ReALM: Reference Resolution As Language Modeling

Joel Ruben Antony Moniz* [1], **Soundarya Krishnan*** [2], **Melis Ozyildirim**[3], **Prathamesh Saraf, Halim Cagri Ates, Yuan Zhang, Hong Yu**[4]
{[1]joelmoniz, [2]skrishnan22, [3]melisozyildirim, [4]hong_yu}@apple.com
Apple

Abstract

Reference resolution is an important problem, one that is essential to understand and successfully handle contexts of different kinds. This context includes both previous turns and context that pertains to non-conversational entities, such as entities on the user's screen or those running in the background. While LLMs have been shown to be extremely powerful for a variety of tasks, their use in reference resolution, particularly for non-conversational entities, remains underutilized. This paper demonstrates how LLMs can be used to create an effective system to resolve references of various types, by showing how reference resolution can be converted into a language modeling problem, despite involving forms of entities like those on screen that are not traditionally conducive to being reduced to a text-only modality. We demonstrate large improvements over an existing system with similar functionality across different types of references, with our smallest model obtaining absolute gains of over 5% for on-screen references. We also benchmark against GPT-3.5 and GPT-4, with our smallest model achieving performance comparable to that of GPT-4, and our larger models substantially outperforming it.

1 Introduction

Human speech typically contains ambiguous references such as "they" or "that", whose meaning is obvious (to other humans) given the context. Being able to understand context, including references like these, is essential for a conversational assistant that aims to allow a user to naturally communicate their requirements to an agent, or to have a conversation with it (Luger and Sellen, 2016; Ljungholm, 2021). In addition, enabling the user to issue queries about what they see on their screen is a crucial step in ensuring a true hands-free experience in voice assistants. For instance, consider the following interactions between a user and an agent shown in Table 1.

Table 1: Sample Interactions between a user and an agent.

Speaker	Dialogue
User	Show me pharmacies near me
Agent	Here is a list I found.
Agent	... (list presented)
User (eg 1)	Call the one on Rainbow Rd.
User (eg 2)	Call the bottom one.
User (eg 3)	Call this number (present onscreen)

Here, it is immediately apparent that it would not be possible for the Agent to understand or complete the user's query without the ability to use and comprehend context. It also stands to reason that there are multiple types of context that are necessary to handle user queries: conversational context, on-screen context, and background entities.

Recent Large Language Models (LLMs) (Stammbach et al., 2022; Touvron et al., 2023; Santhanam et al., 2022; Dettmers et al., 2023) have often enabled end-to-end experiences, perhaps even obviating the need of a traditional multi-stage pipeline that includes reference resolution (Khatri et al., 2018). There are, however, still several real-world cases where a pipeline is valuable, perhaps even essential, and an end-to-end approach falls short. First, when a framework runs completely on-device (for example, for privacy and efficiency reasons) on a system such as a smartphone that has relatively limited computing power, due to the low-power nature of the system and latency constraints, using a single, large, end-to-end model is infeasible: using a single LLM for this task would usually require the use of a large model with long prompts for true end-to-end experiences (Wei et al., 2022). Second, consider the case when the model has to integrate with APIs, has to consume information from components upstream, or has to provide

* Equal contribution

Proceedings of the 25th Annual Meeting of the Special Interest Group on Discourse and Dialogue, pages 51–65
September 18–20, 2024.

information to be consumed downstream: while in these cases it is possible to have an end-to-end approach having the LLM write API calls (Patil et al., 2023; Qin et al., 2023), this often requires a large language model and a complete overhaul of existing pipelines, which might be cumbersome or completely infeasible. Third, the use of a focused model would allow for an existing reference resolution module to be swapped with improved versions in a transparent way, while providing improved ability to hill-climb and improved interpretability, by virtue of the system being modular. Finally, for the task under consideration in this paper, reference resolution does not include solely conversational references, but also includes the ability to reference an on-screen and/or a background entity that is part of what the user currently perceives in their interaction with a device, but has not been a part of the conversational history that results from their direct interaction with the virtual agent in question. There thus continues to be utility in exploring "traditional" NLP tasks such as reference resolution, despite some of the larger language models being able to handle them implicitly. In this work, we thus advocate the use of (relatively) smaller language models, but fine-tuned for specifically and explicitly for the task of reference resolution.

Along similar lines, relying on language modeling alone (Bajaj et al., 2022; Patra et al., 2022; Zheng et al., 2023) has recently shown great promise in being able to handle a variety of tasks (Wang et al., 2018, 2019; Hendrycks et al., 2020; Wei et al., 2021; Chung et al., 2022), such as causal reasoning, linguistic acceptability, question answering, textual entailment and even coreference resolution: Using Language Models (LMs) does exceedingly well on tasks that can be modeled in a sequence-to-sequence fashion. However, the biggest challenge with adopting this technique for the general reference resolution task in the context of a voice assistant lies in resolving references to entities on the screen and using their properties, in other words, getting the LM to, informally speaking, "see". In particular, it is non-obvious how to encode entities on a screen in a manner that is conducive to being resolved by an LM, while also being consistent enough with how conversational entities are encoded to enable the LM to successfully perform reference resolution on both types of entities.

In this work, we propose reconstructing the screen using parsed entities and their locations to generate a purely textual representation of the screen that is visually representative of the screen content. The parts of the screen that are entities are then tagged, so that the LM has context around where entities appear, and what the text surrounding them is (Eg: call the business number). To the best of our knowledge, this is the first work using a Large Language Model that aims to encode context from a screen.

2 Realated Work and Motivation

While traditional reference resolution systems have explored conversational and visual/deictic references in great depth (Kottur et al., 2018; Schwartz et al., 2019; Kang et al., 2019), resolving on-screen references is a domain that has been relatively under-explored. However, as shown above, conversational agents on a mobile device need to understand references to the screen, and to support such experiences, to be truly natural. On screen references differ from visual and deictic references for several reasons: they tend to be more structured and highly textual, which enables the use of a lighter model to treat this as a text-only problem without a visual component; further, user queries around on-screen elements often tend to be more action-oriented rather than QA based; finally, they use synthetic screens rather than natural real-world images, which are much easier to parse, but whose distribution completely differs from that on which larger pre-trained image-based systems (such as CLIP (Radford et al., 2021)) tend to be trained. Further, jointly being able to perform conversational and on-screen reference resolution has been even less explored, with prior work often focusing on images and graphics (Willemsen et al., 2023), or UI elements (You et al., 2024).

Vision transformers (Dosovitskiy et al., 2020; Touvron et al., 2021; Liu et al., 2021; Yu et al., 2021) and other pre-trained models have recently gained prominence as a popular first step in tasks that require visual understanding. However, these tend to be trained on natural, real-world images rather than screenshots of on-screen layouts, which have a very different distribution. In addition, these can be extremely expensive to (pre-)train, requiring a very large number of images and several hundred GPU hours (or more). Further, they tend to not perform as well on images heavily embedded with text, and dedicated textual understanding approaches (Xu et al., 2020, 2021; Hwang et al.,

2021a,b; Hong et al., 2022) tend to heavily rely on multiple modules such as bounding box detection and OCR while also relying on good image quality. Joint vision+text models are also substantially more expensive with respect to parameters and computational cost. Finally, these models would need to parse text to be able to perform function (Eg: "call the business number" needs to extract the number associated with the business landline from the raw image), a process which can be complex and compute intensive when bearing in mind that the underlying text and its location on the screen has been referred by the system, and as a consequence can be relatively easily extracted without large, complex models.

The most closely related work which we are aware of, and which we consequently use as our baseline, is that of Ates et al. (2023), an extension of Bhargava et al. (2023) which deals purely with on-screen references; however, it suffers from several drawbacks, which we address in this work. First, these approaches rely on a dedicated "Category module" to deal with type-based references. This module often requires manually on-boarding entities every time a new type is created (a common occurrence in voice assistants, as the supported functionality of the assistant is expanded over time). In addition, such modules often treat each type as distinct, with the similarity of different types ignored. This, in turn, leaves on the table the potential positive transfer that could have happened between semantically related classes (such as "phone number" and "contact") when data is added for one of those classes. This approach is thus difficult to scale to new entity types and use cases. Second, these systems rely on the use of hand-crafted rule-based textual overlap features, which require heavy feature engineering and tend not to be robust. In addition, these heuristics often do not account for semantic similarity, and are not able to encode real-world understanding or commonsense reasoning. Finally, these methods effectively classify how related each entity is to the query in question independently of all other entities and later threshold them, whereas our current approach directly picks out the most relevant option (or options), while also allowing for no entities to be relevant. Our approach thus additionally has the advantage of removing the reliance on a set threshold, while also providing all the functionality supported in the previous approaches.

3 Task

We formulate our task as follows: Given relevant entities and a task the user wants to perform, we wish to extract the entity (or entities) that are pertinent to the current user query. The relevant entities are of 3 different types:

1. On-screen Entities: These are entities that are currently displayed on a user's screen

2. Conversational Entities: These are entities relevant to the conversation, which predominantly include those that come from a previous turn. For example, let's say that the first turn of the user is "Call Mom.", which is an unambiguous turn that uses a contact called Mom. Shortly after, if the user says "Text her", the reference "her" needs to be resolved to the contact for "Mom" that was brought up in the previous turn; this contact is thus a conversational entity. Another example might involve an interaction in which the user requests for a list of places or alarms to choose from (or the agent presents one for a user turn such as "Show me pharmacies near me"); each item in this list then becomes a conversational entity for subsequent turns.

3. Background Entities: These are relevant entities that come from background processes that might not necessarily be a direct part of what the user sees on their screen or their interaction with the virtual agent; for example, an alarm that starts ringing or music that is playing in the background.

We pose the task of reference resolution as a multiple choice task for the LLM, where the intended output is a single option (or multiple options) from the entities shown on the user's screen. In some cases, the answer could also be "None of these", in which case the model needs to predict "0".

To evaluate this task, we check if the predicted set of options matches the ground truth set; in other words, we allow the model to output the relevant entities in any order, i.e. if the Ground Truth is entities 8, 7, and 4, then we accept any permutation of these three correct entities while evaluating the performance of the model.

Note that as in Ates et al. (2023); Bhargava et al. (2023), we assume that entities along with their types come in from an upstream system (for example, through a mechanism involving entity pullers which are able to extract entities in a high recall

manner or through a donation from a device app, as in Aas et al. (2023)).

4 Datasets

Our datasets comprise data that was either synthetically created, or created with the help of annotators. Each data point contains the user query and a list of entities, along with the ground-truth entity (or set of entities) that are relevant to the corresponding user query. Each entity, in turn, contains information about its type and other properties such as the name and other textual details associated with the entity (the label and time of an alarm, for example). For data points where relevant on-screen context exists, this context is available in the form of the bounding box of the entity, and the list of objects surrounding it along with properties of these surrounding objects such as their types, textual contents and locations. Note that our data collection follows that of Bhargava et al. (2023); Ates et al. (2023); we present an overview here and direct the interested reader to the aforementioned papers for a more detailed description. Note also that each dataset below is somewhat representative of one of our tasks of interest (with our synthetic data bucket being used for both conversational and background entity resolution).

Table 2: Dataset Sizes (Train Set and Test Set)

Dataset	Train	Test
Conversational	2.3k	1.2k
Synthetic	3.9k	1.1k
On-screen	10.1k	1.9k

4.1 Conversational Data

In this case, data is collected for entities that are relevant to the user interaction with the agent. To do this, annotators are shown sample conversations between a user and an agent with synthetic lists of entities provided, and asked to provide queries that unambiguously reference an arbitrarily picked entity in the aforementioned synthetic list. Annotators might thus be provided with a synthesized list of businesses or alarms and asked to refer to a particular entity within that list.

For example, the annotator might be shown a list of businesses that are synthetically constructed, and then asked to refer to a specific one in the list provided; for instance, they might say "Take me to the one that's second from the bottom" or "Call the one on Main Street".

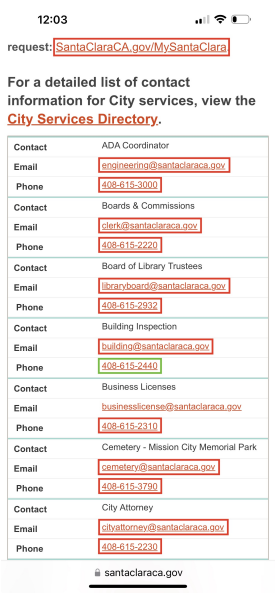
4.2 Synthetic Data

Another approach to obtain data is to rely on synthetic data from templates. This approach is particularly useful for type-based references, when the user query and the entity type are sufficient to resolve the reference, and descriptions are not relied upon. Note that the synthetic nature of this dataset does not preclude it from containing datapoints in which multiple entities can be resolved to a given reference: for example, for the query "play it", "it" can be resolved to all entities of both the types "music" and "video".

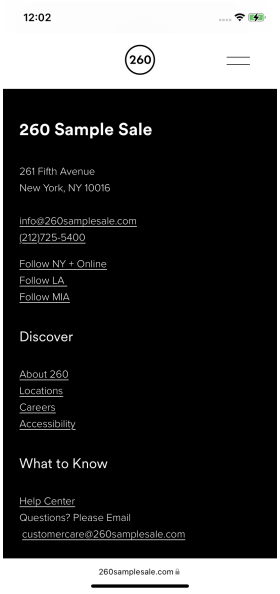
The pipeline used to generate the synthetic data comprises of two parts: a set of templates and a list accompanying each template. The first part, a "language template", contains different variations of queries that can be used for targeted cases, with slots present that can be filled pragmatically from those defined in the "slot list". The second, a "slot list" accompanying the aforementioned template, includes mentions and other possible slot values (often comprising of named entities that aren't mentions, or other slots that can take a large number of possible values such as date-times) if necessary. The slot list also contains the ground truth entity (or entities) that the mentions listed, when filled into the language template, could resolve to.

The data generation pipeline then takes the language template and slot list, and uses them to generate the possible queries. It does this by substituting corresponding values from the slot lists into the language templates to obtain fully formed user queries. The corresponding synthetic data is formed by using these queries and the ground truth entities present in the slot list, and adding in entities of other types into the data to serve as random negatives.

For example, a given language template might consist of phrases like "share [mention] with [name]" and "send [mention] to [name] please". The corresponding slot list might have "[mention]" mapping to "this address" and "that address", "[name]" mapping to various person names, and the ground truth entity tagged as "email address" and "physical address". The pipeline then generates queries like "share that address with Mom" with "email address" and "physical address" entities marked as possible ground truth entity types, and entities of other types marked as negative.



(a) Screenshot example used in first annotation project



(b) Screenshot example used in second annotation project

Figure 1: Sample screenshots used in the annotation of on-screen data. The data was annotated in a two-step process, as described in Section 4.3.

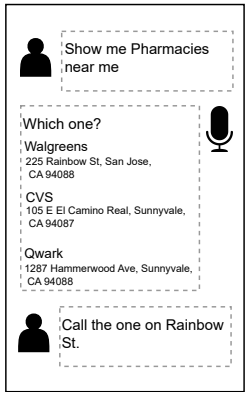
4.3 On-screen Data

As in Bhargava et al. (2023), screen data were collected from various web pages where phone number, e-mail and/or physical address information exist. Our on-screen data annotation comprised of a two-phase process. The first phase was used to obtain queries based on the screens shown, and the second one was for identifying the entities and mention for the given query. In the first grading project, annotators were given a screenshot (Figure 1a) with green and red boxes, and were asked to classify the green boxed data into one of the entities such as phone number, email address, etc. Then, annotators were then asked to provide three unique queries for the green boxed data.

In the second annotation project (Figure 1b), queries collected in the first step were shown to annotators one by one with their corresponding screenshots (but this time, without the bounding boxes), and with all the screen entities as a list. The annotators were asked if the query contains a mention to one of the given visual entities, and if the query sound natural. They were also asked to provide the entities from the list that were referred to in the given query, and to tag the part of the query referring that entity.

5 Models

We compare our proposed model ReALM, described in detail in Section 5.3 below, with two baseline approaches: one based on the reference resolver proposed in MARRS (Section 5.1), which is non-LLM based, and one based on ChatGPT (both GPT-3.5 and GPT-4; Section 5.2).



(a) Conversational User Turns

(b) Onscreen Capture

Figure 2: Technical diagrams representing user turns with a conversational assistant in (a), and a user screen in (b). Shaded rectangles represent various elements shown on the screen detectable by screen parser-extractors.

5.1 MARRS

As a baseline, we compare against the system proposed in Ates et al. (2023), in turn a variation of Bhargava et al. (2023), both of which are non-LLM based approaches. While the latter approach focuses on on-screen entities, MARRS extends it to conversational and background entities as well. For our baseline comparison, we trained a re-implementation of this system with the datasets described in Section 4, which includes conversation, on-screen and synthetic data. Note that in contrast to our approach, which uses a generic off-the-shelf LLM, this baseline we compare against was specifically designed for the task of reference resolution.

5.2 ChatGPT

As another baseline, we run the GPT-3.5 (Brown et al., 2020; Ouyang et al., 2022) and GPT-4 (Achiam et al., 2023) variants of ChatGPT, as available on 2024-01-24, with in-context learning. As in our setup, we aim to get both variants to predict a list of entities from a set that is available. In the case of GPT-3.5, which only accepts text, our input consists of the prompt alone; however, in the

case of GPT-4, which also has the ability to contextualize on images, we provide the system with a screenshot for the task of on-screen reference resolution, which we find helps substantially improve performance. Note that our ChatGPT prompt and prompt+image formulation are, to the best of our knowledge, in and of themselves novel. While we believe it might be possible to further improve results, for example, by sampling semantically similar queries up until we hit the prompt length, this more complex approach deserves further, dedicated exploration, and we leave this to future work.

5.3 Our Approach

In this section, we provide examples of conversational and onscreen reference resolution tasks, followed by how we prompt the model to resolve the same.

We use the following pipeline for fine-tuning an LLM (a FLAN-T5 model (Chung et al., 2022)) in our case. We provide the parsed input to our model, and finetune it. Note that unlike for the baseline, we do not run an extensive hyperparameter search on the FLAN-T5 model, sticking to the default fine-tuning parameters.

Select which among the following entities, if any, are required to understand the user request below. Output 0 if none of the entities are relevant.
User request: Call the one on Rainbow St
User Entities:
0. None
1. Type: Local Business | Name: Walgreens | Address: 225 Rainbow St, San Jose CA 94088
2. Type: Local Business | Name: CVS | Address: 105 E El Camino Real, Sunnyvale, CA 94087
3. Type: Local Business | Name: Qwark | Address: 1287 Hammerwood Ave, Sunnyvale, CA
Relevant entity:

Select which among the following entities, if any, are required to understand the user request below. Output 0 if none of the entities are relevant.
User request: Save the phone number at the bottom-right
Screen:
Your New home!
Steven Realtors Inc.
Trusted by over 5 million
Proud homeowners
Contact Us
Monday - Friday {{1. (206) 198 1999}}
Saturday - Sunday {{2. (206) 198 1699}}
Relevant entity:

Each data point consisting of a user query and the corresponding entities is converted into a sentence-wise format that we can feed to an LLM for training. Examples of the input before and after processing are shown in Appendix Sections A and C, with examples of how we convert entities of different types into text shown in Appendix B. Note that the entities are shuffled before being sent to the model so that the model does not overfit to particular entity positions.

With respect to the output that the model predicts, empirically, we find that the model is consistently able to predict a valid integer (or list of integers), without deviating and outputting any other text. In addition, we observe that the model also respects general output constraints (such as not predicting a '0' that represents 'None of These' at the same time as one or more other entities) as well as those constrains enforced by the input (such as ensuring all predicted entity indices actually exist on the input side). The one exception that we observe is that, on occasion, we find that the model predicts the same entity twice (successively) in its output list. The only post-processing heuristic we apply is thus to convert the model's predictions into a set of unique entities.

5.3.1 Conversational References

For the sake of this work, we assume conversational references to be of two types: type-based and descriptive. Type-based references are heavily reliant on using the user query in conjunction with the types of the entities to identify which entity (of a set of entities) are most relevant to the user query in question: for example, if the user says "play this", we know that they are referring to an entity like a song or a movie, as opposed to a phone number or an address; "call him" likewise refers to a contact or possibly a phone number, as opposed to an alarm. Descriptive references, in contrast, tend to use a property of the entity to uniquely identify it: "The one in Times Square" for example might help uniquely refer to one among a set of addresses or business. Note that it is often the case that references might rely on both types and descriptions to unambiguously refer to a single object: consider the examples "play the one from Abbey Road" vs "directions to the one on Abbey Road", both of which rely on both the entity type and description to identify a song in the first case and address in the second. In our proposed approach, we simply encode the type and various properties of the entity. We show our detailed encoding scheme in Appendix B.

5.3.2 Onscreen References

For onscreen references, as in Bhargava et al. (2023), we assume the presence of upstream data detectors that are able to parse screen text to extract entities. These entities are then available along with their types, bounding boxes and a list of non-entity text elements surrounding the entity in question.

To encode these entities (and thereby, the relevant parts of the screen) into the LM in a manner that involves text alone, we use the novel algorithm given in Algorithm 1. Intuitively, we assume the location of all entities and their surrounding objects to be representible by the center of their respective bounding boxes. We then sort these centers (and thereby, the associated objects) from top-to-bottom (i.e., vertically, along the y-axis), and the use a stable sort to sort from left-to-right (i.e., horizontally, along the x-axis). Next, all objects that are within a `margin` are treated as being on the same line, and are separated from each other by a tab; objects further down outside the margin are placed on the next line, and this is repeatedly, effectively encoding the screen in a left-to-right, top-to-bottom fashion in plain text.

6 Results

Table 3: Model Accuracy for Different Datasets. A prediction is correct if the model correctly predicts all relevant entities, and incorrect otherwise. **Conv** refers to the Conversational Dataset, **Synth** to the Synthetic one, **Screen** to the Onscreen one and **Unseen** to a conversational dataset pertaining to a held-out domain.

Model	Conv	Synth	Screen	Unseen
MARRS	92.1	99.4	83.5	84.5
GPT-3.5	84.1	34.2	74.1	67.5
GPT-4	97.0	58.7	90.1	98.4
ReALM-80M	96.7	99.5	88.9	99.3
ReALM-250M	97.8	99.8	90.6	97.2
ReALM-1B	97.9	99.7	91.4	94.8
ReALM-3B	97.9	99.8	93.0	97.8

We present our results in Table 3. Overall, we find that our approach outperforms the MARRS model in all types of datasets. We also find that our approach is able to outperform GPT-3.5, which has a significantly larger number of parameters than our model by several orders of magnitude. We also find that our approach performs in the same ballpark as the latest GPT-4 despite being a much lighter

Algorithm 1: Onscreen Parse Construction with Turn Object Injection

Data: List of turn objects
Result: Onscreen parse

$onscreen\_parse \leftarrow$ Empty list of onscreen parse elements;
`// Step 0: Get all text boxes present in the screen`
for *each turn object* t*, index* i **do**
  `// Step 1: Get unique surrounding objects`
  $surrounding\_objects \leftarrow$ Set of surrounding objects for t;
  `// Step 2: Insert turn objects into the set`
  $surrounding\_objects \leftarrow surrounding\_objects \cup \{[[i.t]]\}$;
`// Step 3: Sorting the centers of all surrounding objects`
$sorted\_objects \leftarrow$ Sort objects in $surrounding\_objects$ by center (Top $\rightarrow$ Bottom, Left $\rightarrow$ Right);
`// Step 4: Determine vertical levels`
$margin \leftarrow$ Margin for considering objects at the same level;
$levels \leftarrow$ List of vertical levels;
for *each object* o *in* $sorted\_objects$ **do**
  $same\_level \leftarrow$ List of objects at the same level as o;
  for *each object other in* $sorted\_objects$ **do**
    if o *is not the same as other and* $|o.center\_top - other.center\_top| \leq margin$ **then**
      $same\_level \leftarrow same\_level \cup \{other\}$;
  $levels \leftarrow levels \cup \{same\_level\}$;
`// Step 5: Construct onscreen parse`
for *each level* l *in* $levels$ **do**
  $level\_parse \leftarrow$ Empty string;
  for *each object* obj *in* l **do**
    $level\_parse \leftarrow level\_parse + \text{"\t"} + obj$;
  $onscreen\_parse \leftarrow onscreen\_parse + \text{"\n"} + level\_parse$;
return $onscreen\_parse$;

(a) Semantic Understanding

User Request: Call the evening Number

Screen:
{{1. 9 AM - 5 PM}}
{{2. 901.969.3120}}
{{3. 5 PM - 9 PM}}
{{4. 901.969.3391}}

Model Output: 4

(b) Summarisation

User Request: Remind me to get printouts before the tax deadline

Screen:
Tax Deadlines 2023
{{1. Feb 15}}
Reclaim your tax exemption from withholding
{{2. April 18}}
First-quarter estimated tax payment due

Model Output: 2

(c) World Understanding

User Request: Take me to the one in Washington

Screen:
Indian Embassy
{{1. 1701 El Camino Real, Mountain View 94040}}
{{2. 333 Dexter Ave N, Seattle 98109}}
{{3. 8295 Tournament Drive, Memphis, TN 38125}}

Model Output: 2

(d) Commonsense Reasoning

User Request: Save the link to the breakfast Recipe

Screen:
IMAGE
Strawberry Granola
{{1. Recipe link}}
IMAGE
Lavender boba tea
{{2. Recipe link}}

Model Output: 1

Table 4: Qualitative examples that demonstrate the ability of ReALM to adapt to complex use-cases.

(and faster) model. We especially wish to highlight the gains on onscreen datasets, and find that our model with the textual encoding approach is able to perform almost as well as GPT-4 despite the latter being provided with screenshots.

Additionally, we also experiment with models of different sizes. We see that while performance in general improves across all dataset families with an increase in model size, the difference is most pronounced for the onscreen datasets, which alludes to the task being more complex in nature. Interestingly, and contrary to an otherwise consistent trend of larger models performing better, we find that performance on our Unseen dataset, which contains a held-out domain, first decreases with an increase in model size before increasing again. We hypothesize that this is due to the double-descent phenomenon (Nakkiran et al., 2019).

6.1 Analysis

GPT-4 $\approx$ ReALM $\gg$ MARRS for new use-cases: As a case study, we explore zero-shot performance of this model on an unseen domain: Alarms (we show a sample data point in Appendix Table 12). The last column in Table 3 compares the performance of all approaches and baselines on this unseen test set. We find that all of the LLM-based approaches outperform the FT model for this test set. Among the two, we find that the performance of ReALM and GPT-4 are very similar for the unseen domain. Additionally, Table 4 shows completely new experiences enabled by ReALM due to the LLM's superior ability to perform complex understanding of natural language.

Table 5: User Request for Setting or Home Device

User Request: Can you make it brighter?

Entities Shown to User:
1. Type: Settings
2. Type: UserEntity | homeAutomationAccessoryName

GPT-4 Prediction: 1 Ground Truth: 1, 2

ReALM $>$ GPT-4 for domain-specific queries We find that due to finetuning on user requests, ReALM is able to understand more domain-specific questions. Consider Table 5. GPT-4 incorrectly assumes the reference to be about only a setting, whereas the ground truth consists of a home automation device in the background as well, and GPT-4 lacks the domain knowledge to be able to recognise that the device would also be relevant to this reference. ReALM, in contrast, doesn't suffer from this due to being trained on domain-specific data.

7 Conclusion and Future Work

In this work, we demonstrate how large language models, which are typically trained on text alone, can be also be adapted to perform reference resolution to items in an extra-linguistic context. We

do this by encoding entity candidates as natural text; we demonstrate how entities that are present on the screen can be passed into an LLM using a novel textual representation that effectively summarizes the user's screen while retaining relative spatial positions of these entities. Our proposed system is thus able to resolve references in a variety of human-computer interaction settings, such as those involving on-screen, conversational and background entities; we note, however, that our proposed approach focuses primarily on anaphoric and deictic references, and we leave the extension of our system to handle other types of references, such as bridging references, to future work.

In addition, we show that ReALM outperforms previous approaches, and performs roughly as well as the current state-of-the-art LLM, GPT-4, despite consisting of far fewer parameters, even for on-screen references despite being purely in the textual domain. It also outperforms GPT-4 for domain-specific user queries, thus making ReALM an ideal choice for a practical reference resolution system that can exist on-device without compromising on performance.

While our approach is effective in encoding the position of entities on the screen, we find that it may not be able to resolve complex user queries that rely on nuanced positional understanding. We thus believe that exploring more complex approaches such as splitting the screen into a grid and encoding these relative spatial positions into text, while challenging, is a promising avenue of future exploration. In addition, in contrast to a critical assumption of our proposed system, not all on-screen entities are textual. While extending this paper to cover on-screen images, graphics and UI elements is beyond the scope of this work, this is certainly another extension that merits further investigation.

Ethics Statement

While LLMs can generate unexpected output including potentially harmful text, our system offers the ability to constrain decoding or use simple post-processing to ensure this does not happen. Note however that practically we find very little hallucination or even text that deviates from the format that the models were finetuned on, and thus do not constrain the decoding of the LLM.

Acknowledgements

The authors would like to thank Nidhi Rajshree, Stephen Pulman, Leon Liyang Zhang, Jiarui Lu, Jeff Nichols, Shruti Bhargava, Dhivya Piraviperumal, Junhan Chen and the anonymous reviewers for their help, suggestions, and feedback.

References

Cecilia Aas, Hisham Abdelsalam, Irina Belousova, Shruti Bhargava, Jianpeng Cheng, Robert Daland, Joris Driesen, Federico Flego, Tristan Guigue, Anders Johannsen, et al. 2023. Intelligent assistant language understanding on device. *arXiv preprint arXiv:2308.03905*.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Halim Cagri Ates, Shruti Bhargava, Site Li, Jiarui Lu, Siddhardha Maddula, Joel Ruben Antony Moniz, Anil Kumar Nalamalapu, Roman Hoang Nguyen, Melis Ozyildirim, Alkesh Patel, et al. 2023. Marrs: Multimodal reference resolution system. In *Proceedings of The Sixth Workshop on Computational Models of Reference, Anaphora and Coreference (CRAC 2023)*, pages 51–58.

Payal Bajaj, Chenyan Xiong, Guolin Ke, Xiaodong Liu, Di He, Saurabh Tiwary, Tie-Yan Liu, Paul Bennett, Xia Song, and Jianfeng Gao. 2022. Metro: Efficient denoising pretraining of large scale autoencoding language models with model generated signals. *arXiv preprint arXiv:2204.06644*.

Shruti Bhargava, Anand Dhoot, Ing-Marie Jonsson, Hoang Long Nguyen, Alkesh Patel, Hong Yu, and Vincent Renkens. 2023. Referring to screen texts with voice assistants. In *ACL*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias

Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Teakgyu Hong, Donghyun Kim, Mingi Ji, Wonseok Hwang, Daehyun Nam, and Sungrae Park. 2022. Bros: A pre-trained language model focusing on text and layout for better key information extraction from documents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10767–10775.

Wonseok Hwang, Hyunji Lee, Jinyeong Yim, Geewook Kim, and Minjoon Seo. 2021a. Cost-effective end-to-end information extraction for semi-structured document images. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3375–3383, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Wonseok Hwang, Jinyeong Yim, Seunghyun Park, Sohee Yang, and Minjoon Seo. 2021b. Spatial dependency parsing for semi-structured document information extraction. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 330–343, Online. Association for Computational Linguistics.

Gi-Cheon Kang, Jaeseo Lim, and Byoung-Tak Zhang. 2019. Dual attention networks for visual reference resolution in visual dialog. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2024–2033.

Chandra Khatri, Behnam Hedayatnia, Anu Venkatesh, Jeff Nunn, Yi Pan, Qing Liu, Han Song, Anna Gottardi, Sanjeev Kwatra, Sanju Pancholi, et al. 2018. Advancing the state of the art in open domain dialog systems through the alexa prize. *arXiv preprint arXiv:1812.10757*.

Satwik Kottur, José MF Moura, Devi Parikh, Dhruv Batra, and Marcus Rohrbach. 2018. Visual coreference resolution in visual dialog using neural module networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 153–169.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022.

Alice Ljungholm. 2021. Voice interaction vs screen interaction when controlling your music-system. In *Proceedings of the 21st Student Conference in Interaction Technology and Design*, pages 103–108.

Ewa Luger and Abigail Sellen. 2016. " like having a really bad pa" the gulf between user expectation and experience of conversational agents. In *Proceedings of the 2016 CHI conference on human factors in computing systems*, pages 5286–5297.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. 2019. Deep double descent: Where bigger models and more data hurt. In *International Conference on Learning Representations*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2023. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*.

Barun Patra, Saksham Singhal, Shaohan Huang, Zewen Chi, Li Dong, Furu Wei, Vishrav Chaudhary, and Xia Song. 2022. Beyond english-centric bitexts for better multilingual language representation learning. *arXiv preprint arXiv:2210.14867*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3715–3734, Seattle, United States. Association for Computational Linguistics.

Idan Schwartz, Seunghak Yu, Tamir Hazan, and Alexander G Schwing. 2019. Factor graph attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2039–2048.

Dominik Stammbach, Maria Antoniak, and Elliott Ash. 2022. Heroes, villains, and victims, and GPT-3: Automated extraction of character roles without training data. In *Proceedings of the 4th Workshop of Narrative Understanding (WNU2022)*, pages 47–56,

Seattle, United States. Association for Computational Linguistics.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. 2021. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *Transactions on Machine Learning Research*.

Bram Willemsen, Livia Qian, and Gabriel Skantze. 2023. Resolving references in visually-grounded dialogue via text generation. In *Proceedings of the 24th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 457–469.

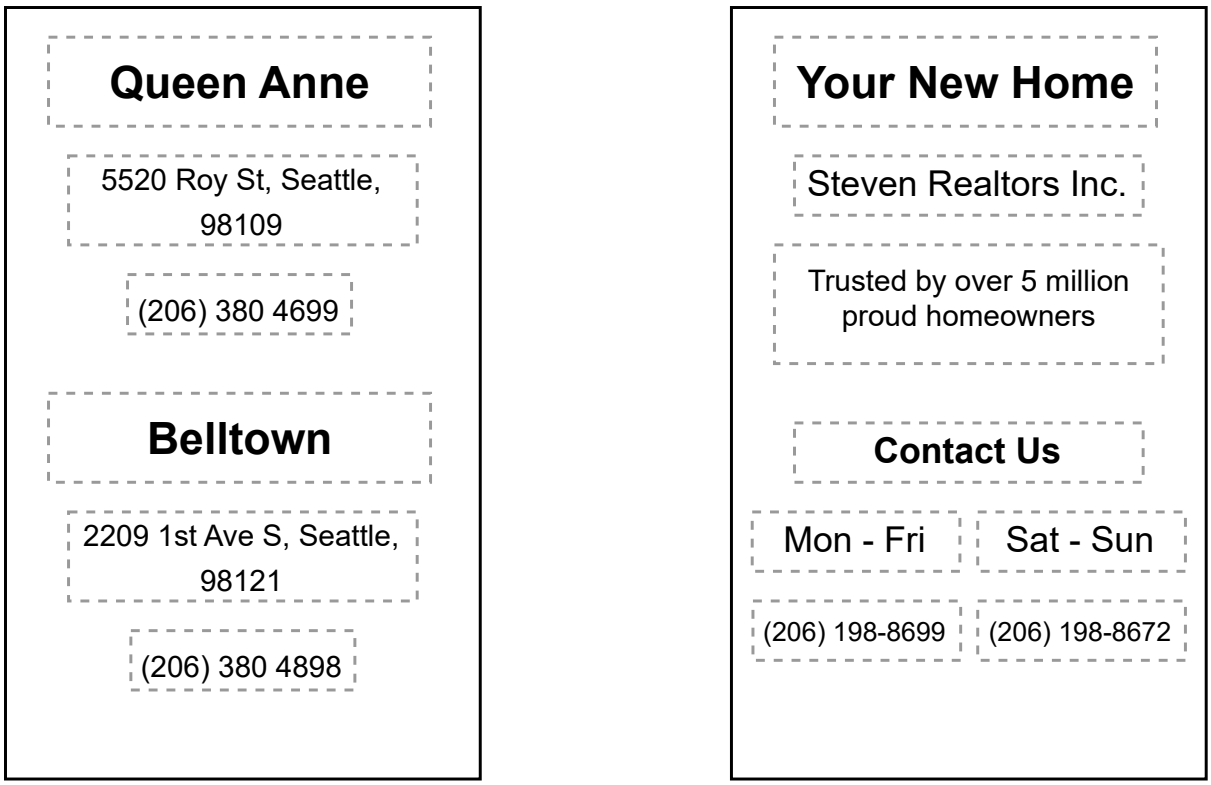
Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, Min Zhang, and Lidong Zhou. 2021. LayoutLMv2: Multi-modal pre-training for visually-rich document understanding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2579–2591, Online. Association for Computational Linguistics.

Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2020. Layoutlm: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1192–1200.

Keen You, Haotian Zhang, Eldon Schoop, Floris Weers, Amanda Swearngin, Jeffrey Nichols, Yinfei Yang, and Zhe Gan. 2024. Ferret-ui: Grounded mobile ui understanding with multimodal llms. *arXiv preprint arXiv:2404.05719*.

Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. 2021. Vector-quantized image modeling with improved vqgan. *arXiv preprint arXiv:2110.04627*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*.



(a) Onscreen Capture 1

(b) Onscreen Capture 2

Figure 3: Technical diagrams representing user screens. Shaded rectangles represent various elements shown on the screen detectable by screen parser-extractors.

A Encoding onscreen entities

First, we show sample representations of what a screen grab might look like, as parsed and exposed to the system. We show these representations in Figure 3

We now describe some other strategies of encoding on-screen elements that we explored.

- **Clustering:** We explored a clustering-based approach wherein we perforedm a spatial clustering of the various surrounding objects present in the screen. We did this to establish semantic clusters wherein a user could refer to nearby bounding boxes (such as the contact information) by a particular title. The detailed approach is given in Algorithm 2, and a sample encoding is shown in Table 6. The biggest drawback of the approach was that the prompt length often explodes as the number of entities in a cluster increases, as each of the objects in the cluster would have every other object in its surrounding objects.

- **Onscreen Grab:** To mitigate this issue, we employed a second approach (similar to our final approach), wherein we parsed the screen as in our final approach, the only difference being that we didn't annotate the turn objects within the parse itself, but provided the turn objects as a list instead (see Table 7).

- **Onscreen Grab with Injected Turn Objects:** Finally, the exact algorithm employed in our final approach is given in 1, and a sample encoding is shown in Table 8.

We show an ablation in Figure 4, in which we

Table 6: Clustering-based encoding

User Request: Get me directions to the branch in Queen Anne
Entities Shown to User: 1. Type: Postal Address | Value: 5520 Roy St, Seattle 98109 | surr_objects: Queen Anne, (206) 380 4699 2. Type: Phone Number | Value: (206) 380 4699 surr_objects: Queen Anne, 5520 Roy St, Seattle 98109 3. Type: Phone Number | Value: (206) 380 4898 surr_objects: Belltown, 2209 1st Ave S, Seattle 98121 4. Type: Postal Address | Value: 2209 1st Ave, Seattle 98121 | surr_objects: Belltown, (206) 380 4898
Ground Truth: 1

Table 7: Onscreen Grab encoding

User Request: Save the phone number at the bottom-right
Screen: Your New home! Steven Realtors Inc. Trusted by over 5 million Proud homeowners Contact Us Monday - Friday (206) 198 1699 Saturday - Sunday (206) 198 1999
Entities Shown to User: 1. Type: Phone Number | Value: (206) 198 1999 2. Type: Phone Number | Value: (206) 198 1699
Ground Truth: 1, 2

show the performance of the various encoding approaches described above (and some other hill-climbing efforts).

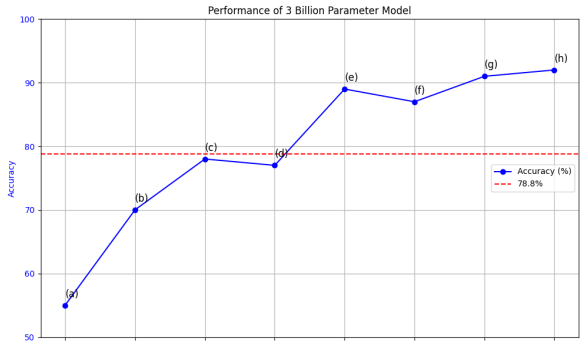


Figure 4: Performance improvements with each experiment – (a) Baseline Finetuned LLM, (b) Obtaining screen elements through OCR, (c) Obtaining screen elements through UI elements and Clustering (d) Adding an extra newline between the instruction and user request, (e) Onscreen Grab, (f) Onscreen Grab with injected turn objects, (g) Onscreen Grab with injected turn object + needing lines to be separated by at least Margin, (h) Separating elements in the same line by a tab

We show the algorithm used to encode onscreen entities, described in Section 5.3.2, in Algorithm 1.

Table 8: Injected Onscreen Encoding (Final Approach)

User Request: Save the phone number at the bottom-right
Screen: Your New home! Steven Realtors Inc. Trusted by over 5 million Proud homeowners Contact Us Monday - Friday {{1. (206) 198 1999}} Saturday - Sunday {{2. (206) 198 1699}}
Ground Truth: 1, 2

Algorithm 2: Surrounding Object Clustering and Prompt Generation

Data: List of MDF turn objects
Result: Updated turn objects with surrounding object prompts

for *each MDF turn object* t **do**
  `// Step 1: Get unique surrounding objects`
  $surrounding\_objects \leftarrow$ Set of unique surrounding objects for t;
  `// Step 2: Spatially cluster surrounding object bounding boxes`
  $clusters \leftarrow DBScan(surrounding\_objects, rect\_distance)$;
  `// Step 3: Predict the cluster for turn object`
  $t\_cluster \leftarrow$ Predicted cluster for t;
  for *each surrounding object* s *in* $surrounding\_objects$ **do**
    if s *belongs to cluster* $t\_cluster$ **then**
      `// Step 4: Process non-overlapping surrounding objects`
      if *no string overlap between* t *and* s **then**
        Add s to the prompt under key ‘surrounding_object‘;
  `// Step 5: Provide global positioning information`
  $t.distance\_from\_top \leftarrow$ Compute distance from the top for t;
  $t.distance\_from\_left \leftarrow$ Compute distance from the left for t;
return *prompt*;

Table 9: Entity Domains and their Representations

Entity Type	After
alarm	Type: Alarm | time: 08:06 PM; label: brush hair; status: Off
app	Type: App | clock
book	Type: Book
date time	Type: DateTime | 1 | 1 | 2021
email address	Type: EmailAddress | membership@ipsa.org
flight number	Type: FlightNumber
general text	Type: GeneralText
home device	Type: UserEntity | heater
home room	Type: UserEntity | Db Bedroom
local business	Type: LocalBusiness | PostalAddress: 15 Broad St, Albany 31701 | Ameris Bank | list_position: 13
media album	Type: MediaItem | MediaItemType: MediaItemType_Album | Mellon Collie
package	Type: Package
painting	Type: Painting
person	Type: Person | Sebastian
phone number	Type: PhoneNumber | 955 545 060
photo	Type: Photo
physical address	Type: PostalAddress | GeographicArea: 814 Elmwood Ave, NY, 14222
plant animal	Type: PlantAnimal
setting	Type: Setting | dark mode
tracking number	Type: TrackingNumber
url	Type: Uri | NY.gov

B Entity Representations

In Table 9, we show some examples of various domains and their representations, as fed into the LLM.

Table 10: Sample input with single ground truth

User Request: Call the one on Rainbow St.
Entities Shown to User: 1. Type: Local Business | Name: Walgreens | Address: 225 Rainbow St, San Jose CA 94088 2. Type: Local Business | Name: CVS | Address: 105 E El Camino Real, Sunnyvale, CA 94087 3. Type: Local Business | Name: Qwark | Address: 1287 Hammerwood Ave, Sunnyvale, CA 94089
Ground Truth: 1

Table 11: Sample input with multiple ground truths

User Request: Save the address.
Entities Shown to User: 1. Type: Postal Address | Value: 225 Rainbow St, San Jose CA 94088 2. Type: Email Address | Value: `contactus@cvs.com` 3. Type: URL | Value: `cvspharmacies.com/usa`
Ground Truth: 1, 2, 3

C Sample Inputs

In this section, we show examples of how inputs into the model have been encoded, in the form of a visual representation in Tables 10, 11 and 12.

Table 12: User Request for Alarms

User Request: Switch off the one reminding me to pick up didi.
Entities Shown to User: 1. Type: Alarm | open laptop 2. Type: Alarm | text Lauren to shower 3. Type: Alarm | pick up didi 4. Type: Alarm | forget this
Ground Truth: 3