3 (2.5)
		Dolch	0.05	19.6 (2.0)	12.8 (5.4)	13.9 (3.2)
4 epochs						
		AllenAI TQ	0.02	19.8 (1.3)	10.5 (1.5)	13.9 (2.9)
		Things	0.03	19.8 (1.3)	13.7 (4.6)	13.4 (2.7)
		Dolch	0.03	19.8 (1.5)	13.3 (5.9)	14.0 (2.8)

Table 4.14.: Results of training with collected 20Q dialogues.

We see that overall, the performance of the models, especially of Llama-3-8B-Instruct is reduced by training on the collected dialogues, in a much stronger way than with the WordNet taxonomy. Phi-3-mini-4k-Instruct loses the most performance on the Things dataset. The reduce in performance is likely due to the low quality of training data, which causes the model to learn worse strategies than what it does untrained. Looking at the dialogues, we find that at the beginning of the dialogues, Llama-3-8B-Instruct is asking the questions from the dataset: “Is it an animal?”, “Is it a vegetable?” Phi-3-mini-4k-Instruct is still unchanged in the first question “Is it a living thing”, but the questions that appeared early in the training games eventually occur later in the dialogues. Overall, learning from the 20Q game is not successful, which is not surprising given that the dataset’s performance on a 20 questions game would be 1%.

4.7.3. Supervised fine-tuning with model distillation

Our third approach for generating dialogue data is to use a bigger model to generate dialogue to train a smaller model, a process known as distillation. As we see promising performance in the untrained Mixtral-8x7B-Instruct-v0.1 (see Table 4.2), we initially attempt to generate dialogue with this model, using the train set of the AllenAI TQ concepts. However, Mixtral-8x7B-Instruct-v0.1 only wins 12.9% of games from the first 303 examples of the training TQ concepts. As a separate source of model generated dialogue, we use the filtered GPT-4o TQ dataset with model-generated dialogues from the Kaggle competition. This dataset contains 9191 dialogues, of which 1634 are won games.

Training setup We adapt the dialogues to our usual prompt format and train Phi-3-mini-4k-Instruct and Llama-3-8B-Instruct on the generated dialogues using the same training setup as for the other questioner datasets. We train both on only the set of won dialogues and on all dialogues. We train all dialogues for 2 epochs (evaluating at 0.5, 1 and 2 epochs) and the won dialogues for 8 epochs (evaluating at 2, 4 and 8 epochs), as it is a much smaller dataset (by a factor of 5.6). Our evaluation is shown on the following pages in Table 4.15 for Phi-3-mini-4k and Table 4.16 for Llama-3-8B-Instruct.

Results We find that Phi-3-mini-4k-Instruct gains performance (from an average of 0.09 to 0.15) after training for 2 epochs on all dialogues, but also slightly improves after 8 epochs on won dialogues (from 0.09 to 0.12) Llama-3-8B-Instruct first gains, then loses performance (from an average of 0.9 to 0.8) after training for 2 epochs on all dialogues. On won datasets, the Llama-3-8B-Instruct gains slight performance after 4 epochs, which stays roughly constant after 8 epochs (from 0.9 to 0.11). Looking at the dialogues, we see no obvious patterns or flaws. Because this is the most promising result so far, we perform joint evaluation with a trained answerer in section 4.8.

Model	Train set/time	Test set	Win (\uparrow)	Turns (\downarrow)	Ts. won (\downarrow)	Yes
Phi-3-mini-4k-instruct						
Untrained						
		AllenAI TQ	0.09	19.2 (2.8)	11.0 (3.7)	3.4 (2.2)
		Things	0.09	19.4 (2.3)	13.2 (3.8)	3.2 (2.1)
		Dolch	0.10	18.8 (4.0)	7.6 (4.3)	4.0 (2.3)
All dialogues						
0.5 epochs						
		AllenAI TQ	0.10	18.9 (3.6)	9.2 (4.9)	2.7 (1.9)
		Things	0.12	19.1 (2.8)	12.6 (4.1)	3.2 (2.5)
		Dolch	0.15	18.2 (4.8)	7.9 (5.1)	3.4 (2.0)
1 epoch						
		AllenAI TQ	0.11	19.0 (3.1)	11.2 (4.2)	5.4 (2.5)
		Things	0.08	19.5 (2.0)	13.6 (3.2)	5.6 (2.2)
		Dolch	0.20	18.1 (4.4)	10.4 (5.0)	5.4 (3.1)
2 epochs						
		AllenAI TQ	0.14	18.7 (3.6)	11.0 (4.7)	7.0 (3.1)
		Things	0.13	19.3 (2.4)	14.3 (4.0)	7.2 (3.0)
		Dolch	0.19	18.2 (4.3)	10.8 (5.2)	6.2 (2.7)
Won dialogues						
2 epochs						
		AllenAI TQ	0.10	19.1 (3.1)	10.7 (4.1)	2.7 (1.1)
		Things	0.05	19.7 (1.6)	14.0 (4.0)	3.7 (3.0)
		Dolch	0.16	18.6 (3.9)	11.4 (5.9)	3.3 (2.1)
4 epoch						
		AllenAI TQ	0.08	19.4 (2.9)	11.9 (6.6)	3.1 (2.2)
		Things	0.07	19.5 (1.9)	13.4 (3.1)	4.1 (2.9)
		Dolch	0.13	18.7 (3.8)	9.8 (4.8)	3.2 (3.0)
8 epochs						
		AllenAI TQ	0.08	19.2 (3.1)	9.5 (4.3)	5.0 (2.3)
		Things	0.12	19.2 (2.4)	13.0 (2.2)	5.4 (2.2)
		Dolch	0.17	18.2 (4.4)	9.6 (4.8)	4.2 (2.0)

Table 4.15.: Results of fine-tuning **Phi-3-mini-4k-Instruct** using dialogue generated by GPT-4o.

4. Experiments and results

Model	Train set/time	Test set	Win (\uparrow)	Turns (\downarrow)	Ts. won (\downarrow)	Yes
Llama-3-8B-Instruct						
Untrained						
		AllenAI TQ	0.10	18.9 (3.4)	9.0 (2.9)	13.4 (3.7)
		Things	0.09	19.2 (2.6)	11.4 (2.6)	13.0 (3.1)
		Dolch	0.17	18.1 (4.4)	9.0 (3.6)	13.3 (4.3)
All dialogues						
0.5 epochs						
		AllenAI TQ	0.15	18.5 (3.9)	9.9 (3.8)	12.2 (4.2)
		Things	0.08	19.3 (2.5)	11.2 (3.1)	12.8 (3.4)
		Dolch	0.24	17.2 (5.3)	8.1 (3.1)	12.1 (5.2)
1 epoch						
		AllenAI TQ	0.07	19.5 (2.0)	13.0 (3.0)	13.2 (2.8)
		Things	0.13	18.5 (3.9)	8.8 (2.9)	12.7 (4.1)
		Dolch	0.10	19.0 (3.3)	9.5 (3.4)	12.7 (3.8)
2 epochs						
		AllenAI TQ	0.06	19.3 (2.8)	8.3 (1.6)	14.4 (3.7)
		Things	0.06	19.5 (2.1)	11.8 (3.1)	13.7 (3.4)
		Dolch	0.13	18.5 (4.0)	8.5 (3.0)	13.8 (4.6)
Won dialogues						
2 epochs						
		AllenAI TQ	0.10	19.1 (3.1)	10.7 (4.1)	2.7 (1.1)
		Things	0.10	19.3 (2.3)	13.1 (3.4)	12.8 (3.1)
		Dolch	0.16	18.6 (3.9)	11.4 (5.9)	3.3 (2.1)
4 epochs						
		AllenAI TQ	0.12	18.7 (3.7)	8.8 (2.1)	12.8 (4.1)
		Things	0.07	19.3 (2.5)	10.6 (2.8)	12.7 (3.5)
		Dolch	0.14	18.5 (4.1)	9.1 (4.4)	13.2 (4.2)
8 epochs						
		AllenAI TQ	0.12	18.8 (3.4)	10.0 (3.1)	13.4 (3.8)
		Things	0.07	19.4 (2.3)	11.9 (3.4)	13.1 (3.4)
		Dolch	0.10	18.8 (3.7)	8.1 (2.8)	13.7 (3.9)

Table 4.16.: Results of fine-tuning **Llama-3-8B-Instruct** using dialogue generated by GPT-4o.

4.7.4. Reinforcement learning

We set up a reinforcement learning environment for the questioner to play the Twenty Questions game using the PPOTrainer from the trl library¹⁰. Our reward modeling for PPO [43] is based on the game score: If the game is won, the reward is proportional to the count of rounds, the shortest possible game scoring a reward of 3 and the longest possible game scoring a reward of 1. If the game is lost, the reward is -1 . The version of PPO we are using requires all activations for all dialogues in a minibatch to be stored in memory, so we can only use a minibatch size of 16 and 1 gradient accumulation step. We are training with a learning rate of 1.0×10^{-5} . Because of the high memory usage, we are only able to train Phi-3-mini-4k-Instruct, which still requires 4 GPUs to train. For both the untrained and fine-tuned model, we find that because of the low initial win rate and the low batch size, there is not enough won games that the model can learn from using the reward signal. The reward signal is mostly -1 except for few won games. After 25 steps, loss begins to increase and the score begins to decrease, as seen in Figure 4.7. There

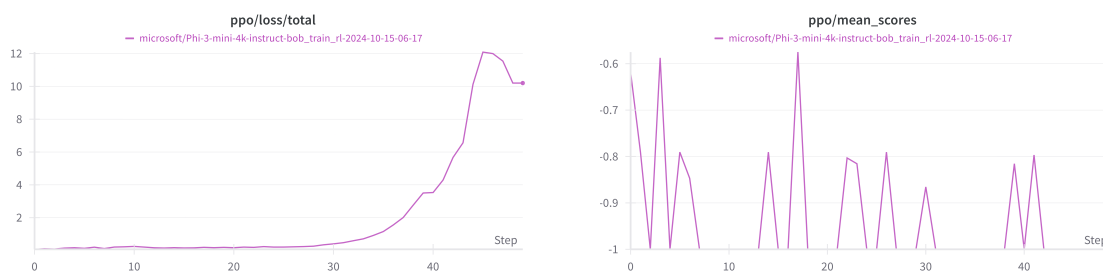


Figure 4.7.: Reinforcement learning training of untrained Phi-3-mini-4k-Instruct. Left: Total Loss over training steps. Right: Mean score over training steps.

is a multitude of possible reasons why the reinforcement learning setup does not work as expected. The most likely reason is that the model is not able to learn from the reward signal, as the reward signal is too sparse and too random. Another reason could be that our reward modeling was not well suited for the PPO algorithm. Zhang et al. [55] successfully uses the trlx library¹¹ which we also attempted to use instead without success. We were not able to perform tests within the time frame of this thesis.

4.8. Joint evaluation of trained questioner and answerer

So far, we have only evaluated a trained answerer with an untrained questioner and vice versa. We are interested in the performance of a trained questioner and answerer together. For this reason, we evaluate the answerer fine-tuned on AllenAI TQ and the questioner trained with model distillation. As we can't evaluate every combination of questioner and answerer, we choose the best-performing questioner and answerer for each task. We evaluate this for the original yes/no-task and using the single question answerer prompt (as that is what the answerer models are fine-tuned for). Our results are shown in Table 4.17.

¹⁰https://huggingface.co/docs/trl/ppo_trainer (last accessed on 12/09/2024)

¹¹<https://github.com/CarperAI/trlx> (last accessed on 12/09/2024)

4. Experiments and results

Model	Ans.	Ques.	Test set	Win (\uparrow)	Turns (\downarrow)	Ts. won (\downarrow)	Yes
Phi-3-mini-4k-Instruct							
Binary choice Task							
All GPT-4o dialogues							
	AllenAI TQ	0.14		19.0 (2.9)	13.2 (4.4)	3.0 (2.8)	
	Things	0.18		18.9 (2.9)	13.7 (3.9)	4.5 (3.2)	
	Dolch	0.3		17.5 (4.7)	11.6 (4.9)	3.2 (2.2)	
Won GPT-4o dialogues							
	AllenAI TQ	0.23		18.2 (4.0)	12.1 (4.6)	5.6 (3.2)	
	Things	0.2		18.9 (2.8)	14.6 (3.9)	5.9 (3.0)	
	Dolch	0.29		17.5 (4.5)	11.5 (4.4)	4.7 (2.6)	
Multiple choice task							
All GPT-4o dialogues							
	AllenAI TQ	0.16		18.9 (3.0)	13.2 (4.4)	2.8 (2.4)	
	Things	0.15		19.0 (2.8)	13.3 (3.6)	4.4 (3.1)	
	Dolch	0.31		17.5 (4.7)	11.8 (5.0)	3.2 (2.5)	
Won GPT-4o dialogues							
	AllenAI TQ	0.23		18.2 (4.0)	12.3 (4.7)	5.4 (3.2)	
	Things	0.19		19.0 (2.7)	14.8 (4.1)	5.9 (3.1)	
	Dolch	0.31		17.5 (4.5)	12.1 (4.6)	4.6 (2.3)	
Llama-3-8B-Instruct							
Binary choice task							
All GPT-4o dialogues							
	AllenAI TQ	0.05		19.5 (2.4)	9.4 (3.5)	11.0 (4.1)	
	Things	0.01		19.9 (1.0)	10.0 (0.0)	10.9 (4.1)	
	Dolch	0.07		19.3 (2.9)	9.6 (4.1)	11.2 (4.8)	
Won GPT-4o dialogues							
	AllenAI TQ	0.02		19.9 (0.4)	17.0 (1.0)	10.2 (4.2)	
	Things	0.03		19.8 (1.4)	12.3 (2.6)	10.2 (3.9)	
	Dolch	0.09		19.2 (2.9)	10.8 (4.1)	10.3 (4.6)	
Multiple choice task							
All GPT-4o dialogues							
	AllenAI TQ	0.26		17.8 (4.3)	11.7 (4.7)	6.6 (3.2)	
	Things	0.2		18.7 (3.3)	13.4 (4.3)	7.1 (3.0)	
	Dolch	0.36		16.9 (4.9)	11.3 (4.4)	6.8 (3.3)	
Won GPT-4o dialogues							
	AllenAI TQ	0.25		18.3 (3.8)	13.2 (4.7)	5.9 (3.0)	
	Things	0.18		18.8 (2.8)	13.6 (3.3)	6.7 (2.1)	
	Dolch	0.33		17.1 (4.9)	11.1 (4.3)	5.4 (2.6)	

Table 4.17.: Joint evaluation of trained questioner (Ques.) and answerer (Ans.) using the binary task.

We find that the best performing setup for the trained Phi-3-mini-4k-Instruct is the answerer trained on the multiple choice task with the questioner fine-tuned on the won dialogues which reaches an average win rate of 0.243. The binary choice task is very close however, with an average win rate of 0.240. For Llama-3-8B-Instruct, the answerer trained on the binary choice task still causes a low win rate due to errors in the first question which are not recoverable. However, the Llama-3-8B-Instruct answerer trained on multiple choice task with the questioner fine-tuned on all dialogues reaches an average win rate of 0.273. This evaluation has yielded the highest scores we've seen so far, especially on the Dolch word list. In comparison to Zhang et al. [55], who found a human win rate of 24% on the Things dataset, our models can both reach 20% at maximum on the same dataset. In comparison to the untrained models running the single question answerer prompt on the AllenAI TQ questions (see Table 4.6), Phi-3-mini-4k-Instruct improves by a factor of 1.6 (from 0.14 to 0.23), while Llama-3-8B-Instruct improves by a factor of 3.7 (from 0.07 to 0.26, note that the untrained Llama-3-8B-Instruct performed worse using this prompt template in comparison to the original evaluation). Comparing the scores from the joint evaluation to the untrained models running the original evaluation, where the models attained an average win rate of 0.067 for Phi-3-mini-4k-Instruct and 0.130 for Llama3-8B-Instruct (see Table 4.3), the trained models improve by a factor of around 3.6 and 2.1, respectively. This shows that the game score is strongly dependent on both the questioner and the answerer, and that training both models can lead to a significant improvement in win rate.

5. Conclusion

In this chapter, we summarize our findings and answer the research questions. We also discuss what limitations apply to our work and suggest directions for future work.

5.1. Answering Research Questions

In this work, we set out to answer three research questions about the ability of foundation language models to play language-based guessing games. In this section, we summarize our findings and answer the research questions.

RQ1: Can language-based guessing games be played by foundation language models?

In this work, we find that foundation language models can play Twenty Questions to some extent, but that it is a challenging task, especially for smaller models. In the role of the answerer, foundation models like Llama3-8B-Instruct and Phi-3-mini-4k-Instruct (that have not been fine-tuned for question answering specifically) reach an F1 score of almost 0.8 on the AllenAI TQ dataset in the binary choice task. In the role of the questioner, these models can also generate questions, but the game scores are relatively low with around 10% success rate for smaller models like Phi-3-mini-4k-Instruct and Llama3-8B-Instruct. Larger instruction-tuned models like Llama3-70B-Instruct perform better at around 23%.

RQ2: What factors are relevant for game performance? How can they be measured?

We find that model size is correlated with both individual answerer performance and game performance, with larger models performing better. Also, we find that the prompt is a sensitive factor for game performance. In our tests, a zero-shot prompt with binary-choice questions perform best. Another important factor is the answerer accuracy, as a single wrong answer can lead to the game being lost. The domain and difficulty of the words in the game are also decisive factors for the models' performance, with simpler words performing better. Constraining the domain by adding additional input like visual input or the first letter of the word make the game much easier. The "I spy" game can be played by the unmodified small multimodal Phi-3-vision-128k-Instruct model at a win rate of 26%. With these constraints, the model's strategy is no longer trying to reduce uncertainty in the game, but to simply list possible guesses, which is an effective strategy.

RQ3: Can the model's playing performance be improved by fine-tuning?

We find that supervised fine-tuning of the both questioner and answerer can improve the model's performance, but the quality of the data is crucial. Training the answerer

on the AllenAI TQ dataset improves the model’s answering accuracy. This accuracy is not directly correlated with the game performance (which depends on questioner and answerer), but it is a necessary condition for the game to be played. The performance on the game especially benefits from answerer training if there are multiple answer options. Generating dialogues from a strict taxonomy like WordNet shows inconclusive results, but is not detrimental to performance. Training the questioner on collected dialogues from 20Q does not improve performance because of the low quality of the data. Model distillation using GPT-4o-generated dialogues is the best performing fine-tuning method for the questioner, which is why we use it in joint evaluation. In joint evaluation of both trained models, we find that our models can both reach much higher win rates. On the things dataset, both reach 20%, in comparison to proposed human win rates of 24% [55]. Averaging the results for the three word lists, the best score of Phi-3-mini-4k-Instruct is an average win rate of 24.3%, while Llama-3-8B-Instruct scores 27.3%.

5.2. Limitations and Future Work

The main limitation of our work is the computational resources available, that limits what experiments can be done. With more resources, hyperparameter optimization could have been done to improve results. Furthermore, larger models could have been trained which may be more capable. Another limitation is data quality, which is the main bottleneck for supervised fine-tuning. We did not have access to human-generated dialogues for supervised fine-tuning, which could have improved the results.

Other possible changes to the game include game length, as the taxonomy based approach and the 20Q collected data shows that a game that is continued after 20 questions can cause more games to be won. It would be interesting to see how the models perform in a longer game. Concerning the answerer, it would be interesting to compare training with AllenAI TQ dataset to training with other QA datasets like BoolQ [8] and whether this improves the model’s playing performance. Concerning the training of the questioner, continuing on working on reinforcement learning could be an interesting avenue for future work, as others have shown that it can improve model performance without the need for human-generated data [55]. Future work could also include more experiments with different games, such as Taboo (which also appears in BIG-bench¹).

Finally, development of LLMs is a very active field and new models are released frequently. During the course of this work, updated versions of the models we evaluate have been released (Phi3.5 was released in August 2024² with a new Mixture-of-Experts variant, Llama was updated to version 3.1 in July³ and 3.2 in September⁴), which may reach better performance in the game.

¹https://github.com/google/BIG-bench/tree/main/bigbench/benchmark_tasks/taboo (last accessed on 12/09/2024)

²<https://techcommunity.microsoft.com/blog/azure-ai-services-blog/discover-the-new-multi-lingual-high-quality-phi-3-5-slm/4225280> (last accessed on 12/09/2024)

³<https://ai.meta.com/blog/meta-llama-3-1/> (last accessed on 12/09/2024)

⁴<https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/> (last accessed on 12/09/2024)

Bibliography

- [1] Marah I. Abdin et al. “Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone”. In: *CoRR* abs/2404.14219 (2024). DOI: 10.48550/ARXIV.2404.14219. arXiv: 2404.14219. URL: <https://doi.org/10.48550/arXiv.2404.14219> (visited on 12/09/2024).
- [2] Joshua Ainslie et al. “GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints”. In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Association for Computational Linguistics, 2023, pp. 4895–4901. DOI: 10.18653/v1/2023.EMNLP-MAIN.298. URL: <https://doi.org/10.18653/v1/2023.emnlp-main.298> (visited on 12/09/2024).
- [3] Yoshua Bengio et al. “A Neural Probabilistic Language Model”. In: *J. Mach. Learn. Res.* 3 (2003), pp. 1137–1155. URL: <https://jmlr.org/papers/v3/bengio03a.html> (visited on 12/10/2024).
- [4] Tom B. Brown et al. “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*. Ed. by Hugo Larochelle et al. 2020. URL: <https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfc4967418bfb8ac142f64a-Abstract.html> (visited on 12/09/2024).
- [5] Maxime De Bruyn et al. “Is It Smaller Than a Tennis Ball? Language Models Play the Game of Twenty Questions”. In: *Proceedings of the Fifth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2022, Abu Dhabi, United Arab Emirates (Hybrid), December 8, 2022*. Ed. by Jasmijn Bastings et al. Association for Computational Linguistics, 2022, pp. 80–90. DOI: 10.18653/v1/2022.BLACKBOXNLP-1.7. URL: <https://doi.org/10.18653/v1/2022.blackboxnlp-1.7> (visited on 12/09/2024).
- [6] Yihong Chen et al. “Learning-to-Ask: Knowledge Acquisition via 20 Questions”. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*. Ed. by Yike Guo and Faisal Farooq. ACM, 2018, pp. 1216–1225. DOI: 10.1145/3219819.3220047.
- [7] Junyoung Chung et al. “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling”. In: *CoRR* abs/1412.3555 (2014). arXiv: 1412.3555. URL: <http://arxiv.org/abs/1412.3555> (visited on 12/09/2024).

- [8] Christopher Clark et al. “BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Association for Computational Linguistics, 2019, pp. 2924–2936. DOI: 10.18653/V1/N19-1300. URL: <https://doi.org/10.18653/v1/n19-1300>.
- [9] Tri Dao. “FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning”. In: *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL: <https://openreview.net/forum?id=mZn2Xyh9Ec> (visited on 12/09/2024).
- [10] Tri Dao et al. “FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness”. In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. Ed. by Sanmi Koyejo et al. 2022. URL: https://proceedings.neurips.cc/paper_files/paper/2022/hash/67d57c32e20fd0a7a302cb81d36e40d5-Abstract-Conference.html (visited on 12/09/2024).
- [11] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/V1/N19-1423. URL: <https://doi.org/10.18653/v1/n19-1423> (visited on 12/10/2024).
- [12] Edward William Dolch. *Problems in Reading*. Garrard Press, 1948.
- [13] Abhimanyu Dubey et al. “The Llama 3 Herd of Models”. In: *CoRR abs/2407.21783* (2024). DOI: 10.48550/ARXIV.2407.21783. arXiv: 2407.21783. URL: <https://doi.org/10.48550/arXiv.2407.21783> (visited on 12/09/2024).
- [14] Suriya Gunasekar et al. “Textbooks Are All You Need”. In: *CoRR abs/2306.11644* (2023). DOI: 10.48550/ARXIV.2306.11644. arXiv: 2306.11644. URL: <https://doi.org/10.48550/arXiv.2306.11644> (visited on 12/09/2024).
- [15] Dan Hendrycks et al. “Measuring Massive Multitask Language Understanding”. In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL: <https://openreview.net/forum?id=d7KBjmI3GmQ> (visited on 12/09/2024).
- [16] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Comput.* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735. URL: <https://doi.org/10.1162/neco.1997.9.8.1735> (visited on 12/09/2024).
- [17] Jordan Hoffmann et al. “Training Compute-Optimal Large Language Models”. In: *CoRR abs/2203.15556* (2022). DOI: 10.48550/ARXIV.2203.15556. arXiv: 2203.15556. URL: <https://doi.org/10.48550/arXiv.2203.15556> (visited on 12/09/2024).

-
- [18] Edward J. Hu et al. “LoRA: Low-Rank Adaptation of Large Language Models”. In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL: <https://openreview.net/forum?id=nZeVKeeFYf9> (visited on 12/09/2024).
- [19] Huang Hu et al. “Playing 20 Question Game with Policy-Based Reinforcement Learning”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*. Ed. by Ellen Riloff et al. Association for Computational Linguistics, 2018, pp. 3233–3242. DOI: 10.18653/V1/D18-1361. URL: <https://doi.org/10.18653/v1/d18-1361> (visited on 12/09/2024).
- [20] Aaron Hurst et al. “GPT-4o System Card”. In: *CoRR* abs/2410.21276 (2024). DOI: 10.48550/ARXIV.2410.21276. arXiv: 2410.21276. URL: <https://doi.org/10.48550/arXiv.2410.21276>.
- [21] Albert Q. Jiang et al. “Mistral 7B”. In: *CoRR* abs/2310.06825 (2023). DOI: 10.48550/ARXIV.2310.06825. arXiv: 2310.06825. URL: <https://doi.org/10.48550/arXiv.2310.06825> (visited on 12/09/2024).
- [22] Albert Q. Jiang et al. “Mixtral of Experts”. In: *CoRR* abs/2401.04088 (2024). DOI: 10.48550/ARXIV.2401.04088. arXiv: 2401.04088. URL: <https://doi.org/10.48550/arXiv.2401.04088> (visited on 12/09/2024).
- [23] Denis Kocetkov et al. “The Stack: 3 TB of Permissively Licensed Source Code”. In: *Trans. Mach. Learn. Res.* 2023 (2023). URL: <https://openreview.net/forum?id=pxpbTdUEpD> (visited on 12/09/2024).
- [24] Ranjay Krishna et al. “Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations”. In: *Int. J. Comput. Vis.* 123.1 (2017), pp. 32–73. DOI: 10.1007/S11263-016-0981-7. URL: <https://doi.org/10.1007/s11263-016-0981-7> (visited on 12/09/2024).
- [25] Yuanzhi Li et al. “Textbooks Are All You Need II: Phi-1.5 Technical Report”. In: *CoRR* abs/2309.05463 (2023). DOI: 10.48550/ARXIV.2309.05463. arXiv: 2309.05463. URL: <https://doi.org/10.48550/arXiv.2309.05463> (visited on 12/09/2024).
- [26] Haotian Liu et al. “Visual Instruction Tuning”. In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh et al. 2023. URL: https://proceedings.neurips.cc/paper_files/paper/2023/hash/6dcf277ea32ce3288914faf369fe6de0-Abstract-Conference.html (visited on 12/09/2024).
- [27] Xiaoyong Liu and W. Bruce Croft. “Statistical Language Modeling for Information Retrieval”. In: *Annu. Rev. Inf. Sci. Technol.* 39.1 (2005), pp. 1–31. DOI: 10.1002/ARIS.1440390108. URL: <https://doi.org/10.1002/aris.1440390108> (visited on 12/10/2024).

- [28] Pan Lu et al. “Learn to Explain: Multimodal Reasoning via Thought Chains for Science Question Answering”. In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. Ed. by Sanmi Koyejo et al. 2022. URL: http://papers.nips.cc/paper_files/paper/2022/hash/11332b6b6cf4485b84afadb1352d3a9a-Abstract-Conference.html.
- [29] Thomas Mesnard et al. “Gemma: Open Models Based on Gemini Research and Technology”. In: *CoRR abs/2403.08295 (2024)*. DOI: 10.48550/ARXIV.2403.08295. arXiv: 2403.08295. URL: <https://doi.org/10.48550/arXiv.2403.08295> (visited on 12/09/2024).
- [30] Tomás Mikolov et al. “Efficient Estimation of Word Representations in Vector Space”. In: *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2013. URL: <http://arxiv.org/abs/1301.3781> (visited on 12/09/2024).
- [31] George A. Miller. “WordNet: A Lexical Database for English”. In: *Commun. ACM* 38.11 (1995), pp. 39–41. DOI: 10.1145/219717.219748.
- [32] OpenAI. “GPT-4 Technical Report”. In: *CoRR abs/2303.08774 (2023)*. DOI: 10.48550/ARXIV.2303.08774. arXiv: 2303.08774. URL: <https://doi.org/10.48550/arXiv.2303.08774> (visited on 12/09/2024).
- [33] Long Ouyang et al. “Training Language Models to Follow Instructions with Human Feedback”. In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. Ed. by Sanmi Koyejo et al. 2022. URL: http://papers.nips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html (visited on 12/09/2024).
- [34] Parth Parikh and Anisha Gupta. “Reversing The Twenty Questions Game”. In: *CoRR abs/2301.08718 (2023)*. DOI: 10.48550/ARXIV.2301.08718. arXiv: 2301.08718. URL: <https://doi.org/10.48550/arXiv.2301.08718> (visited on 12/09/2024).
- [35] Razvan Pascanu, Tomás Mikolov, and Yoshua Bengio. “On the Difficulty of Training Recurrent Neural Networks”. In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*. Vol. 28. JMLR Workshop and Conference Proceedings. JMLR.org, 2013, pp. 1310–1318. URL: <http://proceedings.mlr.press/v28/pascanu13.html> (visited on 12/09/2024).
- [36] Alec Radford et al. “Learning Transferable Visual Models From Natural Language Supervision”. In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 8748–8763. URL: <http://proceedings.mlr.press/v139/radford21a.html> (visited on 12/09/2024).

-
- [37] Rafael Rafailov et al. “Direct Preference Optimization: Your Language Model Is Secretly a Reward Model”. In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh et al. 2023. URL: http://papers.nips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html.
- [38] Colin Raffel et al. “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”. In: *Journal of Machine Learning Research* 21 (2020), 140:1–140:67. URL: <https://jmlr.org/papers/v21/20-074.html>.
- [39] Joseph Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91.
- [40] Victor Sanh et al. “DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter”. In: *CoRR abs/1910.01108* (2019). arXiv: 1910.01108. URL: <http://arxiv.org/abs/1910.01108>.
- [41] Victor Sanh et al. “Multitask Prompted Training Enables Zero-Shot Task Generalization”. In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL: <https://openreview.net/forum?id=9Vrb9D0WI4>.
- [42] Victor Sanh et al. “Multitask Prompted Training Enables Zero-Shot Task Generalization”. In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL: <https://openreview.net/forum?id=9Vrb9D0WI4> (visited on 12/12/2024).
- [43] John Schulman et al. “Proximal Policy Optimization Algorithms”. In: *CoRR abs/1707.06347* (2017). arXiv: 1707.06347. URL: <http://arxiv.org/abs/1707.06347> (visited on 12/09/2024).
- [44] Rico Sennrich, Barry Haddow, and Alexandra Birch. “Neural Machine Translation of Rare Words with Subword Units”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016. DOI: 10.18653/V1/P16-1162. URL: <https://doi.org/10.18653/v1/p16-1162> (visited on 12/09/2024).
- [45] Noam Shazeer. “GLU Variants Improve Transformer”. In: *CoRR abs/2002.05202* (2020). arXiv: 2002.05202. URL: <https://arxiv.org/abs/2002.05202> (visited on 12/09/2024).
- [46] Aarohi Srivastava et al. “Beyond the Imitation Game: Quantifying and Extrapolating the Capabilities of Language Models”. In: *Trans. Mach. Learn. Res.* 2023 (2023). URL: <https://openreview.net/forum?id=uyTL5Bvosj> (visited on 12/09/2024).
- [47] Jianlin Su et al. “RoFormer: Enhanced Transformer with Rotary Position Embedding”. In: *Neurocomputing* 568 (2024), p. 127063. DOI: 10.1016/J.NEUCOM.2023.127063. URL: <https://doi.org/10.1016/j.neucom.2023.127063> (visited on 12/09/2024).

- [48] Hugo Touvron et al. “Llama 2: Open Foundation and Fine-Tuned Chat Models”. In: *CoRR* abs/2307.09288 (2023). DOI: 10.48550/ARXIV.2307.09288. arXiv: 2307.09288. URL: <https://doi.org/10.48550/arXiv.2307.09288> (visited on 12/09/2024).
- [49] Hugo Touvron et al. “LLaMA: Open and Efficient Foundation Language Models”. In: *CoRR* abs/2302.13971 (2023). DOI: 10.48550/ARXIV.2302.13971. arXiv: 2302.13971. URL: <https://doi.org/10.48550/arXiv.2302.13971> (visited on 12/09/2024).
- [50] Ashish Vaswani et al. “Attention Is All You Need”. In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon et al. 2017, pp. 5998–6008. URL: <https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html> (visited on 12/09/2024).
- [51] Jason Wei et al. “Finetuned Language Models Are Zero-Shot Learners”. In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL: <https://openreview.net/forum?id=gEZrGCozdqR> (visited on 12/09/2024).
- [52] Xiang Yue et al. “MMMU: A Massive Multi-Discipline Multimodal Understanding and Reasoning Benchmark for Expert AGI”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*. IEEE, 2024, pp. 9556–9567. DOI: 10.1109/CVPR52733.2024.00913. URL: <https://doi.org/10.1109/CVPR52733.2024.00913>.
- [53] Rowan Zellers et al. “HellaSwag: Can a Machine Really Finish Your Sentence?” In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Ed. by Anna Korhonen, David R. Traum, and Lluís Màrquez. Association for Computational Linguistics, 2019, pp. 4791–4800. DOI: 10.18653/V1/P19-1472. URL: <https://doi.org/10.18653/v1/p19-1472> (visited on 12/09/2024).
- [54] Peiyuan Zhang et al. “TinyLlama: An Open-Source Small Language Model”. In: *CoRR* abs/2401.02385 (2024). DOI: 10.48550/ARXIV.2401.02385. arXiv: 2401.02385. URL: <https://doi.org/10.48550/arXiv.2401.02385> (visited on 12/09/2024).
- [55] Yizhe Zhang, Jiarui Lu, and Navdeep Jaitly. “Probing the Multi-turn Planning Capabilities of LLMs via 20 Question Games”. In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*. Ed. by Lun-Wei Ku, Andre Martins, and Vivek Srikumar. Association for Computational Linguistics, 2024, pp. 1495–1516. DOI: 10.18653/V1/2024.ACL-LONG.82. URL: <https://doi.org/10.18653/v1/2024.acl-long.82> (visited on 12/09/2024).
- [56] Wayne Xin Zhao et al. “A Survey of Large Language Models”. In: *CoRR* abs/2303.18223 (2023). DOI: 10.48550/ARXIV.2303.18223. arXiv: 2303.18223. URL: <https://doi.org/10.48550/arXiv.2303.18223> (visited on 12/09/2024).

-
- [57] Lianmin Zheng et al. “Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena”. In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh et al. 2023. URL: https://proceedings.neurips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html (visited on 12/09/2024).

A. Appendix

A.1. Datasets

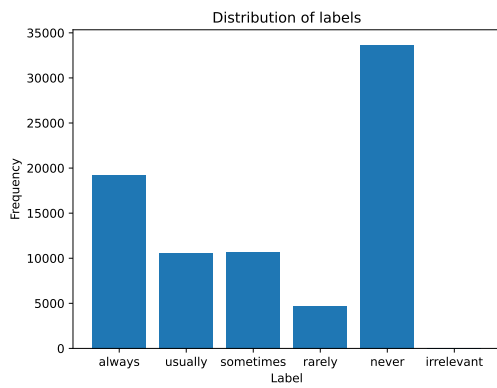


Figure A.1.: Distribution of answer options in unfiltered AllenAI TQ dataset (37,122 entries)

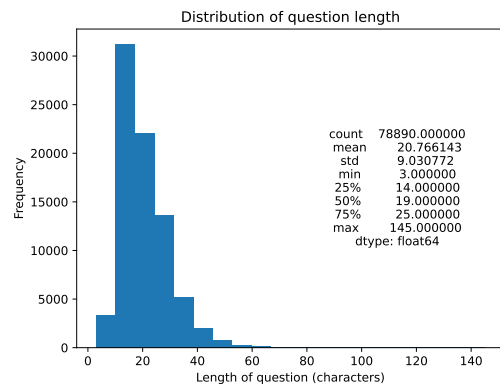


Figure A.2.: Question length distribution of unfiltered AllenAI TQ dataset

A.2. Evaluation

Alice and Bob are playing a game of Twenty Questions with each other. Bob will ask a series of questions, and Alice will respond to each question with a 'Yes' or 'No', until Bob is able to guess the concept that Alice is thinking of.

The concept Alice is thinking of and wants to communicate to Bob is 'car'.

This is Alice and Bob's conversation, starting with Bob's first question.

Bob: Is the concept you're thinking of a living thing?

Alice: No

Bob: Is the concept you're thinking of man-made?

Alice: Yes

Bob: Does the concept you're thinking of have wheels?

Alice: Yes

Bob: Is the concept you're thinking of used for transportation?

Alice: Yes

Bob: Does the concept you're thinking of have a motor?

Alice: Yes

Bob: Is the concept you're thinking of a type of car?

Figure A.3.: Example dialogue from the answerer's perspective for Twenty Questions game in original BIG-bench [46] game. Blue text is what the answerer model generated, everything else is part of the prompt.

Alice and Bob are playing a game of Twenty Questions with each other. Bob will ask a series of questions, and Alice will respond to each question with a 'Yes' or 'No', until Bob is able to guess the concept that Alice is thinking of.

This is Alice and Bob's conversation, starting with Bob's first question.

Bob: Is the concept *you're thinking of a living thing?*
 Alice: No
 Bob: *Is the concept you're thinking of man-made?*
 Alice: Yes
 Bob: *Does the concept you're thinking of have wheels?*
 Alice: Yes
 Bob: *Is the concept you're thinking of used for transportation?*
 Alice: Yes
 Bob: *Does the concept you're thinking of have a motor?*
 Alice: Yes
 Bob: *Is the concept you're thinking of a type of car?*

Figure A.4.: Abbreviated example dialogue from the questioner's perspective for Twenty Questions game in original BIG-bench [46] game. The original dialogue contained ten questions. Teal text is what the questioner model generated, everything else is part of the prompt.

coarse objects: "apple", "television", "dinosaur", "airplane", "house",
 "tree", "coat", "shoes", "car", "train", "shower", "frisbee", "cow"

refined objects: "cosmic crisp apple", "giganotosaurus", "siberian huskey",
 "glass micropipette"

verbs: "jog", "catch", "defenestrate", "eat", "apologize", "operate",
 "pretend"

emotions: "anger", "love", "hate", "contentment", "jealousy", "surprise",
 "disgust", "hopefulness"

abstract concepts: "global poverty", "phase transition", "positive sum game",
 "beauty", "representative democracy", "tall", "mauve", "perspicacious"

Figure A.5.: List of concepts in original Twenty Questions task from BIG-Bench [46], see repository at https://github.com/google/BIG-bench/tree/main/bigbench/benchmark_tasks/twenty_questions (last accessed on 12/09/2024)

```
<|system|>
Alice and Bob are playing a game of Twenty Questions with each other. Bob
    will ask a series of questions, and Alice will respond to each question
    with a 'Yes' or 'No', until Bob is able to guess the concept that Alice
    is thinking of.
```

The concept Alice is thinking of and wants to communicate to Bob is 'apple'.

```
This is Alice and Bob's conversation, starting with Bob's first question.<|
    end|>
```

```
<|user|>
Bob: Is the concept a living thing?<|end|>
<|assistant|>
Alice: No<|end|>
<|user|>
Bob: Is it found on Earth?<|end|>
<|assistant|>
Alice: Yes<|end|>
<|user|>
Bob: Is it man-made?<|end|>
<|assistant|>
Alice: No<|end|>
<|user|>
[...]
Bob: Is it a plant?<|end|>
<|assistant|>
Alice: Yes<|end|>
<|user|>
Bob: Is it a tree?<|end|>
<|assistant|>
Alice: No<|end|>
<|user|>
Bob: Is it a flowering plant?<|end|>
<|assistant|>
Alice: Yes<|end|>
<|user|>
Bob: Is it a fruit-bearing tree?<|end|>
<|assistant|>
Alice: Yes<|end|>
<|user|>
Bob: Is it an apple tree?<|end|>
```

Figure A.6.: Example dialogue from the answerer's perspective for Twenty Questions game in updated game with correct prompt format for Phi-3-mini-4k-instruct. Teal text is what the questioner model generated, blue text is what the answerer model generated, everything else is part of the prompt. Abbreviated.

Overlap between Things concepts / AllenAI TQ concepts:

Apple
Continent
Drink
Guitar
Spider

Overlap between AllenAI TQ concepts / Dolch concepts:

Air
Apple
Car
Cat
Dog
Father
Picture
School
Snow
Tree
Wind

Overlap between Dolch concepts / Things concepts:

Apple
Boat
Egg
Watch

A.3. Experiments

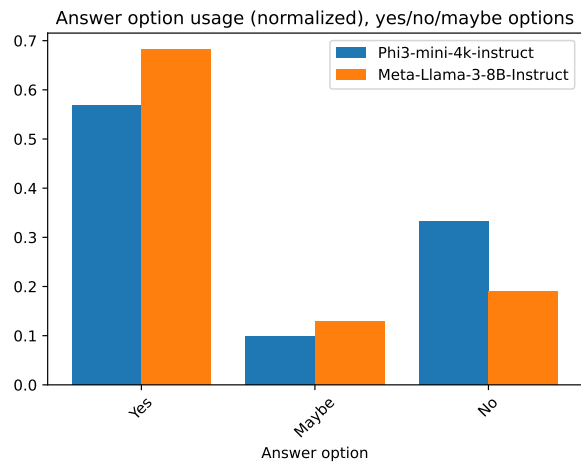


Figure A.7.: Usage of answer options in Twenty Questions task for yes/no/maybe options

```
<|system|>
Alice and Bob are playing a game of Twenty Questions with each other. Bob
    will ask a series of questions, and Alice will respond to each question
    with Yes, No, Unknown, Irrelevant, Sometimes, Maybe, Probably, Doubtful,
    Usually, Depends, Rarely or Partly, until Bob is able to guess the
    concept that Alice is thinking of.

Here is the first of two example dialogues:

---
Bob: Is it a living thing?
Alice: Yes
Bob: Is it an Animal?
Alice: Yes
Bob: Is it small?
Alice: No
[...]
Bob: Does it live in large populations?
Alice: Yes
Bob: I am guessing that it is an elephant?
Alice: Yes
---

Here is the second example dialogue:

---
Bob: Is it a living thing?
Alice: No
Bob: Is it a plant?
Alice: No
Bob: Is it a mineral?
Alice: Partly
[...]
Bob: Is it found on a desk?
Alice: No
Bob: Does it make noise?
Alice: No
Bob: I am guessing that it is the Sun?
Alice: Yes
---

Now it is your turn to play.

This is Alice and Bob's conversation, starting with Bob's first question.
<|end|>
```

Figure A.8.: System prompt from questioner's perspective for few-shot approach containing two example dialogues and correct prompt format for Phi-3-mini-4k-instruct. Abbreviated dialogue, the system prompt contains the full dialogue⁷¹

```
<|system|>
You are thinking of the concept '{concept}'.

Your friend is trying to guess what you are thinking of by asking questions.
Answer the following question by only answering {answer_options}:<|end|>
<|user|>
Question: {question}<|end|>
<|assistant|>Answer:
```

Figure A.9.: Single question answerer prompt for measuring the answerer’s performance for each entry of the AllenAI TQ test dataset (Prompt format is for Phi-3-mini-4k-Instruct). The same format is used for training of the answerer on the AllenAI TQ train dataset.

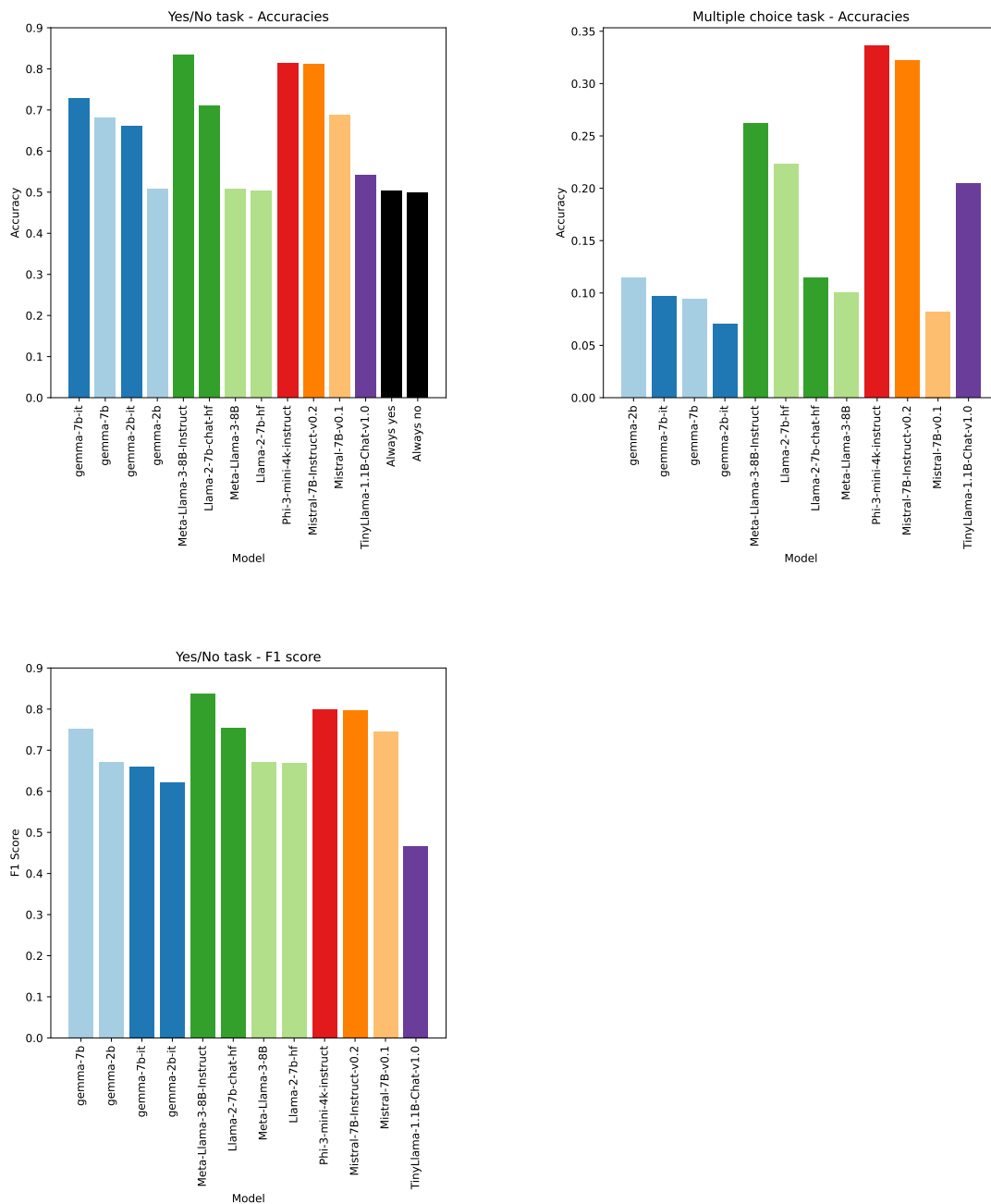


Figure A.10.: Results of the answerer’s performance. Top left: Accuracies of yes/no task. Top right: Accuracies of multiple choice task. Bottom left: F1 score of yes/no task. Model families are in a common color with solid shades for instruction-tuned models and lighter shades for non-instruction-tuned models. All evaluations are using the unfiltered dataset, evaluations on the unfiltered dataset can be found in Figure 4.3

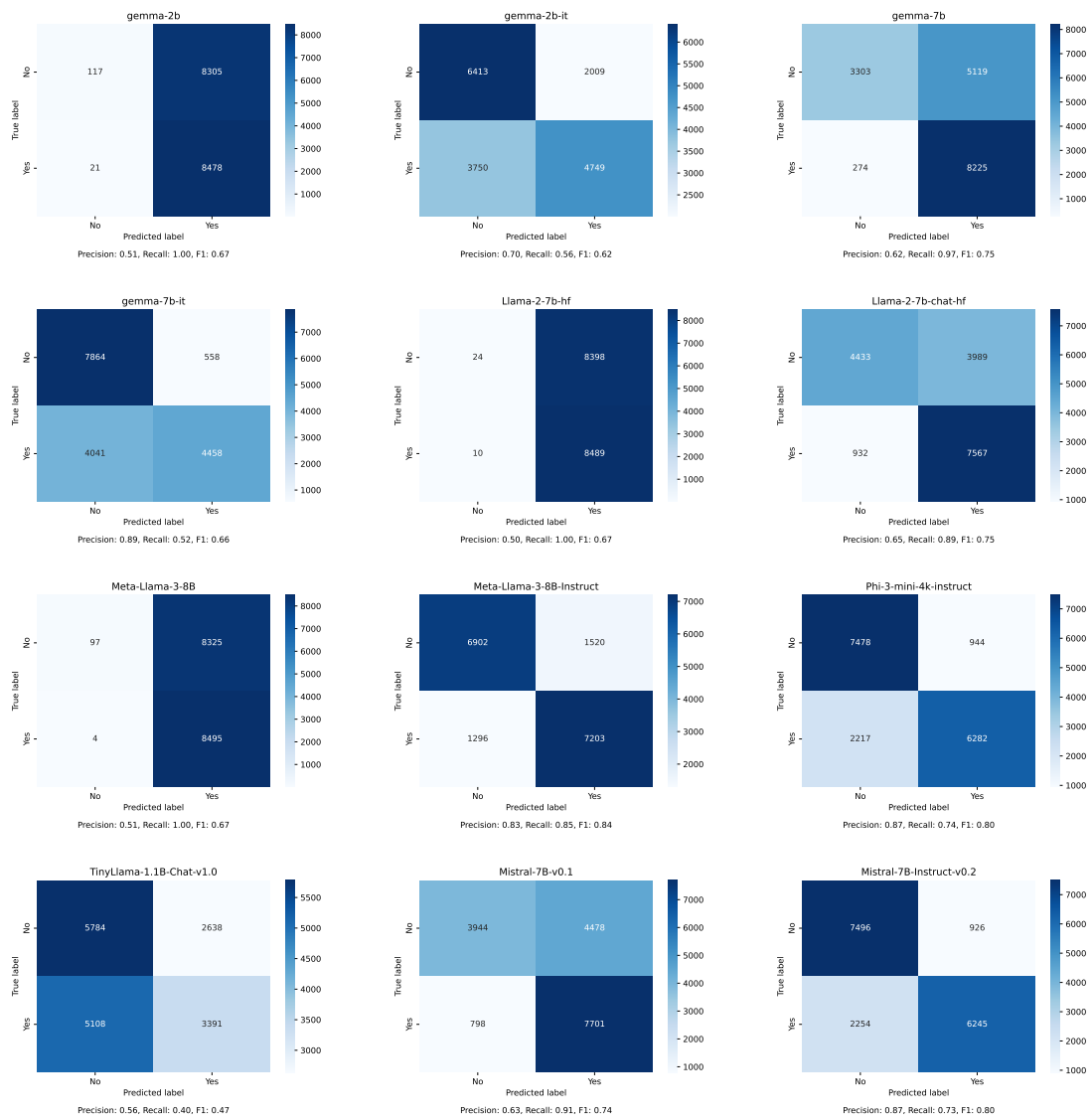


Figure A.11.: Confusion matrices for binary answerer evaluation using the AllenAI TQ dataset.

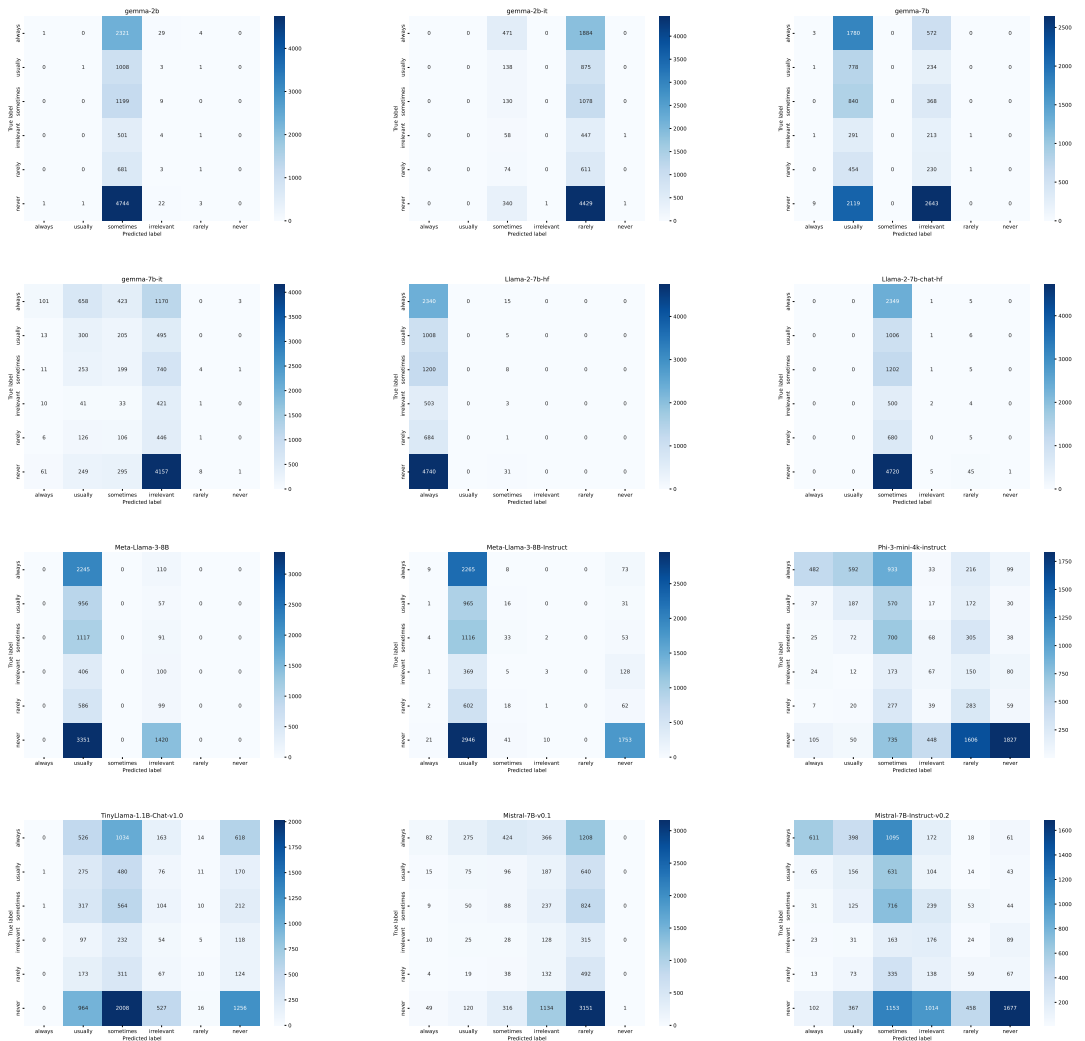


Figure A.12.: Confusion matrices for multiple choice answerer evaluation using the AllenAI TQ dataset.

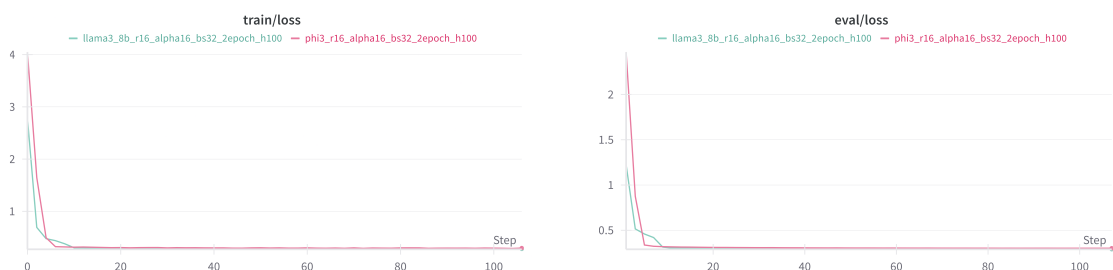


Figure A.13.: Training and evaluation loss for multi-choice answerer.

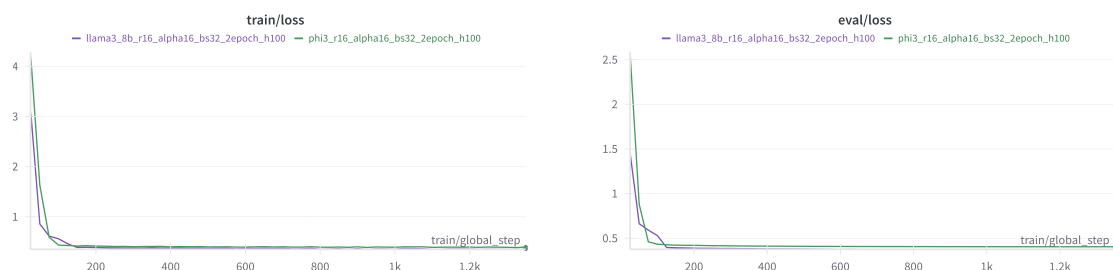


Figure A.14.: Training and evaluation loss for multi-choice answerer.

```

<|system|>
Your job is to decide whether a given concept is known to a 5-year-old. You
  reply with only saying 'Yes' or 'No'.

Example: Is 'tree' is known to a 5-year-old?
Answer: Yes.

Counterexample: Is 'hyperbolic paraboloid' known to a 5-year-old?
Answer: No.<|end|>
<|user|>
Is 'table' known to a 5-year-old?"<|end|>
<|assistant|>
Answer: Yes
  
```

Figure A.15.: Prompt for filtering WordNet concepts for supervised fine-tuning formatted for Phi-3-mini-4k-instruct.