Automated Situated Task Guidance with Foundation Models

Elijah Joseph Gutierrez

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy in Machine Learning and Machine Intelligence

Clare College

August 2024

Declaration

I, Elijah Joseph Gutierrez of Clare College, being a candidate for the MPhil in Machine Learning and Machine Intelligence, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

The software used for this thesis is written in Python. I built on the Watch, Talk, and Guide (Bao et al., 2023) repository[1] to accommodate various experimental setups and models. The key-frame detection algorithm introduced in Section 4.3.1 was taken from the video-keyframe-detector repository[2]. Minor changes were made to this script to incorporate it into the project. Finally, to fine-tune Vicuna (Zheng et al., 2023), I made minor adjustments to the fine-tuning script from the Alpaca LoRA repository[3]. This software is used in Sections 4.3.2 and 5.3. All other software was written in standard Python from scratch.

The word count, excluding declarations, bibliography, photographs, and diagrams, but including tables, footnotes, figure captions, and appendices, is 14,843 words.

Elijah Joseph Gutierrez
August 2024

[1] https://github.com/sled-group/Watch-Talk-and-Guide/tree/main
[2] https://github.com/joelibaceta/video-keyframe-detector/tree/master
[3] https://github.com/tloen/alpaca-lora

Acknowledgements

I would like to thank my supervisors, Dr. Kate Knill, Dr. Svetlana Stoyanchev, and Dr. Catalin Zorila for always providing helpful advice and proposing interesting ideas throughout this challenging but rewarding project. I learned a lot and enjoyed our weekly chats in the office. I would also like to thank Rama Doddipatla for his support and suggestions. Finally, I would like to thank my friends and family for helping make my time at Cambridge truly memorable.

Abstract

As large-scale foundation models rapidly develop, we begin to consider their applicability to tasks that require knowledge of the physical world. The recently released Watch, Talk, and Guide (WTaG) (Bao et al., 2023) benchmark examines the particular problem of situated task guidance, where an Instructor assists a User through a physical task in real-time, such as baking a cake. Due to the real-time nature of the task, guidance in this context must be accurate and relevant to the situation at hand. This thesis explores the use of Large Language Models (LLMs) and Vision-Language Models (VLMs) to tackle this complex problem.

Evaluating task guidance quality is challenging because it is largely subjective. WTaG thus proposes a set of 7 auxiliary classification tasks that are straightforward to measure. The assumption is that a model that performs well on these auxiliary tasks would also perform well as a situated task guidance system, since it exhibits the necessary skills to do so.

Currently, ChatGPT serves as the sole zero-shot baseline for WTaG. However, this thesis conducts a suite of experiments with open-source models to a) assess whether these can produce comparable zero-shot results to ChatGPT and b) ensure that the reported results are reproducible. Upon establishing these baselines we explore subsequent methods of improving performance, such as prompt optimisation and instruction tuning.

Table of contents

List of figures

List of tables

Terminology

Acronyms / Abbreviations

GT	Ground Truth
IDM	Instructor Decision Making
KD	Knowledge Distillation
LLaVA	Large Language and Vision Assistant
LLM	Large Language Model
NLG	Natural Language Generation
NLU	Natural Language Understanding
QP	Query Point
SEQ-KD	Sequence-level Knowledge Distillation
IT	Instruction Tuning/Instruction Tuned
TOD	Task Oriented Dialogue
UEU	User and Environment Understanding
VLM	Vision-Language Model
WTaG	Watch, Talk, and Guide

Chapter 1

Introduction

1.1 Motivation

The advent of Transformer-based models in the last decade have led to the widespread use of Large Language Models (LLMs) today. The general-purpose capabilities of LLMs make them suitable for many potential applications, such as scanning and summarising legal documents, analysing medical records, or translating multilingual dialogue between two speakers. However, as Transformer models exhibit increasingly complex abilities, a new range of more advanced applications become possible. One such interesting application is automated situated task guidance.

Often, people may want to perform tasks they have never seen before or have little prior experience with. These tasks could be anything from baking a cake to changing a light bulb. In order to ensure that the task is completed correctly, there would ideally be an expert instructor present to offer real-time, *situated* guidance. 'Situated' in this context means that the guidance is relevant to the task performer's immediate situation, as opposed to more general advice on completing the task. Of course, we cannot expect a human instructor to be constantly available whenever a task needs to be performed. Thus, it is worth considering methods to automate the guidance process instead.

There are many ways to build an automated situated task guidance system, but this thesis focuses on leveraging recent large foundation models to perform task guidance. We are particularly interested in LLMs like Vicuna or Vision-Language Models (VLMs) such as LLaVA (Liu et al., 2024b). The advantage of these models lies in the implicit world knowledge and common sense they have access to after being trained on vast amounts of text data. This rich implicit knowledge can be probed in many interesting ways. In our case, we would like to find out whether foundation models can make use of their implicit common sense knowledge to reason about a given egocentric (i.e. first-person) scene. We probe this capability by prompting the models in various ways, the details of which will be explained in later sections.

Providing situated guidance requires a system to perform multiple sub-tasks. Any agent (human or otherwise) that serves as an instructor must keep track of many things. For example, which step the user is currently on, detecting if a mistake has occurred, or whether or not an intervention is necessary. Naturally, evaluating such complex functionality can be challenging. For this project, we use the Watch, Talk, and Guide (WTaG) benchmark dataset developed by Bao et al. (2023) for evaluation. WTaG is a multimodal benchmark that is designed specifically to test the task guidance capabilities of foundation models. Most existing benchmarks focus on evaluating foundation models as 'Users', particularly for tasks like Vision-Language Navigation (VLN) (Anderson et al., 2018; Fried et al., 2018). In this case, a 'User' or 'Follower' is given an ordered set of instructions to complete an arbitrary task. In contrast, WTaG aims to evaluate foundation models as 'Instructors' or 'Speakers' (i.e. agents that *issue* instructions) instead. This setup makes task guidance more challenging. Thus, treating these models as Instructors provides a great opportunity to test their higher-level understanding and reasoning.

Prior work in task guidance has typically focused on designing bespoke models for specific tasks. However, foundation models do not need to be explicitly trained on a specific task to be successful since they are designed to be general systems. Indeed, it has been widely observed that foundation models can have strong 'zero-shot' abilities (Xu et al., 2021). This means they can perform reasonably well on a wide variety of tasks as is, with no additional training. But even if a model might not have strong zero-shot performance on a task, they can still improve on a target task by being fine-tuned on task-specific examples. This 'pre-train, fine-tune paradigm' has been the dominant practice in recent years, especially as LLMs grow in size.

Furthermore, the original WTaG paper used ChatGPT Turbo 3.5, a closed-source model as its zero-shot baseline. In the interest of producing work that is reproducible, this work employs open-source models over proprietary models in order to promote their use as assistants, since they are generally more transparent and modifiable compared to their 'closed' counterparts.

1.2 Research Questions

Given the convenient abilities of foundation models, this thesis aims to answer the following research questions:

1. How do instruction-tuned open-source models perform on the WTaG benchmark under a zero-shot setting?
2. How do foundation models synthesise various pieces of information?
3. To what extent does further instruction tuning improve performance on the WTaG benchmark?
4. How relevant and helpful are the instructions that open foundation models generate for unseen real-world tasks?

Collectively, these questions probe the reasoning capabilities of open-source models in non-trivial real-world scenarios. To answer these questions, we run a suite of experiments and analyses. We answer 1) by conducting a systematic study of the zero-shot capabilities of current state-of-the-art models on the WTaG tasks. We start by attempting to replicate the performance of ChatGPT on the benchmark, then proceed to explore potential methods to improve this baseline performance.

Answering 2) requires analysing model outputs to get a better picture of how a foundation model reasons. For instance, did it make use of the immediate visual information, its conversation with the User, or both? We believe that these questions are interesting because they shed light on the ability of current foundation models to synthesise information presented to them, which is key for a complex task like WTaG.

While obtaining a zero-shot baseline for the open-source models is useful for comparison to ChatGPT, we would also like to determine the benefits of further instruction tuning on task guidance performance. Thus, to answer 3) we fine-tune the Vicuna model (Zheng et al., 2023) on outputs generated by Mixtral (Jiang et al., 2024) from a subset of WTaG videos. While instruction-tuned models can be powerful, they may perform a task in a way that is not aligned with the intention of the original instruction. We hope to mitigate this problem by tuning Vicuna on data that is specific to WTaG.

Finally, to answer 4) we manually examine the quality of a small number of model-generated instructions. We do this manually because it is difficult to quantify how 'helpful' an instruction is, especially since there are usually several ways to issue a valid instruction to a User. We believe this brief qualitative analysis would help us gauge the extent to which a model's performance on the WTaG benchmark correlates with its final generated instructions.

1.3 Roadmap

Having contextualised the premise of this work and the research questions it seeks to address, the subsequent chapters will include the following:

- **Chapter 2** reviews the current literature, and establishes the background behind Transformer-based models and instruction tuning to better understand the design and implementation of the LLMs and VLMs that will be used in later chapters.
- **Chapter 3** introduces the WTaG benchmark in full detail. This is done to provide a clear description of its purpose, contents and statistics. The evaluation procedures we use are also introduced here.
- **Chapter 4** covers our preliminary investigations to gauge the capabilities of various open-source models on WTaG. We establish a few partial baselines (i.e. only a small subset of all possible experiments were run) to save on compute resources.
- **Chapter 5** narrows the focus of the thesis, establishes full baselines, and presents our attempts to improve on these baselines through a combination of prompt tuning and instruction tuning.
- **Chapter 6** includes results, analyses and discussion regarding the suite of experiments that were run in Chapters 4 and 5.
- **Chapter 7** concludes the thesis and suggests directions for future work.

Chapter 2

Background

2.1 From Task Oriented Dialogue to Task Guidance

Completing tasks with the help of an automated system is a long-standing research area in AI, especially within the context of conversational agents. Traditionally, Task-Oriented Dialogue (TOD) systems have been (and continue to be) used for the purpose of assisting a User to complete a fixed, well-defined task. For example, a User might want to book a haircut appointment, or make a hotel reservation. While TOD systems have enjoyed much success, they are designed for a specific use case and are consequently not adaptable to new tasks. A typical TOD system is shown in Figure 2.1. In the context of TOD, a dialogue is confined to a certain domain. The system can then only make a limited number of actions within that domain. As dialogue progresses, the system tracks the human user's current state and intention(s) and executes a given task once it has collected all the information it needs from the User. At a high level, TOD systems need to be able to understand a user's request, make a decision based on this request, and respond accordingly. These high level steps are similar to what is needed for situated task guidance, except the problem becomes more difficult in this case because a) interactions are less structured (i.e. no specific domain or slot-filling templates) and b) the system is not querying a database to retrieve information for a User but is actually the one giving instructions, which requires high-level decision making.

LLMs are promising candidates because they can perform the functions of a TOD system end-to-end, with virtually no restriction on the tasks they can be used for. 'End-to-end' here means that the TOD problems of user understanding, decision making, and response generation are solved all at once, rather than sequentially. Hosseini-Asl et al. (2020) demonstrated the potential of an end-to-end, unified approach with SimpleTOD, based on GPT-2 (Radford et al., 2019). By recasting the TOD sub-problems as a single sequence-

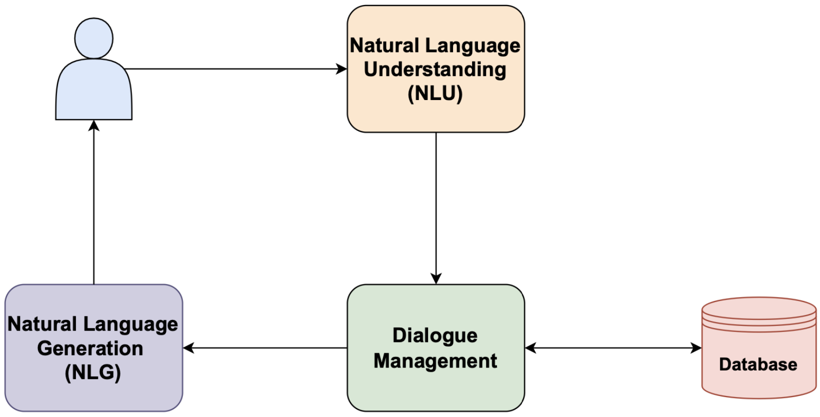


Fig. 2.1 A classic Task Oriented Dialogue System.

to-sequence generation problem, SimpleTOD was able to improve TOD performance by leveraging the pre-trained knowledge of its GPT-2 backbone.

Although LLMs show promise as situated task guidance systems, they are often prone to hallucinations. An incorrect or irrelevant instruction could confuse a User and prevent them from successfully completing the given task. Still, the free-form outputs of LLMs have great potential for tailored, natural interactions, and thus it is worth exploring whether these models can be adopted for task guidance.

2.2 Automatic Methods for Task Guidance

Previous work has focused on viewing automated agents as actors who follow instructions of an Instructor to perform physical tasks. Thus, relatively little has been done on the reverse scenario, where the agent acts as the Instructor to a human User who is performing a physical task. Fried et al. (2018) consider the traditional VLN task, where an embodied agent is given instructions to follow from a human Instructor. However, the authors also develop an Instructor model, which they call a *Speaker*. The Speaker model is designed for data augmentation and generates navigation instructions for the Follower to execute during training. The key difference between this setup and WTaG is that it only considers a scenario involving two embodied agents, whereas WTaG considers a human Follower/User. In some ways, having a human User could simplify the situation, since we can be confident that they will execute a task well if they are given a good enough instruction. But providing an instruction that is ‘good enough’ is precisely what is difficult about the WTaG task, because this requires abstract reasoning and environment understanding to do well.

Another relevant work is Task-driven Embodied Agents that Chat (TEACh) (Padmakumar et al., 2022). TEACh is a dataset on interactive gameplay sessions between two people in a simulated environment. One person assumes the role of the Commander, while the other is the Follower. The collected human demonstrations and instructions can be used to benchmark the capabilities of both Commander and Follower models. One of the benchmark tasks, Two-Agent Task Completion (TATC), concerns both the Commander and Follower. In this task, the models are given only descriptions of the environment and must interact with each other through language generation to complete a task. However, in their baseline experiments for TATC the authors opted for a rule-based approach, where both models operate under a set of hand-crafted rules. In WTaG, an LLM is used as the Commander, instead of a rule-based system.

There have also been works that tackle task guidance with an emphasis on either its visual or linguistic aspects. For example, Lu and Mayol-Cuevas (2019) build a Hand-object Interactive Guidance System (HIGS) in a mixed reality setting with bespoke CNN-based visual components to monitor and provide guidance for a specific task like operating a sewing machine. This setup is similar to WTaG, except the system is designed for specific tasks, and thus its applicability to other tasks is more limited than an LLM.

TG approaches	General model?	Multimodal?	Model as Instructor?	Real world?
TEACh		✓	✓	
Speaker-Follower		✓	✓	
ChatCRS			✓	✓
HIGS			✓	✓
WTaG	✓	✓	✓	✓

Table 2.1 Comparison of task guidance approaches

On the language side, Li et al. (2024) explore the use of an LLM to give recommendations during conversations and develop a model they call ChatCRS. ChatCRS features knowledge-retrieval and goal-planning modules that are designed to improve recommendation quality. The recommendation task could be seen as being similar to providing guidance during a conversation, except ChatCRS does not consider the multimodal setting.

Table 2.1 summarises the approaches to task guidance we have just highlighted. We can see that WTaG provides a unique opportunity to test the general capabilities of LLMs as Instructors in multimodal, real-world scenarios.

2.3 Transformer Preliminaries

This thesis primarily works with sequence-to-sequence models based on the popular Transformer-based architecture (Vaswani et al., 2017). This section introduces the technical aspects of the Transformer model to contextualise much of the underlying functionality of the models used for this project. We follow the presentation of the Transformer in Turner (2023).

2.3.1 Transforming the Data: The Encoder

The Transformer is a model that can be applied to any sequence-to-sequence problem. Although the original Transformer design features an encoder-decoder architecture, either component can be used as standalone models. We thus focus first on the encoder, which is the base architecture for popular models like BERT (Devlin et al., 2019). Let f_{θ_T} be a transformer block as in Figure 2.2, parameterised with θ_T. The goal of f_{θ_T} is to transform a $D \times N$ input matrix X representing an N-length sequence of D-dimensional tokens to a refined representation X':

$$X' = f_{\theta_T}(X) \tag{2.1}$$

To perform this transformation, the Transformer block first attends to the sequence via an $N \times N$ attention matrix A. $A_{n,n'}$ indicates the relative importance of the nth token $X_{:,n}$ in the sequence to either itself or another token $X_{:,n'}$. In particular,

$$\mathrm{SA}(X) = X_{d,:}A_{:,n} \tag{2.2}$$

Where $A_{:,n}$ denotes the nth column of A and SA is the so-called 'Self-Attention' module. Equation 2.2 effectively computes the weighted average (i.e. dot product) between the dth row of X and the nth column of A. The operation is called Self-Attention because A is computed from the input X itself:

$$A_{n,n'} = \frac{\exp(X_{:,n}^{\top} W_q^{\top} W_k X_{:,n'})}{\sum_{n''=1}^{N} \exp(X_{:,n''}^{\top} W_q^{\top} W_k X_{:,n'})} \tag{2.3}$$

Equation 2.3 shows that each column of A is effectively a softmax distribution, where each column entry represents the normalised weight of how important the token at position n is to the token at position n'. Typically, attention is thought of in terms of queries, keys and values. Query and key vectors are used to compute attention weights, which are then applied to the value vector to obtain the refined representation. In self-attention, the same vector is used as query, key, and value. Usually, a set of learnable parameters W_q, W_k, and W_v is also introduced for greater expressiveness.

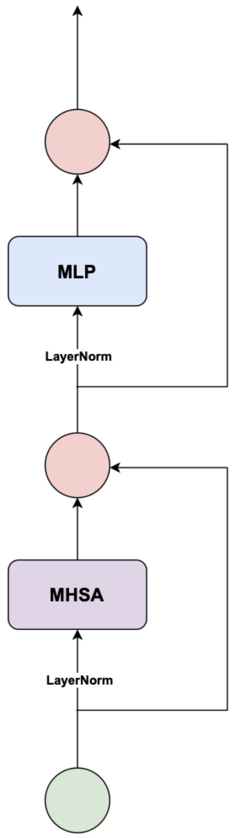


Fig. 2.2 The standard Transformer Block, as depicted in Turner (2023). The MHSA and MLP layers are the core components of this block. Inputs are sent through a residual connection and normalised via LayerNorm (Ba et al., 2016) to stabilise the learning process.

In practice, Vaswani et al. (2017) actually use multi-head self-attention (MHSA), which simply replicates the attention mechanism across H 'heads'. The result is a mechanism that can attend to different regions of the input, since each head can learn to focus on modelling different aspects of the input text. Once $\mathrm{SA}(X)$ has been obtained, it is run through an additional MLP layer to obtain the block-level representation of the input, X'. Between the MHSA and MLP stages, there are residual connections and normalisation operations (Ba et al., 2016) to aid the deep learning process (see Figure 2.2 below). Running the input through B Transformer blocks (usually 12) then yields the final encoding.

2.3.2 Autoregressive Prediction: The Decoder

Transformers with decoder-based architectures generate language in an autoregressive fashion. The goal here is to predict the next word of the text output given all previous words:

$$\hat{w}_n \sim p(w_n|\hat{w}_{1:n-1}) \tag{2.4}$$

Where $\hat{w_n}$ is the observed word at position n, and $\hat{w}_{1:n-1}$ is the previous observed word sequence up to position n. To obtain $p(w_n|\hat{w}_{1:n-1})$, we can simply add a final classification head to the base architecture from Figure 2.2 to project the final representations to the dimensionality of the target vocabulary. However, since the decoder has access to the entire sequence, we can also introduce 'masking' to prevent future tokens from influencing the model's prediction at position n. This modification is called 'masking', and essentially sets the portion of the matrix corresponding to future positions to 0. With masking and the final softmax layer, the base Transformer architecture becomes suitable for causal language modelling. With masking, the observed word sequence so far $\hat{w}_{1:n-1}$ is represented as $\mathbf{x}_{n-1}$. This vector captures causal information across the sequence, and can be used to incrementally generate the next word $\hat{w}_n$. This process is repeated and the model continues to generate one word at a time until some stopping criterion is met.

Prominent generative models such as GPT-3 (Brown et al., 2020) and LLaMA (Touvron et al., 2023) are built on this decoder-only architecture, and in fact the latter is what most of the models used for this project are based on.

2.4 Instruction Tuning

Transformer-based LMs may be able to learn a strong general model of language, but a significant advancement in the real-world applicability of these models stems from the practice of instruction tuning (IT). IT drives the development of the popular chat systems used today, namely ChatGPT, Gemini, Claude, etc. The idea is to pass in pairs of prompt instructions and associated desired responses to a standard autoregressive model like GPT-3. In doing so, the ability of a standard LM to follow instructions is augmented. IT can also be cast as an alignment problem, where an LM should *align* with a human user's intent. Standard LLMs may generate unhelpful or even toxic content depending on which data was used for training. Instruction tuning aims to bridge the gap between a model's internal linguistic knowledge and its ability to align with human intent. Ouyang et al. (2022) introduced InstructGPT, and demonstrated the benefits of IT by explicitly fine-tuning GPT-3 on data aligned with human feedback.

Of course, collecting IT data is expensive. Since LLMs are generative, a natural option is to leverage them to cheaply generate new synthetic datasets. These datasets can subsequently be used to further train other models in a supervised fashion. In fact, the data generated from an LLM could even be used to fine-tune itself, as Wang et al. (2022) demonstrate with the SelfInstruct framework.

2.4.1 Relation to Knowledge Distillation

Performing IT on synthetic data can be viewed as a form of knowledge distillation (KD) from a teacher model to a student model. The typical KD setup involves a student minimising the Kullback-Leibler (KL) divergence of its distribution to the teacher distribution. However, in cases where the full teacher distribution may be inaccessible to the student, the teacher-generated outputs can be viewed as approximating the mode of this distribution. Kim and Rush (2016) introduce sequence-level knowledge distillation (SEQ-KD), and formulate an objective over all possible generated text sequences:

$$\mathcal{L}_{SEQ-KD} = -\sum_{t \in \mathcal{T}} q(\mathbf{t}|\mathbf{s}) \log p(\mathbf{t}|\mathbf{s}) \tag{2.5}$$

Where $q(\mathbf{t}|\mathbf{s})$ is the teacher distribution and $p(\mathbf{t}|\mathbf{s})$ is the student distribution, $\mathbf{t}$ is the 'target' or output sequence, and $\mathbf{s}$ is the 'source' or input sequence. The set $\mathcal{T}$ is the set of all possible sequences. If we approximate q with its mode, we get:

$$\begin{aligned}\mathcal{L}_{SEQ-KD} &\approx -\sum_{t \in \mathcal{T}} \mathbb{1}\{\mathbf{t} = \hat{\mathbf{t}}\} \log p(\mathbf{t}|\mathbf{s}) \\ &= -\log p(\hat{\mathbf{t}}|\mathbf{s}) \end{aligned} \tag{2.6}$$

Assuming a greedy decoding strategy, $\hat{\mathbf{t}}$ would be equivalent to the teacher-generated output. Thus, Equation 2.6 shows that performing supervised training on teacher-generated outputs is a special case of SEQ-KD. This form of KD applied to IT data drives much of the progress around the recent open-source foundation models that are used for this project, and in fact we will perform further SEQ-KD on Vicuna with Mixtral (Jiang et al., 2024) in Section 4.3.2.

Most of the models we use build on Vicuna (Zheng et al., 2023). Vicuna is a variant of LLaMA that has been fine-tuned on data from the ShareGPT[1] website. This data consists of pairs of user-shared prompts and the corresponding responses from ChatGPT, thus providing

[1] https://sharegpt.com

a good opportunity to perform SEQ-KD. In this case, ChatGPT is the indirect 'teacher' and LLaMA is the student. Essentially, Zheng et al. (2023) are leveraging the IT data from ShareGPT to teach LLaMA how to follow prompts in a similar way to its indirect teacher, ChatGPT. We now turn to the ways in which Vicuna has been further fine-tuned to perform tasks in multimodal settings.

2.4.2 Multimodal Instruction Tuning

Extending an LLM like Vicuna to multimodal tasks requires modifying the current model architecture to handle prompt-response pairs with different modalities. In this thesis, we only consider the scenario where prompts are multimodal and responses are always text, but recent works have explored multimodal responses as well (Koh et al., 2024; Wu et al., 2023).

A well-known example of an instruction-tuned model in the vision domain is the aforementioned Large Language and Visual Assistant (LLaVA) model Liu et al. (2024b). Liu et al. (2024b) take Vicuna as a starting point and fine-tune it with a synthetic IT dataset[2] that is curated for image understanding tasks. But how can an LLM be fine-tuned for a task that requires processing an image?

It turns out that the image can simply be passed to a *separate* Transformer that has been specifically trained to encode images. Usually, this would not be possible since a raw image is not a sequence. However, if we note that an image can be represented as a sequence of its flattened, projected patches, it becomes possible to apply a pure Transformer encoder to obtain an encoding of the image that makes full use of MHSA. This insight from Dosovitskiy et al. (2020) led to the development of the aptly named Vision Transformer (ViT).

ViT forms the visual backbone of LLaVA and many other multimodal models. The model was originally used for image classification, but when used as the encoder for a multimodal instruction-tuned agent, it is typically pre-trained using a contrastive objective (Radford et al., 2021). This objective guides ViT to bring image-text pairs that are semantically similar closer together in the shared embedding space, while pushing pairs that are less similar further apart. Through this strategy, the resulting image representation becomes tied to the representation of the text caption it was paired with, which is useful for many downstream tasks such as VQA and image captioning.

Thus far we have introduced an LLM backbone Vicuna, and a vision backbone ViT. The last step is to design a mechanism to connect the representations from these two models together, to form the unified architecture of LLaVA. Early versions of LLaVA implemented a simple projection matrix W to project the visual representation to the language embedding

[2]The data was generated with GPT-4 (Achiam et al., 2023)

space, while more recent versions such as LLaVA-1.5 (Liu et al., 2023) use an MLP layer instead for higher-quality results. Figure 2.3 illustrates the complete LLaVA architecture.

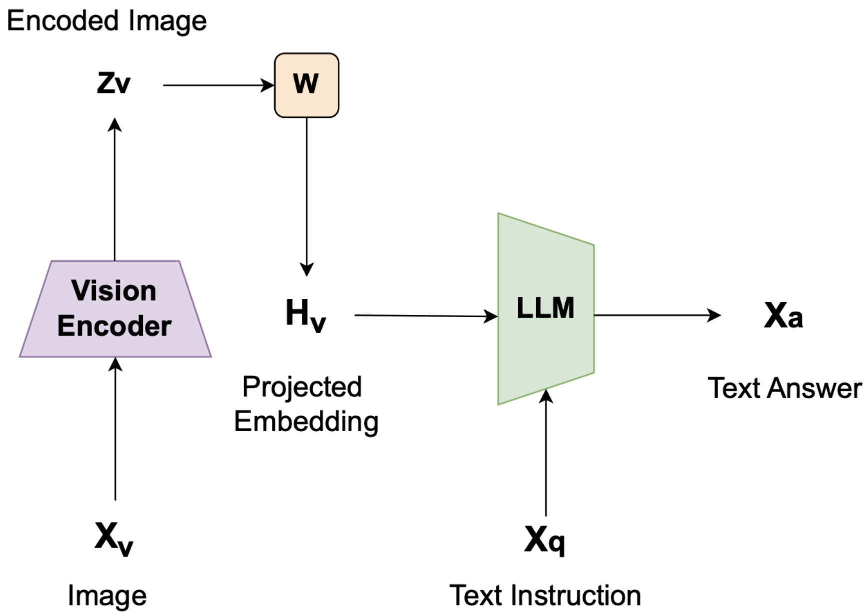


Fig. 2.3 The LLaVA architecture. Image-text pairs are fed to a frozen vision encoder and LLM that are connected through a projection matrix.

To train LLaVA for instruction-following tasks, Liu et al. (2024b) design a custom synthetic dataset for multimodal IT, LLaVA-Instruct-158k. During fine-tuning, only the parameters of the base LLM and the projection layer are updated. Naturally, we should also note that projection layers and MLPs are not the only way to connect different models. The recent BLIP-2 and PandaGPT models (Li et al., 2023; Su et al., 2023) for instance employ a Query-Transformer (Q-former) and ImageBind (Girdhar et al., 2023) respectively. However, the underlying idea remains the same. First, connect powerful pre-trained unimodal models via some mechanism to bootstrap a strong multimodal model. Then, fine-tune on IT data to obtain a multimodal instruction-following agent.

2.4.3 Instruction Tuning for Task Guidance

Having discussed the underlying background behind most of what is required for the project, we explain how this knowledge is relevant to task guidance. Recall that IT gives foundation models strong instruction-following capabilities. From an application perspective, having a general-purpose model that is good at following instructions is powerful, because we can ask it to perform arbitrary tasks. This makes IT models highly versatile and well-suited for situated task guidance. Leveraging IT models for task guidance is particularly interesting because we are essentially assessing whether they can be *instructed to instruct*.

2.5 Summary

This chapter introduced much of the background needed to understand the rest of the thesis. By bringing together the fundamental ideas of Transformer models, instruction tuning, and multimodal bootstrapping, we can apply a range of open-source models to the WTaG benchmark and evaluate their zero-shot capabilities on different aspects of the instruction generation process.

Chapter 3

Watch, Talk, and Guide

3.1 Overview

The Watch, Talk, and Guide (WTaG) benchmark (Bao et al., 2023) was constructed in an effort to assess whether state-of-the-art foundation models can act as Instructors when Users are performing a physical task. The benchmark is multimodal, and features egocentric videos of a User performing a task accompanied with text annotations. During data collection, the User wears a camera-enabled headset and is required to complete three physical tasks related to cooking. Specifically, the User is tasked to make:

1. Five pinwheels.
2. A coffee.
3. A cake with chocolate icing.

Throughout the tasks the User is guided by a human Instructor in real-time. The dialogues between Instructor and User are transcribed via ASR, and manually corrected. Ground truth (GT) annotations of 'step' segments and their associated time-stamps in the video are included as well.

While smaller than other related datasets, WTaG aims to contribute data featuring more natural human-to-human interactions and real-world egocentric video. The biggest egocentric dataset is Ego4D Grauman et al. (2022), however it does not feature interactive sessions. Other datasets such as TEACh (Padmakumar et al., 2022) and ALFRED (Shridhar et al., 2020) conduct sessions in simulated environments, and thus lack the complexity of in-the-wild scenarios. WTaG therefore compensates for its relatively smaller scale by bridging the gap between the shortcomings of other similar datasets.

3.1.1 Statistics

WTaG consists of 56 egocentric videos, collected by 17 User participants and 3 Instructors. There are 19 pinwheel videos, 18 coffee videos, and 19 cake videos. The video lengths vary from 5 to 18 minutes, with a median length of 10 minutes. For the original experiments conducted by Bao et al. (2023), 6 videos were reserved for prompt tuning, while the rest of the 50 were used for the experiments. However, the publicly available version of WTaG used in this thesis only has 48 of the original 56 videos. This is because the tuning set of 6 videos is not included, and because 2 of the original videos contained sensitive information.

3.2 Tasks

Recall from Chapter 2.1 that like a TOD system, a task guidance system should be able to:

I. Understand User state and intent.

II. Make decisions as an Instructor.

III. Generate a relevant response.

Since it is difficult to quantify the quality of task guidance directly (point III), Bao et al. (2023) define more objectively measurable sub-tasks that test for a model's capabilities regarding points I) and II). The assumption is that a model that performs well on sub-tasks related to I) and II) would indirectly signal that a foundation model is suitable for III).

Concretely, the sub-tasks proposed by Bao et al. (2023) are all framed as classification problems split into two broad categories. These are:

- **User and Environment Understanding (UEU)**
    1. Step Detection: K classes for a K-step recipe.
    2. Mistake Detection: 2 classes (yes/no).
    3. Mistake Type: If applicable, 3 classes (Wrong State, Object, or Action).
    4. User Intent Prediction: 6 classes (Question, Answer, Confirmation, Self Description, Hesitation, Other).

- **Instructor Decision Making (IDM)**

  1. When To Talk: Should the Instructor talk? 2 classes (yes/no).
  2. Instructor Intent Prediction: 5 classes (Instruction, Question, Answer, Confirmation, Other).
  3. Instruction Type: If applicable, 4 classes (Current Step, Next Step, Details, Mistake Correction).

User and Environment Understanding (UEU) tasks map to point I) above, with Instructor Decision Making (IDM) tasks mapping to II).

3.3 Experimental Setup

In order to understand the setup of the WTaG experiments, it is helpful to run through an example experiment to better understand how results were collected for WTaG. Sessions in WTaG are labelled as their video code followed by the recipe type they belong to (e.g. 'T10_pinwheel' or 'T55b_cake').

An experiment consists of running through a WTaG video session frame by frame and prompting the foundation model (originally ChatGPT) at certain Query Points (QPs) to answer a set of questions pertaining to the UEU and IDM tasks. Ideally, the model would be prompted continuously at every frame, but this is unrealistic in practice due to the compute it would require. Thus, Bao et al. (2023) prompt the model under 3 specific circumstances, which we call 'QPs'. A frame F of the video is a QP if:

a The User talks (a GT User utterance begins at frame F).

b The Instructor talks (a GT Instructor utterance begins at frame F).

c No one has talked in the 10 seconds preceding frame F (no GT utterance has been observed in the last 10 seconds).

One session might have 50 QPs, meaning there would be 50 outputs from the model. Across the set of 48 videos available to us, there are a total of 5297 QPs. The prompts at every QP follow a specific format. Prompts begin with an introduction to the task, then show the GT recipe being followed. The chat history up to the current frame is also included, to give the model a sense of what has been talked about. Optionally, additional information may be included to provide further context to the model. In the original paper, BLIP-2 (Li et al., 2023) was used as an image captioner to provide information about the current frame to ChatGPT. Finally, the model is asked a suite of questions to obtain its predictions for the UEU and IDM tasks described previously. Figure 3.1 illustrates the full setup for WTaG, and an example of the full prompting scheme is available in Appendix A.

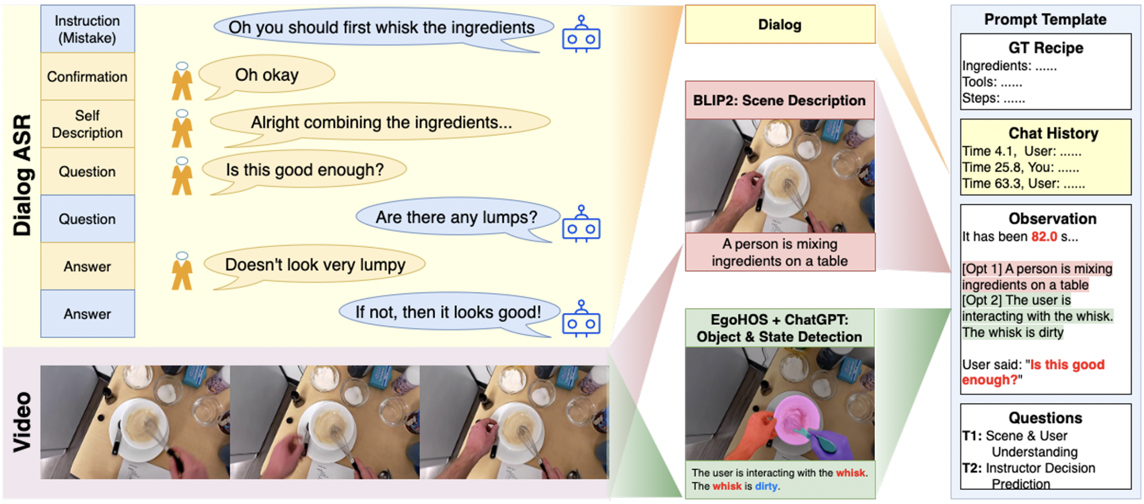


Fig. 3.1 The full WTaG setup, as shown in Bao et al. (2023). The prompt template includes the GT recipe, dialogue history, and an optional BLIP-2 caption, followed by the WTaG questions. We do not consider the EgoHOS setup in this thesis.

3.4 Evaluation

Now that the experimental setup for WTaG has been explained in detail, we turn to the matter of evaluating the model outputs. The key challenge here is post-processing the outputs to ensure they are clean for metric calculation. Once this is done, objective evaluation of the UEU and IDM tasks is straightforward, because they are all computed with a standard F1 score. The F1 score for a set of model predictions is the harmonic mean of its Precision and Recall:

$$\mathrm{F1} = \frac{2PR}{P+R} \tag{3.1}$$

Where P is Precision and R is Recall. P is computed as:

$$P = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \tag{3.2}$$

Where TP is the number of True Positives in the prediction set and FP is the number of False Positives.

As for R, this is computed as:

$$R = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \tag{3.3}$$

Where FN is the number of False Negatives. Precision thus rewards the model for identifying few FPs, while Recall rewards the model for minimising FNs. F1 is a balance of these metrics, meaning a model could have high Precision but a low F1 score if its Recall is low. Equations 3.1 to 3.3 are typically designed for binary class problems. However, some of the WTaG tasks are multi-class problems, such as User Intent Prediction. In these cases, we compute the *Micro* F1 score instead. To compute Micro F1 score, we simply sum up TPs, FPs, and FNs across all K classes and compute P, R and F1 as normal.

More concretely, we can derive a sequence of reference labels from the GT annotations, then compare this sequence to the predicted labels from the models. For Step Detection and Mistake Detection, the reference sequence spans all the QPs across all experiments. We obtain this sequence by concatenating the references from each experiment into one big sequence. This reference sequence is then compared with the predicted sequence, which is a concatenation of all the predicted labels across all QPs. The F1 score is then computed through Equation 3.1. Other tasks like User Intent Prediction and Instructor Intent Prediction are concerned with only a subset of all QPs, since they are only relevant to QPs where the User or Instructor speak, respectively.

3.5 A Note on the Mistake Detection Task

While replicating the original WTaG setup (the exact methodology is described in the next chapter), we found that mistake detection was the most difficult to evaluate in practice compared to the other WTaG tasks. Remember that for a video with N QPs, the mistake detection annotations are essentially an N-length sequence with entries of either 0 or 1, where a 0 denotes no mistake at QP n and 1 denotes a mistake. However, upon inspecting the original GT annotations, it turns out that mistakes are only marked on Instructor utterances, as the WTaG authors made the assumption that a mistake *always elicits* a response from the Instructor. The problem is that annotating a mistake this way means that we are effectively marking the *end* of a mistake interaction, which means we cannot recover the exact point where the mistake began. For example, if a User places a tortilla wrap on a plate instead of a cutting board, there may be a significant delay (for example, a few query points) between the User making the mistake and the Instructor addressing it. In other words, mistakes are marked at a single Instructor query point, when in reality a mistake could span multiple QPs.

This means that models could be penalised for detecting a mistake before the ground truth QP, even though this may actually be correct.

Aside from the annotation issue, there is another problem related to the inclusion of the dialogue history. In the current WTaG setup, the model is asked ‘Did the user make a mistake?’ to test for mistake detection (see Appendix A). Since the full dialogue history is included in the original setup, there is the risk that a model would continue to answer ‘Yes’ to this question even after a mistake has occurred, because evidence of the mistake is still present in the dialogue history. This is only a problem for mistakes that are strongly signalled in the dialogue, but still poses a risk to the integrity of the collected results.

3.5.1 Mistake Bitmaps

To illustrate the problems described above, we plot some bitmaps for sessions T10_pinwheel, T11_coffee, and T4_cake. These show the reference mistake sequence derived from the GT Instructor annotations, and the corresponding model predictions. For session T10_pinwheel (see Figure 3.2), we see that there is a GT mistake at QP 18, but both LLaVA and Vicuna have flagged a mistake at QP 11. The same occurs at QP 34 for LLaVA:

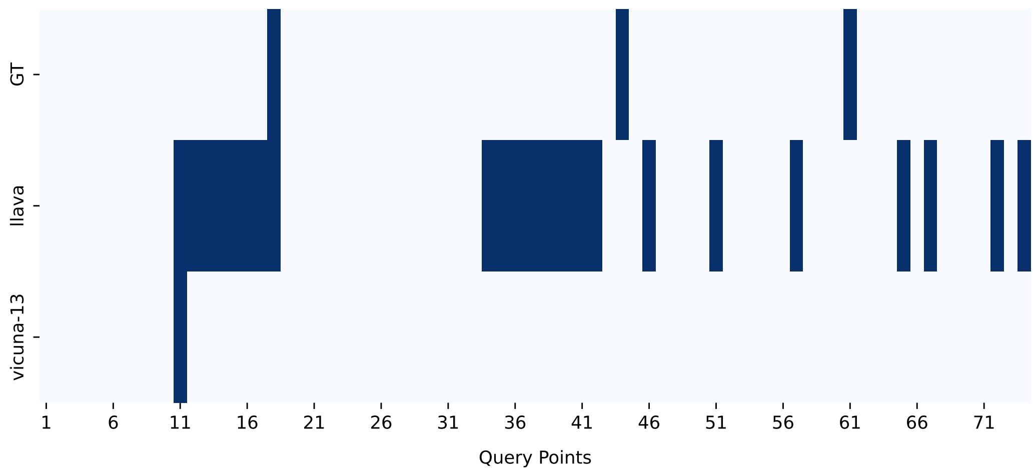


Fig. 3.2 Mistake bitmap for T10_pinwheel. From top row to bottom: GT, LLaVA, Vicuna.

Taking a closer look at QP 11 in Figure 3.3, we see that the Instructor says ‘but you should be placing the tortilla on a cutting board’ at time 79.3. This utterance may mark the beginning of a mistake, but it has not been annotated in the GT. However, even if it had been annotated, evidence of the mistake is kept in the dialogue history. Thus, there is always the risk that the model will fixate on this mistake, depending on how strongly it was originally signalled.

T10_pinwheel, Frame 2671 (QP 11)

```
Chat History:
Time: 50.6, User: we'll start the recipe now
Time: 61.1, User: the recipe says to place a tortilla on the
bowl but i have a plate
Time: 66.5, User: is that okay?
Time: 68.8, You: if you need extra stuff
Time: 73.7, User: thank you
Time: 79.3, You: but you should be placing the tortilla on a
cutting board
Time: 91.4, User: ohh
It has now been 92.9 seconds since the start of the recipe.
The user just said: 'they're all stuck together'
```

Fig. 3.3 Chat history of a GT mistake in T10_pinwheel at QP 11.

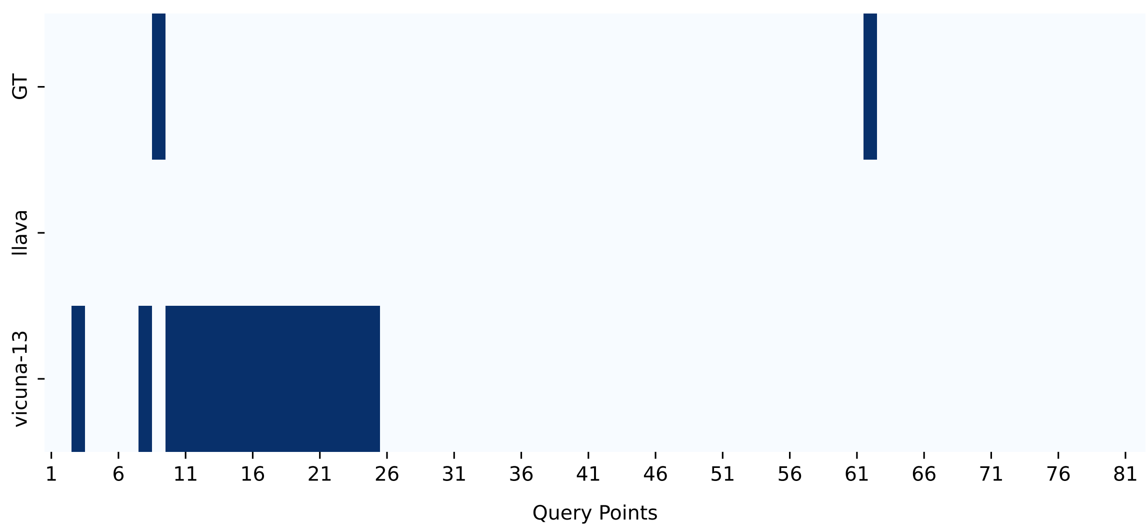


Fig. 3.4 Mistake bitmap for T11_coffee.

'Mistake fixation' is a significant problem because it means we cannot be sure if the model is detecting the most recent mistake or a previous one it has fixated on. T11_coffee shows a clear example of this issue. In this session, the first GT mistake is marked at QP 9 (see Figure 3.6). Here the Instructor says 'oh we should only need 12 ounces'. From

Figure 3.6 we see that Vicuna then immediately detects the mistake from QP 10 onwards, and continues to detect it until QP 25. The issue with the current setup is that we would penalise the model for QPs 10 to 25, even though it is technically not wrong.

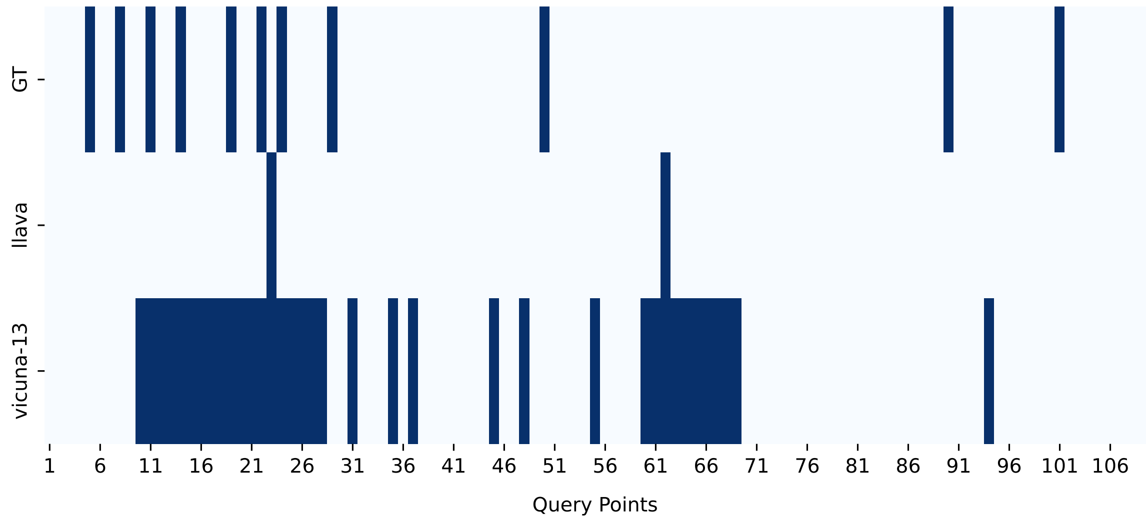


Fig. 3.5 Mistake bitmap for T4_cake.

T11_coffee, Frame 1488 (QP 10)

```
Chat History:
Time: 10.6, User: 15 ounces of cold water, transfer to kettle
Time: 17.8, User: this water?
Time: 21.0, You: yes, it’s not hot
Time: 44.3, You: how much water are we pouring
Time: 47.5, User: 14 ounces
Time: 49.3, You: oh we should only need 12 ounces
It has now been 51.6 seconds since the start of the recipe.
The user just said: ’what should i do with the extra water?’
```

Fig. 3.6 Chat history of a GT mistake in T11_coffee at QP 10.

Finally, Figures 3.5 and 3.7 illustrate a scenario where many GT mistakes occur in quick succession (QPs 6 to 30), potentially causing a model (e.g. Vicuna) to simply fixate on one. The problem with this is that although the model is ‘correctly’ detecting GT mistakes, it is not necessarily distinguishing them from one another. At QP 23 for example, the User made the mistake of using too much salt. However, the model fixates on an earlier mistake, namely the incorrect quantity of baking powder previously used. The response from Vicuna is included for reference.

T4_cake, Frame 3241 (QP 23)

```
Chat History:
Time: 86.1, User: wait
Time: 87.3, You: no baking powder is that one
Time: 95.6, You: how much are you adding
Time: 96.6, User: one teaspoon
Time: 97.8, You: oh no you only use a quarter teaspoon
Time: 100.1, User: oh
Time: 116.2, You: uh only one pinch

Vicuna: The user made a mistake by adding too much
baking powder. The correct amount is a quarter teaspoon,
not one teaspoon.
```

Fig. 3.7 Chat history of a GT mistake in T4_cake at QP 23.

3.6 Summary

This chapter introduced the WTaG benchmark and its associated tasks. Measuring the quality of task guidance objectively is challenging, so the benchmark aims to frame tasks related to instruction generation as classification problems to more easily quantify a foundation model's task guidance capabilities.

Chapter 4

Methodology

4.1 Replicating Watch, Talk, and Guide

The original WTaG approach featured ChatGPT as the core reasoning agent, with optional visual information being provided by BLIP-2 via an image caption. However, ChatGPT (specifically, ChatGPT 3.5 Turbo) is a unimodal LLM, and it is closed source. The recent efforts on multimodal IT raise natural questions about how open-source models would fare on the WTaG benchmark. As a result, we begin by establishing preliminary open-source baselines on WTaG. Note that the baselines established in these preliminary investigations are partial. This means that results were collected only for a small subset of all sessions. This was done to save compute as we noticed that inference was slow when replicating the original WTaG setup exactly (~30 minutes per session), presumably because of prompts that grow long due to the full dialogue history being included.

For the preliminary baselines, we use models previously covered in Chapter 2, namely Vicuna (Zheng et al., 2023), LLaVA (Liu et al., 2024b), and LLaMA-2 (sizes 7B and 13B) (Touvron et al., 2023). This preliminary model set allows us to test the impact of important design choices such as model architecture (LLMs vs. LLaVA), size (LLaMA 13B vs. LLaMA 7B), and training set (Vicuna v. LLaMA) on WTaG performance.

4.1.1 Challenges

The main challenges with replicating the setup stemmed from inference constraints and the lack of an evaluation script. Even with Flash Attention (Dao et al., 2022), inference times for each session ranged from 30 minutes to over an hour. This is clearly impractical and cannot be scaled up to all sessions across all models, which is why we chose to perform preliminary experiments on only a subset of the full dataset. Additionally, no evaluation script for the

WTaG tasks was publicly released, and so this was implemented from scratch. The hope is that the custom implementation replicates the original version as closely as possible.

4.2 Narrowing the Project Scope

The preliminary baseline results will be discussed in more detail in Chapter 5.1, but they essentially demonstrate that the open-source models are most lacking in step detection and mistake detection. This is not surprising, as out of the 7 tasks in WTaG, we believe that these tasks require the most multimodal knowledge and reasoning. User and Instructor intents do not rely as much on visual information, because they can largely be inferred from the dialogue history alone. Although figuring out 'when to talk' *could* rely on visual information, it is a more subjective task as an Instructor could plausibly talk at different QPs to the GT, yet still deliver adequate guidance.

Since step detection and mistake detection are the areas of the WTaG benchmark that open-source models struggle most with, we initially set out to improve performance on both tasks. However, due to the issues discussed in Chapter 3.5 related to the mistake annotations and 'model fixation', we chose to focus on step detection alone, and leave mistake detection for future work.

Thus, Section 4.3 formally narrows the focus of the thesis to improving step detection performance on WTaG. The step detection problem is akin to an action segmentation problem, which has been tackled by many previous works. The key difference in this thesis is that we make use of instruction-tuned foundation models, as opposed to the more bespoke architectures that related works such as VideoCLIP (Xu et al., 2021), Cross-Task (Zhukov et al., 2019), and NeuralSeg (Xing et al., 2024) have used.

4.3 Avenues for Improving Step Detection

This section presents the systematic study we conducted to explore potential methods for improving step detection performance on WTaG. We explore a) designing a simpler prompt template, b) incorporating scene captions, and c) performing instruction tuning Vicuna on WTaG outputs generated by Mixtral. First, we replace the default prompt template with our simpler design, then modify this new template to either augment it with a scene caption or use it for synthetic data generation. Approaches a) and b) are described in the following sections, while approach c) is presented separately in Section 4.3.2.

4.3.1 Prompt Design

Simplified Prompt Scheme

The first approach we consider is reducing the complexity and length of the prompt. With the default scheme, prompts can get lengthy because of the full dialogue history and detailed steps in the GT recipe. For instance, Step 2 of the default pinwheel recipe is: ‘Use a butter knife to scoop nut butter from the jar. Spread nut butter onto tortilla, leaving 1/2-inch uncovered at the edges.’ This description breaks down the action of spreading peanut butter on a tortilla into smaller sub-steps, but is fairly granular. The detail about ‘leaving 1/2-inch uncovered at the edges’ is not needed to detect the step. Thus, we simplify Step 2 as ‘Scoop and spread nut butter on the tortilla’. This simplification captures the key purpose of the step, and omits unnecessary details. We also remove the ‘Tools and Utensils’ section of the GT recipe as this information is encapsulated in the steps.

As for the dialogue history, we truncate this to include only the 10 previous turns between Instructor and User. This is done to prevent prompt lengths from exceeding the context length of the models, especially for the longer videos in the dataset. More importantly, we removed temporal information from the prompt as well, since we noticed that Vicuna and LLaVA were confusing the timestamps as steps (see Figure 4.1).

Vicuna Step Hallucination

```
Time: 60.8, User: alright
Time: 73.0, You: you should place the tortilla on the cutting
board
Time: 76.7, User: ok
Time: 83.3, You: oh you should use a knife instead of the fork

Vicuna: The user is at step 76.7, which is the step where the
user should use a knife instead of a fork to spread the nut
butter on the tortilla.
```

Fig. 4.1 An example of Vicuna conflating the a timestamp included in the history as a step.

Finally, focusing on step detection means we only need to ask the model one question. We believe that asking the other questions at the same time complicates the setup, and could affect performance and inference times, since the prompt is longer. We also stop asking the model to structure its responses in JSON format, as we hypothesise that this additional instruction could be contributing to the problem.

Figure 4.2 overleaf shows the general template of our simplified prompting scheme. The template includes an optional image (discussed further in the next section). A full example of a prompt following this template is available in Appendix A for comparison to the default prompting scheme.

To summarise, we simplify the default prompt in the following ways:

- Step descriptions are reduced to one sentence that captures the main goal of the step. Unnecessary details are also omitted.
- Temporal information and the 'Tools and Utensils' section of the GT recipe are removed
- Chat history is truncated to last 10 turns
- Only ask the model 1 question, as opposed to 7

The expectation is that by adopting this simpler prompt, models can more adequately follow the instructions they are given and perform better on the step detection task.

General Structure

```
<TASK CONTEXT>
<INGREDIENTS>
<SIMPLIFIED STEPS>
<TRUNCATED CHAT HISTORY>
<IMAGE CAPTION>
<QUESTION>
```

Fig. 4.2 The general structure of the simplified prompt scheme described in this section.

Captioning and Frame Selection

Another way of manipulating an input prompt is to augment it with information from the current scene as a caption. This was implemented by the original WTaG authors, with BLIP-2 as the captioner and ChatGPT as the LLM backbone. Majumdar et al. (2024) refer to this type of system as a **Socratic LLM w/ Caption**. Figure 4.3 shows the general setup of an LLM with an external captioning module. The captioning module (in our case a VLM) captions the image, then the LLM guesses the current step based on the dialogue context and the additional context from the caption. We additionally build on the original approach by implementing a 'frame selector'. In WTaG, models cannot have access to future frames in the video, because this would not simulate a real-time task guidance setup. As such, we assume that models only have access to the video from the first frame up to and including the current frame. In the original approach, when a model was prompted at a QP, only the current frame at that QP was passed to BLIP-2 (Li et al., 2023) for captioning. However, upon closer examination we noticed that these frames are not always relevant to the current step, or are of poor quality (see Figures 4.4a and 4.4b).

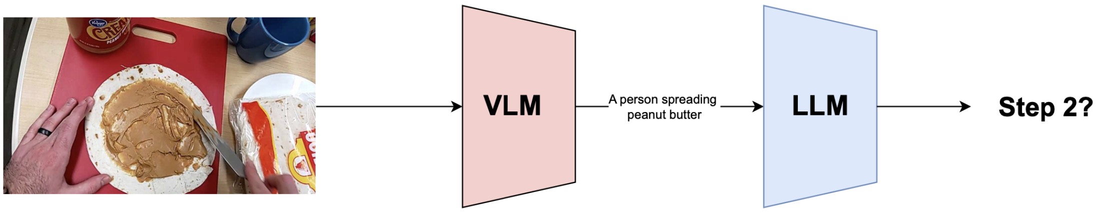


Fig. 4.3 A 'Socratic' LLM with an external captioning module.

(a) A blurry frame from T4_cake. (b) An unhelpful frame from T10_pinwheel.

Fig. 4.4 Examples of sub-optimal frames in the T4 and T10 sessions.

From Figure 4.4 it would seem that the visual information in the scene is not being exploited as much as it could be. We thus implement a light-weight frame selection module in an effort to select more optimal frames than the original set. This frame selector uses peak estimation to identify the closest key frame to the current frame[1]. Then, the sharpest image within 3 seconds of this key frame is selected as the final frame to pass to the captioner. A measure of blurriness is obtained by applying a Laplacian filter to the (grey-scale) frame and computing the variance of the result (Pech-Pacheco et al., 2000). High variance indicates a sharp image.

Across all our experiments that use a **Socratic LLM w/ Caption**, we use LLaVA-1.5 (13B) (Liu et al., 2023) as the captioner. The model is quantised to 4-bit precision so that the experiments can run on a single A100 80GB GPU. We feed LLaVA the following prompt: '*Describe the key ingredients and utensils that are being used in this image, if any*'.

[1]https://github.com/joelibaceta/video-keyframe-detector/tree/master.

4.3.2 Instruction Tuning

In addition to manipulating or augmenting the input prompt, we also investigate explicitly fine-tuning Vicuna on outputs generated by Mixtral, in an attempt to address the shortcomings of Vicuna on the step detection task. The idea is to perform SEQ-KD (refer back to Section 2.4.1) with Vicuna as the student and Mixtral as the more capable teacher model.

To implement this approach, we select a subset of WTaG consisting of 14 out of the 48 available videos (5 pinwheels, 5 coffees, 4 cakes). The exact list is in Appendix C. Then, we use Mixtral to generate responses as in previous experiments, except we also give the model access to the GT step. The prompt follows the simplified template described in Figure 4.2, but replaces the final <QUESTION> with: '*Based on the situation described, explain in detail why the user is on Step* <GT STEP> *of the recipe*'. The core idea is to fine-tune Vicuna to approximate the way Mixtral 'reasons' about the current situation to improve step detection. By injecting information about the GT step into the prompt, we steer Mixtral to focus on aspects of the input that are relevant to that step.

In the subset we have chosen, there are a total of 1549 QPs. Since we have QPs from each session type, we can group those of the same type together, yielding 3 small subsets: $\mathcal{D}_{\text{pin}}$, $\mathcal{D}_{\text{cof}}$, and $\mathcal{D}_{\text{cak}}$. Due to sessions being unequal lengths, these datasets have different sizes (534, 536, and 479 respectively). In practice, we interleave[2] the subsets to yield the final dataset, $\mathcal{D}_{\text{mix}}$. Interleaving the samples allows us to control the proportions of each subset we would like in $\mathcal{D}_{\text{mix}}$. Initially, the proportions are kept uniform. More formally, $\mathcal{D}_{\text{mix}}$ is defined as:

$$\mathcal{D}_{\text{mix}} = \{(x,y)_i\}_{i=1}^{N}, \text{ where } (x,y)_i \sim \{\mathcal{D}_{\text{pin}} \vee \mathcal{D}_{\text{cof}} \vee \mathcal{D}_{\text{cak}}\} \tag{4.1}$$

Where N is the size of the dataset, and (x,y) is a data pair sampled independently from one of the subsets with pre-defined interleaving probabilities. A data pair consists of an input prompt x and the corresponding Mixtral response y (see Appendix C for an example). Once $\mathcal{D}_{\text{mix}}$ has been constructed it is then used to fine-tune Vicuna in a supervised fashion with standard cross-entropy loss.

We investigate two experiments with this instruction tuning approach. The first trains Vicuna on a dataset $\mathcal{D}_{\text{mix}}$ with uniform proportions of QPs from each subset. The second trains the model on QPs from only one subset (we choose pinwheels as this generally proves to be the most challenging task). In the first experiment, we expect Vicuna to improve across all tasks, since the 3 session types are equally represented. The hope is that the model would be able to generalise patterns it learns from being exposed to various types of QPs and

[2] https://huggingface.co/docs/datasets/v2.20.0/en/package_reference/main_classes.

situations represented in the $\mathcal{D}_{\text{mix}}$. On the other hand, the second experiment studies the effect of using only pinwheel QPs to address the relatively weak step detection performance of Vicuna on this task.

We conduct training by making small changes to the commonly used script for replicating Stanford Alpaca[3] (Taori et al., 2023). This script is designed to fine-tune a LLaMA-based model with LoRA (Hu et al., 2021). LoRA trains a large model efficiently by representing a parameter update as a decomposition of low-rank matrices:

$$W_{\tau+1} = W_{\tau} + BA \tag{4.2}$$

Where W represents the now frozen parameters of the pre-trained model and A and B are learnable low-rank matrices that are injected into each layer of the model. Updating the low-rank matrices instead of the full model reduces the parameters needed for training, without much loss in performance. Training details and results are discussed in Section 5.3.

4.4 Model Set

We summarise the full set of models we use across all our experiments:

- **Blind LLM**: Vicuna (13B), LLaMA-2 (7B, 13B), Mixtral (47B)
- **Single-Frame VLM**: LLaVA (13B), LLaVA-NeXT (13B)
- **Socratic LLM w/ Caption**: Vicuna (13B) + LLaVA (13B), Vicuna (13B) + LLaVA-NeXT-Video (7B), Mixtral (47B) + LLaVA (13B)
- **Fine-Tuned LLM**: Vicuna (13B) + LoRA

In addition to the models used for the initial experiments, we consider Mixtral (Jiang et al., 2024) and LLaVA-NeXT (Liu et al., 2024a). Mixtral is a sparse mixture of experts LLM, and features 8 experts at each layer. At inference time, the model uses a router network to propagate outputs from only 2 of the 8 experts, thus improving effiency. In practice, we quantise Mixtral to 4-bit precision so that all experiments can be run on a single A100 GPU. On the other hand, LLaVA-NeXT improves on LLaVA by being able to handle higher-resolution images, and has been tuned on a higher-quality IT dataset. In sum, Mixtral allows us to investigate the effects of scale and architecture on step detection, while LLaVA-NeXT sheds insight on the importance of training data quality.

[3]https://github.com/tloen/alpaca-lora/blob/main/finetune.py

Due to resource constraints, we cannot run an exhaustive study. However, we believe that the experiments we conduct are informative and highlight the key effects of different design choices on step detection performance.

4.5 Summary

This chapter covered the methodology we followed to a) establish preliminary open-source baselines on WTaG and b) conduct a systematic study aimed at improving step detection performance. To achieve this we manipulate the input prompt, implement frame selection, and perform further instruction tuning of the Vicuna model.

Chapter 5

Results and Discussion

This chapter compiles the results from the various experiments that were described in Chapter 4. We begin with the results from the preliminary experiments, then move to our step detection experiments. Note that **all tabular results in this chapter should be interpreted as binary/micro F1 scores**.

5.1 Preliminary Experiments

To conduct the preliminary experiments, I adapted the WTaG codebase to handle multiple models and prompt templates. The preliminary investigations were run with a subset of 3 sessions (1 for each recipe) on one A100 80GB GPU. Experiments were also seeded[1], to ensure reproducability. The prompt template was kept the same, except we additionally asked the models to arrange their final answers into JSON format in order to save on post-processing efforts at evaluation.

Model Type	Models	User Intent	Mistake Existence	Mistake Type
Baselines	Chance	16.67	50.00	33.33
	ChatGPT (LanOnly)	37.45	6.30	40.60
Blind LLM	Vicuna	30.88	**14.08**	12.50
	LLaMA-2 (7B)	25.00	0	**25.00**
	LLaMA-2 (13B)	**33.82**	11.67	18.75
Single-Frame VLM	LLaVA	**33.82**	4.88	12.50

Table 5.1 Preliminary results for User Environment and Understanding (UEU) tasks, except for step detection. The performing open-source models are shown in bold.

[1]The seed used across all experiments was 42.

Tables 5.1 to 5.3 show our partial results for the 7 tasks that constitute WTaG. Step detection is reported separately for each recipe type as in the original paper. The results are ‘partial’ because they were obtained for a subset of the evaluation data, including T10_pinwheel, T11_coffee, and T4_cake. While these partial baselines are not directly comparable to those reported by Bao et al. (2023), they nonetheless provide a gauge for open-source model performances on WTaG.

Table 5.1 shows that the open models were only able to perform better than random chance on user intent prediction. We also observe that the 13B models generally perform better than the 7B LLaMA-2 model, which could be an early indication of scaling laws.

Model Type	Models	Pinwheel	Coffee	Cake
Baselines	Chance	8.33	12.50	8.33
	ChatGPT (LanOnly)	42.09	47.27	38.23
Blind LLM	Vicuna	4.17	3.75	0
	LLaMA-2 (7B)	1.39	2.50	2.83
	LLaMA-2 (13B)	11.11	**16.25**	9.43
Single-Frame VLM	LLaVA	**29.17**	7.50	**15.09**

Table 5.2 Preliminary results for step detection across the 3 recipe types. Best open models are in bold.

Table 5.2 shows the preliminary baselines for step detection. We aim to improve on these in Section 5.2. We observe that all the models perform poorly on all 3 recipe types. Vicuna struggles to detect steps because of the hallucination issue shown in Figure 4.1. Indeed, this is one of the problems that the simplified prompt template aims to address.

LLaVA is the only model that performs significantly above random chance, and performs best on the pinwheel and cake recipes. Moreover, although LLaVA uses Vicuna as its language backbone, its further fine-tuning has prevented it from making the same hallucinations as Vicuna. LLaVA’s relatively strong performance on the step detection task seems to indicate that the model can ground its response in the image it can directly ‘see’. However, we find in Section 5.2 that this observation does not always hold when considering the full WTaG benchmark.

Model Type	Models	When To Talk	Instructor Intent	Instruction Type
Baselines	Chance	50.00	20.00	25.00
	ChatGPT (LanOnly)	50.14	37.89	29.35
Blind LLM	Vicuna	**54.14**	**36.23**	44.44
	LLaMA-2 (7B)	49.40	33.33	**66.67**
	LLaMA-2 (13B)	51.13	17.39	22.22
Single-Frame VLM	LLaVA	51.81	34.78	**66.67**

Table 5.3 Preliminary results for IDM tasks.

As for the Instructor Decision Making (IDM) tasks in Table 5.3, all the models exhibit about chance-level performance when deciding 'when to talk', which is similar to the reported ChatGPT performance. On instructor intent and instruction type prediction, most models perform better than chance, with Vicuna performing best on 'When To Talk' and instructor intent prediction.

In sum, the baseline results highlight the deficiencies of widely used open-source models on the WTaG benchmark. We have shown that these models can be comparable to ChatGPT for most of the WTaG tasks, achieving similar F1 scores to ChatGPT on the small subset of WTaG we consider. However, these preliminary observations also show that the open-source models generally struggle with step detection. The following section presents the results we obtained from our various attempts to address this issue and improve step detection performance.

5.2 Step Detection Improvements

We begin by presenting the results from manipulating the input prompt, then report our results from the explicit instruction-tuning approach with Vicuna. The results in this section have been obtained across the full WTaG benchmark, meaning they are directly comparable to those reported by Bao et al. (2023).

5.2.1 Prompt Design

Simplified Prompt Scheme

Table 5.4 shows step detection performance across the Blind LLMs when using the simplified prompt scheme. With this new scheme, Vicuna and Mixtral strikingly outperform ChatGPT on both the coffee and cake tasks. ChatGPT still performs better on the pinwheel task, which suggests that the open-source models find step detection for the pinwheel recipe most challenging.

Prompt Style	Blind LLM	Model Size	Pinwheel	Coffee	Cake
Default	ChatGPT (LanOnly)	-	42.09	47.27	38.23
	Vicuna	13B	5.26	1.99	4.87
Simplified	Vicuna	13B	30.62	**48.63**	42.54
	LLaMA-2	7B	29.54	17.56	24.44
	LLaMA-2	13B	26.59	42.15	31.56
	Mixtral	47B	**38.18**	41.84	**45.57**

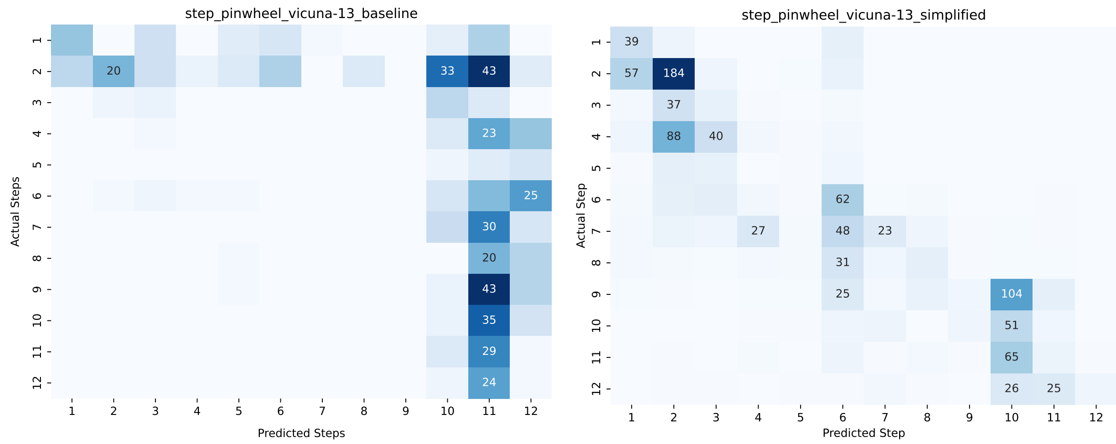
Table 5.4 F1 scores for the 'Blind' open-source LLMs.

Again, we observe the benefits of large model size. Mixtral performs best on average, while the 13B models generally outperform LLaMA-2 7B. This suggests that scale is an important factor in ensuring that a model has the minimum level of reasoning required to perform situated task guidance. A model that is too small is unlikely to be powerful enough to perform a complex task like step detection.

The drastic performance improvement from Vicuna demonstrates that the complexity of the default prompt scheme was previously impairing Vicuna's ability to perform the task. Moreover, the fact that Vicuna outperforms LLaMA-2 indicates the importance of the IT dataset used for fine-tuning. In this case, it seems that fine-tuning on ShareGPT data is an effective strategy to align the base LLaMA model more closely with the WTaG task instructions.

To visualise the differences in the Vicuna responses when using the default vs. simplified templates, we plot confusion matrices in Figure 5.1. The default template causes Vicuna to predict step 11 for most QPs. On the other hand, the simplified prompt brings out a more reasonable distribution of step predictions. Examining the diagonal entries of Figure 5.1b, we see that while steps 2, 6, and 10 garner the most correct predictions, they are also heavily confused with later steps. This indicates that these steps are strongly signalled by the dialogue history, since this is the only context the Blind LLMs have access to. For reference, Step 2 is 'Scoop and spread nut butter onto the tortilla', Step 6 is 'Roll the tortilla from one end to the other into a log shape, about 1.5 inches thick', and Step 10 is 'Slice the tortilla roll with the floss'.

It could be argued that the above steps are the most semantically distinct in the recipe. We term these 'landmark' steps. The other steps, such as cleaning the knife or sliding the floss under the tortilla (see Appendix A), are termed 'transition' steps. It would seem that transition steps are either semantically too close to a landmark or are too weakly signalled to be detected by the foundation models. A similar trend can be observed for the other recipes, though we include them in Appendix B for brevity.



(a) (b)

Fig. 5.1 Confusion matrices for Vicuna step predictions across pinwheel sessions. 5.1b shows the simplified prompt setting.

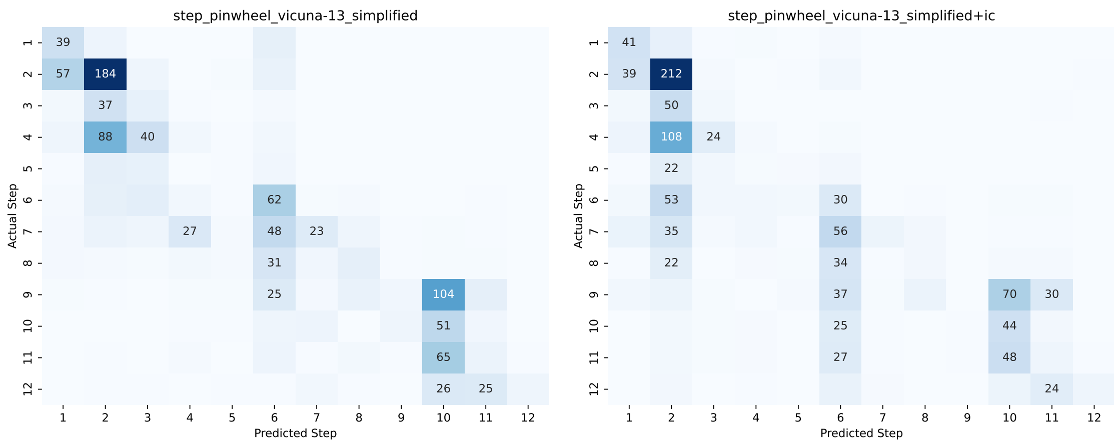
Single-Frame Captions and Frame Selection

Table 5.4 demonstrates that the Blind LLMs can achieve step detection performance that is superior to ChatGPT, provided that the input prompt is simple. This suggests that the dialogue history alone can provide strong cues about the current step. However, this does not make use of the visual information available in WTaG. This section reports our attempts to incorporate the vision modality into the step detection process. **It can be assumed that experiments run from this point on use the simplified prompting scheme**, as this clearly provides better results than the previous default template.

Our first approach was to replicate the 'Scene Description' setup described in the original WTaG paper (Bao et al., 2023). In our case, the LLM backbone is Vicuna and the captioner is LLaVA. The results are shown in Table 5.5. The main observation is that Vicuna+LLaVA achieves lower F1 scores than Vicuna alone. This suggests that image captions are not helping but rather hurting performance. We also note that apart from the pinwheel task, ChatGPT+BLIP-2 performs about as well as Vicuna, further casting doubt on the utility of captions.

Model Type	Model	Pinwheel	Coffee	Cake
Blind LLM	Vicuna	30.62	48.63	42.54
Socratic LLM w/ Caption	ChatGPT+BLIP-2	37.99	48.64	41.75
	Vicuna+LLaVA	27.16	45.77	42.49

Table 5.5 Socratic LLM step detection scores across recipe types.



(a) (b)

Fig. 5.2 The ‘Step reinforcement’ effect of image captions on model predictions across all pinwheel sessions, shown by 5.2b.

We plot Figure 5.2 to better understand the effect of image captions on model predictions. Interestingly, we see that captions bias Vicuna towards the landmark steps of the recipe (steps 2, 6, and 10 for the pinwheel recipe). In other words, captions are often providing redundant information that reinforces the already strong signal provided by the landmark steps. This ‘step reinforcement’ effect causes the model to predict landmark steps more often, as seen in Figure 5.2b.

More optimistically, Figure 5.3 shows an image whose caption helps Vicuna predict the correct step. Without the caption, the model predicts step 10 (the GT is step 12). However, LLaVA provides the caption: ‘*[...] The person is also holding a spoon, which might be used for serving the cake or for eating it. The scene suggests that the person is preparing to enjoy a delicious dessert*’. This caption successfully steers Vicuna to predict step 12.

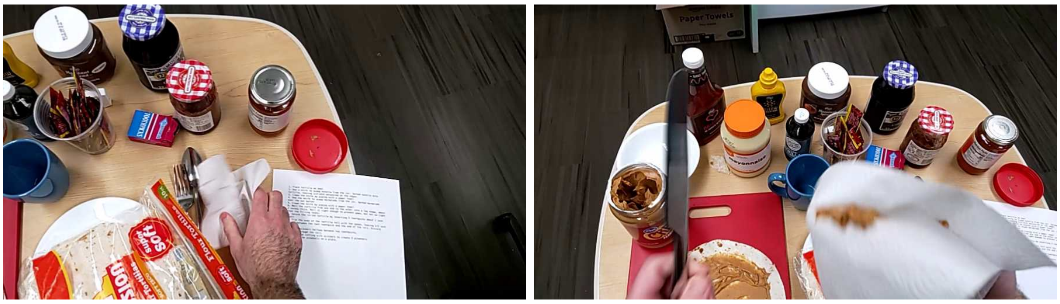
Fig. 5.3 Example frame from session T4_cake that was well-captioned by LLaVA.

In Section 4.3.1 it was observed that with the original WTaG setup, the frames we pass to the captioner can be blurry or irrelevant to the current step. At best, this noisy information could simply be ignored by Vicuna. At worst, it could cause the LLM to incorrectly predict a step that it would have otherwise predicted correctly *without* the caption. In an attempt to select more relevant frames for the captioner model, we implement a light-weight frame selection module. The module is light-weight so that it can efficiently select process frames *on-the-fly*, as the WTaG setup requires. Other methods may be more powerful and select more optimal frames, but may be impractical to use in a real-time setting.

(a) (b)

Fig. 5.4 Example of a frame selection improvement in session T10_pinwheel at frame 13042. 5.4b shows the improved frame.

Figures 5.4 and 5.5 show examples where the frame selector has improved the image passed to the model. Figure 5.4a shows an unclear image focusing on an irrelevant portion of the scene, and Figure 5.5a shows a frame where a key object (the knife) is obscured during the step ‘Clean the knife with a paper towel’. Figures 5.4b and 5.5b address both these problems. Of course, this will not be the case for every image selected by the frame-selector, but we believe that it is at least a more systematic method of extracting relevant frames from the video compared to the previous approach.



(a) (b)

Fig. 5.5 Frame selection improvement session T10_pinwheel at frame 7293. 5.5b shows the improved frame.

Table 5.6 reports F1 scores for the **Socratic LLM w/ Caption** setup, with and without frame selection. We observe that frame selection marginally compensates for the performance drop from the Blind LLM on the pinwheel and coffee recipes (recall Table 5.5), but hurts performance further for the cake recipe. These results suggest that the frame selection module may not be effective enough yet to make a significant difference to the outcomes of the predicted steps. However, it should be noted that frame selection is only one part of the decision pipeline for a Socratic LLM. There is always the inherent risk that the captioner will fail to extract key information from the image, or hallucinate and produce a caption that does not faithfully describe the scene. These challenges make adopting Socratic LLM systems for situated task guidance difficult, as every stage in the decision pipeline can be a point of failure.

Setting	**Socratic LLM w/ Caption**	**Pinwheel**	**Coffee**	**Cake**
Without FS	ChatGPT+BLIP-2	37.99	48.64	41.75
	Vicuna+LLaVA	27.16	45.77	**42.49**
With FS	Vicuna+LLaVA	**27.45**	**46.45**	39.27

Table 5.6 Step detection F1 scores for Socratic LLMs with and without the frame selection (FS) module.

Direct Multimodal Processing

One way to alleviate the problems associated with the **Socratic LLM w/ Caption** model is to use the captioner as the reasoning agent itself. This is possible because LLaVA has been explicitly fine-tuned on multimodal IT data, whereas BLIP-2 has not. Thus, considering this **Single-Frame VLM** setting can be instructive in understanding whether it is more beneficial to process visual information directly or as a text caption for situated task guidance.

Prompt Style	Single-Frame VLM	Pinwheel	Coffee	Cake
Default	LLaVA	16.14	15.26	14.93
Simplified	LLaVA	23.85	45.14	42.49
	LLaVA-NeXT	**27.09**	**49.69**	**42.72**
No History	LLaVA	16.71	30.39	15.76
	LLaVA-NeXT	19.16	29.95	27.17

Table 5.7 F1 scores for the Single-Frame VLMs used in this project, LLaVA and LLaVA-NeXT.

Table 5.7 shows some interesting results. First, we notice that LLaVA-NeXT outperforms both the previous and Simplified LLaVA baselines across all 3 recipes. Strikingly, we also observe that for the coffee recipe, LLaVA-NeXT produces the highest score out of all other models, including ChatGPT. This suggests that high-quality IT data is crucial for building a good general-purpose multimodal system that is robust to hallucination.

Finally, we equally ablate the dialogue history to determine its impact on performance. As expected, doing so leads to substantial drops in performance for both models (although LLaVA-NeXT makes better use of the visual information). This demonstrates that the models still heavily rely on dialogue cues to infer the current step. However, the fact that LLaVA-NeXT outperforms its 'Blind' Vicuna counterpart is promising, as it is an indication that the model is to an extent capable of exploiting the immediate scene to make more grounded decisions about the current step.

5.2.2 Connecting the Dots

In the previous sections, we conducted a suite of experiments to investigate methods of improving step detection performance on WTaG. We compile the key results in Table 5.8.

Model Type	Model	Model Size	Pinwheel	Coffee	Cake
Blind LLM	ChatGPT	-	42.09	47.27	38.23
	Vicuna	13B	30.62	48.63	42.54
	Mixtral	47B	**38.18**	41.84	**45.57**
	LLaMA-2	7B	29.54	17.56	24.44
	LLaMA-2	13B	26.59	42.15	31.56
Socratic LLM w/ Caption	ChatGPT	-	37.99	48.64	41.75
	Vicuna+LLaVA (w/o FS)	13B	27.16	45.77	42.49
	Vicuna+LLaVA (w/ FS)	13B	27.45	46.45	39.27
Single-Frame VLM	LLaVA	13B	23.85	45.14	42.49
	LLaVA-NeXT	13B	27.09	**49.69**	42.72

Table 5.8 Summary of key results from the various prompt design experiments we conducted. Best scores among the open-source models (regardless of type) are shown in bold.

From these results we can make a number of observations. The most obvious is that using the simplified prompting scheme leads to a substantial gain in performance across the board. This shows the importance of a clear prompt when issuing instructions to a model, and demonstrates that open-source models can produce comparable performance to ChatGPT for the step detection task. We also note that scaling laws generally apply, as we see performance tend to increase as models get bigger.

The performance gap between Vicuna and LLaMA-2 13B across all recipes again highlights the benefits of instruction-tuning on ShareGPT data. While there are issues with model imitation as mentioned previously (Gudibande et al., 2023), imitating ChatGPT (see Section 2.4.1) seems to lead to tangible performance gains in our case.

Regarding leveraging the visual information to detect the current step, we explored a) augmenting an LLM with an image caption and b) using instruction-tuned VLMs. The **Socratic LLM w/ Caption** results show that incorporating captions hurts performance. Upon further analysis we deduce that this is because the captions bias the model towards certain landmark signals and cause the LLM to 'double down' on its decision.

Incorporating the vision modality also introduces the problem of frame selection. We adapted a simple key-frame detection algorithm as a means to extract a more relevant set of frames for the foundation models. This led to only marginal improvements for the Socratic LLM on the pinwheel and coffee tasks. This suggests that more work needs to be done to further optimise frame selection and improve multimodal reasoning across foundation models.

Finally, for the **Single-Frame VLM** experiments we observe that LLaVA-NeXT exhibits higher performance than LLaVA, and for the coffee and cake tasks even outperforms Vicuna. This demonstrates the benefits of fine-tuning on high-quality multimodal IT data, as this has led to LLaVA-NeXT being more robust to errors compared to its predecessor. The model proves to be the most adept at 'connecting the dots' between the visual information and the dialogue history, and thus demonstrates that direct multimodal processing is a promising research direction to improve performance on the step detection task.

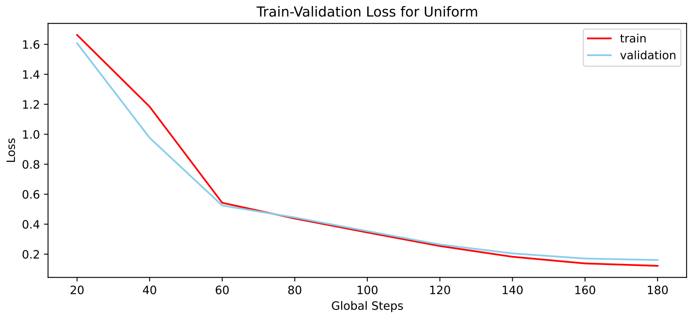
5.3 Instruction Tuning Vicuna

This section presents the results from the experiments described in Section 4.3.2. Since the $\mathcal{D}_{\text{mix}}$ training set is formed from a subset of 14 WTaG videos, we re-evaluate the Blind Vicuna and Mixtral baselines on the remaining 34 videos, so that results are comparable to the fine-tuned model. Note that during training, Vicuna was quantised to 4-bit precision for efficiency.

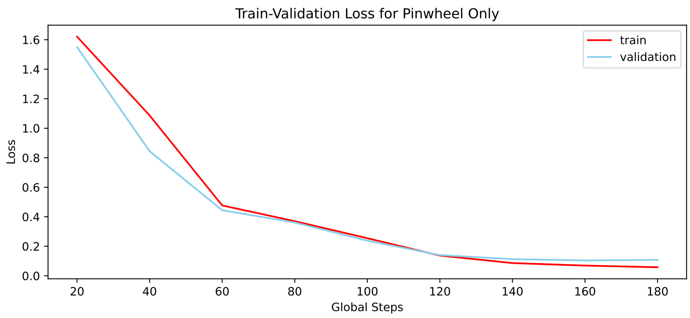
For the first experiment, $\mathcal{D}_{\text{mix}}$ contains uniform proportions of the $\mathcal{D}_{\text{pin}}$, $\mathcal{D}_{\text{cof}}$, and $\mathcal{D}_{\text{cak}}$ subsets, and consists of 1439 samples. Vicuna is then trained on $\mathcal{D}_{\text{mix}}$ for 10 epochs with a scheduled learning rate that warms up to 0.0003 in 100 global steps and decays linearly afterwards. Regarding the LoRA hyperparameters r and α, these are set to 8 and 16 respectively to follow standard practice. Dropout regularisation with a probability of 5% is also applied to the LoRA matrices to mitigate the risk of overfitting. 20% of the data in $\mathcal{D}_{\text{mix}}$ was held out for validation. The train-validation loss curves are shown in Figure 5.6a. Training seems to converge at 180 global steps, with minimal signs of overfitting.

As for the 'Pinwheel Only' setting, the model is only trained on the pinwheel data from the training set, so the dataset is smaller with only 534 samples, 20% of which were again held out for validation. The batch size was adjusted 32, the warmup steps to 120, and the epochs to 15 to account for the smaller dataset size. Dropout regularisation was also applied. From Figure 5.6b we can see that training again converages between steps 160-180, and that the losses are generally lower than those in Figure 5.6a.

Table 5.9 shows the final results from these fine-tuning experiments. We see similar trends from before, namely that Vicuna outperforms Mixtral on the coffee task but exhibits lower F1 step detection scores on the other tasks. Results for the Uniform setting indicate



(a)



(b)

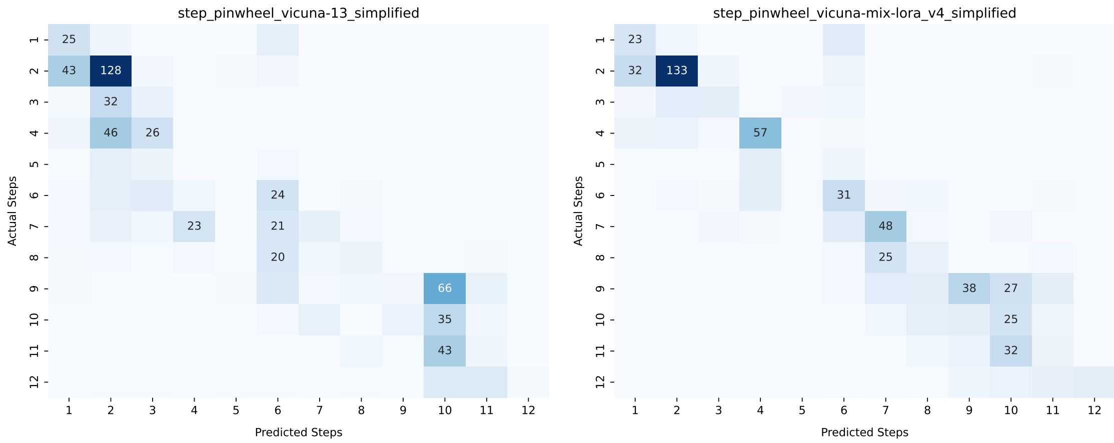
Fig. 5.6 Loss curves for the 'Uniform' and 'Pinwheel-Only' experiments.

that the model is indeed answering prompts in a similar way to Mixtral, as performance on pinwheel sessions improves, while performance on coffee sessions declines, presumably due to the natural weakness of Mixtral on this task. We also notice that the cake F1 score is higher than those of either baseline model. This is not surprising, as the baseline performances are zero-shot and are thus not explicitly aligned to this specific downstream task yet.

Even more intriguing however are the results for the Pinwheel Only setting. Here we observe that despite being exposed to data exclusively from pinwheel sessions (and as such a smaller dataset), F1 scores improve on the baselines across all tasks. This suggests that for step detection, constructing a smaller dataset with homogeneous samples is more beneficial than constructing a larger one with a mix of samples from the different tasks. It would seem that the diverse dataset provides Vicuna with a more noisy learning signal compared to the pinwheel-only dataset, which contains data points which are more consistent with each other (e.g. they have the same underlying structure, possibly similar interactions between Instructor and User, etc.). Of course, Vicuna still improves under the Uniform setting. However, this improvement is just not as pronounced as in the Pinwheel-Only setting. The substantial gains that come out of training on the pinwheel-only samples could indicate that Vicuna is able to learn more general associations between the dialogue and the current step. In other words, since these tasks are predictable, sequential procedures Vicuna can learn to make better use of dialogue cues that often appear during training.

Model Type	**Setting**	**Model**	**Pinwheel**	**Coffee**	**Cake**
Blind LLM	Baseline	Vicuna	29.17	53.43	43.90
		Mixtral	36.57	44.91	47.77
Fine-Tuned LLM	Uniform	Vicuna+LoRA	35.07	50.00	48.53
	Pinwheel Only	Vicuna+LoRA	**47.11**	**54.26**	**51.76**

Table 5.9 Step detection performance from fine-tuning Vicuna vs. the Blind baseline.



(a) (b)

Fig. 5.7 Effect of fine-tuning on the confusion matrix for the pinwheel sessions. 5.7b shows the confusion matrix for Vicuna+LoRA.

Figure 5.7 shows that fine-tuned Vicuna is able to recognise more ‘transition steps’ in the pinwheel task, namely Steps 4, 7, and 9. These were often overlooked by vanilla Vicuna because they resembled a landmark step too closely (e.g. spreading peanut butter, rolling a tortilla). However, it becomes clear that with instruction tuning, Vicuna learns to focus on dialogue cues that are more relevant to the current step. These cues could be something obvious, such as the mention of a key noun (for example ‘toothpick’ for Step 7) or something more subtle on the dialogue level.

Finally, Figure 5.8 shows a qualitative example where plain Vicuna predicts the wrong step (2 instead of 4) while Vicuna+LoRA guesses correctly. In this case, before fine-tuning Vicuna acknowledges the necessary piece of information to infer the current step, but does not make the connection and even hallucinates, since jelly is not needed for Step 3 (cleaning the knife). However, after fine-tuning we see that Vicuna is able to focus on the correct cue and predict the correct step.

T10_pinwheel, Frame 7880

```
User: this is nut butter right
Instructor: yes
User: the recipe says to use marmalade
The user just said: 'this marmalade or this marmalade'

Vicuna: Based on the situation described, it seems
that the user is currently on Step 2 of the recipe. [...]
The user mentioned that they have marmalade instead of
jelly, which is the ingredient called for in Step 3 of the
recipe.

Vicuna+LoRA:Based on the situation described, I believe
the user is currently on Step 4 of the recipe, which is to
spread jelly over the nut butter. This is because
the user mentioned that the recipe says to use
marmalade instead of jelly [...]
```

Fig. 5.8 Example of Vicuna+LoRA correctly detecting a step that Vicuna previously failed to detect (GT is Step 4).

In sum, instruction tuning proves to be a promising direction for improving the base capabilities of Vicuna to pick out salient cues in the dialogue history. Interestingly, we find that a dataset of only about 500 synthetic samples is enough for the model to significantly improve step detection performance across all tasks, provided that the dataset consists of similar samples. We now consider the instruction generation aspect of WTaG, which is what the benchmark was built for in the first place.

5.4 Instruction Generation

This thesis has predominantly focused on improving step detection performance on WTaG. However, although this task is interesting in its own right, it is auxiliary to the core purpose of WTaG, which is to assess instruction generation. In this section, we include some qualitative examples of instructions generated by Vicuna to briefly assess its instruction generation capabilities. To generate the instructions, we replace the final instruction in the simplified prompt template with '*Based on the situation described, what would you instruct the user? Please be brief and natural*'.

Figures 5.9 and 5.10 demonstrate that Vicuna can generate instructions of different types, namely confirmations/encouragement ('*Great!*') and questions ('*Can you check the size of the remaining batter?*').

T10_pinwheel, Frame 3917

```
User: OK
User: it's on the cutting board

Vicuna: Great! Now, make sure to scoop and
spread the nut butter onto the tortilla evenly.
```

Fig. 5.9 Example of Vicuna issuing a confirmation and encouraging the User.

T4_cake, Frame 13165

```
Instructor: can you look down more
User: oh yeah sorry
Instructor: uh how large
User: oh that was one third

Vicuna: Great, leave the liner on while it cools. Can you
check the size of the remaining batter?
```

Fig. 5.10 Example of Vicuna issuing asking a question to the User.

Moreover, Figure 5.11 demonstrates that Vicuna can also be flexible about the current situation. In this scenario, the User has placed the tortilla on the 'wrong' object. However, Vicuna acknowledges that both a plate and a bowl are valid objects on which to place a tortilla, and encourages the User to use the cutting board instead. In real-world situations, many unexpected state changes may occur during the process of completing a task, but as long as the final state is correct, the task can be considered successful. This example shows that Vicuna can be robust to deviations from the original recipe, which is key for smooth task guidance.

T10_pinwheel, Frame 2279

```
User: we'll start the recipe now
User: the recipe says to place a tortilla on the bowl
but i have a plate
User: is that okay?

Vicuna: It's okay to use a plate instead of a bowl, but
it's better to use a cutting board for stability.
```

Fig. 5.11 Example of Vicuna being robust to deviations from the original recipe.

T11_coffee, Frame 604

```
User: 15 ounces of cold water, transfer to kettle
User: this water?

Vicuna: Actually, you should measure 12 ounces of cold
water and transfer it to the kettle.
```

Fig. 5.12 Example of Vicuna correcting a mistake.

Finally, in Figure 5.12 we observe that Vicuna can *implicitly* recognise a mistake and issue an appropriate instruction. This suggests that basing evaluations on the auxiliary task may in fact misrepresent the model's true task guidance capabilities, since in this case the low mistake detection F1 scores would have led us to believe that the model is not able to detect mistakes, even though this may not necessarily be true. Nevertheless, the auxiliary tasks are still interesting in their own right and should be investigated further, as we have done here for step detection.

Thus, from Figures 5.9 to 5.12 we can see that Vicuna can generate reasonable instructions. The instructions are not only relevant to the current situation, but are varied as well. Importantly, Vicuna demonstrates its ability to correct mistakes, calling into question whether performance on the auxiliary WTaG tasks necessarily correlates with a foundation model's ability to generate situated guidance. Nevertheless, the auxiliary tasks are still interesting in their own right and should be investigated further, as we have done here for step detection.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This thesis has tackled the problem of situated task guidance from a number of different angles. The core aim was to leverage large open-source foundation models that could be 'instructed to instruct' another human User. Having conducted various experiments and analyses, we are now in a position to address the research questions posed in Section 1.2:

1. How do instruction-tuned open-source models perform on the WTaG benchmark under a zero-shot setting?

2. How do foundation models synthesise various pieces of information?

3. To what extent does further instruction tuning improve performance on the WTaG benchmark?

4. How relevant and helpful are the instructions that open foundation models generate for unseen real-world tasks?

Regarding 1), our experiments demonstrate that current open-source models can yield comparable zero-shot performance to ChatGPT on the WTaG tasks, and when given suitable prompts can even outperform it in step detection. However, this does not mean that the models are viable for this use case yet, as performances remain generally low.

For 2), we find that Blind LLMs are adept at extracting cues from the dialogue history, as their zero-shot performances are relatively strong without any visual context. However, leveraging this visual context is challenging, because we need to consider the relevance of a frame to the current step. For the Socratic LLM setup where we pass a caption to a core LLM (Vicuna in our case), we also have to consider the truthfulness of this caption, because it could lead the LLM astray in its reasoning. Indeed, we empirically observe a ‘step reinforcement’ effect that biases the LLM towards ‘landmark’ steps that have stronger intrinsic signals in the recipe. In other words, we observe that captions often provide redundant or false information to the core LLM instead of complementing the dialogue history. As such, we explore the use of Single-Frame VLMs, as these eliminate the error-prone captioning step. We find that although LLaVA can directly ‘see’ the frame, it does not fare better than the Socratic LLM. However, the latest version of LLaVA, LLaVA-NeXT, not only outperforms the Socratic LLM, but also outperforms the Blind LLMs on the coffee task. This demonstrates the importance of curating high-quality IT training data, and the potential of direct multimodal models to solve difficult vision-language problems such as real-time step detection.

In a similar way, our additional instruction-tuning of Vicuna demonstrates that the model can be ‘taught’ to focus on important aspects of the input that may be relevant to the task. We show that only about 500 data points from 5 pinwheel sessions are sufficient for the model to reasonably adapt to the downstream step detection task, since we observe substantial performance gains across all session types under this experimental setting. A more diverse training set also leads to performance gains, but not to the same extent.

Finally, our qualitative observations in 5.4 demonstrate that Vicuna (and other models like it) can indeed generate instructions that are varied, helpful, and situated. For instance, we find that the model can be flexible to User deviations from the give task, give encouragement, and correct a mistake. These are all traits of a capable Instructor, yet Vicuna could still be deemed a poor task guidance system if it fails to perform well on one of the auxiliary tasks in WTaG. Thus, our observations warrant further investigation into how exactly the performance of a model on the WTaG tasks correlates with its final generated instructions, as it may be the case that a model is a good instructor, but a poor step/mistake detector.

6.2 Future Work

There are a number of possible directions for future work. First, it would be interesting to convert WTaG to a format that is suitable for success detection (Du et al., 2023). Under this approach, we would simply ask a VLM whether or not the User has successfully completed a step in the recipe (similar to ‘Progress Check’ in (Padmakumar et al., 2022)). We believe that this approach could be helpful for detecting both steps and mistakes, since a mistake could technically be thought of as a step that was executed unsuccessfully.

Another direction would be to fine-tune either Single-Frame or Multi-Frame VLMs on WTaG. In Section 5.3, we fine-tuned Vicuna to better exploit dialogue cues and boost performance across all recipe types. The natural extension to this is to investigate fine-tuning VLMs. In early experiments we experimented with using an open-source video VLM (LLaVA-NeXT-Video-DPO (7B)) as the core reasoning agent, but concluded that this model did not have sufficient zero-shot reasoning to be applied to step detection. If such a model were able to better leverage the video information in WTaG, we could potentially see significant boosts in the current zero-shot performances we observe. As for Single-Frame VLMs, one interesting option would be to explore Localised Symbolic KD (Park et al., 2024), which provides more focused and granular captions about an image.

References

Achiam, J., Adler, S., Agarwal, S., et al. (2023). GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*.

Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., Reid, I., Gould, S., and Van Den Hengel, A. (2018). Vision-and-Language Navigation: Interpreting Visually-Grounded Navigation Instructions in Real Environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer Normalization.

Bao, Y., Yu, K., Zhang, Y., Storks, S., Bar-Yossef, I., de la Iglesia, A., Su, M., Zheng, X., and Chai, J. (2023). Can Foundation Models Watch, Talk and Guide You Step by Step to Make a Cake? In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12325–12341.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language Models are Few-shot Learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

Dao, T., Fu, D., Ermon, S., Rudra, A., and Ré, C. (2022). Flashattention: Fast and Memory-Efficient Exact Attention with IO-Awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929.

Du, Y., Konyushkova, K., Denil, M., Raju, A., Landon, J., Hill, F., de Freitas, N., and Cabi, S. (2023). Vision-Language Models as Success Detectors. *arXiv preprint arXiv:2303.07280*.

Fried, D., Hu, R., Cirik, V., Rohrbach, A., Andreas, J., Morency, L.-P., Berg-Kirkpatrick, T., Saenko, K., Klein, D., and Darrell, T. (2018). Speaker-Follower Models for Vision-and-Language Navigation. *Advances in Neural Information Processing Systems*, 31.

Girdhar, R., El-Nouby, A., Liu, Z., Singh, M., Alwala, K. V., Joulin, A., and Misra, I. (2023). ImageBind: One Embedding Space to Bind Them All. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15180–15190.

Grauman, K., Westbury, A., Byrne, E., et al. (2022). Ego4D: Around the World in 3,000 Hours of Egocentric Video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18995–19012.

Gudibande, A., Wallace, E., Snell, C., Geng, X., Liu, H., Abbeel, P., Levine, S., and Song, D. (2023). The False Promise of Imitating Proprietary LLMs. *arXiv preprint arXiv:2305.15717*.

Hosseini-Asl, E., McCann, B., Wu, C.-S., et al. (2020). A Simple Language Model for Task-Oriented Dialogue. *Advances in Neural Information Processing Systems*, 33:20179–20191.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2021). LoRA: Low-Rank Adaptation of Large Language Models. *arXiv preprint arXiv:2106.09685*.

Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., Casas, D. d. l., Hanna, E. B., Bressand, F., et al. (2024). Mixtral of Experts. *arXiv preprint arXiv:2401.04088*.

Kim, Y. and Rush, A. M. (2016). Sequence-Level Knowledge Distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327.

Koh, J. Y., Fried, D., and Salakhutdinov, R. R. (2024). Generating Images with Multimodal Language Models. *Advances in Neural Information Processing Systems*, 36.

Li, C., Deng, Y., Hu, H., Kan, M.-Y., and Li, H. (2024). Incorporating External Knowledge and Goal Guidance for LLM-based Conversational Recommender Systems. *arXiv preprint arXiv:2405.01868*.

Li, J., Li, D., Savarese, S., and Hoi, S. (2023). BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. In *International conference on machine learning*, pages 19730–19742. PMLR.

Liu, H., Li, C., Li, Y., and Lee, Y. J. (2023). Improved Baselines with Visual Instruction Tuning.

Liu, H., Li, C., Li, Y., Li, B., Zhang, Y., Shen, S., and Lee, Y. J. (2024a). LLaVA-NeXT: Improved reasoning, OCR, and world knowledge.

Liu, H., Li, C., Wu, Q., and Lee, Y. J. (2024b). Visual Instruction Tuning. *Advances in Neural Information Processing Systems*, 36.

Lu, Y. and Mayol-Cuevas, W. (2019). HIGS: Hand Interaction Guidance System. In *2019 IEEE international symposium on mixed and augmented reality adjunct (ISMAR-Adjunct)*, pages 376–381. IEEE.

Majumdar, A., Ajay, A., Zhang, X., Putta, P., Yenamandra, S., Henaff, M., Silwal, S., Mcvay, P., Maksymets, O., Arnaud, S., et al. (2024). OpenEQA: Embodied Question Answering in the Era of Foundation Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16488–16498.

Ouyang, L., Wu, J., Jiang, X., et al. (2022). Training Language Models to Follow Instructions with Human Feedback. *Advances in neural information processing systems*, 35:27730–27744.

Padmakumar, A., Thomason, J., Shrivastava, A., Lange, P., Narayan-Chen, A., Gella, S., Piramuthu, R., Tur, G., and Hakkani-Tur, D. (2022). TEACh: Task-driven Embodied Agents that Chat. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 2017–2025.

Park, J. S., Hessel, J., Chandu, K., et al. (2024). Localized Symbolic Knowledge Distillation for Visual Commonsense Models. *Advances in Neural Information Processing Systems*, 36.

Pech-Pacheco, J. L., Cristóbal, G., Chamorro-Martinez, J., and Fernández-Valdivia, J. (2000). Diatom Autofocusing in Brightfield Microscopy: a Comparative Study. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 3, pages 314–317. IEEE.

Radford, A., Kim, J. W., Hallacy, C., et al. (2021). Learning Transferable Visual Models From Natural Language Supervision. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language Models are Unsupervised Multitask Learners. *OpenAI blog*, 1(8):9.

Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., Zettlemoyer, L., and Fox, D. (2020). Alfred: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749.

Su, Y., Lan, T., Li, H., Xu, J., Wang, Y., and Cai, D. (2023). PandaGPT: One model to instruction-follow them all. In Hazarika, D., Tang, X. R., and Jin, D., editors, *Proceedings of the 1st Workshop on Taming Large Language Models: Controllability in the era of Interactive Assistants!*, pages 11–23, Prague, Czech Republic. Association for Computational Linguistics.

Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B. (2023). Stanford Alpaca: An Instruction-following LLaMA model. https://github.com/tatsu-lab/stanford_alpaca.

Touvron, H., Lavril, T., Izacard, G., et al. (2023). LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971*.

Turner, R. E. (2023). An Introduction to Transformers. *arXiv preprint arXiv:2304.10557*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is All You Need. *Advances in Neural Information Processing Systems*, 30.

Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. (2022). Self-Instruct: Aligning Language Models with Self-generated Instructions. *arXiv preprint arXiv:2212.10560*.

Wu, S., Fei, H., Qu, L., Ji, W., and Chua, T.-S. (2023). NExT-GPT: Any-to-Any Multimodal LLM. *arXiv preprint arXiv:2309.05519*.

Xing, L., Tran, Q., Caba, F., Dernoncourt, F., Yoon, S., Wang, Z., Bui, T., and Carenini, G. (2024). Multi-Modal Video Topic Segmentation with Dual-Contrastive Domain Adaptation. In *International Conference on Multimedia Modeling*, pages 410–424. Springer.

Xu, H., Ghosh, G., Huang, P.-Y., Okhonko, D., Aghajanyan, A., Metze, F., Zettlemoyer, L., and Feichtenhofer, C. (2021). VideoCLIP: Contrastive Pre-training for Zero-shot Video-Text Understanding. *arXiv preprint arXiv:2109.14084*.

Zheng, L., Chiang, W.-L., Sheng, Y., et al. (2023). Judging LLM-as-a-judge with MT-Bench and Chatbot Arena.

Zhukov, D., Alayrac, J.-B., Cinbis, R. G., Fouhey, D., Laptev, I., and Sivic, J. (2019). Cross-Task Weakly Supervised Learning from Instructional Videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3537–3545.

Appendix A

Recipes and Prompts

Simplified Pinwheel Recipe

```
Ingredients:
1 8-inch flour tortilla
Jar of nut butter
Jar of jelly

Recipe:
Step 1: Place a tortilla on the cutting board.
Step 2: Scoop and spread nut butter onto the tortilla.
Step 3: Clean the knife with a paper towel.
Step 4: Scoop and spread jelly over the nut butter.
Step 5: Clean the knife with a paper towel.
Step 6: Roll the tortilla from one end to the other into a log
shape, about 1.5 inches thick.
Step 7: Secure the rolled tortilla by inserting 5 toothpicks
about 1 inch apart.
Step 8: Trim the ends of the tortilla roll.
Step 9: Slide floss under the tortilla.
Step 10: Slice the tortilla roll with the floss.
Step 11: Continue slicing with floss to create 5 pinwheels.
Step 12: Place the pinwheels on a plate.
```

Simplified Coffee Recipe

```
Ingredients:
12 oz water
25 grams whole coffee beans

Recipe:
Step 1: Measure 12 ounces of cold water and transfer to a kettle.
Step 2: Assemble the filter cone.
Step 3: Place the filter cone in the dripper.
Step 4: Grind the coffee beans.
Step 5: Check that the water has boiled.
Step 6: Pour a small amount of water to wet the coffee grounds.
Step 7: Fill the paper filter with water.
Step 8: Drain the coffee into the mug.
```

Simplified Cake Recipe

```
Ingredients
2 tablespoons all-purpose flour
1.5 tablespoons granulated sugar
0.25 teaspoon baking powder
A pinch of salt
2 teaspoons canola or vegetable oil
2 tablespoons water
0.25 teaspoon vanilla extract
Container of chocolate frosting

Recipe:
Step 1: Place the cake liner in the mug.
Step 2: Add flour, sugar, baking soda, and salt to the mixing bowl.
Step 3: Whisk the ingredients.
Step 4: Add oil, water, and vanilla to the mixing bowl.
Step 5: Whisk the ingredients with no lumps remaining.
Step 6: Pour batter into the mug.
Step 7: Microwave the batter in the mug.
Step 8: Check that the cake is done with a toothpick.
Step 9: Invert the mug to release the cake on a plate.
Step 10: Scoop 4 spoonfuls of chocolate frosting into a bag.
Step 11: Cut one corner of the frosting bag to create a small opening.
Step 12: Apply the frosting around the base of the cake.
```

Default Prompt Example

```
You are a helpful instructor tasked with helping a user
make a pinwheel. The recipe is:
Ingredients:
1 8-inch flour tortilla
Jar of nut butter or allergy-friendly alternative (such as
sunbutter, soy butter, or seed butter) Jar of jelly, jam,
or fruit preserves

Tools and Utensils:
cutting board
butter knife
paper towel
toothpicks
~12-inch strand of dental floss plate

Steps:
1. Place tortilla on cutting board.
2. Use a butter knife to scoop nut butter from the jar.
Spread nut butter onto tortilla, leaving 1/2-
inch uncovered at the edges.
3. Clean the knife by wiping with a paper towel.
[...]
12. Place the pinwheels on a plate.


Chat History:
Time: 50.6, User: we'll start the recipe now
Time: 61.1, User: the recipe says to place a tortilla on the
bowl but i have a plate
Time: 66.5, User: is that okay?
Time: 68.8, You: if you need extra stuff
Time: 73.7, User: thank you
Time: 79.3, You: but you should be placing the tortilla on a
cutting board
It has now been 89.4 seconds since the start of the recipe.
Answer the following questions:
1. Based on the provided chat history, which step of the
recipe is the user at? Give your answer as an integer.
2. What is the user's dialogue intention? Choose between:
Question, Answer, Confirmation, Hesitation,
```

```
Self Description, and Other
3. Should you say anything? Yes or no?
3.1. If yes, choose your dialogue intention between:
Instruction, Confirmation, Question, Answer, or Other.
3.2. If your dialogue intention is Instruction, is it about:
current step, next step, details, or mistake correction?
4. Did the user make a mistake? Yes or no?
4.1 If yes, choose between: wrong object, wrong state,
wrong action.

Structure your answer in the following JSON format:
{"1": "xxx", "2": "xxx", "3": "xxx", "3.1": "xxx",
"3.2": "xxx", "4": "xxx", "4.1": "xxx"}

where "xxx" is your answer for each question. If an answer is
not applicable, answer with "N/A" in the JSON instead.
Explain your reasoning for each answer step by step.
```

Simple Prompt Example

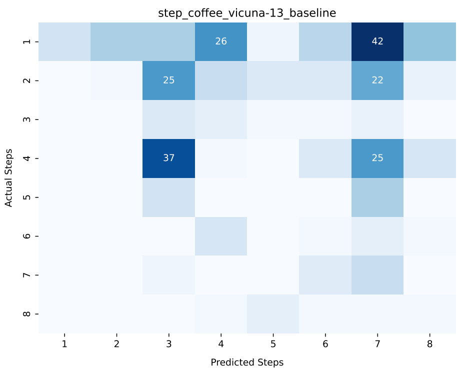
In the example, below, the red text indicates an optional image caption and the blue text indicates an optional video caption. **Not that only one of these captions are used at a time**.

```
An instructor is helping a user make a pinwheel.
The ingredients required and the steps to complete are:
Ingredients:
1 8-inch flour tortilla
Jar of nut butter
Jar of jelly

Recipe:
Step 1: Place a tortilla on the cutting board.
Step 2: Scoop and spread nut butter onto tortilla.
Step 3: Clean the knife with a paper towel.
[...]
Step 12: Place the pinwheels on a plate.

The following is a summary of the current situation:
User: we'll start the recipe now
User: the recipe says to place a tortilla on the bowl but i
have a plate
```

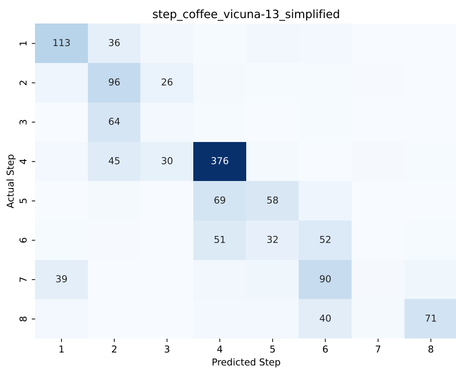
```
User: is that okay?
Instructor: if you need extra stuff
User: thank you
Instructor: but you should be placing the tortilla on a
cutting board

In the image, there is a table with various food items and
utensils. The table is covered with a red tablecloth, and
there are several plates, cups, and bowls placed on it. Some
of the food items include a sandwich, a cake, and a hot dog.
There are also multiple bottles, possibly containing
condiments or beverages, and a spoon. The table appears to be
set for a meal or a gathering, with the food and utensils
arranged for easy access and enjoyment.

Based on the situation described, which step of the recipe do
you think the user is currently on? Explain why.
```

Appendix B

Confusion Matrices

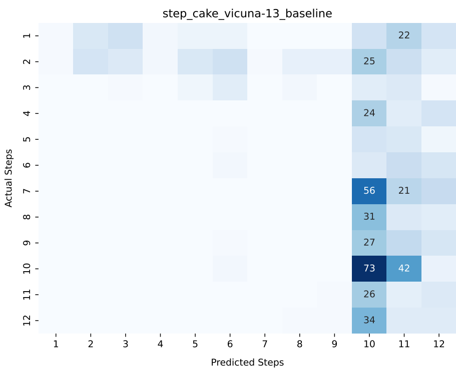


(a)

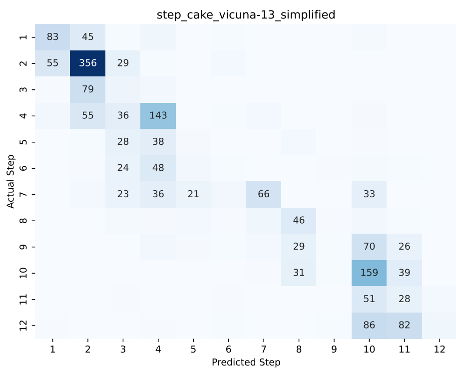


(b)

Fig. B.1 Confusion matrices for Vicuna step predictions across coffee sessions. B.1b shows the simplified prompt setting. Only raw counts greater than 20 are included.



(a) Caption for Image 1



(b) Caption for Image 2

Fig. B.2 Confusion matrices for Vicuna step predictions across coffee sessions. B.2b shows the simplified prompt setting.

Appendix C

Instruction Tuning Data

C.1 Train Set

- T13_pinwheel, T30_pinwheel, T40_pinwheel, T53_pinwheel, T34_pinwheel
- T18_coffee, T28_coffee, T37_coffee, T42_coffee, T21_coffee
- T16_cake, T36_cake, T6_cake, T22_cake

C.2 Training Data Example

The following shows an example of a data point in our small synthetic IT dataset. An input x would look like:

```
An instructor is helping a user make a pinwheel.
The ingredients required and the steps to complete are:
Ingredients:
1 8-inch flour tortilla
Jar of nut butter
Jar of jelly

Recipe:
Step 1: Place a tortilla on the cutting board.
Step 2: Scoop and spread nut butter onto the tortilla.
Step 3: Clean the knife with a paper towel.
[...]
Step 12: Place the pinwheels on a plate.
```

```
The following is a summary of the current situation:
User: I’m not sure if this is fine?
Instructor: yeah that should be fine
User: ok
User: clean the knife
User: so I use the paper to clean my knife?
Instructor: yeah
User: ok
User: alright jelly
User: but there’s two jelly which one I should use?
Instructor: uh any type of jelly is fine
The user just said: ’ok’

Based on the situation described, explain in
detail why the user is on Step 4 of the recipe.
```

The corresponding *y* for this input is Mixtral’s response to *x*:

```
MIXTRAL ANSWER: The user is on Step 4 of the recipe,
which is to spread jelly on the tortilla, because
they have already completed the previous steps. In
the conversation provided, the user can be seen
acknowledging and following the steps and
instructions given by the instructor.

First, the user placed a tortilla on the cutting
board (Step 1), and then spread nut butter on it
(Step 2). The user also cleaned the knife with a
paper towel after spreading the nut butter (Step
3) and asked for confirmation from the instructor,
who affirmed that it was fine.

Next, the user spread jelly on top of the nut
butter (Step 4) after asking the instructor which
type of jelly to use. The instructor replied that
any type of jelly is fine, and the user
acknowledged this by saying "ok." Therefore, based
on the conversation and the recipe’s steps, it can
```

```
be concluded that the user has reached Step 4 of
the recipe.
```