UNIVERSITY OF AMSTERDAM

MSc ARTIFICIAL INTELLIGENCE
MASTER THESIS

Belief State for Visually Grounded, Task-Oriented Neural Dialogue Model

by

TIM BAUMGÄRTNER
11043683

April 5, 2019

36 European Credits
March 2018 - April 2019

Examiner:
Dr Raquel Fernández

Supervisor:
Dr Elia Bruni

Assessor:
Dr Iacer Calixto

INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION

Acknowledgement

I would like to thank all professors, lecturers, teaching assistants and the course administration at the University of Amsterdam, who contributed to my great experience as a student of the Masters of Artificial Intelligence program. They have always understood to make the learning experience fun and challenging.

My sincere thanks to Raquel Fernández, for fruitful discussions about research directions, her intermediate feedback on this work, and for her patience with me. Further, for providing me with opportunities to do research in her lab, letting me present my work at the Dialogue Modeling Group and taking me along to the ParlAI workshop in New York.

I am profoundly grateful to Elia Bruni who created an environment that let me experience AI research at first hand. He always understood to provide me with opportunities in which I could grow personally and professionally and experience what it is like to do research. I owe my sincere gratitude to him and Raquel for inspiring me to pursue a research-oriented career.

Lastly, heartfelt thanks go to my parents. Without their love and support, this work would not have been possible.

Abstract

The development of intelligent, conversational agents that can interact with humans in natural language is a longstanding goal of artificial intelligence. An approach that has recently gained traction is to learn conversational agents end-to-end. Thereby, the agent has to learn not only the language but also an intermediate representation of the dialogue state that is required to produce high-quality utterances. This approach is in contrast to the more traditional work on spoken dialogue systems, specifically dialogue agents with a dialogue manager that have an explicit representation of the current state of the dialogue. However, this requires interpretation and expensive annotation of the intermediate utterances.

In this work, we investigate an approach to equip an end-to-end system with an explicit dialogue state. This state provides a representation of the dialogue after each new utterance pair of the interlocutors. However, it is trained only from complete dialogues that lead to task success, therefore avoiding expensive intermediate annotations. Subsequently, the language generation is conditioned on the dialogue state. Specifically, we use the GuessWhat?! task, a two-player visually grounded, task-oriented, dialogue dataset. In this game, one agent is assigned an object in an image, and another agent tries to identify that object through a series of yes/no questions that the first agent answers. We apply the explicit dialogue state representation to the question generating agent, compare the effectiveness of different state representations and provide a detailed analysis of the task performance and generated language.

Contents

List of Figures

List of Tables

1 Introduction

The development of intelligent, conversational agents that can interact with humans in natural language is a longstanding goal of artificial intelligence. Applications of dialogue systems are already ubiquitous in our everyday life in the form of personal assistants such as Amazon's Alexa and Apple's Siri, chatbots on Facebook's Messenger platform , Twitter bots or dialogue systems in call centres helping a customer directly or routing the call to a respective human expert. However, these systems are usually far from perfect, often lacking the ability to hold an engaging and coherent conversation over multiple utterances or only useful on a narrow set of domains.

Spoken dialogue systems use a pipeline approach, which is generally composed of three units: First, a module that takes acoustic user utterances as input, and produces hypotheses what has been said. Second, a dialogue manager that keeps track of the information provided by the user and the user's goal. It also decides the next action to take by the system and how to respond to the user. This information is represented in the belief state [Young et al., 2010, Jurafsky and Martin, 2019] (also called dialogue state or information state). Finally, the third unit generates and verbalizes the next system utterance based on the belief state. The dialogue state tracking challenge (DSTC) [Williams et al., 2013], is particularly concerned about the central component of dialogue modelling, the dialogue manager. It introduces a unifying testbed and dataset for the dialogue manager to make the different approaches better comparable. However, while the DSTC operates on the input from automatic speech recognition and natural language understanding modules (i.e. the first unit in the Spoken Dialogue System pipeline), it does not regard further downstream task such as the generation of the next system utterance. Note that in this work, we focus on text-based dialogue systems, which in principle can be extended to speech-based systems.

Recently, deep learning based methods have significantly advanced the field of artificial intelligence in many of its domains [Krizhevsky et al., 2012, Mnih et al., 2013] including natural language processing (NLP) [Goldberg and Hirst, 2017]. While deep learning methods require large datasets, they have outperformed rule-based and statistical approaches. The data-driven models learn end-to-end, taking as input the raw data, learning their own task-specific, internal representations and eventually returning an output according to the task at hand. For example, in [Vinyals and Le, 2015] this approach is applied to conversations in an IT helpdesk scenario and movie dialogues. While these approaches are typically able to produce syntactically correct responses, they have difficulties generating diverse and engaging replies [Li et al., 2016] or long term coherent replies for task-oriented systems. Thanks to remarkable advancements in computer vision, many NLP tasks have been enriched by visually grounding them. Adding visual perception to language tasks allows building more human-like semantic representations, which are deeply intertwined with perception [Barsalou, 2008]. New tasks and datasets have been proposed on the intersection of language and vision such as image captioning [Rashtchian et al., 2010, Young et al., 2014, Lin et al., 2014], machine translation [Elliott et al., 2016, Elliott et al., 2017], question answering [Antol et al., 2015] and also dialogue [Das et al., 2017, De Vries et al., 2017a, de Vries et al., 2018].

Deep learning based dialogue systems are now omnipresent in artificial intelligence research. However, an explicit module representing the task of the dialogue manager has not been widely adopted. In this work, we attempt to enrich the successful deep learning based dialogue systems with the dialogue manager idea from more traditional dialogue system research. We propose a deep learning based, visually grounded dialogue system that has an explicit belief state representation that is grounded in the visual perception both participants share. Analogous to a dialogue manager, the belief state is updated with each new interaction between the dialogue partners. It further represents the information conveyed in the dialogue thus far in an effective manner. Downstream, it is utilized to focus the perception of the system, as well as condition the language generation. The working of the model is principally demonstrated on the task-oriented, visually grounded GuessWhat?! [De Vries et al., 2017a] dataset. However, its application is not limited to this task and can be in principle employed to other task-oriented dialogue settings.

1.1 Overview and Contributions

Our main contributions are the following:

- We present an end-to-end trainable dialogue system with an explicit belief state representation inspired by the dialogue systems pipeline.
- We conduct elaborate experiments and evaluate different representations and applications of the belief state in the GuessWhat?! scenario.
- To the best of our knowledge, our model generates dialogues with the highest task success rate in GuessWhat?! when only learning from the original dataset[1].
- We present a detailed analysis of the task performance and linguistic measures of the baseline model and our approach.
- We make our entire code base public to reproduce our experiments, results and analyses and facilitate further research[2]. We also provide a web-based tool for comparing the belief state representations of different dialogue models at every turn.

In chapter 2 we review deep learning fundamentals for this work. Further, we discuss the motivation and related work for visually grounded natural language processing tasks and dialogue systems in general. In chapter 3 we formally introduce the GuessWhat?! game and review recent advancements. Next, in chapter 4 we start by introducing the GuessWhat?! baseline models, and then motivate and formally describe the belief state model and its variants. Subsequently, in chapter 5 we describe the experimental setup and report the results. Chapter 6 provides a detailed analysis of the results and investigates the generated language by the models. Finally, chapter 7 provides a conclusion and an outlook on future work.

[1]In other words, without exposing the agent to more training examples other than those in the train set, as for example done in reinforcement learning [Strub et al., 2017, Zhang et al., 2018] or cooperative learning [Shekhar et al., 2019].

[2]`https://github.com/timbmg/belief`

2 Background and Related Work

2.1 Deep Learning

Deep learning has recently significantly advanced state of the art in artificial intelligence. The success is also due to its compositional properties. Building blocks of different abstraction levels can be stacked on top of each other, forming new architectures to solve problems. The emergence of frameworks that provide easy entry points for practitioners, such as PyTorch [Paszke et al., 2017] and Tensorflow [Abadi et al., 2016], further facilitate the accessibility of the technology.

Essentially, deep learning is concerned around the idea of function approximation. A set of parameters θ transform an input x into an output $\hat{y}$, such that it is close to a desired output y, i.e. $f(x|\theta) = \hat{y} \approx y$. During the training process, the parameters θ are changed such that the distance between $\hat{y}$ and y is minimized. In the following sections, the most important deep learning buildings blocks for this work will be reviewed.

2.1.1 Multi-Layer Perceptron

At the core of deep learning is a vector-matrix multiplication also referred to as a *linear transformation*. It consists of an input vector $\mathbf{x} \in \mathbb{R}^n$, where $\mathbb{R}^n$ denotes that the vector has n elements and all are in $\mathbb{R}$, and a matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$ and a bias vector $\mathbf{b} \in \mathbb{R}^m$, both containing trainable weights (see equation 2.1). Note that the boldface small characters denote a vector and boldface capital characters represent a matrix. The weight matrix and bias are also referred to as parameters which are updated during the training process.

$$
\begin{aligned}
\mathbf{W}\,\mathbf{x}+\mathbf{b} &= \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,n} \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \\
&= \begin{bmatrix} \sum_{i=1}^{n}(w_{1,i} * x_i) + b_1 \\ \sum_{i=1}^{n}(w_{2,i} * x_i) + b_2 \\ \cdots \\ \sum_{i=1}^{n}(w_{m,i} * x_i) + b_m \end{bmatrix}
\end{aligned} \tag{2.1}
$$

In many occasions, the linear transformation is followed by a non-linear element-wise operation, also called activation. The most commonly used are listed below.

- sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$
- hyperbolic tangent: $\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
- rectified linear unit: $\text{relu}(x) = \max(0, x)$ [Nair and Hinton, 2010]

The application of the non-linear activation function after the linear transformation allows for also approximating non-linear relations between the input $\mathbf{x}$ and output $\mathbf{y}$. Before the introduction of these functions, a major criticism was that non-linear problems could not be solved [Minsky and Papert, 1969] (e.g. a simple function such as XOR cannot be learned).

When stacking multiple layers of non-linear transformations (i.e. a linear transformation followed by a non-linear operation) on top of each other, a Multi-Layer Perceptron (MLP) is formed (also referred to as feed-forward neural network). Intermediate outputs of the non-linear transformations are referred to as hidden layers. In mathematical terms, an MLP with a single hidden layer and a relu activation function can be expressed as follows:

$$\hat{\mathbf{y}} = \mathbf{W}_1\ (\text{relu}(\mathbf{W}_2\mathbf{x} + \mathbf{b}_2)) + \mathbf{b}_1 \tag{2.2}$$

Many problems in artificial intelligence are treated as classification problems (e.g. recognition of handwritten digits [LeCun et al., 1998] or generation of the next word in sequence [Mikolov et al., 2010]). In order for the final output of an MLP to represent a valid probability distribution over the classes, the output vector $\hat{\mathbf{y}}$ has to be normalized such that $\sum_i \hat{y}_i = 1$. To do so, the softmax function is used:

$$\hat{y}_i = \frac{e^{a_i}}{\sum_j e^{a_j}} \tag{2.3}$$

where a_i is the unnormalized output at the i^{th} index.

2.1.2 Recurrent Neural Networks

Recurrent Neural Networks (RNN) [Rumelhart et al., 1985] play a tremendous role in Natural Language Processing as they are able to model sequences by design. Many state of the art models are based on recurrent networks (e.g. part-of speech-tagging [Bohnet et al., 2018], text classification or sentiment analysis [Howard and Ruder, 2018]) or have only recently been outperformed by transformer models [Vaswani et al., 2017] (e.g. language modelling [Merity et al., 2017])[3].

Formally, an RNN processes a sequence $\mathbf{X} = \mathbf{x}_1, ...\mathbf{x}_t..., \mathbf{x}_m$ of length m by updating a hidden state $\mathbf{h}_t$ at each timestep based on the hidden state of the previous timestep $\mathbf{h}_{t-1}$ and the current input $\mathbf{x}_t$.

$$\mathbf{h_t} = f(\mathbf{x_t}, \mathbf{h_{t-1}}) \tag{2.4}$$

The reason this type of network is called a *recurrent* neural network is obvious from equation 2.4. The output at time $t-1$ is also part of the input to the next timestep t. Another convenient feature of RNNs is that the functions applied at every timestep are shared over the time dimension. This allows the network to be applied to variable size sequences and recognize patterns more independently over the time dimension. This becomes clear when trying to use an MLP for sequence processing. First, the input to the MLP needs to be of a fixed dimension, which makes it impractical to process inputs with variable sizes. Second, especially in the language domain, the

[3]State of the art retrieved from https://github.com/sebastianruder/NLP-progress. Accessed on 1st April 2019.

same information can be expressed with different word order (e.g. "Germany won the world cup in 2014" and "In 2014, Germany won the world cup."). A system based on an MLP would have to learn to detect the linguistic features at every timestep since they can occur theoretically everywhere. In contrary, an RNN shares its parameters over timesteps, i.e. the same function is applied at every timestep [Goodfellow et al., 2016, Chapter 10].

There are many different versions of RNNs, most commonly are the Elman Recurrent Networks [Elman, 1990] also referred to as vanilla RNNs, Long Short-Term Memory (LSTM) network [Hochreiter and Schmidhuber, 1997] and Gated Recurrent Units (GRU) [Cho et al., 2014]. The last two's main advantage is that they overcome the vanishing gradient problem [Hochreiter, 1991, Bengio et al., 1994] in vanilla RNNs and therefore are beneficial when modelling long term dependencies. In the sections below, Elman RNNs and LSTMs are briefly reviewed.

Elman Recurrent Neural Networks

Elman RNNs are a simple form of recurrent neural network. At every timestep, the hidden state $\mathbf{h}_t$ is updated and an output $\mathbf{y}_t$ is produced according to the following equations:

$$\mathbf{h_t} = \tanh(\mathbf{W_{hh}h_{t-1}} + \mathbf{W_{xh}x_t} + \mathbf{b_h}) \tag{2.5}$$

$$\mathbf{\hat{y}_t} = f(\mathbf{W_{hy}h_t} + \mathbf{b_y}) \tag{2.6}$$

Where f can be any function appropriate to the use case (e.g a softmax for classification), $\mathbf{h}_t \in \mathbb{R}^n$, $\mathbf{x_t} \in \mathbb{R}^m$, $\mathbf{y_t} \in \mathbb{R}^o$ and therefore $\mathbf{W}_{hh} \in \mathbb{R}^{n\times n}$, $\mathbf{W}_{xh} \in \mathbb{R}^{n\times m}$, $\mathbf{b}_h \in \mathbb{R}^n$, $\mathbf{W}_{hy} \in \mathbb{R}^{o\times m}$ and $\mathbf{b}_y \in \mathbb{R}^o$. Figure 2.1 illustrates the working of the Elman RNN.

While in theory, this network is able to learn to predict for example the stock price (time series prediction) or the next word in a sentence (language modelling), practically the network suffers from the vanishing gradient problem. That is, as we obtain the gradient for inputs that lie further in the past, the gradient gets smaller and smaller. This is due to the derivative of the activation functions that are in $(0, 0.25]$ for sigmoid and $(0, 1]$ for tanh. The gradient is multiplied with this, therefore, for each layer that is activated, it will become smaller. Essentially, this inhibits the model from learning longer sequences.

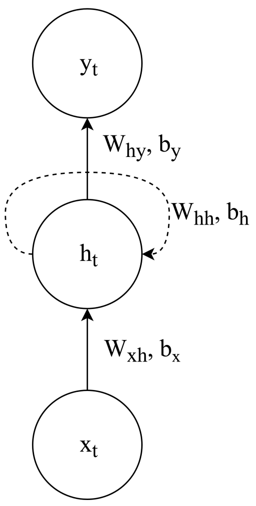


Figure 2.1: Elman RNN Graph

Long Short-Term Memory Networks

Long Short-Term Memory Networks [Hochreiter and Schmidhuber, 1997] are defined by the following equations 2.7 - 2.12.

$$\mathbf{f}_t = \sigma(\mathbf{W}_{fh}\mathbf{h}_{t-1} + \mathbf{W}_{fx}\mathbf{x}_t + \mathbf{b}_f) \tag{2.7}$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_{ih}\mathbf{h}_{t-1} + \mathbf{W}_{ix}\mathbf{x}_t + \mathbf{b}_i) \tag{2.8}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{oh}\mathbf{h}_{t-1} + \mathbf{W}_{ox}\mathbf{x}_t + \mathbf{b}_o) \tag{2.9}$$

$$\widetilde{\mathbf{c}}_t = \tanh(\mathbf{W}_{ch}\mathbf{h}_{t-1} + \mathbf{W}_{cx}\mathbf{x}_t + \mathbf{b}_c) \tag{2.10}$$

$$\mathbf{c}_t = \widetilde{\mathbf{c}}_t * \mathbf{i}_t + \mathbf{f}_t * \mathbf{c}_{t-1} \tag{2.11}$$

$$\mathbf{h}_t = \mathbf{o}_t * \tanh(\mathbf{c}_t) \tag{2.12}$$

Where $\mathbf{h}_t \in \mathbb{R}^n$ and $\mathbf{x}_t \in \mathbb{R}^m$, therefore $\mathbf{W}_{\cdot,h} \in \mathbb{R}^{n \times n}$, $\mathbf{W}_{\cdot,x} \in \mathbb{R}^{n \times m}$, $\mathbf{b}_{\cdot} \in \mathbb{R}^m$ and $\mathbf{f}_t, \mathbf{i}_t, \mathbf{o}_t, \mathbf{c}_t, \mathbf{h}_t \in \mathbb{R}^n$.

LSTMs are centred around the idea of gates. At every timestep a forget vector $\mathbf{f}_t$ (equation 2.7), an input vector $\mathbf{i}_t$ (equation 2.8) and an output vector $\mathbf{o}_t$ (equation 2.9) is computed. Each of the elements in these vectors is in $(0, 1)$ due to the sigmoid activation function. These gates are computed based on the current input $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{t-1}$. Based on the input and forget gate, the cell state $\mathbf{c}_t$ is updated. First, a new propositional cell state is computed (2.10). This new cell state is then element-wise multiplied with the input vector, which now becomes clear acts as a gate to the new cell state. Besides, the forget vector is element-wise multiplied

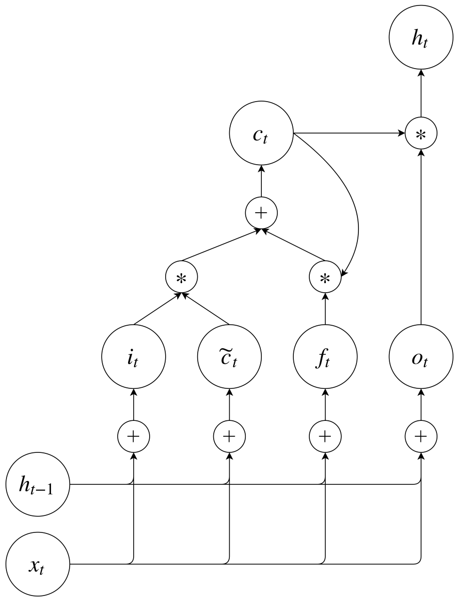


Figure 2.2: LSTM Graph

with the previous cell state (right half of equation 2.11), and thereby acting the other way round, controlling which parts of the previous cell state shall be kept in memory and which shall be replaced. Finally, a new hidden state is computed with the output gate and the tanh-activated cell state (equation 2.12).

2.1.3 Gradient-Based Learning

In the previous sections 2.1.1 and 2.1.2, linear transformations have been used extensively. In this section, the main method for updating the weights and bias shall be explained.

Generally, the goal of a machine learning model is to perform "well" (according to an evaluation metric, e.g. accuracy for classification problems) on new, previously unseen data. However, we usually do not have access to the true data distribution and therefore cannot optimize the model towards that goal directly. Rather, we have access to a dataset $\mathcal{D}$ containing samples (x_i, y_i) from that distribution, where x_i is some input and, in case of supervised learning, y_i a corresponding output (e.g. x_i is a house location and year, and y_i its price, or x_i is a sentence and y_i its sentiment). Since we only have access to a limited number of samples, but we want to optimize the model performance on unseen data, usually the data is split into a training, validation and test set ($\mathcal{D}_{train}$, $\mathcal{D}_{valid}$, $\mathcal{D}_{test}$). The training set will be used to directly optimize the parameters of the model. In turn, the validation set will be used for intermediate model performance evaluations. Finally, the test set is the surrogate for the true data distribution and therefore it will only be used once hyperparameters of the model have been chosen according to the evaluation metric on the validation set. This procedure is referred to as Empirical Risk Minimization. Formally, we want to minimize the empirical risk $\mathcal{J}$ (also called objective function) on the training dataset of the expected loss $\mathcal{L}$, between the predicted output $f(x|\theta) = \hat{y}$ and y:

$$\mathcal{J}(\theta) = \mathbb{E}_{(x_i,y_i)\sim\mathcal{D}_{train}}[\mathcal{L}(f(x_i|\theta), y_i)] \tag{2.13}$$

$$= \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(f(x_i|\theta), y_i) \tag{2.14}$$

When interpreting the model output probabilistically and taking the logarithm we obtain:

$$\mathcal{J}(\theta) = -\mathbb{E}_{(x_i,y_i)\sim\mathcal{D}_{train}} \log p_{model}(x_i, y_i) \tag{2.15}$$

The objective is now to find the parameters θ^* such that:

$$\theta^* = \arg\min_{\theta} - \sum_{i=1}^{n} \log p_{model}(x_i, y_i) \tag{2.16}$$

This can be accomplished by changing the model parameters θ in the negative direction of the gradient of $\mathcal{J}(\theta)$, which points into the direction of the steepest ascent. We therefore minimize the objective function.

$$\theta^{t+1} = \theta^t - \alpha\nabla\mathcal{J}(\theta^t) \tag{2.17}$$

where α is the learning rate.

In the Stochastic Gradient Descent (SGD) algorithm, the gradient $\nabla\mathcal{J}(\theta^t)$ is estimated from a minibatch of examples, instead of the entire dataset. This has several advantages, namely: First, in order to update the parameters, the model does not have to be evaluated on the entire dataset, which usually is expensive particularly with large datasets. Second, empirically, the noisy gradient estimation also helps to overcome potential local minima in the loss surface.

In this work optimization algorithms that are based on SGD are used, namely Adam [Kingma and Ba, 2015]. Compared to SGD, Adam keeps an adaptive learning rate for each parameter in the model based on the magnitude of the gradient.

2.2 Visually Grounded NLP

By virtue of advancements in computer vision, many natural language tasks have been enriched with a visual context. That is *grounding* the language used in visual perception such as a single image or even video. Tasks range from visually grounded image and video captioning [Rashtchian et al., 2010, Young et al., 2014, Lin et al., 2014] [Chen and Dolan, 2011, Rohrbach et al., 2015, Torabi et al., 2015], reference resolution [Kazemzadeh et al., 2014], machine translation [Elliott et al., 2016, Elliott et al., 2017], question answering [Das et al., 2017] to also dialogue [Das et al., 2017, De Vries et al., 2017a, Mostafazadeh et al., 2017, Kim et al., 2017a, de Vries et al., 2018].

The idea of visually grounding language is based on the observation that humans also do not learn language in isolation but in a highly multimodal way. That is, while learning, i.e. becoming intelligent, we are constantly exposed to all our sensory input such as sound, smell, touch and also our visual perception of the physical world. Further [Harnad, 1990] points out, that symbols cannot be solely defined by other symbols (symbol grounding problem). In babies, it has been shown that this multimodal input is heavily intertwined with language learning [Smith and Gasser, 2005]. Further, research in cognitive science has shown that comprehending a word also activates the perceptual system [Zwaan et al., 2002, Barsalou, 2008]. Therefore suggesting that the two modalities go hand in hand.

In order to advance artificial agents, grounding their understanding is a necessity. Adding visual perception to artificial agents has also been a long-standing goal [Winograd, 1972]. However, only recently adding visual context has become more widely adopted, mainly due to efficient representations learned by computer vision models.

2.3 Dialogue Systems

2.3.1 Task-Oriented Dialogue Systems

As opposed to chit-chat dialogue systems, task-oriented dialogue systems assist an user in accomplishing a task through a conversational interface. An early successful dialogue system was the GUS system [Bobrow et al., 1977] that introduced the idea of frames. Frames are sets of slot-value pairs that the system needs to fill to perform a task (such as booking a flight). Based on what information was already provided through the dialogue and saved in the frames, and what information is still missing, the system produces the next utterance. Based on this idea a pipeline approach for dialogue developed. Generally speaking, more traditional dialogue systems consist of a natural language understanding unit, a dialogue manager and a natural language generation module. If the system interacts with its user through speech, additionally speech recognition and text to speech modules are added [Jurafsky and Martin, 2019, Jokinen and McTear, 2009] (see Figure 2.3).

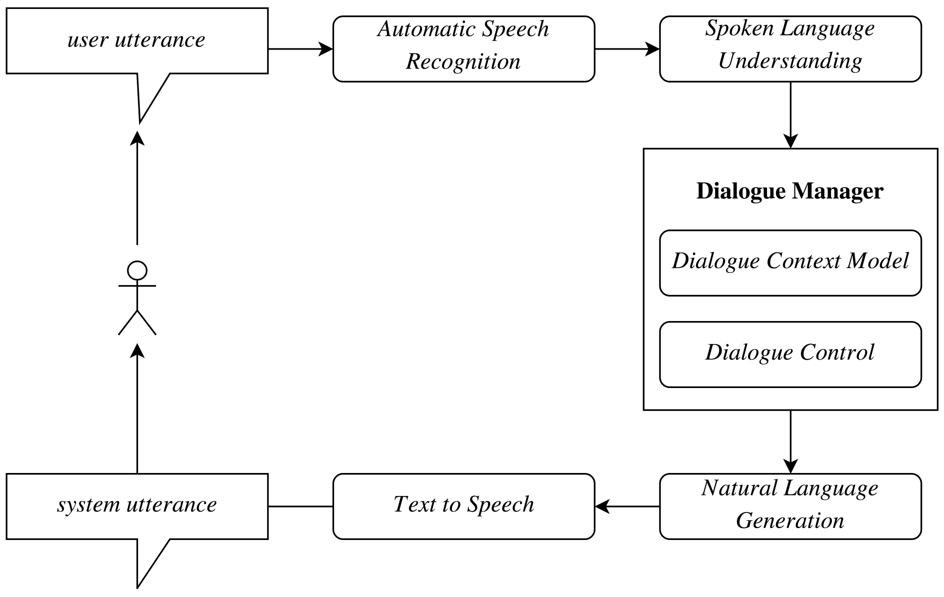


Figure 2.3: Spoken Dialogue System Pipeline

The dialogue manager module is of particular interest to our work. [Jokinen and McTear, 2009] further break down the module into two parts. First, the dialogue context model (also known as dialogue state tracker) that keeps track of information provided in the dialogue for the task at hand. It holds the information in an internal representation, called the belief state or dialogue state. Further, it might provide context to the dialogue by interfacing to external systems or input from other modalities. Second, the dialogue control unit (also known as dialogue policy), that decides what action to take next. Essentially, it determines the dialogue act of the system. For example, the system might ask a clarification question, provide information to the user or restart or end the conversation.

The dialogue state tracking challenge [Williams et al., 2013, Henderson et al., 2014a, Henderson et al., 2014b, Kim et al., 2017b, Kim et al., 2016b] has specialized in the dialogue manager component. It provides a testbed for general dialogue state tracking modules and several datasets. However, it regards dialogue state tracking in isolation. While it does incorporate the uncertainty coming from speech recognition and natural language understanding, it is not concerned about natural language generation.

Training all modules of the pipeline in a data-driven process is a major challenge since usually labels for each module are required. Moreover, when the modules are trained separately, the input from a previous module might be flawed. The approach of [Young et al., 2013] acknowledges this uncertainty and therefore models the dialogue as a partially observable Markov decision process. The dialogue manager models the dialogue state transition function and what action to choose next with stochastic models. Albeit state and action are modelled by probabilistic functions, still symbolic variables are ultimately used. The system is trained through reinforcement learning, i.e. via task success. Notably, the dialogue management component we propose for the GuessWhat?! system has a continuous dialogue state and therefore, the uncertainty can be incorporated in the downstream modules.

2.3.2 End-to-End Dialogue Systems

Another line of research is that of end-to-end dialogue systems. These models purely learn from the language, without any intermediate representations like dialogue acts or handcrafted features. End-to-end learning has shown many improvements in artificial intelligence, for example in machine translation.

Based on [Ritter et al., 2011] data-driven approach to response generation in the context of social media, [Vinyals and Le, 2015, Shang et al., 2015, Sordoni et al., 2015b] employ the sequence to sequence framework [Sutskever et al., 2014] that has been successful in machine translation, for dialogue modelling. The past utterances are represented in a fixed size vector through an encoder recurrent neural network. From this representation, the next utterance is decoded. The advantage of these models is that they are generative, therefore in principal being able to produce a wide range of utterances. Results show good syntactical capabilities, however, they tend to generate inconsistent utterances, i.e. contradicting themselves (see the example below). Furthermore, also the diversity of the utterances is problematic, for example when employed in question answering scenarios, these model tend to generate save answers like "I don't know" [Li et al., 2016].

> **Human**: *what is your job ?* **Machine**: *i 'm a lawyer .*
> **Human**: *what do you do ?* **Machine**: *i 'm a doctor .*
>
> Example taken from [Vinyals and Le, 2015]

A model building on the vanilla sequence to sequence framework is the hierarchical recurrent encoder decoder [Serban et al., 2016]. Here, the encoder is modelled hierarchically on the word and utterance level. The lower level recurrent network encodes a single utterance word by word. The output is then processed by the higher level recurrent network, that encodes the dialogue history utterance by utterance. From the hidden state of the utterance encoder, the system reply is generated. This approach has also been tried in GuessWhat?! [De Vries et al., 2017a], however, it did not improve over a simple recurrent language model that is additionally conditioned on the image.

Another notable end-to-end system is that of [Bordes and Weston, 2016], where memory networks [Weston et al., 2015, Sukhbaatar et al., 2015] are used on a task-oriented dialogue setting. The model can read and write to a dedicated memory via an attention mechanism [Bahdanau et al., 2015]. It is tested the on babi tasks, a toy dataset in the restaurant domain [Bordes and Weston, 2016]. While it excels at the simple tasks like correctly querying a knowledge base, the model still struggles with correctly conducting full dialogues.

In summary, end-to-end systems rely on a latent representation. These are not learned explicitly but only through the dialogues themselves. While this allows for less expensive data collection, the models do show drawbacks on consistency.

2.3.3 Multimodal Dialogue Systems

Another line of related work is that of systems incorporating other modalities, specifically visual perception. A predominant approach to adding visual perception to a dialogue agent is to extract visual features from an object classification network such as VGG [Simonyan and Zisserman, 2015] or ResNet [He et al., 2016]. A simple approach is to use the activation of one of the final, fully connected layers as feature representation [De Vries et al., 2017a, Das et al., 2017]. Another approach is to take outputs from the convolutions layer and reduce them via an attention mechanism [Yang et al., 2016, Kazemi and Elqursh, 2017]. This allows the model to focus on specific areas of the image. Recently another method has shown promising results in obtaining a visual representation for downstream language applications. [De Vries et al., 2017b, Strub et al., 2018] condition the visual pipeline on a language encoding, such that it can focus from the beginning on relevant features.

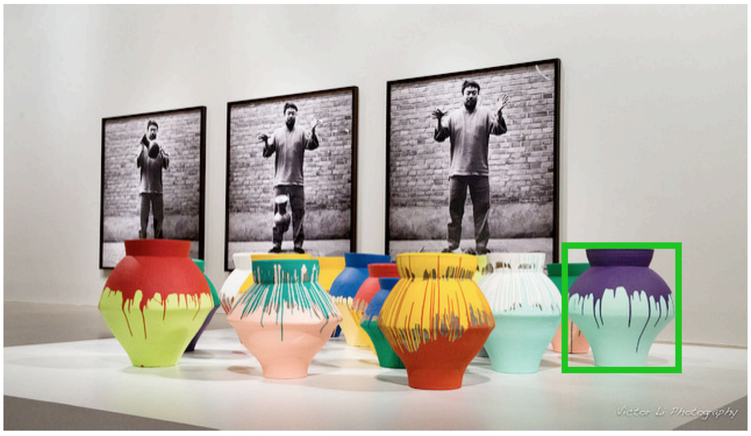
3 GuessWhat?!

3.1 Task Setup

GuessWhat?! [De Vries et al., 2017a] is a two-player, visually grounded, dialogue dataset. The goal is to determine an object on an image through a series of yes-no questions. Therefore, it requires both visual understanding, as well as language capabilities.

More specifically, the players adopt two different roles: that of a Questioner and an Oracle. The Oracle is randomly assigned an object on the image. On the other hand, the Questioner does not know the assignment and has to ask yes-no questions in order to determine it. Once the Questioner is confident of having determined the target object, the Questioner can proceed from to the *guessing phase* in which an overlay on the image marks all possible objects to choose from. If the Questioner selects the correct object, the game is considered successful and unsuccessful otherwise. Independent of the result the game terminates at this stage, i.e. more questions and another guess is not permitted. Figure 3.1 shows an example game. On `https://guesswhat.ai/` the game can be played and the dataset explored.

The dataset has been collected via Amazon Mechanical Turk and consists of a total of 155,280 dialogues, of which 144,434 have been finished (i.e. none of the two humans players have aborted before the guess was taken) and 131,394 ended successful (90.97%). After preprocessing the human dialogues (for details see [De Vries et al., 2017a]), the vocabulary size of tokens used at least three times is 4,919. The dataset is based on images from MS COCO [Lin et al., 2014], in total 66,537 different images are used. Therefore there are about 2.3 dialogues per image. The dataset is split in a 70:15:15 ratio into a train, validation and test set. The validation set consists only of images which are also in the train-set. However, new target



Questioner	**Oracle**
Is it a vase?	Yes
Is it partially visible?	No
Is it in the left corner?	No
Is it the turquoise and purple one?	Yes

Figure 3.1: Example game from GuessWhat?!. Figure from [De Vries et al., 2017a].

objects have been selected. On the other hand, the test-set contains completely new images, neither present in the training nor validation split. Therefore, the validation split evaluates how well the model generalizes to new objects, while the test-set measures the performance on new images.

Formally, a game $\mathcal{G}$ consists of a dialogue $\mathcal{D}$ with a variable number of question-answer pairs $[q_1, a_1, ..., q_t, a_t, ..., q_n, a_n]$. Each question-answer pair consists of a variable number of words $[w_{t,1}, ..., w_{t,\tau}, ..., w_{t,m}]$. Besides, also an image $\mathcal{I} \in \mathbb{R}^{3 \times W \times H}$ with a set of annotated objects $o_i \in \mathcal{O}$ of which one is the target object o_{target} is part of the game. Each object is specified by a set of points, forming a polygon around the object. Each object is assigned a category $c \in \mathcal{C}$, and $|\mathcal{C}| = 81$. Both, the polygon and category information come from the MS COCO annotations [Lin et al., 2014].

3.1.1 Evaluation

In [De Vries et al., 2017a], baseline versions of the Questioner and Oracle agents are proposed. Thereby, the Questioner is split into a Question Generator and a Guesser. The former is responsible for generating the next question given the dialogue history and the image. The Guesser comes into play when the dialogue is finished. Given the entire conversation and the list of objects, it has to select the target object. The evaluation of the Guesser and Oracle performance on the dataset is straight forward, as simply the total number of correctly guessed games, or in case of the Oracle correctly answered questions can be calculated based on the human dialogues. On the other hand, the evaluation of the Question Generator is more difficult. While the perplexity can be measured, it might not be as revealing due to the fact that there is a big space of potentially correct next words. This is a common problem in dialogue as opposed to, for example, machine translation where the space of correct translations is much smaller.

Since GuessWhat?! is a task-oriented setting, we can, however, evaluate the Question Generator on the task. Therefore, all the modules are evaluated alongside each other. In order to do so, the Question Generator predicts the next word based on its previous generation (for details see section 4.1.3). In this way, new questions are generated which in turn can be answered by the Oracle. Finally, after a fixed number of questions, the Guesser uses the generated dialogue between the Question Generator and the Oracle to predict the target object. The accuracy resulting from this process can be used to measure the performance of the Question Generator.

3.2 Advancements in GuessWhat?!

3.2.1 Guesser and Oracle Improvements

Building on the Feature-wise Linear Modulation (FiLM) line of research [De Vries et al., 2017c, Perez et al., 2018], [Strub et al., 2018] apply this technique to the Oracle and Guesser model. Thereby, the visual pipeline is manipulated through language, by shutting down or amplifying feature detectors. Further, an attention mechanism on the language itself allows for multi-step reasoning in the visual pipeline. Thereby, the Oracle performance improves to 83.1% (from 78.5%) and the Guesser to 69.5% (from 62.6%) on the test set[4].

[4]Details of the baseline Oracle and Guesser model are provided in section 4.1.1 and 4.1.2.

3.2.2 Question Generator Improvements

In [Shekhar et al., 2018] we address the issue of the fixed number of questions that the baseline Question Generator has to produce during inference. While humans can stop the game at any point and proceed to the guessing phase, for lack of a better mechanism, the baseline generates a fixed number of questions. By introducing a Decider module that predicts whether to stop the dialogue after each question-answer pair, the dialogues become more natural due to fewer repeated questions.

In [Strub et al., 2017] reinforcement learning is applied to the Question Generator. The authors start from the pretrained baseline models of the Oracle, Question Generator and Guesser (detailed in section 4.1). The authors frame the dialogue generation as a Markov Decision Process in order to apply the policy gradient algorithm REINFORCE [Williams, 1992]. Thereby, questions are generated from the Question Generator by sampling from the categorical distribution over the vocabulary. A zero-one reward is defined, namely if the Guesser module is able to correctly identify the target object from the sampled dialogue, a positive reward is provided. Thereby, a task accuracy of 60.3% on the validation and 58.4% on the test set is achieved, by evaluating the agent with a greedy decoding strategy.

In [Zhang et al., 2018], the performance is further improved by adding intermediate rewards on a question level. Thereby, dialogues are rewarded that have only few questions before the correct target is identified. Further, questions that improve the Guesser certainty on the target object are encouraged. Finally, questions that do not help to disambiguate between the objects are penalized. Therefore, the system is encouraged to solve the game with fewer questions, which therefore have to be more informative. This system achieves a task accuracy of 63.6% and 60.7% on the validation and test set respectively.

In [Shekhar et al., 2019], we propose a joint architecture for the Question Generator and Guesser based on a shared, visually grounded language encoder. Further, we achieve on par state of the art performance, by applying a novel learning scheme named cooperative learning that schedules the updates of the different output modules of the Questioner (viz. Guesser and Question Generator). On the validation set, this architecture achieves 63.3% and 60.7% on the test set. Table 3.1 provides an overview of the performances of the different works.

Model	Validation New Objects	Test New Images
[De Vries et al., 2017a]	43.5%	40.8%
[Strub et al., 2017]	60.3%	58.4%
[Zhang et al., 2018]	63.6%	60.7%
[Shekhar et al., 2019]	63.3%	60.7%

Table 3.1: Task performance of different Question Generator improvements. All results are based on greedy decoding.

4 Model

4.1 Baseline Models

The model presented in this work builds on the baseline models introduced in [De Vries et al., 2017a]. These are described in the following three sections.

4.1.1 Oracle

The Oracle agent faces a multimodal (language and vision), question answering problem. Typically, question answering problems can either be solved by a discriminative or generative model. In the former case, a set of possible answers over the entire training set has to be chosen, then a classifier is trained. For the latter, e.g. an RNN can be used to generate the answer given a question and possibly other inputs (both these approaches are for example explored in the VQA baseline [Antol et al., 2015]). In the case of GuessWhat?! a discriminative approach is natural due to the low number of classes. The game is designed such that only *yes*, *no* or *not applicable (n/a)* can be chosen as answer by the Oracle.

In the baseline setup by [De Vries et al., 2017a], the Oracle takes as input the target object's spatial location as bounding box derived from the polygon and its category c as one hot vector. The spatial location is provided by MS COCO annotations in form of the points of a polygon of the object borders. These points are transformed following [Hu et al., 2016, Yu et al., 2016] into an 8-dimensional spatial feature vector $\mathbf{s}_{target}$. The features contain the upper left and lower right point of the rectangular bounding box encapsulating the object, as well as, the centre point and the bounding box width and height. Therefore, the spatial location of the target object is represented as follows: $\mathbf{s}_{target} = [x_{min}, y_{min}, x_{max}, y_{max}, x_{centre}, y_{centre}, w_{box}, h_{box}]$. Note that before processing, the bounding box is normalized, by setting the centre of the image to the origin and the width and height in the interval $[-1, 1]$. The category is represented through a dense embedding $\mathbf{c}_{target}$ retrieved from a learned embedding matrix $\mathbf{E}_{category}$. The current question q_t is processed by a Recurrent Neural Network, specifically an LSTM [Hochreiter and Schmidhuber, 1997]. Each word $w_{t,\tau}$ in q_t is represented by an one hot vector, such that it can be processed into an embedding by multiplying the representation with the word embedding matrix $\mathbf{E}_{word}$. The word embeddings are sequentially processed by the recurrent neural network. The last hidden state of the LSTM is used as a language representation $\mathbf{h}_t$. Note that in the baseline version the dialogue history is not exploited in the Oracle, i.e. only the current question q_t is processed. The three representations (language, category and spatial) are concatenated and passed through an MLP with one hidden layer, activated with a ReLU. Finally, a softmax layer is applied to the three outputs of the MLP, resulting in a probability for each of the answers. The architecture is shown in Figure 4.1. Formally, the model can be expressed as follows:

$$\mathbf{s}_{target} = [x_{min}, y_{min}, x_{max}, y_{max}, x_{centre}, y_{centre}, w_{box}, h_{box}] \tag{4.1}$$

$$\mathbf{c}_{target} = \mathbf{E}_{category}\ c \tag{4.2}$$

$$\mathbf{h}_t = \text{LSTM}(q_t) \tag{4.3}$$

$$p(a_i) = \text{softmax}_i(\text{MLP}([\mathbf{s}_{target}, \mathbf{c}_{target}, \mathbf{h}_t])) \tag{4.4}$$

where $[\cdot, \cdot]$ denotes concatenation.

During training, the Negative Log Likelihood of the correct answer is minimized. Therefore the loss is defined as follows: $\mathcal{L}(\theta) = -\log p(a|\theta)$, where $p(a|\theta)$ is the probability of the correct answer class under the current set of parameters. The parameters are optimized with Adam [Kingma and Ba, 2015]. The model achieves an accuracy of 78.9% and 78.5% on the validation and test set in [De Vries et al., 2017a] and 78.82% and 78.33% in our re-implementation of the model respectively. All hyperparameters can be found in Appendix B.1.

4.1.2 Guesser

The Guesser has to select an object o_i from the set of objects $\mathcal{O}$. Each is represented by its location in the image via the spatial feature vector $\mathbf{s}_i$, which is obtained in the same way as done in the Oracle, and its object category via an embedding $\mathbf{c}_i$. Besides the object representation, the selection is based on the question-answer pairs in the

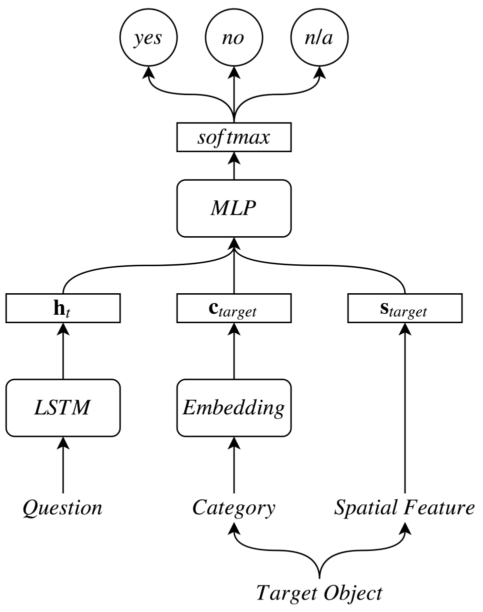


Figure 4.1: Oracle Architecture

dialogue $\mathcal{D}$. The dialogue $\mathcal{D}$ is processed by an LSTM, similar as in the case of the Oracle. The words in the questions, as well as the answers, are represented through a learned embedding $\mathbf{E}_{word}$. These representations are then sequentially processed through the LSTM. The last hidden state of the LSTM $\mathbf{h}_D$ is used as the representation of the dialogue. For each object in the image, a separate representation is computed combining $\mathbf{c}_i$ and $\mathbf{s}_i$. Therefore, the spatial feature vector $\mathbf{s}_i$, as well as a separately learned embedding for the object category $\mathbf{c}_i$ is concatenated to represent the object o_i. This representation is then processed by an MLP to obtain a final object representation $\mathbf{r}_i$ of the same dimensionality as the hidden state of the Guesser's LSTM. The weights of the MLP are shared among the variable number of objects in a game (i.e. the same weights are used to compute $\mathbf{r}_i$ for all objects). The dot product between the dialogue representation and each object representation leads to a score for each object. This score vector is finally processed by a softmax function, leading to a probability for each object, i.e. $p(o_i|\mathcal{D}, \mathbf{s}_i, \mathbf{c}_i) = \text{softmax}_i(\mathbf{h}_D \times \mathbf{r}_i)$. The architecture is shown in Figure 4.2. Formally, the Guesser model can be summarized in the following equations:

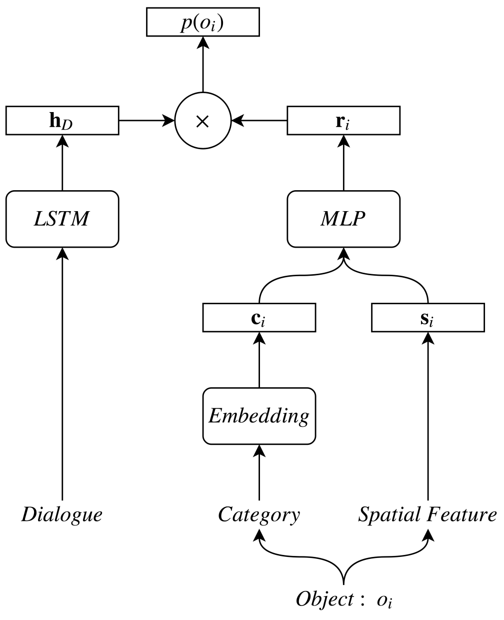


Figure 4.2: Guesser Architecture

$$\mathbf{s}_i = [x_{min}, y_{min}, x_{max}, y_{max}, x_{centre}, y_{centre}, w_{box}, h_{box}] \tag{4.5}$$
$$\mathbf{c}_i = \mathbf{E}_{category} c_i \tag{4.6}$$
$$\mathbf{h}_D = \text{LSTM}(\mathcal{D}) \tag{4.7}$$
$$\mathbf{r}_i = MLP([\mathbf{s}_i, \mathbf{c}_i]) \tag{4.8}$$
$$p(o_i) = \text{softmax}_i(\mathbf{h}_D \mathbf{r}_i^T) \tag{4.9}$$

All parameters of the Guesser are optimized by minimizing the Negative Log Likelihood of the target object, i.e. $\mathcal{L}(\theta) = -\log p(o_{target}|\theta)$. The parameters are optimized with Adam [Kingma and Ba, 2015]. The model achieves an accuracy of 62.1% and 61.3% on the validation and test set respectively in [De Vries et al., 2017a]. Our re-implementation scores 64.0% and 62.8% respectively. All hyperparameters can be found in Appendix B.2.

4.1.3 Question Generator

The Question Generator's task is to generate a question given the previous question-answer pairs and the image. The generated question has to be grounded in the image, be coherent with dialogue history (i.e. previous question-answer pairs) and most importantly, eventually the question-answer pairs are used to perform the task, identifying the target object.

In [De Vries et al., 2017a] different architectures have been experimented with, including the Hierarchical Recurrent Encoder Decoder (HRED) model [Sordoni et al., 2015a]. However, a simple RNN language model [Mikolov et al., 2010] setup that is additionally conditioned on an image feature vector achieved the best results.

This model has a visual pipeline processing the image $\mathcal{I}$ with a pretrained VGG16 [Simonyan and Zisserman, 2015] network to obtain a visual representation $\mathbf{v}$ of the image. The VGG16 network has been optimized on ImageNet [Deng et al., 2009] to categorize an image into one of 1000 classes[5]. For GuessWhat?! the final output of the network (also referred to as FC8) is used to obtain the visual representation.

In order to generate the next question q_t in the dialogue, an LSTM is used. The LSTM generates the next word in the sequence, by taking in an embedding of the previous word $w_{t,\tau-1}$ obtained through the embedding matrix $\mathbf{E}_{word}$, and the visual representation $\mathbf{v}$. Further, the previous hidden state $\mathbf{h}_{t,\tau-1}$ and cell state $\mathbf{c}_{t,\tau-1}$ are used and updated. This results in a sequence of hidden states $\mathbf{H}_t = [\mathbf{h}_{t,1}, ..., \mathbf{h}_{t,\tau}, ...\mathbf{h}_{t,m}]$ for each word in the question. This hidden state in the sequence is linearly transformed and followed by a softmax to obtain a conditional probability distribution at each timestep over all possible next tokens $p(w_{t,\tau}|w_{1,1}, ..., w_{t,\tau-1}, \mathcal{I})$ given the sequence so far and the image. The architecture is shown in Figure 4.3. Formally, the Question Generator model can is summarized in the following equations:

$$\mathbf{v} = \mathbf{W}_v \, \text{VGG16}(\mathcal{I}) \tag{4.10}$$

$$\mathbf{l}_{t,\tau} = \mathbf{E}_{word} w_{t,\tau-1} \tag{4.11}$$

$$\mathbf{h}_{t,\tau} = \text{LSTM}(\mathbf{l}_{t,\tau}, \mathbf{v}, \mathbf{h}_{t,\tau-1}, \mathbf{c}_{t,\tau-1}) \tag{4.12}$$

$$p(w_{t,\tau+1}) = \text{softmax}(\mathbf{W}\mathbf{h}_{t,\tau} + \mathbf{b}) \tag{4.13}$$

During training, the LSTM is trained with teacher-forcing [Williams and Zipser, 1989]. In this setting, the input word is always taken from the data. In order for the LSTM to predict the next token in the sequence, the first input word is a start-of-sequence token and therefore, the first output of the LSTM predicts the first actual token of the sequence.

[5]On ImageNet VGG16 achieves a top-1 error of 27.00% and top-5 error of 8.80%

The network is optimized by minimizing the Negative Loglikelihood Loss, i.e. $\mathcal{L}(\theta) = -\log p(w_{t,\tau+1}|\theta)$ and the gradients are determined according to the back-propagation through time algorithm [Rumelhart et al., 1986]. The parameters are optimized with Adam [Kingma and Ba, 2015]. All hyperparameters of the Question Generator can be found in Appendix B.3.

During inference, human dialogues are not used. Therefore, the word input to the LSTM at each timestep is the previously generated token. In [De Vries et al., 2017a], that is determined by sampling, greedy decoding and beam search. In this work, we only compare to greedy decoding.

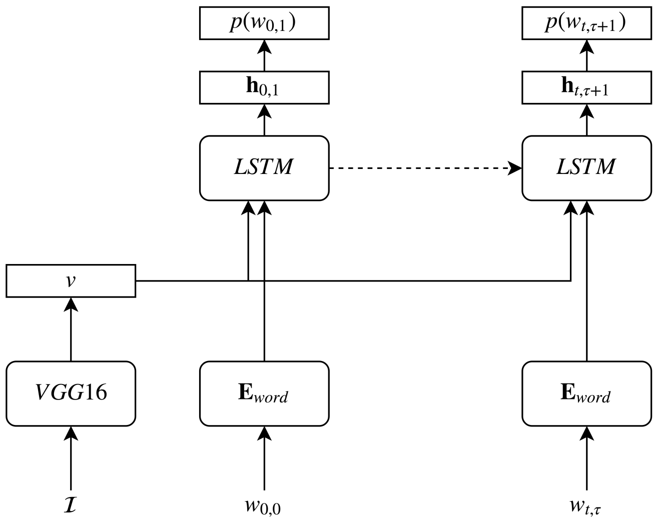


Figure 4.3: Question Generator baseline architecture. During training $w_{t,\tau}$ comes from the human dialogues, during inference, the previously generated token is used, i.e. in case of greedy decoding $w_{t,\tau} = \arg\max p(w_{t,\tau-1})$.

4.2 Belief State

4.2.1 Motivation

As discussed in the previous section, the task of the Question Generator module is to ask the next question given the previous question-answer pairs and the image. What question to ask next is learned from the human dialogues. However, learning just from the statistics of word occurrences is hard, particularly in dialogue, where the space of potential next acceptable words is vast. This also applies to GuessWhat?![6]. We argue that the process that generates the question is based on the uncertainty of the questioner agent. That is, its *belief* about what objects are still candidate target objects, given the dialogue history so far drives the language generation process.

Modelling this observation is a common approach in artificial intelligence and deep learning in particular to improve model performance. For example in machine translation, one can argue that the process that generates a target word, is heavily connected to its alignment in the source. This has been successfully addressed and modelled in the attention mechanism [Bahdanau et al., 2015]. Similarly, we also address this in the Question Generator, by providing it with an explicit representation of the current belief. That is, the process that generates the next question is the uncertainty of the questioner agent over the potential target objects. We represent this in the belief state. Therefore, the model will be equipped with a component similar to the dialogue manager in the more traditional dialogue modelling pipeline (see section 2.3). This should improve the *signal-to-noise* ratio the Question Generator experiences during training. When exploiting the belief representation, the space of potential next questions is smaller. For example, when narrowing down the set of objects by grouping them. At first, the belief state would contain an equal distribution over all objects in the image. Allowing to group them for example by finding common properties. Further, when the object category is already guessed (e.g. after the question-answer *is it a person? yes*), the model can focus on questions that disambiguate between objects of that category.

Another point of view of adding a belief state of potential target objects is that it separates *what* (i.e. semantics) to ask, from *how* (i.e. syntax) to ask. The belief state represents the objects that are still considered potential target objects under the current dialogue history and defines *what* the question should be about. Vice versa, another important skill is to verbalize this belief, by producing a question about it that can be understood by the Oracle and therefore leads to correct answers. Both skills are required to perform well on the GuessWhat?! task, and excel at any task-oriented dialogue in general.

From these perspectives, it also becomes clear, why it should be easier for the

[6]For example when imagining a plausible follow up question after the first turn in the example in Figure 3.1. Valid questions could for example query the colour or the location. However, the Questioner chooses to ask *is it partially visible?*

Question Generator to learn an effective language. Given a belief state, it is less required to reason about the dialogue history anymore for determining what to ask, as this is already done and efficiently represented in the belief state. In contrast, without a belief state, the model might need to carry on information from the first question as it might become relevant again in a later turn.

4.2.2 Model

We define the belief state as a probability distribution over the objects in the image, given the dialogue history. To obtain this, we evaluate the Guesser module at each new question-answer pair. Note that we do not use the Guesser output to determine whether the correct object has been guessed, merely we leverage it as a representation of the current uncertainty over the objects given the dialogue so far. Formally,

$$p_t(o_i) = \text{GUESSER}(q_{0:t}, a_{0:t}, \mathcal{O}) \tag{4.14}$$

where $0 : t$ is short for all timesteps up to t.

The probability distribution by itself is not a useful input to the Question Generator module yet. That is because the probabilities cannot be linked to the objects. In each game, there are a variable number of objects. Therefore, an object representation attached with the object probability is used as input to the Question Generator. In order to obtain a single belief state representation, independent of the number of objects, we opt for a weighted bag of words approach (or *bag of objects* in this case). Therefore, each object is represented by a continuous, fixed-size vector $\mathbf{r}_i$, and $\mathbf{R}$ contains all object representations of the game. This representation is then multiplied by the respective probability. Finally, all the weighted representations are summed up. This can be efficiently computed by the dot product between the matrix of object representations $\mathbf{R}$ and the probability vector over the objects, resulting in the belief state $\mathbf{b}_t$.:

$$\mathbf{b}_t = \sum_i \mathbf{r}_i * p_t(o_i) = \mathbf{R}^T \times \mathbf{p}_t \tag{4.15}$$

The belief state is provided to the Question Generator by concatenating it to the input at every timestep. Therefore, equation 4.12 is updated such that the new hidden state is also conditioned on the belief state:

$$\mathbf{h}_{t,\tau} = \text{LSTM}(\mathbf{l}_{t,\tau}, \mathbf{v}, \mathbf{b}_t, \mathbf{h}_{t,\tau-1}, \mathbf{c}_{t,\tau-1}) \tag{4.16}$$

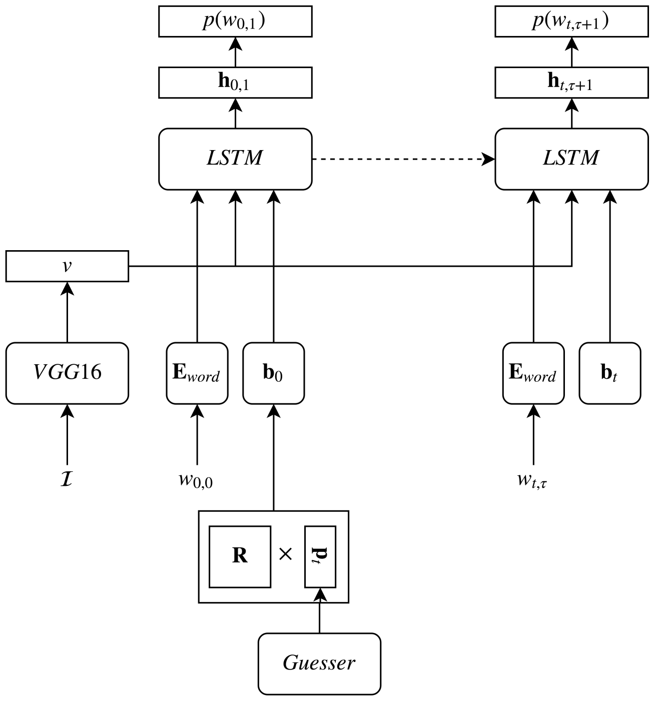


Figure 4.4: Architecture of Question Generator with explicit belief state representation, updated after each new question-answer pair.

Figure 4.4 shows the architecture visually. Note that the belief state is updated after each new question-answer pair.

Besides directly concatenating the belief state to the word embedding, we also further use it to refine the visual representation. Therefore, we extract visual features from the *conv_4* layer of a ResNet-152 [He et al., 2016], pretrained on ImageNet [Deng et al., 2009][7]. The resulting features for each image are in $\mathbb{R}^{14\times14\times1024}$. Since we do not obtain the visual features from a fully connected layer as in the baseline setting, the visual features still have spatial information. We make use of an attention mechanism, that lets the network learn on which spatial dimension it should focus. We choose the Multimodal Lowrank Bilinear (MLB) [Kim et al., 2016a] attention mechanism, which has shown promising results in multimodal settings [Kafle and Kanan, 2017]. We flatten the visual annotation to a 2d matrix $A \in \mathbb{R}^{196\times1024}$. We then obtain scores for each of the image regions, by first linearly transforming them and the belief state into vectors in $\mathbb{R}^{h\times1}$, where h is the hidden size of the attention mechanism. Further, the results are activated with tanh and then element-wise multiplied,

[7]On ImageNet ResNet-152 achieves a top-1 error of 22.16% and a top-5 error of 6.16%.

and again linearly transformed into a scalar for each image region. The scalars are now processed into a probability distribution with a softmax. Finally, the dot product between the region probabilities and the visual features results in a visual feature vector $\mathbf{v}_t \in \mathbb{R}^{1024}$. Note that the belief state changes after each question-answer pair. Therefore, we also recompute the attention and resulting visual features. Formally:

$$s_i = \mathbf{w}_s \times (\tanh(\mathbf{W}_a \times \mathbf{A}_i^T) * \tanh(\mathbf{W}_b \times \mathbf{b}_t^T)) \tag{4.17}$$

$$\mathbf{v}_t = \text{softmax}(\mathbf{s}) \times \mathbf{A} \tag{4.18}$$

where $\mathbf{A}_i \in \mathbb{R}^{1\times 1024}$ is the annotation vector in location i, $\mathbf{W}_a \in \mathbb{R}^{h\times 1024}$, $\mathbf{W}_b \in \mathbb{R}^{h\times|\mathbf{b}|}$, $\mathbf{w}_s \in \mathbb{R}^h$ and $\mathbf{s} \in \mathbb{R}^{196}$. This visual vector is then used as in the baseline version, i.e. passed through a linear layer and then concatenated to the LSTM input.

4.2.3 Application to other Task-Oriented Dialogue Settings

In the previous section, we described how the model can be applied to the GuessWhat?! task. In this section, we provide more detail on how the approach can be transferred to other task-oriented settings.

Any task-oriented dialogue system is defined by its task $\mathcal{K}$, that usually can be automatically evaluated. Further, the task might be factored into multiple subtasks $k_1, ..., k_n$ that can be performed individually. For example in a restaurant setting, where the dialogue system shall recommend a restaurant to a user, subtasks might be to determine the user's preferences for location, type of cuisine and price [Bordes and Weston, 2016].

From a dataset of dialogues with labels for the task, classifiers $\mathcal{C}_k$ can be trained to produce a probability distribution p_k over the task outcomes. By factoring the task into multiple subtasks this can be represented in a smaller space. In the above restaurant setting, instead of producing a probability distribution over all restaurants, a distribution over each of the properties (location, cuisine type, price, etc.) is more efficient. Further, this will also help the agent as tasks that still have a high uncertainty are separated from solved subtasks. Eventually, as detailed for the GuessWhat?! setting, the language generation can be conditioned on these probability distributions by learning a representation $\mathbf{r}_{k,i}$ for each task outcome, and weighing it with the respective task probability (see equation 4.15).

5 Experiments and Results

5.1 Belief State Object Representations

In the previous chapter, we introduced the general architecture of the Question Generator with an explicit belief state. We now conduct several experiments for different representations.

5.1.1 Object Representations from Annotations

One way to represent the set of objects is by learning an embedding for each category $\mathbf{E}_{category}$, and we will, therefore, refer to this belief state representation as *Category*. Practically, by only distinguishing objects by their category, all games can be solved where the target object is the only instance of that category. The Question Generator can enumerate all the object categories present in the image, and query each. When the Oracle responds with *yes* to a question about an object category, the game can be solved confidently (e.g. if there is only a single dog on the image, the question identifying the target object category will suffice to solve the game). However, this also exposes the disadvantage of this representation. It is not possible to disambiguate between objects of the same category.

Therefore, we can make use of another object annotation, namely the spatial location of each object. Since we will eventually sum up the object representations weighted by the belief probabilities, using only the spatial location is not practical[8]. Therefore, we can extend the above approach by concatenating the spatial feature vector to the category embedding and processing it through an MLP. We will refer to

[8]For example, it would not be possible to distinguish between two objects in the centre of the image and an object on the left and right. If the probabilities for objects are similar, the resulting hypothesis of the object location would be in the centre in both cases.

this approach as *Category + Spatial*. The object is represented in the same fashion as done in case of the Oracle and the Guesser.

Finally, instead of learning the object representation from scratch, the representation learned by the Guesser can also be reused. Therefore, we retrieve the output of the Guesser's MLP for each object in the image (see equation 4.8). Subsequently, we apply a linear transformation to obtain the object representation for the belief state. We refer to this approach as *Guesser Object Representation* and to the model as *Belief*.

The latter two approaches allow the Question Generator to also do simple spatial reasoning of the objects. The absolute locations are directly accessible. Therefore, distinguishing between two objects of the same category, but with different locations in the image (i.e. left and right) is possible. However, distinguishing objects by their relative location to each other remains difficult, since it requires a multi-hop reasoning process. For example in the question *is it below the table?*, first the context, in this case the table, has to be retrieved, only then the objects spatial location. However, the belief state does not provide information on the objects in context.

Experimental Setup

For all object representations, we search the parameters of the visual representation (i.e. the size of $\mathbf{W}_v$ in equation 4.10) and the object representation (i.e. the size of $\mathbf{R}$ in equation 4.15). The former we search in $[0, 64, 128, 256, 512]$, therefore also allowing no visual representation from the VGG16 FC8 features. We search the object representation in $[64, 128, 256, 512]$. Note that depending on these sizes, also the input size of the LSTM changes (i.e. $\mathbf{W}_{fx}, \mathbf{W}_{ix}, \mathbf{W}_{ox}$ and $\mathbf{W}_{cx}$ in equation 2.7 to 2.10)[9]. We evaluate the model performance as done with the baseline Question Generator, by tracking the loss on the validation set. Subsequently, we use the model with the lowest validation loss and test its performance on the task by generating questions. We report the performance of the model when asking a total of 5 questions in order to compare with the baseline. Further, we search the number of questions n in $\{5, ..., 12\}$ of the best models on the validation set and report the of the best number of questions on both sets.

[9]An overview of the number of parameters of the best performing models is provided in the appendix in Table C.1

Results

The results for the best performing model in the hyperparameter search are reported in Table 5.1. We can see that all models with a belief state perform well above the baseline performance of [De Vries et al., 2017a]. Besides task success, also the cross entropy improves. We find that the best models do not use the visual representation from VGG16 FC8. We conjecture that this because the belief state in this setup and the FC8 features highly overlap in what they represent, namely the object categories of the image. Note that FC8 is the final layer of VGG16, therefore on the ImageNet setting, it provides a probability distribution over the respective categories. For a more detailed analysis of the results we refer to chapter 6.

Belief State Representation	$\|\mathbf{W}_v\|$	$\|\mathbf{R}\|$	Cross Entropy	Validation Accuracy (n=5)	Test Accuracy (n=5)	Validation Accuracy (best n)	Test Accuracy (best n)
None / Baseline	512	n/a	1.475	42.90%	42.55%	43.05% (n=6)	42.55%
Category	0	64	1.443	49.48%	48.30%	49.94% (n=8)	49.60%
Category + Spatial	0	256	1.433	50.00%	**49.49%**	50.78% (n=8)	**50.23%**
Guesser Obj. Rep.	0	256	1.436	49.16%	48.57%	50.06% (n=8)	49.06%

Table 5.1: Results for Question Generator with explicit belief state with varying object representations. Cross entropy is measured on the validation set. $|\mathbf{W}_v|$ represents the size of the visual embedding, $|\mathbf{R}|$ the size of the object representation.

5.1.2 Fine Tuning

In previous experiments we kept the parameters of the Guesser, i.e. the parameters generating the belief state, fixed. In this experiment, we do not stop the gradient at the level of the Guesser. We unfreeze the parameters and update them through the language modelling loss of the Question Generator, therefore, fine-tuning them end-to-end. Note that we restart the training procedure, i.e. we randomly initialize the Question Generator weights. We refer to this model as *Belief+FineTune*.

Experimental Setup

For this experiment, we take the hyperparameter settings of the eight best performing models of the *Category*, *Category + Spatial* and *Guesser Object Rep.* setting. The hyperparameters and results for all these models with the frozen belief state can be found in the Appendix (see Table B.4). Note that as common in fine tuning, we reduce the learning rate updating the Guesser parameters ($1e^{-5}$). Further, this will require to copy the Guesser network. One module will be fine-tuned and produces the belief state. Another will be kept fixed and performs the final guessing of the object.

Results

The results for models with a fine-tuned Guesser can be found in Table 5.2. All models further improve over their settings with a fixed belief state. The cross entropy improves for the *Category* setting and stays the same for the other belief state representations. In contrast to the models with a fixed belief state, two of the best performing models use the VGG16 FC8 visual representation. We conjecture, that the fine-tuned belief state now learns a complementary representation. For example, instead of having a uniform distribution over the objects, the belief probabilities might be more focused. Therefore, the VGG16 features provide a holistic view of the image, while the belief state provides the best guess and its representation under the current dialogue. Again, we refer to the analysis chapter 6 for a detailed review of the results.

Belief State Representation	$\lvert\mathbf{W}_v\rvert$	$\lvert\mathbf{R}\rvert$	Cross Entropy	Validation Accuracy (n=5)	Test Accuracy (n=5)	Validation Accuracy (best n)	Test Accuracy (best n)
Category	0	64	1.428	54.85%	53.83%	55.65% (n=8)	54.65%
Category + Spatial	128	512	1.432	55.08%	**54.75%**	56.15% (n=7)	**55.63%**
Guesser Obj. Rep.	256	512	1.437	54.73%	54.46%	55.37% (n=7)	55.22%

Table 5.2: Results for Question Generator with fine-tuned belief state.

5.1.3 Visual Attention

As described in section 4.2.2, we experiment with a visual attention mechanism. We use the belief state as query, and annotations from the conv_4 layer of ResNet-152.

Experimental Setup

For all experiments, we use the belief state representation that achieved the best results in the previous section, i.e. the belief state representation that learns a representation from the object category and a spatial embedding from scratch and has 512 dimensions (row 2, Table 5.2).

We investigate how the model performs with visual features from the attention mechanism, with and without concatenating the belief state to the input. Further, we research the impact of fine-tuning the belief state representation in this setting. For all experiments, we search the size of the visual embedding in $[64, 128, 256, 512]$. Additionally, as a baseline, we train a model that does not query with the belief state, but with the hidden state of the Question Generator. We, therefore, use the hidden state after each answer token input. For this model, we also search for the size of the visual embedding in $[64, 128, 256, 512]$.

Results

The results for the Question Generator with attention over the visual locations are reported in Table 5.3. First of all, the model that queries the visual representation with the hidden state of the Question Generator performs better than the baseline by [De Vries et al., 2017a] and our re-implementation. We find that all models that query the visual features with the belief state outperform the visual attention baseline model. Further, we observe that both additionally concatenating the belief state and fine-tuning it improves performance. However, the performance of the model that combines these settings is only on par with the belief model with the static visual features from VGG16 FC8 (see Table 5.2).

Attention Query	Fine Tuning	$\lvert\mathbf{W}_v\rvert$	$\lvert\mathbf{R}\rvert$	Cross Entropy	Validation Accuracy (n=5)	Test Accuracy (n=5)	Validation Accuracy (best n)	Test Accuracy (best n)
Hidden State	n/a	256	n/a	1.450	43.78%	43.19%	43.84% (n=6)	43.13%
Category + Spatial	✗	64	0	1.445	45.30%	44.38%	45.69% (n=6)	45.06%
Category + Spatial	✓	256	0	1.440	45.75%	45.86%	46.09% (n=6)	46.04%
Category + Spatial	✗	64	512	1.430	48.49%	47.66%	49.38% (n=8)	48.78%
Category + Spatial	✓	128	512	1.422	55.21%	**54.23%**	56.10% (n=7)	**55.11%**

Table 5.3: Results for Question Generator with visual attention. The first model serves as a baseline, the hidden state of the LSTM is used to query the visual annotations. The models in row 2 and 3 do not have the belief state concatenated to the LSTM input but only use it to query the visual annotations. Finally, the models in row 4 and 5 also have the belief state as input to the LSTM.

5.2 Ablation Studies

5.2.1 Bag of Objects

In the previous experiments, the proposed model was provided with the list of objects in the image. However, when the dataset was collected, this list is not explicitly provided to the human player. Instead, only the image is given, from which a human player then detects the objects in the scene.

One way of providing the list of objects to the Question Generator that is similar to the belief state approach is to use an unweighted bag of objects approach over the object representation. This is equivalent to the belief state with a uniform probability distribution at every timestep. Therefore, all object embeddings are summed with the same weight. This can be thought of, as a perfect visual perception for the categories, as they are available to the Question Generator at all times and do not need to be extracted from the visual features. We concatenate the bag of objects to the word embedding at every timestep. Similar to section 5.1.1, we search the visual embedding in $[0, 64, 128, 256, 512]$ and the size of the bag of objects embedding dimension in $[64, 128, 256, 512]$. This experiment will demonstrate how much performance simply comes from being provided with the list of objects, and how much gain comes from the belief state itself.

5.2.2 Belief State over All Object Categories

In case of a bag of objects representation, we rely on the object annotations. Another approach is to not assume this list, instead provide a belief state over all possible 81 object categories in GuessWhat?!. Thereby, we do not provide the Question Generator with any interpretation of the image. Thus, we train a Guesser with the goal to only predict the category of the target object from the human dialogues. This model achieves an accuracy of 80.52% on the validation set. Subsequently, the probability distribution over all objects is used as belief state and the Question Generator is trained.

The results of both experiments are shown in Table 5.4. A more detailed analysis is provided in chapter 6.

Model	Belief State Representation	$\|\mathbf{W}_v\|$	$\|\mathbf{R}\|$	Cross Entropy	Validation Accuracy (n=5)	Validation Accuracy (best n)
Baseline	n/a	512	n/a	1.475	42.90%	43.05% (n=6)
Belief	Category	0	64	1.443	49.48%	49.94% (n=8)
All Categories	Category	64	64	1.456	44.52%	44.75% (n=8)
Belief	Category+Spatial	0	256	1.433	50.00%	50.78% (n=8)
Bag of Objects	Category+Spatial	64	64	1.431	45.32%	46.40% (n=7)

Table 5.4: Results for ablation studies and the respective belief model for comparison.

6 Analysis

We now perform detailed analyses of the results of the previous chapter. We, therefore, choose our implementation of [De Vries et al., 2017a] for comparison (which we refer to as *Baseline*, Table 5.1 row 1) and the best performing model with a belief state obtained from the original Guesser (*Belief*, Table 5.1 row 3) and a fine-tuned belief state (*Belief+FineTune*, Table 5.2 row 2). The two belief models both use the *Category+Spatial* object representation. Note that we do not use models with a visual attention mechanism, as the results are on par with the simpler visual representation proposed in [De Vries et al., 2017a]. All analyses are conducted on the validation set. Further, for comparison, we use the outcome after 8 questions if not stated otherwise.

6.1 Task Success

We first take an overview of the successful and unsuccessful games by the different models. Figure 6.1 shows Venn diagrams of successful (left) and unsuccessful (right) games.

We notice that the majority of games that are solved are solved by all models (28.7%), similarly for unsuccessful games, where 29.76% are not completed successfully by all models. Further, the amount of games only the baseline solves is small (4.8%) and the number of games only the baseline fails is quite large (12.6%). From these numbers, we can conclude that the models mainly learned additional ways to solve the task at hand and keep the skills learned by the baseline.

We now take a closer look at the statistics of the games solved by each model (see Table 6.1). When comparing the statistics of the successful games to the entire validation set, we notice that in general easier games are solved. That is games with

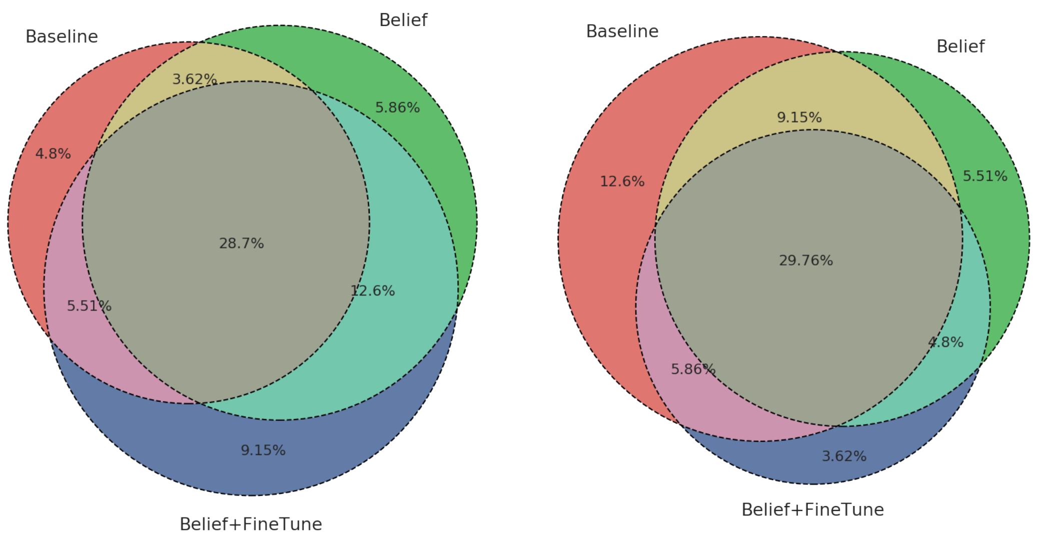


Figure 6.1: Venn diagram of the relative number of games that are solved (left) and that are guessed incorrectly (right) by the different models.

fewer number of objects, fewer number of objects with the same category as the target and bigger target object areas. When we compare the statistics of the baseline to the belief models, we can see that they are better able to deal with more objects and games with many different object categories. In the appendix, we provide further statistics on games solved by one model but not by another (see Table C.2).

6.1.1 Task Success by Number of Questions

Next, we compare how effective the generated questions are in terms of task success. We, therefore, take the dialogue up to turn t and evaluate it with the Guesser (see Figure 6.2). We notice that the baseline model solves the majority of its games within four rounds (42.36% task success). Barely any additional games are solved after the fourth round. For the models with belief state, we observe that task performance plateaus about 1 question-answer pair later.

This means that the belief models learn the long term dependencies in the dialogue more effectively. This can be concluded as a game can only switch to successful if the produced question and answer add enough information to disambiguate the target object. Therefore, the questions at later timesteps of the belief models must be more informative to the Guesser, as questions by the baseline.

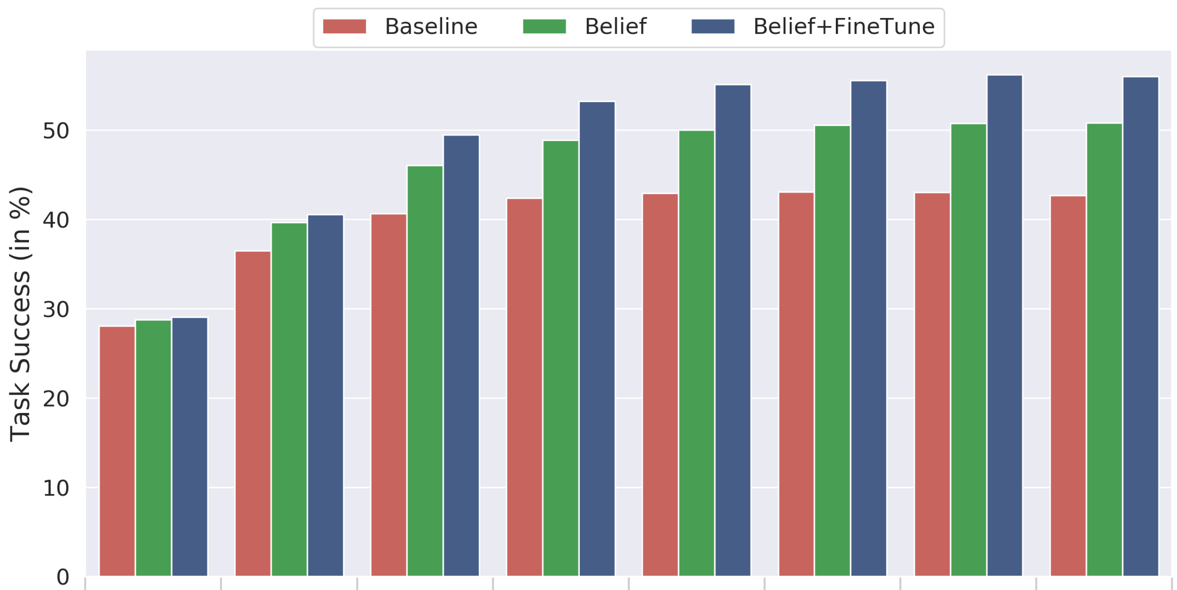
	All Games	Baseline	Belief	Belief +FineTune
Num Games	23,739	10,122	12,056	13,284
Num Objects	8.54 (±4.67)	6.84 (±4.07)	7.02 (±4.19)	7.07 (±4.14)
Num Object Categories	3.49 (±1.72)	3.28 (±1.49)	3.38 (±1.56)	3.51 (±1.61)
Num Instances of Target Cat.	3.99 (±3.59)	2.66 (±2.62)	2.68 (±2.63)	2.52 (±2.50)
Log of Target Object Area	8.64 (±2.00)	9.13 (±2.04)	9.17 (±2.02)	9.08 (±2.02)

Table 6.1: Game statistics of successfully solved games by model. Except for *Num Games*, all values are averages and standard deviations are in parentheses.

6.1.2 Oracle for Number of Questions

By evaluating the Guesser after each question-answer pair, we can also investigate what the influence of the fixed number of questions is. In [De Vries et al., 2017a] as well as in this work, we let the Question Generator ask a fixed number of questions, neglecting the fact that humans have the option to ask a variable number of questions. We search the optimal number of questions per game by evaluating the Guesser at every question-answer pair and stop the generation when the Guesser is able to solve the game correctly. Note that we do not go beyond the maximum of 8 questions. We also perform this procedure on the human dialogues with the trained Guesser (*Human+Guesser*). Results are shown in Table 6.2.

Interestingly we generally see a big jump in performance when we can stop the dialogue prematurely. Most interesting however is the comparison to the *Human+Guesser* result. We observe that the model with a fine-tuned belief state even slightly surpasses the Guesser performance on human dialogues. This means, that from a task success perspective, this model cannot learn to produce better questions anymore by getting closer to the human data. When we unroll this analysis by question-answer pair, i.e. investigate when then model solves the game correctly the first time, we can observe a difference between the model with fine-tuned belief state and the human dialogues (see Figure 6.3). The questions asked by the fine-tuned belief state model in the first three rounds surpass the performance of the human dialogues, however, in later rounds this effect is inverse. This shows that the questions in later rounds of humans are better than in the generations of our model.



	1	2	3	4	5	6	7	8
Baseline	28.07	36.48	40.64	42.36	42.9	43.05	42.99	42.64
Belief	28.75	39.65	46.03	48.86	50.0	50.55	50.72	50.79
Belief+FineTune	29.03	40.51	49.45	53.19	55.08	55.56	56.16	55.96

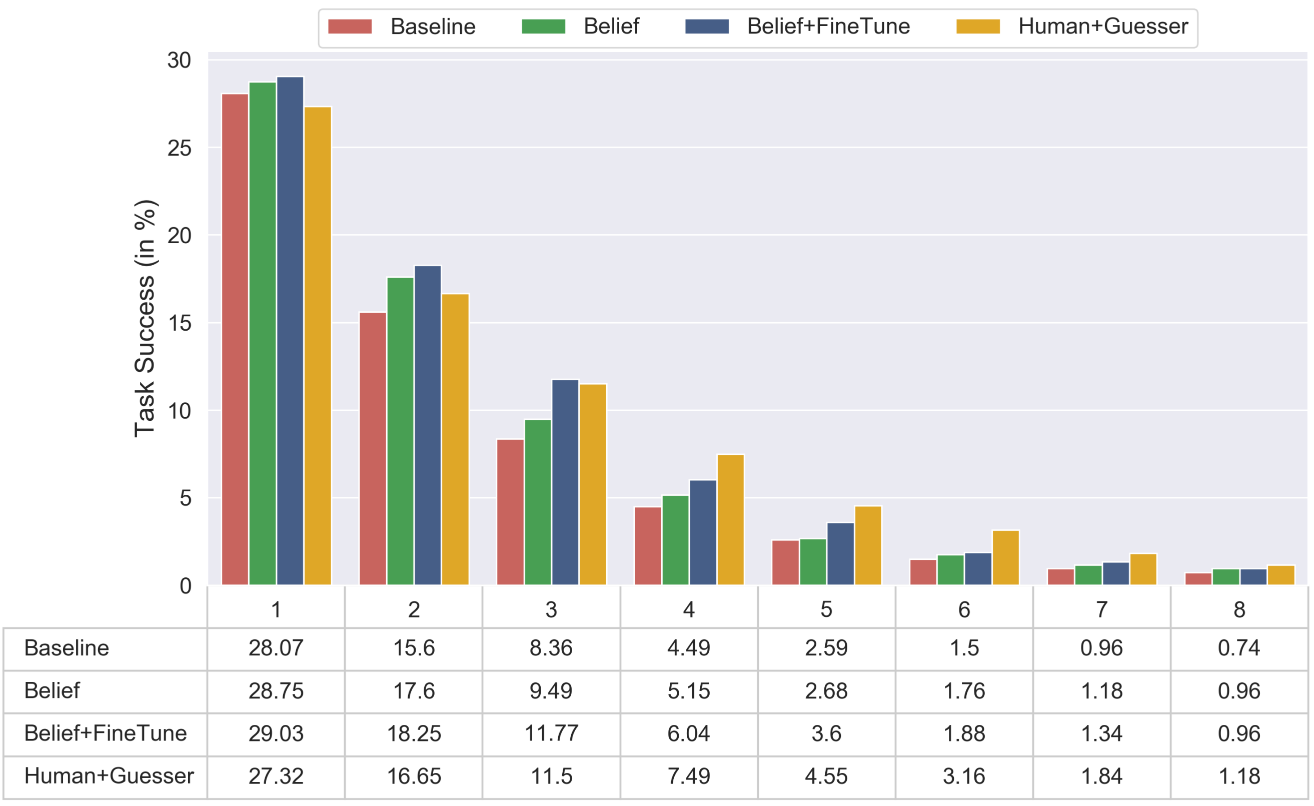
Figure 6.2: Task success of the different models after each question-answer pair.

The results also unveil the potential of our approach in [Shekhar et al., 2018], where we introduce a Decider module to predict when to stop the dialogue. A model that can perfectly predict when to stop the dialogue can not only improve the language but also task performance.

However, the Human+Guesser result is also concerning. By qualitatively looking at human dialogues we observe that in the majority of cases humans initiate the guessing phase when the object is clear, only rarely humans ask one additional question confirming their hypothesis (e.g. in the example in Figure 3.1, it is impossible to confidently predict the right object before the last/fourth question). Therefore, human dialogues should not be solvable by the Guesser module with much fewer questions on average. Most likely, the Guesser overfits on the data, for example exploiting the unbalanced target categories.

	Baseline	Belief	Belief+ FineTune	Human+ Guesser	Human
Task Success	62.30%	67.58%	72.88%	72.13%	90.80%
Avg. (StD.) Num Q	2.19 (±1.55)	2.26 (±1.59)	2.35 (±1.60)	2.61 (±1.96)	5.07 (±3.23)

Table 6.2: Task success with an Oracle on the number of questions on the validation set.



	1	2	3	4	5	6	7	8
Baseline	28.07	15.6	8.36	4.49	2.59	1.5	0.96	0.74
Belief	28.75	17.6	9.49	5.15	2.68	1.76	1.18	0.96
Belief+FineTune	29.03	18.25	11.77	6.04	3.6	1.88	1.34	0.96
Human+Guesser	27.32	16.65	11.5	7.49	4.55	3.16	1.84	1.18

Figure 6.3: Relative number of games that are solved the first time after t question-answer pairs.

6.1.3 Number of Objects

One dimension of complexity in GuessWhat?! is the number of objects in the image. Therefore, we analyse the model performance with respect to the number of objects, and the number of objects of the same category as the target object (see Figure 6.4).

In order to relate these results, we provide an overview of the number of games by the number of objects and the number of objects with the same category as the target object in the Appendix (see Figure C.1 and C.2).

In case of the number of objects, we see that all models consistently perform well above a random baseline. Not surprisingly, performance decreases with more objects. Further, the Belief model outperforms the baseline with any number of objects, and in turn, the model with a fine-tuned belief state outperforms the Belief model.

For the analysis on the number of objects with the same category as the target object, we notice a strong performance gain for the Belief+FineTune model when there is no other object of the same category as the target object. The model solves 88.04% of these games as opposed to the Belief model with 74.14% and the Baseline with 63.82% successful games. This explains a big part of the overall performance improvement, as 31.02% of all games in the validation set fall into this category. For games with 5 or more objects with the same category as the target object, we observe that fine-tuning the belief state does not lead to improvements, the Belief model performs on par, or even slightly better. This effect occurs as for the part of the belief state representation that comes from the object category, a fine-tuned belief state will not help because the disambiguation between objects cannot happen on a category level. Therefore, the disambiguation must happen on the spatial level. However, representing that many locations is hard.

6.2 Quantitative Linguistic Analysis

In this section, we analyse the language of the generated dialogues. Table 6.3 shows linguistic measures calculated on the generations of the different models. We observe that the size of the vocabulary compared to humans is much smaller. Recall that humans use $4,919$ different tokens that occur at least 3 times after tokenization[10]. Nevertheless, the vocabulary of the Belief+FineTune model is 22% bigger than that of the baseline and Belief model. Further, it generates a greater number of unique questions. Also, we notice that the average number of tokens slightly increases with respect to the baseline.

[10]Tokenization performed with NLTK [Bird et al., 2009] Tweet Tokenizer.

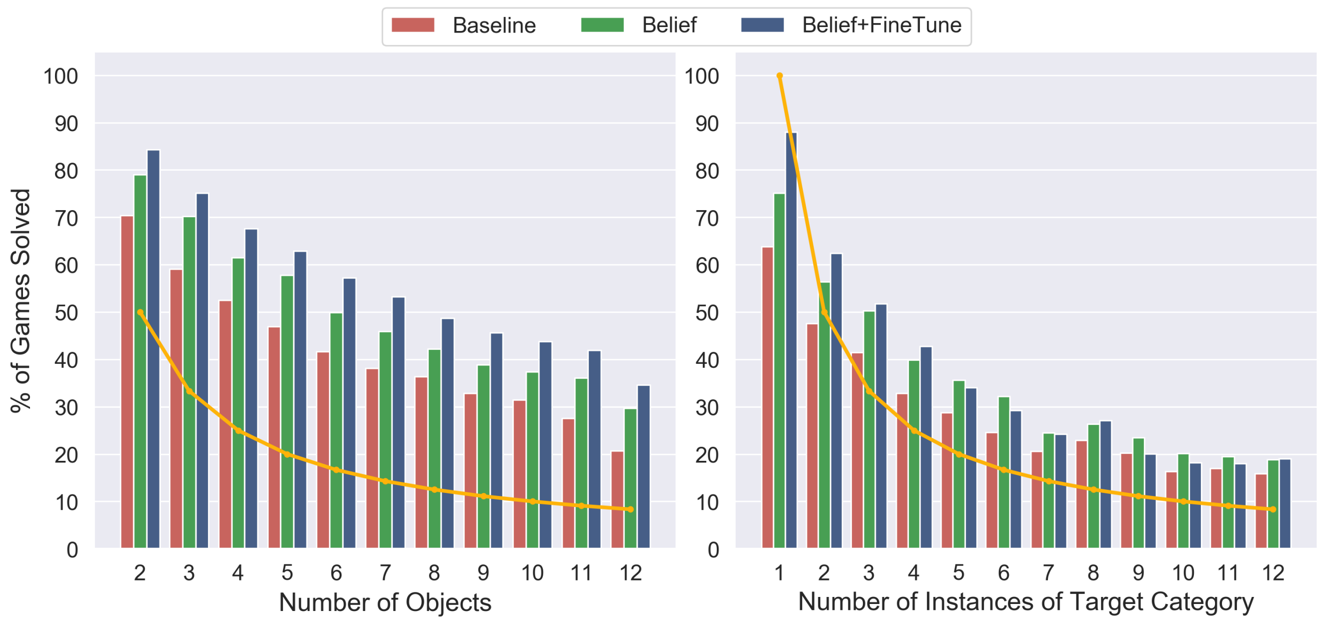


Figure 6.4: Relative number of games solved with respect to the number of objects (left) and the number of objects with the same category as the target object (right). The yellow line indicates random performance. On the right side, random performance is defined over the number of objects with the same category.

We further take a look at the contents of the questions, by checking if a certain word is present. We use the same word list for super categories, categories, and colours as in [Shekhar et al., 2019].

6.2.1 Definite and Indefinite Determiners

We further analyse the language by providing sunburst diagrams of all models (see section C.5 in the appendix). We notice that there is a shift in the distribution of the use of indefinite and definite determiners between the baseline (Figure C.8) and the two belief models (see Figure C.9 and C.10). We investigate this further by measuring the number of questions with an indefinite determiner (*a*, *an* and *any*) and a definite determiner (*the*) by dialogue turn (see Figure 6.5). We notice that all models change from indefinite to definite determiner over the course of the dialogue. This is logical, particularly for games with multiple objects of the same category as the target. After determining the object category, the agent needs to refer to a specific

	Baseline	Belief	Belief +FineTune
Vocabulary Size	368	368	449
Unique Questions	2,145	1,896	3,161
Avg. (StD.) Question Length	6.32 (±1.7)	6.59 (±1.8)	6.60 (±1.8)
Questions answered w/ yes	38.86%	58.30%	62.63%
Questions w/ Super Category	26.30%	31.68%	22.51%
Questions w/ Category	53.88%	49.03%	48.53%
Questions w/ Colour	8.38%	6.86%	7.98%

Table 6.3: Quantitative Measures on dialogues generated by different models. Token list for Super Category, Category and Color taken from [Shekhar et al., 2019].

instance of an object, therefore switching from indefinite to definite determiner. For the belief models, this shift is amplified. From the 5th question, only less than 20% of the questions contain an indefinite determiner and more than 80% a definite determiner.

6.2.2 Repetitions

Similar to [Shekhar et al., 2019] we analyse the number of repetitions of questions generated by the models. Therefore, we check for exact string matches in the generated questions. We measure the number of games that contain repetitions and the number of repeated questions in total. First, we take a look at full games, i.e. we regard all questions in the dialogue (see first row in Table 6.4).

We notice that almost all games contain at least one repetition. Further for the baseline, 44.88% of all questions are repetitions. Here we can observe an improvement for the belief models, where only 38.77% (Belief) or even 32.94% (Belief+FineTune) of questions are repetitions.

From looking at qualitative examples we observe that repetitions mostly occur at the end of dialogues, often when the game is already solved (e.g. see Figure 6.6). Note that, the models are not trained on this phenomena, as the training data does not contain questions after the game has been solved because human players can stop the game at any time. We conjecture that this is one of the reasons the models

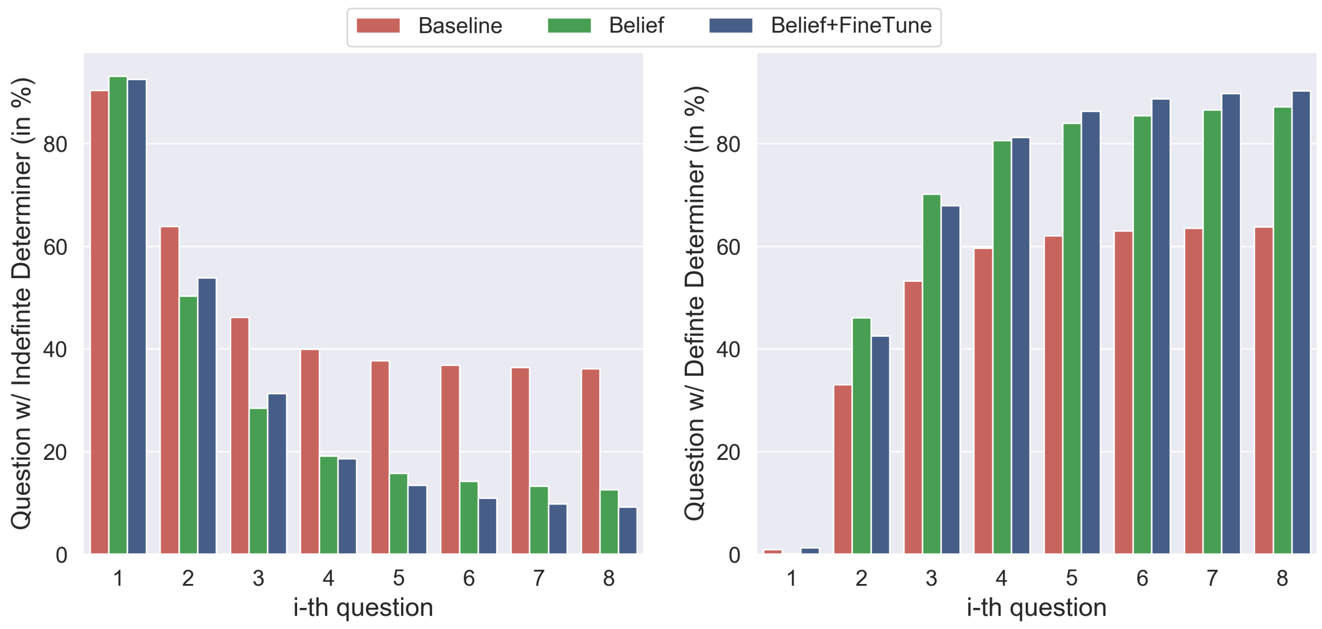


Figure 6.5: Relative frequency of questions with an indefinite determiner (left) (*a*, *an*, *any*) and a definite determiner (right) (*the*).

generate repetitions in the first place. Therefore, we also look at the repetitions when we disregard questions generated after the Guesser predicted the correct target object for the first time (see second row in Table 6.4).

The number of games that contain repetitions now drastically decreases for all models. Further, the difference between the models also becomes clearer. While the baseline still has 51.97% of games with repetitions, the number drops 7.71% points for the Belief model, and 8.08% for the Belief+FineTune model. Similarly, for the total number of repetitions. This observation confirms our hypothesis, that the belief models are better able to learn long time dependencies as they are able to longer generate questions without repeating themselves.

These observations are also shown qualitatively in Figure 6.6. We see that the baseline repeats questions (*is it a skateboard?* and *is it the one on the right?*). Note that these repetitions occur before the target is identified. It also fails after identifying the target category to ask a discriminative question to distinguish between the different cars. In contrast, the belief models generate a greater variety of questions and are eventually both able to identify the target object. Note that, in the belief model there are also repeated questions, however, they appear at the end when the

		Baseline	Belief	Belief+ FineTune
Full Dialogues	Games w/ Repeated Q	23,540 (99.16%)	22,978 (96.79%)	23,025 (96.99%)
	Total Repeated Q	10,655 (44.88%)	9,204 (38.77%)	7,819 (32.94%)
Up to 1st correct Guess	Games w/ Repeated Q	98,694 (51.97%)	84,062 (44.26%)	83,354 (43.89%)
	Total Repeated Q	43,714 (41.00%)	32,580 (33.29%)	27,243 (29.55%)

Table 6.4: Number of games with repetitions and the total number of questions that are repeated within a dialogue. The first row shows results for full dialogues, i.e. including all 8 questions. Contrary, for the second row, we only regard questions that are generated before the Guesser is able to predict the target object from the dialogue. Note that therefore, the denominator for the total number of repeated questions in order to obtain the relative frequency differs from model to model.

target object is already identified by category (*car*) and location (*left*). Further, in this example we can also observe the switch from indefinite determiner to definite determiner. The definite determiner is used after the target category is identified.

6.2.3 Influence of Belief State on Category Questions

We now analyse whether the belief state has a direct influence on the question that is generated (Table 6.5). Therefore, we take the $\arg\max$ of the belief probabilities after every turn in the dialogue and determine what category is currently predicted by the belief module. For the baseline, we evaluate the Guesser. We then check if the category string of the $\arg\max$ is used in the question[11]. We perform this first on all games in the validation set (*All Games*). However, the problem with using all games is that games where there are multiple objects of the same category as the target, the generated questions also have to disambiguate between these objects. This is most likely done by referring to object properties or their spatial location, but not necessarily by their object category. Therefore, we also perform the same analysis on games with only one object of the target category (we refer to this as *Single Target*

[11]A positive example would be: The $\arg\max$ of Guesser probabilities yields *person* and the question is *is it a person ?*. Note that we also allow partial matches, for example, if the category is *tennis racket* and the question contains *racket*.

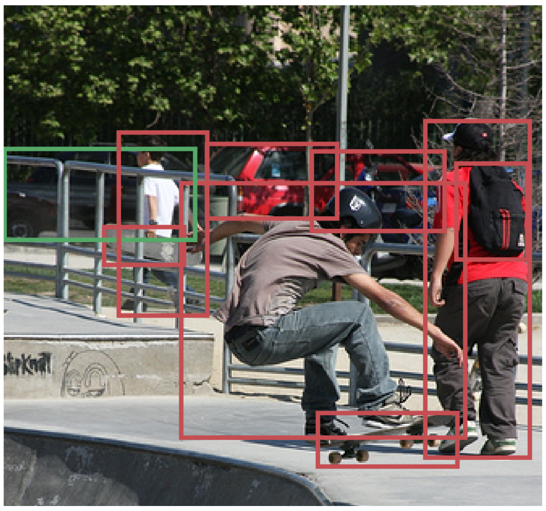


Figure 6.6: Game ID: 199807, Target Category: Car

Baseline	**Belief**	**Belief+FineTune**
is it a person? no	is it a person? no	is it a person? no
is it a skateboard? no	is it a skateboard? no	is it a car? yes
is it a skateboard? no	is it a car? yes	is it the one in the middle? no
is it a car? yes	is it the one on the left? yes	is it the one on the right? no
is it the one on the right? no	is it the whole car? yes	is it the one on the left? yes
is it the one on the right? no	is it the whole car? yes	the one that is cut off? yes
is it the one on the right? no	is it the car on the left? yes	the whole car? yes
is it the one on the right? no	is it the whole car? yes	the whole car? yes
→ Failure	→ Success	→ Success

Cat. Instance in the Table). These are about 31% of all games. We further slice the analysis by considering full dialogues, and only dialogues up to the turn where the Guesser is able for the first time to correctly predict the object.

First, we notice that the relative frequencies for the baseline in all settings is lower than that of the belief models. Note that overall, the models ask about the same number of questions about object categories (see Table 6.3). This clearly shows that the belief state has a direct influence on what the Question Generator asks about.

Furthermore, we notice in games where it is only necessary to determine the object category, about every second question of Belief+FineTune has a match with the predicted category of the belief state. These frequencies are more than twice as

Games	Up to 1st correct Guess	Baseline	Belief	Belief+ FineTune
All Games	✗	29.99%	50.04%	42.43%
All Games	✓	26.14%	43.63%	39.63%
Single Target Cat. Instance	✗	26.06%	47.70%	52.11%
Single Target Cat. Instance	✓	17.16%	33.70%	49.72%

Table 6.5: Relative number questions that contain the object category of $\arg\max$ of the belief probabilities.

high as those of the baseline model.

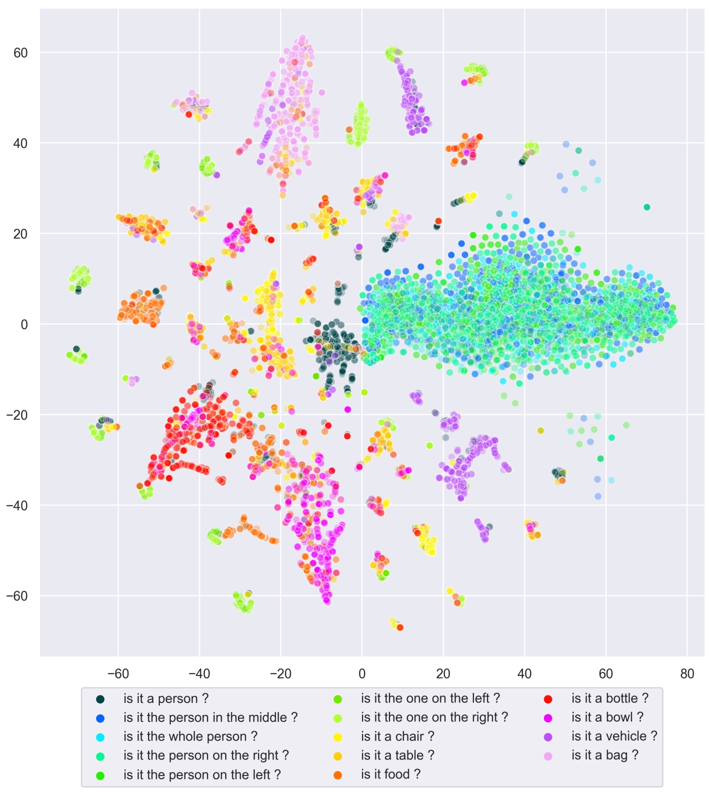
For qualitative examples, we refer to the example section in the append C.4. At the bottom of each example, we provide a table that shows the $argmax$ of the respective probabilities, and in colour code if the generated question contains the category or not.

In addition, we also visualize the belief state and the question that is generated via t-SNE [Maaten and Hinton, 2008]. t-SNE is a dimensionality reduction method that aims at preserving the local distances between points in high dimensional space. Therefore, points that are close in the t-SNE result in 2d space, are also close in high dimensional space. We consider all questions that occur at least 1000 times in dialogues up to the turn where the Guesser is able to correctly identify the target for the first time. Further, in order to balance among the different questions, we take the belief state of each question 1000 times. I.e., for each question, we analyse 1000 belief states. The results are shown in Figure 6.7. We colour each question differently and see that clusters by question appear. This means, that the belief states that produced that question are similar. This provides further evidence, that the belief state highly influences the generated question.

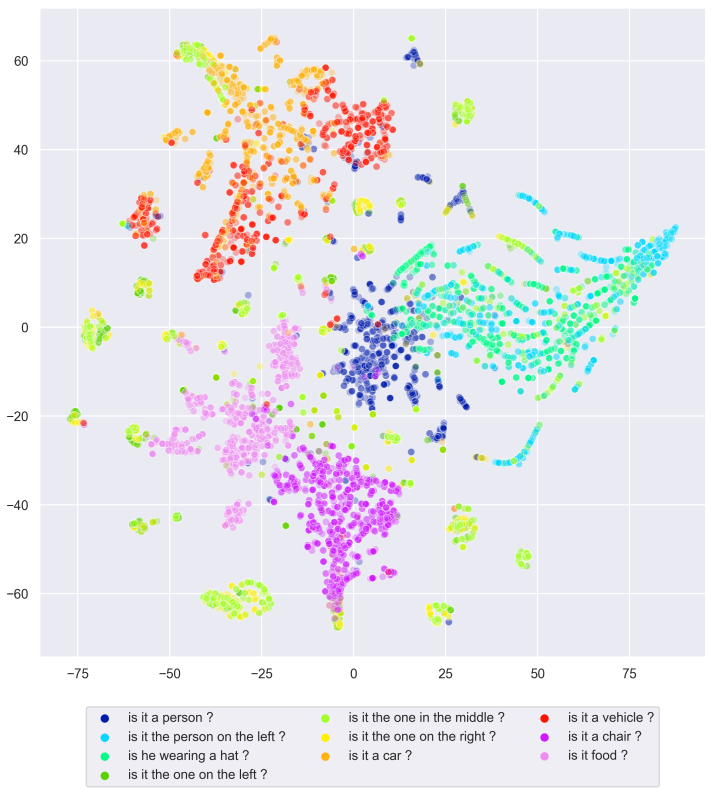
6.3 Ablation Study Analysis

In section 5.2 we conducted ablation studies to further investigate the performance improvements by the belief models.

From the results (Table 5.4) we can interpret where performance improvements



(a) *Belief*



(b) *Belief+FineTune*

Figure 6.7: t-SNE on the belief state of the different models, coloured by question.

are coming from. When removing the dynamic belief probabilities, and providing a uniform distribution (Bag of Objects), the ablation model's performance decreases 4.38% compared to the Belief model. This provides evidence on how important the belief state representing the uncertainty over the objects is. Vice versa, the ablation model improves over the baseline by 3.35%. This gain can be contributed to the improved visual perception.

Similar in case of the ablation model that has a belief state, but without receiving the list of objects. This ablation model improves 1.7% over the baseline. We conjecture that this improvement is low because of the much larger space of object categories (81) compared to the average number of objects in the image (8.54). Therefore, the belief state becomes harder to interpret. Nevertheless, we observe an improvement over the baseline, which can be contributed to the dynamic belief probabilities. That is because, in this setting, we do not provide a list of objects in

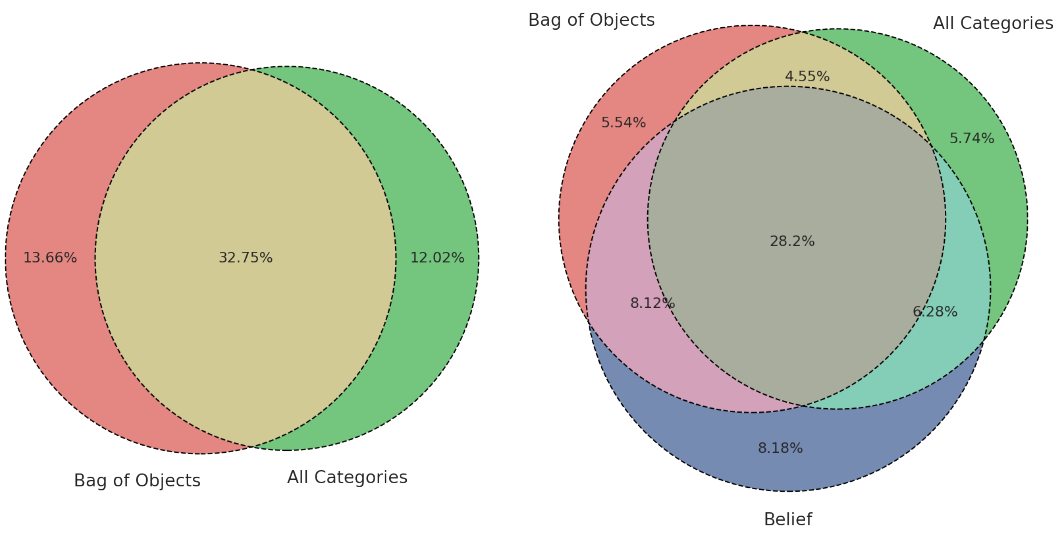


Figure 6.8: Venn diagram of solved games for the two ablation study models (left) and including the Belief model (right).

the image.

Figure 6.8 (left) shows a Venn diagram of the solved games by the two ablation study models. While 32.75% of games are solved by both models, 25.68% of games are only solved by one model, and the mass is about equally distributed.

Next, we add the set of games solved by the Belief model to the Venn diagram (Figure 6.8 right). We see that the majority of the games solved by the Belief model are also solved by either of the ablation models. Further, there are 8.18% of games that are not solved by the ablation models, but by the Belief model. We conclude that the Belief model learns an efficient way of combining the improved perception (bag of objects) and the dynamic belief state (all categories) in order to solve these games.

6.4 Qualitative Examples

We provide further qualitative example by picking 5 games at random. We, therefore, take the original order of the games IDs provided by [De Vries et al., 2017a] on the validation set and sample game IDs with replacement. We use numpy version 1.15.4 and set the seed to 1. This results in the following game IDs: 5635, 125592, 63722, 170089, 116089. We choose this approach to minimize cherry picking. The games with their generated dialogues and images can be found in the appendix in section C.4. Further examples can be easily viewed with the web tool we also publish (see section A).

7 Conclusion

We presented a principal method of equipping an end-to-end, task-oriented neural dialogue model with a belief state similar to the tasks of a dialogue manager in the traditional dialogue systems pipeline. Our model is end-to-end trainable and does not require intermediate annotations of dialogue states.

We tested the model on GuessWhat?!, in a visually grounded, task-oriented dialogue setting. We conducted an elaborate search on different possible representations for the belief state for the task at hand. Our best performing model achieves 55.63% on the GuessWhat?! test set. That is an absolute improvement compared to the baseline of 13.08% and a relative improvement of 30.74%[12].

We conducted a detailed analysis of different variants of the model, shading light on where task success improvements are coming from. At the task at hand, we show that our model mainly extends the capabilities of the baseline. Main improvements come for example from games where the target object is the only instance of its category. Further, we showed that models equipped with an explicit belief state representation ask better questions in later rounds compared to the baseline. In terms of task success, this is shown by breaking down the performance over the turns, linguistically, this also shown by the fact that our models produce fewer repetitions. In addition, our best model learns extremely effective from the human dialogues, we show that for early rounds the learned language is even slightly more effective than that of humans in terms of task success evaluated by the Guesser. We further showed the direct influence of the belief state on the question that will be generated. Our ablation studies further show that improvements do not only come from improved perception but provide evidence that our models learn an efficient representation combining the improved vision and the dynamic belief state.

[12]Comparison conducted to the task performance of our implementation of the baseline with the best number of questions.

7.1 Future Work

In this work, we extensively addressed how the belief state can be used in a supervised learning framework. While this model is to the best of our knowledge state of the art when regarding supervised approaches, we leave it for future work to explore the efficiency when applying reinforcement learning [Strub et al., 2017, Zhang et al., 2018] or cooperative learning as in [Shekhar et al., 2019]. Besides our novel learning paradigm, also our joint architecture for the Questioner could benefit from the explicit belief state. It could be added to both the common encoder and the Question Generator. The belief state would be concatenated on a word level, and the probability distribution would be produced by the Guesser that is also conditioned on the common encoder. It will be interesting to explore if similar results or improvements can be achieved in this setting, as the Guesser (and therefore the belief state) is trained jointly with the Question Generator. Moreover, the Decider module we proposed in [Shekhar et al., 2018] can be conditioned on the belief state. The proposed models could be extended (e.g. by concatenating the belief state to the Decider input), or also a module based on the belief state alone can be explored.

In addition, the proposed architecture can easily be transferred to other task-oriented setups. For example in the navigation setting of [de Vries et al., 2018], the tourist agent can have a belief state over the possible locations. Another exemplary application is the mutual friend task [He et al., 2017]. The belief state can be over the different properties of the friends, such as school, major or company.

Further, learning an efficient, end-to-end representation of the image that can be used by the Question Generator is still not addressed adequately. Our model that uses lower-level visual features that in principle should be more expressive does not outperform the simpler approach of using VGG16 FC8 features. Potentially, approaches such as [Strub et al., 2018] can be employed also in the Question Generator.

Bibliography

[Abadi et al., 2016] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

[Antol et al., 2015] Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Lawrence Zitnick, C., and Parikh, D. (2015). Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433.

[Bahdanau et al., 2015] Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[Barsalou, 2008] Barsalou, L. W. (2008). Grounded cognition. *Annu. Rev. Psychol.*, 59:617–645.

[Bengio et al., 1994] Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

[Bird et al., 2009] Bird, S., Klein, E., and Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

[Bobrow et al., 1977] Bobrow, D. G., Kaplan, R. M., Kay, M., Norman, D. A., Thompson, H., and Winograd, T. (1977). Gus, a frame-driven dialog system. *Artificial intelligence*, 8(2):155–173.

[Bohnet et al., 2018] Bohnet, B., McDonald, R. T., Simões, G., Andor, D., Pitler, E., and Maynez, J. (2018). Morphosyntactic tagging with a meta-bilstm model over context sensitive token encodings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2641–2651.

[Bordes and Weston, 2016] Bordes, A. and Weston, J. (2016). Learning end-to-end goal-oriented dialog. *CoRR*, abs/1605.07683.

[Chen and Dolan, 2011] Chen, D. L. and Dolan, W. B. (2011). Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 190–200. Association for Computational Linguistics.

[Cho et al., 2014] Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

[Das et al., 2017] Das, A., Kottur, S., Gupta, K., Singh, A., Yadav, D., Moura, J. M., Parikh, D., and Batra, D. (2017). Visual dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2.

[de Vries et al., 2018] de Vries, H., Shuster, K., Batra, D., Parikh, D., Weston, J., and Kiela, D. (2018). Talk the walk: Navigating new york city through grounded dialogue. *CoRR*, abs/1807.03367.

[De Vries et al., 2017a] De Vries, H., Strub, F., Chandar, S., Pietquin, O., Larochelle, H., and Courville, A. C. (2017a). Guesswhat?! visual object discovery through multi-modal dialogue. In *CVPR*, volume 1, page 3.

[De Vries et al., 2017b] De Vries, H., Strub, F., Mary, J., Larochelle, H., Pietquin, O., and Courville, A. C. (2017b). Modulating early visual processing by language. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 6594–6604. Curran Associates, Inc.

[De Vries et al., 2017c] De Vries, H., Strub, F., Mary, J., Larochelle, H., Pietquin, O., and Courville, A. C. (2017c). Modulating early visual processing by language. In *Advances in Neural Information Processing Systems*, pages 6594–6604.

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee.

[Elliott et al., 2017] Elliott, D., Frank, S., Barrault, L., Bougares, F., and Specia, L. (2017). Findings of the second shared task on multimodal machine translation and multilingual image description. In *Proceedings of the Second Conference on Machine Translation, WMT 2017, Copenhagen, Denmark, September 7-8, 2017*, pages 215–233.

[Elliott et al., 2016] Elliott, D., Frank, S., Sima'an, K., and Specia, L. (2016). Multi30k: Multilingual english-german image descriptions. In *Proceedings of the 5th Workshop on Vision and Language, hosted by the 54th Annual Meeting of the Association for Computational Linguistics, VL@ACL 2016, August 12, Berlin, Germany*.

[Elman, 1990] Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

[Goldberg and Hirst, 2017] Goldberg, Y. and Hirst, G. (2017). *Neural Network Methods in Natural Language Processing*. Morgan & Claypool Publishers.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

[Harnad, 1990] Harnad, S. (1990). The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1-3):335–346.

[He et al., 2017] He, H., Balakrishnan, A., Eric, M., and Liang, P. (2017). Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1766–1776.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[Henderson et al., 2014a] Henderson, M., Thomson, B., and Williams, J. D. (2014a). The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272.

[Henderson et al., 2014b] Henderson, M., Thomson, B., and Williams, J. D. (2014b). The third dialog state tracking challenge. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 324–329. IEEE.

[Hochreiter, 1991] Hochreiter, S. (1991). Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Howard and Ruder, 2018] Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 328–339.

[Hu et al., 2016] Hu, R., Xu, H., Rohrbach, M., Feng, J., Saenko, K., and Darrell, T. (2016). Natural language object retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4555–4564.

[Jokinen and McTear, 2009] Jokinen, K. and McTear, M. (2009). *Spoken dialogue systems*, volume 2. Morgan & Claypool Publishers.

[Jurafsky and Martin, 2019] Jurafsky, D. and Martin, J. H. (2019). *Speech and language processing*. Third, draft edition. unpublished draft, retrieved January 2019.

[Kafle and Kanan, 2017] Kafle, K. and Kanan, C. (2017). Visual question answering: Datasets, algorithms, and future challenges. *Computer Vision and Image Understanding*, 163:3–20.

[Kazemi and Elqursh, 2017] Kazemi, V. and Elqursh, A. (2017). Show, ask, attend, and answer: A strong baseline for visual question answering. *CoRR*, abs/1704.03162.

[Kazemzadeh et al., 2014] Kazemzadeh, S., Ordonez, V., Matten, M., and Berg, T. (2014). Referitgame: Referring to objects in photographs of natural scenes. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 787–798.

[Kim et al., 2016a] Kim, J., On, K. W., Lim, W., Kim, J., Ha, J., and Zhang, B. (2016a). Hadamard product for low-rank bilinear pooling. *CoRR*, abs/1610.04325.

[Kim et al., 2017a] Kim, J., Parikh, D., Batra, D., Zhang, B., and Tian, Y. (2017a). Codraw: Visual dialog for collaborative drawing. *CoRR*, abs/1712.05558.

[Kim et al., 2016b] Kim, S., D'Haro, L. F., Banchs, R. E., Williams, J. D., Henderson, M., and Yoshino, K. (2016b). The fifth dialog state tracking challenge. In *Spoken Language Technology Workshop (SLT), 2016 IEEE*, pages 511–517. IEEE.

[Kim et al., 2017b] Kim, S., D'Haro, L. F., Banchs, R. E., Williams, J. D., and Henderson, M. (2017b). The fourth dialog state tracking challenge. In *Dialogues with Social Robots*, pages 435–449. Springer.

[Kingma and Ba, 2015] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[Li et al., 2016] Li, J., Galley, M., Brockett, C., Gao, J., and Dolan, B. (2016). A diversity-promoting objective function for neural conversation models. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 110–119.

[Lin et al., 2014] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

[Maaten and Hinton, 2008] Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

[Merity et al., 2017] Merity, S., Keskar, N. S., and Socher, R. (2017). Regularizing and optimizing LSTM language models. *CoRR*, abs/1708.02182.

[Mikolov et al., 2010] Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

[Minsky and Papert, 1969] Minsky, M. and Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, USA.

[Mnih et al., 2013] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. A. (2013). Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602.

[Mostafazadeh et al., 2017] Mostafazadeh, N., Brockett, C., Dolan, B., Galley, M., Gao, J., Spithourakis, G. P., and Vanderwende, L. (2017). Image-grounded conversations: Multimodal context for natural question and response generation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Taipei, Taiwan, November 27 - December 1, 2017 - Volume 1: Long Papers*, pages 462–472.

[Nair and Hinton, 2010] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

[Paszke et al., 2017] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch.

[Perez et al., 2018] Perez, E., Strub, F., De Vries, H., Dumoulin, V., and Courville, A. (2018). Film: Visual reasoning with a general conditioning layer. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[Rashtchian et al., 2010] Rashtchian, C., Young, P., Hodosh, M., and Hockenmaier, J. (2010). Collecting image annotations using amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 139–147. Association for Computational Linguistics.

[Ritter et al., 2011] Ritter, A., Cherry, C., and Dolan, W. B. (2011). Data-driven response generation in social media. In *Proceedings of the conference on empirical methods in natural language processing*, pages 583–593. Association for Computational Linguistics.

[Rohrbach et al., 2015] Rohrbach, A., Rohrbach, M., Tandon, N., and Schiele, B. (2015). A dataset for movie description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3202–3212.

[Rumelhart et al., 1985] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533.

[Serban et al., 2016] Serban, I. V., Sordoni, A., Bengio, Y., Courville, A. C., and Pineau, J. (2016). Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, volume 16, pages 3776–3784.

[Shang et al., 2015] Shang, L., Lu, Z., and Li, H. (2015). Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language*

Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers, pages 1577–1586.

[Shekhar et al., 2018] Shekhar, R., Baumgärtner, T., Venkatesh, A., Bruni, E., Bernardi, R., and Fernández, R. (2018). Ask no more: Deciding when to guess in referential visual dialogue. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 1218–1233.

[Shekhar et al., 2019] Shekhar, R., Venkatesh, A., Baumgärtner, T., Bruni, E., Bernardi, R., and Fernández, R. (2019). Beyond task success: A closer look at jointly learning to see, ask, and guesswhat. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. to appear.

[Simonyan and Zisserman, 2015] Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[Smith and Gasser, 2005] Smith, L. and Gasser, M. (2005). The development of embodied cognition: Six lessons from babies. *Artificial Life*, 11(1-2):13–29.

[Sordoni et al., 2015a] Sordoni, A., Bengio, Y., Vahabi, H., Lioma, C., Grue Simonsen, J., and Nie, J.-Y. (2015a). A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 553–562. ACM.

[Sordoni et al., 2015b] Sordoni, A., Galley, M., Auli, M., Brockett, C., Ji, Y., Mitchell, M., Nie, J., Gao, J., and Dolan, B. (2015b). A neural network approach to context-sensitive generation of conversational responses. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 196–205.

[Strub et al., 2017] Strub, F., de Vries, H., Mary, J., Piot, B., Courville, A. C., and Pietquin, O. (2017). End-to-end optimization of goal-driven and visually grounded dialogue systems. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 2765–2771.

[Strub et al., 2018] Strub, F., Seurin, M., Perez, E., de Vries, H., Mary, J., Preux, P., and CourvilleOlivier Pietquin, A. (2018). Visual reasoning with multi-hop feature modulation. In *The European Conference on Computer Vision (ECCV)*.

[Sukhbaatar et al., 2015] Sukhbaatar, S., Weston, J., Fergus, R., et al. (2015). End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

[Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

[Torabi et al., 2015] Torabi, A., Pal, C. J., Larochelle, H., and Courville, A. C. (2015). Using descriptive video services to create a large data source for video annotation research. *CoRR*, abs/1503.01070.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010.

[Vinyals and Le, 2015] Vinyals, O. and Le, Q. V. (2015). A neural conversational model. *CoRR*, abs/1506.05869.

[Weston et al., 2015] Weston, J., Chopra, S., and Bordes, A. (2015). Memory networks. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[Williams et al., 2013] Williams, J., Raux, A., Ramachandran, D., and Black, A. (2013). The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413.

[Williams, 1992] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

[Williams and Zipser, 1989] Williams, R. J. and Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

[Winograd, 1972] Winograd, T. (1972). Understanding natural language. *Cognitive psychology*, 3(1):1–191.

[Yang et al., 2016] Yang, Z., He, X., Gao, J., Deng, L., and Smola, A. (2016). Stacked attention networks for image question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 21–29.

[Young et al., 2014] Young, P., Lai, A., Hodosh, M., and Hockenmaier, J. (2014). From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78.

[Young et al., 2010] Young, S., Gašić, M., Keizer, S., Mairesse, F., Schatzmann, J., Thomson, B., and Yu, K. (2010). The hidden information state model: A practical framework for pomdp-based spoken dialogue management. *Computer Speech & Language*, 24(2):150–174.

[Young et al., 2013] Young, S., Gašić, M., Thomson, B., and Williams, J. D. (2013). Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

[Yu et al., 2016] Yu, L., Poirson, P., Yang, S., Berg, A. C., and Berg, T. L. (2016). Modeling context in referring expressions. In *European Conference on Computer Vision*, pages 69–85. Springer.

[Zhang et al., 2018] Zhang, J., Wu, Q., Shen, C., Zhang, J., Lu, J., and Van Den Hengel, A. (2018). Goal-oriented visual question generation via intermediate rewards. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 186–201.

[Zwaan et al., 2002] Zwaan, R., A Stanfield, R., and H Yaxley, R. (2002). Language comprehenders mentally represent the shapes of objects. *Psychological science*, 13:168–71.

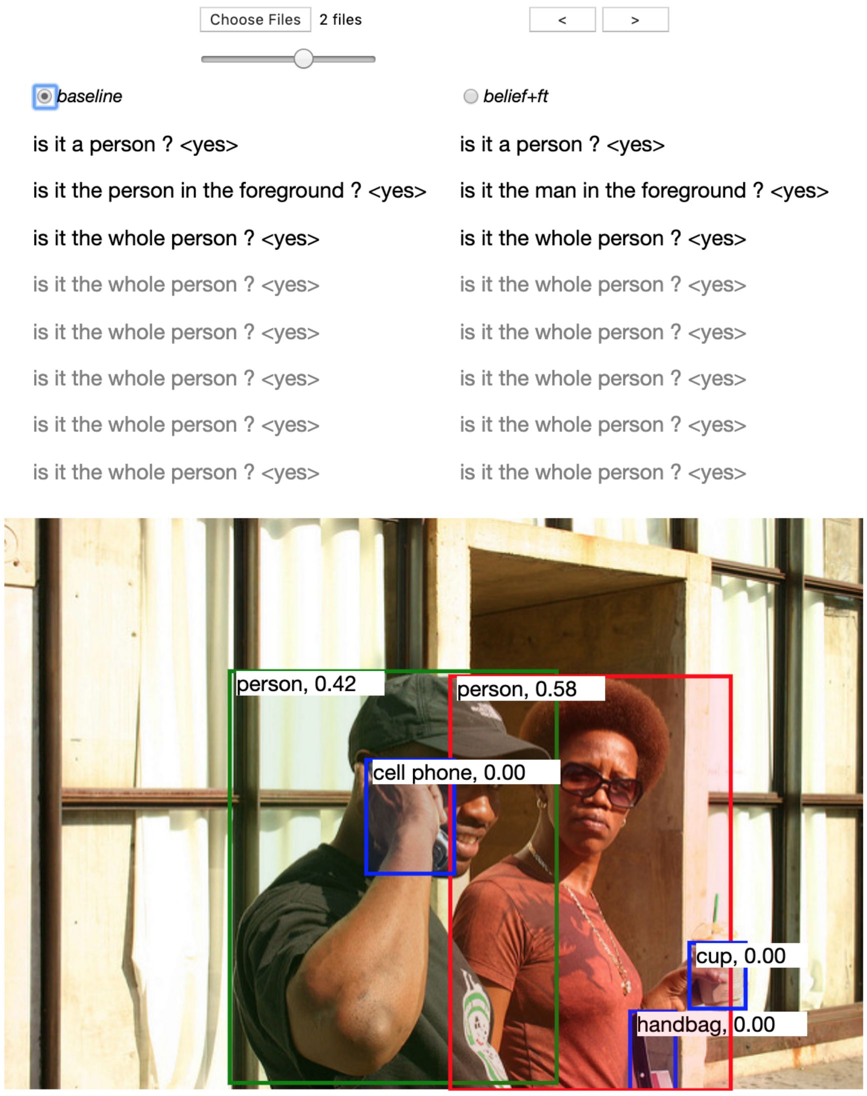
A Reproducibility

We make our entire code base public in order to ensure the reproducibility of our experiments and facilitate future research. The code, models and log files can be found at github.com/timbmg/belief.

We provide the parameters of the following best performing models:

- Our implementation of the baseline models
- Belief model
- Belief+FineTune model
- Belief model with visual attention
- Belief+FineTune model with visual attention

Moreover, we provide the log files of the models we conducted our analysis on, namely the Baseline, Belief and Belief+FineTune model. Along, we publish the jupyter notebook for generating the plots used in this thesis. Lastly, we provide a web tool for comparing up to two dialogue models and their belief probabilities at every turn. For an example see Figure A.1.



● *baseline* ○ *belief+ft*

baseline	belief+ft
is it a person ? <yes>	is it a person ? <yes>
is it the person in the foreground ? <yes>	is it the man in the foreground ? <yes>
is it the whole person ? <yes>	is it the whole person ? <yes>
is it the whole person ? <yes>	is it the whole person ? <yes>
is it the whole person ? <yes>	is it the whole person ? <yes>
is it the whole person ? <yes>	is it the whole person ? <yes>
is it the whole person ? <yes>	is it the whole person ? <yes>
is it the whole person ? <yes>	is it the whole person ? <yes>

Figure A.1: Web tool for browsing and comparing dialogues of different models. The bounding boxes represent the MS COCO annotations, the red box encloses the object guessed by the selected model at the end of the dialogue (if different from the target object), the green box the actual target object and all other objects have a blue bounding box. Besides the object category, also the probability of the belief is displayed for every object in the bounding box. This is the probability the Guesser assigns at the current turn, which can be chosen with the slider. The arrow buttons at the top allow for changing to the next game.

B Hyperparameters

In the following sections, the hyperparameters for the various models are listed. Any not specified hyperparameters are set according to the suggested default settings in the original paper (e.g. in case of the optimizer). All models have been trained until the validation loss was consistently rising. Subsequently, the model of the epoch achieving the lowest loss on the validation set has been selected.

B.1 Oracle Hyperparameters

Hyperparameter	Value
Optimizer	ADAM [Kingma and Ba, 2015]
Batch Size	32
Learning Rate	0.001
Gradient Norm Clip	3
Word Embedding Size	300
Category Embedding Size	512
LSTM Hidden Size	512
MLP Hidden Size	512

Table B.1: Hyperparameters of the Oracle model. The same hyperparameters have been chosen as in [De Vries et al., 2017a].

B.2 Guesser Hyperparameters

Hyperparameter	Value
Optimizer	ADAM [Kingma and Ba, 2015]
Batch Size	64
Learning Rate	0.0001
Word Embedding Size	512
Category Embedding Size	256
LSTM Hidden Size	512
MLP Hidden Size	512

Table B.2: Hyperparameters of the Guesser model. The same hyperparameters have been selected as in [De Vries et al., 2017a].

B.3 Question Generator Hyperparameters

Hyperparameter	Value
Optimizer	ADAM [Kingma and Ba, 2015]
Batch Size	64
Learning Rate	0.0001
Word Embedding Size	512
Visual Embedding Size	512
LSTM Hidden Size	512

Table B.3: Hyperparameters of the Question Generator model. The same hyperparameters have been chosen as in [De Vries et al., 2017a] for the baseline model. For additional hyperparameters of the belief models, and their search space, please see chapter 5.

B.4 Comprehensive Belief State Results

Belief State Representation	$\lvert\mathbf{W}_v\rvert$	$\lvert\mathbf{R}\rvert$	Cross Entropy	Validation Accuracy (n=5)
Category	0	64	1.443	49.48%
Category	0	512	1.444	49.40%
Category	128	128	1.439	48.21%
Category	256	256	1.446	48.12%
Category	512	256	1.453	47.93%
Category	0	256	1.443	47.62%
Category	128	256	1.441	47.60%
Category	256	128	1.445	47.53%
Category+Spatial	0	256	1.433	50.00%
Category+Spatial	128	512	1.440	48.75%
Category+Spatial	0	512	1.434	48.44%
Category+Spatial	256	256	1.441	48.32%
Category+Spatial	128	256	1.437	48.24%
Category+Spatial	64	128	1.432	47.87%
Category+Spatial	64	512	1.436	47.86%
Category+Spatial	256	512	1.445	47.76%
Guesser Obj. Rep.	0	256	1.436	49.16%
Guesser Obj. Rep.	64	512	1.436	48.35%
Guesser Obj. Rep.	0	64	1.436	47.93%
Guesser Obj. Rep.	0	512	1.435	47.71%
Guesser Obj. Rep.	128	64	1.438	47.56%
Guesser Obj. Rep.	256	512	1.444	47.51%
Guesser Obj. Rep.	64	64	1.433	47.50%
Guesser Obj. Rep.	128	128	1.436	47.38%

Table B.4: Results and hyperparameter setting for the 8 best (by validation accuracy) Belief models for each belief state representation.

C Analysis

C.1 Number of Parameters

Table C.1 shows the number of parameters of each model. Note that the Belief model is listed twice. Since the Guesser used for generating the belief state is not updated the same module can be used to perform the task evaluation. Therefore we report the number of parameters when considering the Guesser part of the model and when excluding it. Further, note that the Belief model excluding the Guesser's parameters has fewer parameters as the baseline, as the size of the input to the Question Generator's LSTM is smaller for the belief model ($|\mathbf{W}_v| + |\mathbf{R}| + |\mathbf{E}_{word}| = 0 + 256 + 512 = 768$) as in the baseline ($|\mathbf{W}_v| + |\mathbf{E}_{word}| = 512 + 512 = 1024$). All numbers are excluding the visual pipeline.

Model	Test Accuracy	Parameters
Baseline	42.55%	16.46M
Belief (excluding Guesser)	49.49%	15.19M
Belief (including Guesser)	49.49%	20.23M
Belief+FineTune	54.75%	19.15M
Belief+FineTune + VisAttn	54.23%	23.00M

Table C.1: Number of Parameters of the different Models

C.2 Game Statistics

Games solved by:	Baseline	Belief	Baseline	Belief+FT	Belief	Belief+FT
But not by:	Belief	Baseline	Belief+FT	Baseline	Belief+FT	Belief
Num Games	2,448	4,382	1,999	5,161	2,252	3,480
Num Objects	8.36	8.19	8.66	8.13	9.05	8.50
Num Object Categories	3.51	3.69	3.32	3.88	3.55	3.96
Num Instances of Target Cat.	3.75	3.32	4.28	2.92	4.26	3.08
Log of Target Object Area	8.45	8.85	8.44	8.73	8.70	8.54

Table C.2: Game statistics of the set of games solved by the model in the first row, but not by the model in the second row. Note that Belief+FineTune is abbreviated with Belief+FT for space reasons.

	All Games	Uniform	All Categories
Num Games		11,018	10,628
Num Objects	8.54	6.85	6.25
Num Object Categories	3.49	3.33	3.33
Num Instances of Target Cat.	3.99	2.56	2.67
Log of Target Object Area	8.64	9.20	9.20

Table C.3: Game statistics of the set of games solved by the ablation study models.

Games solved by:	Uniform	All Categories
But not by:	All Categories	Uniform
Num Games	3,243	2,853
Num Objects	8.01	8.43
Num Object Categories	3.58	3.61
Num Instances of Target Cat.	3.24	3.75
Log of Target Object Area	8.74	8.66

Table C.4: Game statistics of the set of games solved by the ablation study model in the first row, but not by the model in the second row.

C.3 Number of Games by Number of Objects

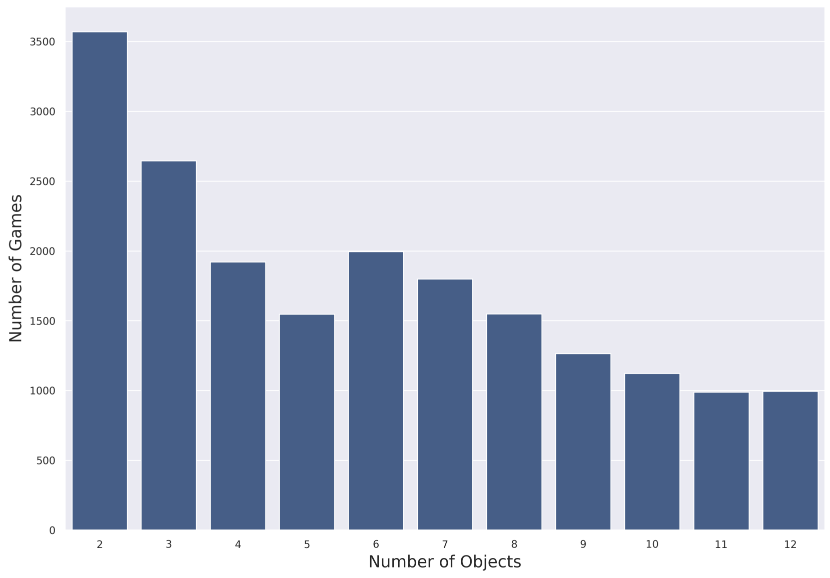


Figure C.1: Number of games by number of objects on the validation set.

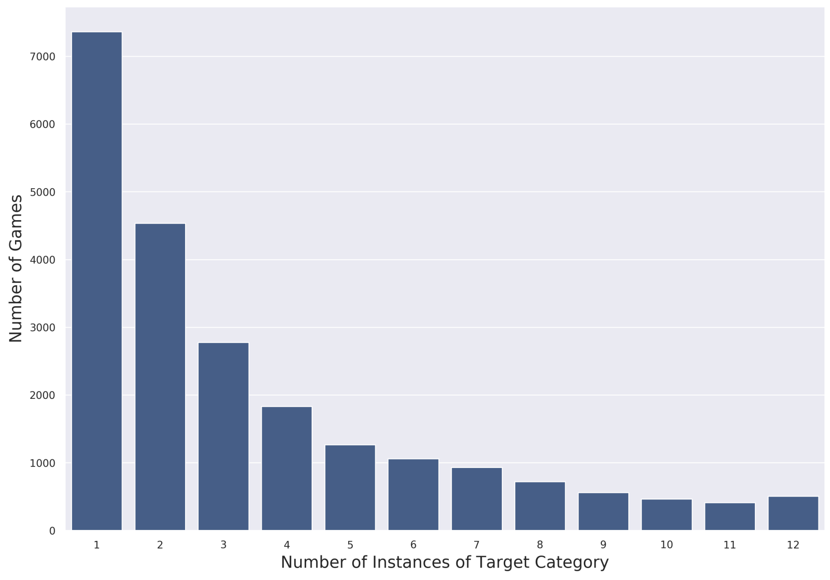


Figure C.2: Number of games by number of objects with the same category as the target object on the validation set.

C.4 Qualitative Examples

The following pages show different games, the generated dialogue by each model and the *argmax* of the belief probabilities, or in case of the baseline, of the Guesser probabilities.

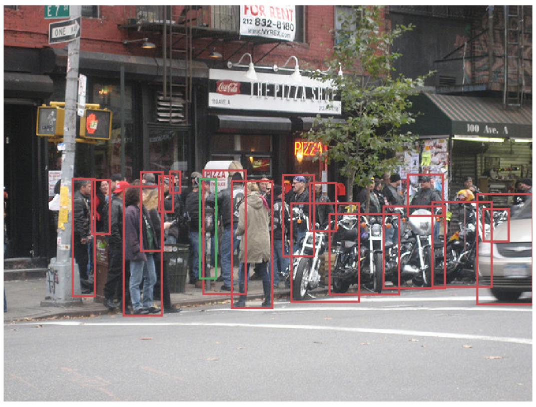


Figure C.3: Game ID: 5635, Target Category: Person

Baseline	**Belief**	**Belief+FineTune**
is it a person? yes	is it a person? yes	is it a person? yes
is it a woman? no	is it the person on the motorcycle? no	is it a man? yes
is it a woman? no	is it the person on the motorcycle? no	is he wearing a hat? yes
is it a woman? no	is it the person on the left? yes	is he wearing a hat? yes
is it a woman? no	is it the person on the left? yes	is he wearing a hat? yes
is it a woman? no	is it the whole person? yes	is he wearing a hat? yes
is it a woman? no	is it the person on the left? yes	is he wearing a hat? yes
is it a woman? no	is it the whole person? yes	is he wearing a hat? yes
→ Failure	→ Failure	→ Failure

	1	2	3	4	5	6	7	8
Baseline	person	person	person	person	person	person	person	person
Belief	person	person	person	person	person	person	person	person
Belief+FineTune	person	person	person	person	person	person	person	person

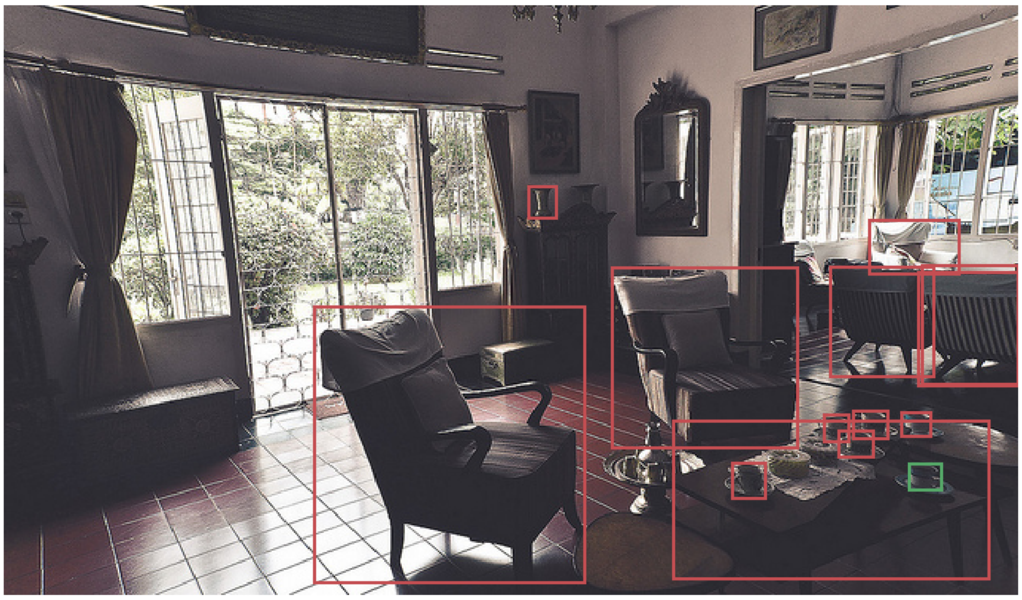


Figure C.4: Game ID: 125592, Target Category: Cup

Baseline	**Belief**	**Belief+FineTune**
is it a piece of furniture? no	is it a chair? no	is it a chair? no
is it a piece of furniture? no	is it a table? no	is it a table? no
is it a piece of furniture? no	is it a glass? yes	is it a table? no
is it a piece of furniture? no	is it the one on the right? yes	is it a lamp? no
is it a piece of furniture? no	is it the one on the right? yes	is it a lamp? no
is it a piece of furniture? no	is it the one on the right? yes	is it a lamp? no
is it a piece of furniture? no	is it the one on the right? yes	is it a table? no
is it a piece of furniture? no	is it the one on the left? no	is it a chair? no
→ Failure	→ Success	→ Failure

	1	2	3	4	5	6	7	8
Baseline	vase	cup	dining table	dining t.	dining t.	dining t.	dining t.	dining t.
Belief	vase	cup	cup	cup	cup	cup	cup	cup
Belief+FineTune	chair	dining table	vase	vase	vase	vase	couch	vase

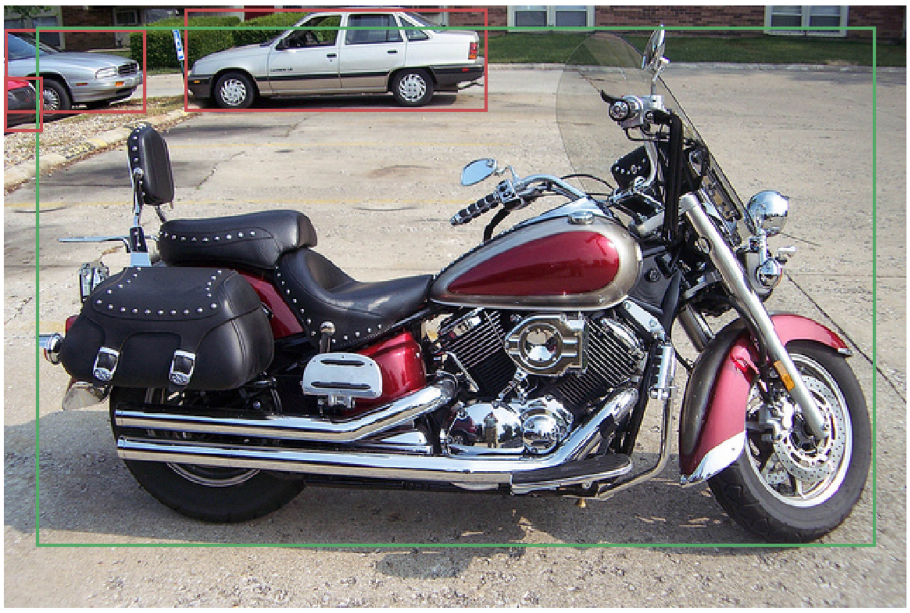


Figure C.5: Game ID: 63722, Target Category: Motorcycle

Baseline	**Belief**	**Belief+FineTune**
is it a vehicle? yes	is it a motorcycle? yes	is it a vehicle? yes
is it the motorcycle? yes	is it the whole motorcycle? yes	is it a motorcycle? yes
is it the whole motorcycle? yes	is it the whole motorcycle? yes	is it the whole motorcycle? yes
is it the whole motorcycle? yes	is it the whole motorcycle? yes	is it the whole motorcycle? yes
is it the whole motorcycle? yes	is it the whole motorcycle? yes	is it the whole motorcycle? yes
is it the whole motorcycle? yes	is it the whole motorcycle? yes	is it the whole motorcycle? yes
is it the whole motorcycle? yes	is it the whole bike? yes	is it the whole motorcycle? yes
is it the whole motorcycle? yes	is it the whole bike? yes	is it the whole motorcycle? yes
→ Success	→ Success	→ Success

	1	2	3	4	5	6	7	8
Baseline	motorcycle	motorcy.	motorcy.	motorcy.	motorcy.	motorcy.	motorcy.	motorcy.
Belief	motorcycle	motorcy.	motorcy.	motorcy.	motorcy.	motorcy.	motorcy.	motorcy.
Belief+FineTune	motorcycle	motorcy.	motorcy..	motorcy.	motorcy.	motorcy.	motorcy.	motorcy.

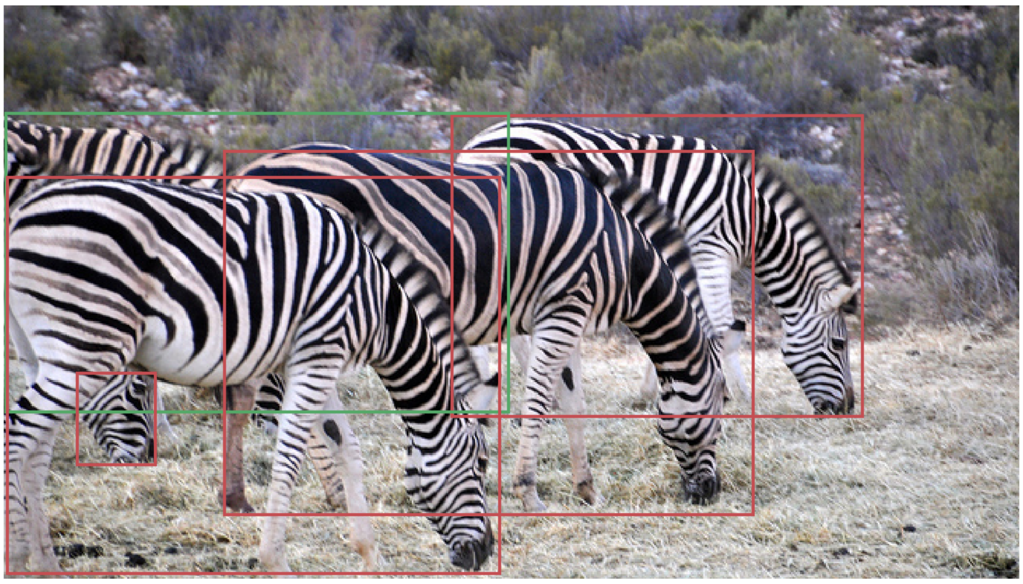


Figure C.6: Game ID: 170089, Target Category: Zebra

Baseline	**Belief**	**Belief+FineTune**
is it a zebra? yes	is it a zebra? yes	is it a zebra? yes
is it the one in the middle? no	is it the one on the left? yes	is it the one on the left? yes
is it the one on the right? no	is it the whole zebra? yes	is it the one in the middle? no
is it the one on the right? no	is it the whole zebra? yes	is it the one on the right? no
is it the one on the right? no	is it the one on the left? yes	is it the one on the left? yes
is it the one on the right? no	is it the whole zebra? yes	is it the one in the middle? no
is it the one on the right? no	is it the whole zebra? yes	is it the one on the left? yes
is it the one on the right? no	is it the one on the left? yes	is it the one on the left? yes
→ Failure	→ Failure	→ Failure

	1	2	3	4	5	6	7	8
Baseline	zebra	zebra	zebra	zebra	zebra	zebra	zebra	zebra
Belief	zebra	zebra	zebra	zebra	zebra	zebra	zebra	zebra
Belief+FineTune	zebra	zebra	zebra	zebra	zebra	zebra	zebra	zebra

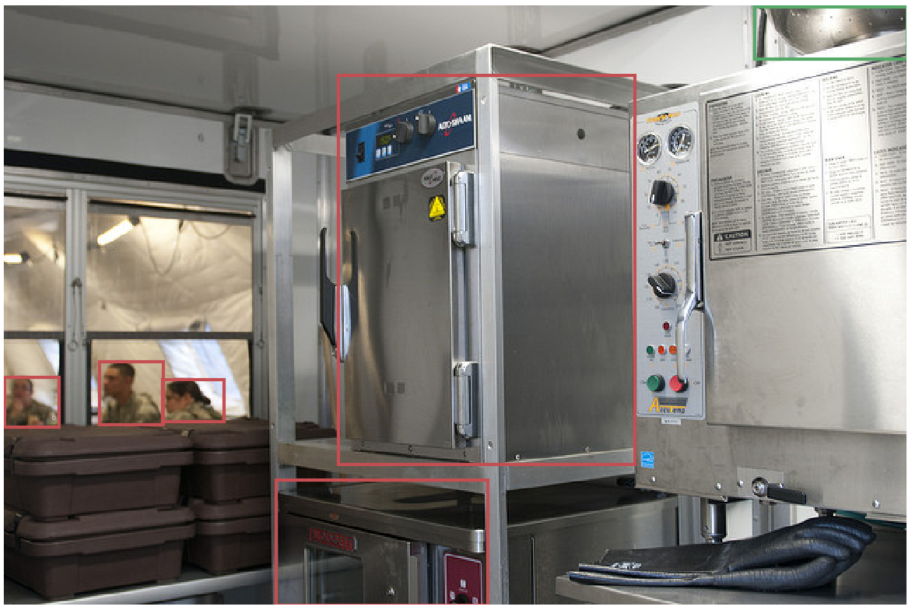


Figure C.7: Game ID: 116089, Target Category: Bowl

Baseline	**Belief**	**Belief+FineTune**
is it a fridge? no	is it a person? no	is it a person? no
is it a bottle? no	is it the fridge? no	is it a fridge? no
is it a bottle? no	is it the fridge? no	is it a stove? no
is it a bottle? no	is it on the fridge? no	is it a stove? no
is it a bottle? no	is it on the fridge? no	is it a fridge? no
is it a bottle? no	is it the fridge? no	is it a bowl? yes
is it a bottle? no	is it the fridge? no	is it the one with the white stuff in it? no
is it a bottle? no	is it the fridge? no	is it the one with the white stuff in it? no
→ Success	→ Failure	→ Success

	1	2	3	4	5	6	7	8
Baseline	bowl	person	person	person	person	bowl	bowl	bowl
Belief	bowl	bowl	bowl	person	bowl	bowl	person	person
Belief+FineTune	refrigerator	refrigerator	oven	refrigerator	refrigerator	bowl	bowl	bowl

C.5 Sunburst Diagrams

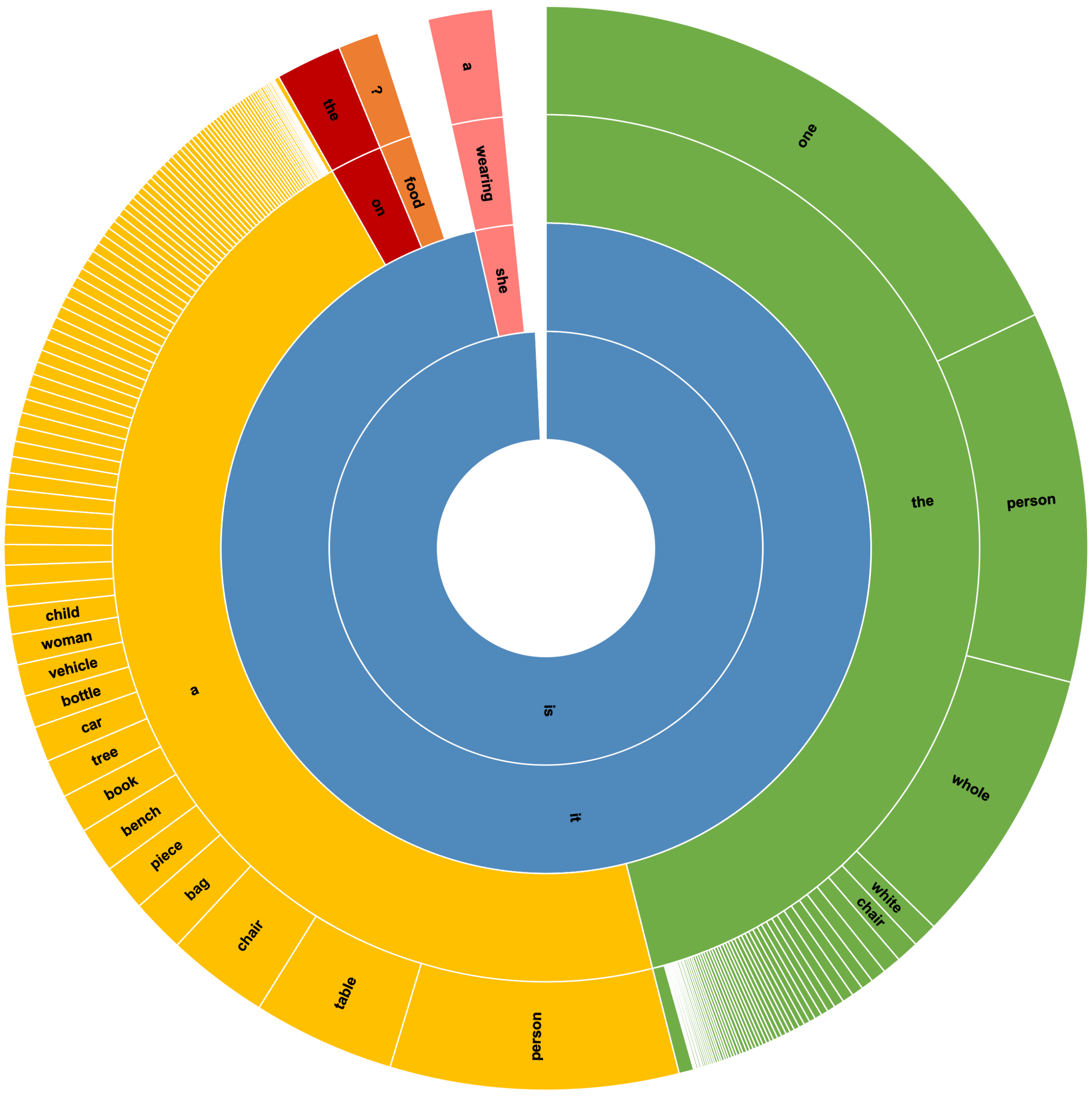


Figure C.8: Sunburst Diagram of *Baseline* Question Generator with 8 questions per game.

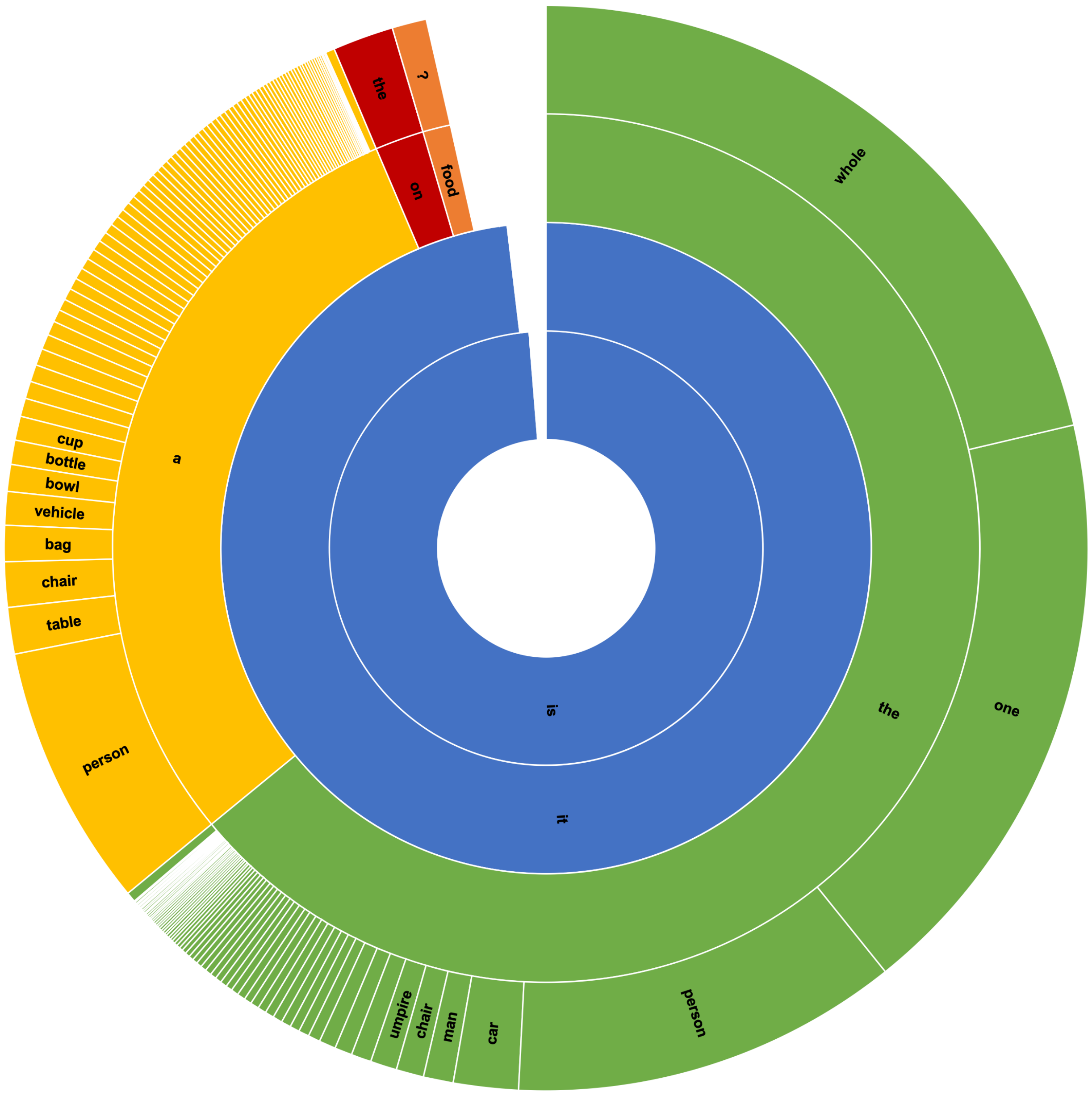


Figure C.9: Sunburst Diagram of *Belief* Question Generator with 8 questions per game.

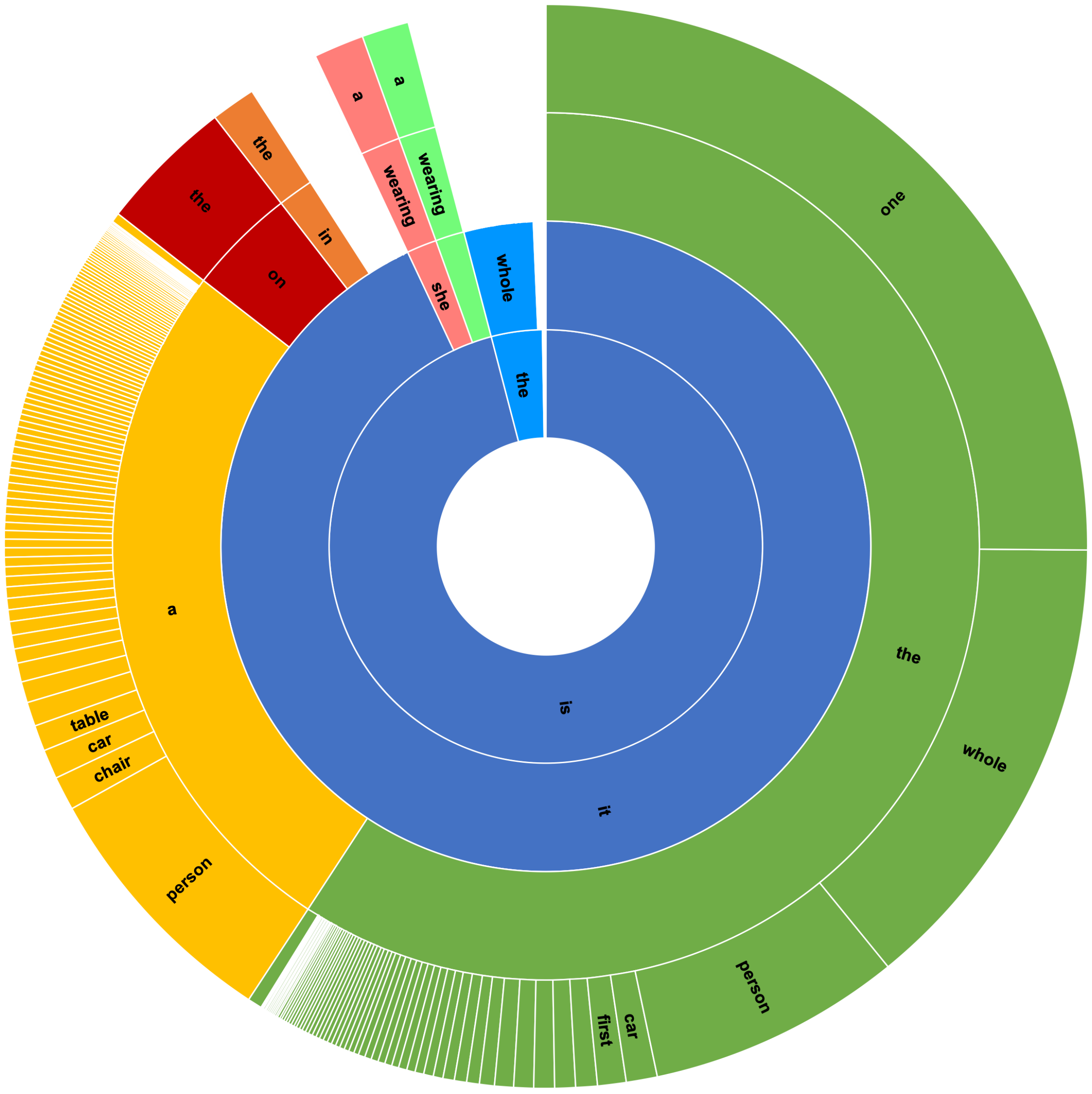


Figure C.10: Sunburst Diagram of *Belief+FineTune* Question Generator with 8 questions per game.