Referring to Screen Texts with Voice Assistants

Shruti Bhargava, Anand Dhoot, Ing-Marie Jonsson, Hoang Long Nguyen, Alkesh Patel, Hong Yu, Vincent Renkens
Apple Inc.
{shruti_bhargava, adhoot, ingmarie, romanhoangnguyen_long, alkesh.patel, hong_yu, vrenkens}@apple.com

Abstract

Voice assistants help users make phone calls, send messages, create events, navigate and do a lot more. However assistants have limited capacity to understand their users' context. In this work, we aim to take a step in this direction. Our work dives into a new experience for users to refer to phone numbers, addresses, email addresses, urls, and dates on their phone screens. Our focus lies in reference understanding, which becomes particularly interesting when multiple similar texts are present on screen, similar to visual grounding. We collect a dataset and propose a lightweight general purpose model for this novel experience. Due to the high cost of consuming pixels directly, our system is designed to rely on the extracted text from the UI. Our model is modular, thus offering flexibility, improved interpretability, and efficient runtime memory utilization.

1 Introduction

With the advent of internet and smartphones, the world came to our fingertips. And with the emergence of voice assistants (VAs), everything became even more accessible. VAs have become pervasive in the smartphones as they offer natural means of communication to the user. They able a user to perform tasks faster with natural language instead of several taps, app switches, scrolls and typing. However, they are limited in their ability to understand the user's context.

Let us look at an example. In Fig. 1, a user wants to share a number from a webpage to a friend. They might do either of the following:

- memorize the number → go to messages → new message to friend → type the number from memory → send
- select the number → copy → go to messages → new message to friend → paste → send

One solution might be that the user can read out the number to the VA. However reading out may be cumbersome and unnatural as this is not how one would communicate with a person standing next to them. Further, it may create unwarranted ASR errors, especially for texts like URLs or emails. Our work explores how to make this simpler by enabling users to refer to screen elements in requests made to voice assistants. References make conversations more natural and succinct, thus allowing the user to say: "Send the middle number to Tim".

We conduct a user study to explore how users would make requests involving screen elements. Participants are shown screenshots, each containing multiple entities of a category (eg. 3 phone numbers), and asked to type requests for a VA to act on one of them. The study reveals that a majority of users (57%) prefer to use references like "Send that office number to Tim' instead of repeating the full text.

For supporting such experiences, voice assistants need to resolve the references. In this work, we focus on such reference resolution. Specifically, we consider requests referring to phone numbers, addresses, email addresses, URLs, date/time. We

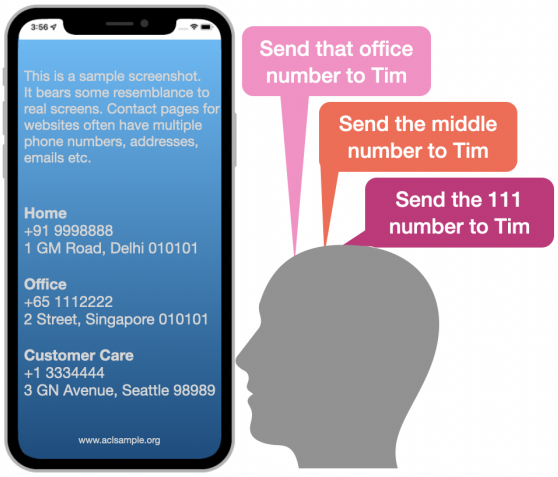


Figure 1: Suppose a user wants to share a number on their screen. We aim to support this in a natural and succinct way by enabling users to refer to screen elements in interactions with Voice Assistants.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 5: Industry Track, pages 752–762

choose these *actionable text* since, ∼50% of screen elements are texts (Zhang et al., 2021), and these categories are commonly acted upon. Users often call or message numbers, share contact details, navigate to addresses.

To understand and evaluate the task, we collect ScreenRef, a dataset of 14k requests, with references to *actionable text* or entities. ScreenRef contains two collections. First, *Descriptive Data*, which is based off screens with multiple similar entities to get descriptive references like 'call the Apple Business Manager number'. This is similar to how visual grounding datasets (Kazemzadeh et al., 2014; Mao et al., 2016) focus on images with multiple objects of the same category to get challenging references. Second, *Category level Data*, includes simpler references to a category without disambiguating within a category, eg. 'call this number'. These are described in Sec. 4.

With the purpose of deploying in real world, it is critical to design solutions with low latency. To this end, we design Screen Reference Resolver, or SRR, a modular attention based architecture for reference understanding. We focus on privacy, hence our model is lightweight and executable on-device. Our network re-uses existing signals available from upstream including request embedding, and text scraped from the UI. We also discuss a heuristic-based baseline (designed for quick prototyping).

Overall, our main contributions are:

1. We explore a novel experience for Voice Assistant users to execute tasks on *actionable text* on the phone screen by using references.
2. We conduct a user study to analyse users' interactions with entities on screen. This reveals interesting insights about usage of references.
3. We design efficient data collection schemes for collecting requests with references to *actionable text* on screen and collect a dataset.
4. To understand references to entities on screen, we propose a heuristic-based baseline and a modular attention-based network, SRR. The model has a small memory footprint, low latency, can run on device, and drastically boosts performance compared to the baseline.

2 Related Work

Grounding to UI elements. Past works have explored mapping natural language commands to UI elements for Chrome web pages (Pasupat et al., 2018), grounding executable actions for UI navigation (Li et al., 2020) and user interaction (Xu et al., 2021). These works primarily focus on navigational commands, thus target buttons, links and input boxes. Our goal is to explore screen referencing capability for common VA tasks, thus we target 'actionable text entities' like phone numbers. Hsiao et al. (2022) propose ScreenQA with questions about UI elements including text, which could also benefit from UI grounding. Wang et al. (2022); Rozanova et al. (2021) investigate LLM abilities for UI grounding. Li et al. (2021); Li and Li (2022) use vision and language transformers for the task. However, we only use the screen texts and no pixels directly. Our solution design focuses on low latency, less memory and privacy-preserved inference that can be run on device.

Voice assistants and Multimodal Interactions. The power of replacing multiple low-level actions by natural language has been explored for webpage designing (Kim et al., 2022), image editing (Laput et al., 2013). Users use VAs for controlling screen content, particularly the visually challenged (Vtyurina et al., 2019). Ljungholm; Luger and Sellen (2016) discuss how lack of context understanding makes VA usage unnatural. Bolt (1980) employed a point-and-speak approach for desktops. Prior works have explored tracking user gaze for multimodal interactions (Drewes et al., 2007), for digital screens (Hutchinson et al., 1989; Mardanbegi and Hansen, 2011) as well as for external, real-world objects (Mayer et al., 2020). In this work, we explore using language to reduce the low-level actions needed to interact with certain text categories on phone screens and thereby increase the context understanding of VAs.

Grounding to objects and text in open scenes. A related task to ours is visual grounding (Kazemzadeh et al., 2014; Mao et al., 2016; Yu et al., 2018), resolving references to physical objects in scenes. The physical form and semantics of text is much different, resulting in different reference forms. Rong et al. (2019, 2017) look at references to text in scenes. However, a lot of their references are of the form 'the text on ...', thus grounding requires less knowledge of text and more of physical objects. Also, the major challenge in open scenes is text localisation and recognition, which is much simpler on phone screens. On the other hand, screens are challenging as they contain a lot more text. TextVQA, from Singh et al. (2019);

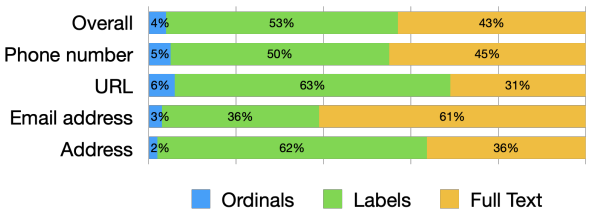


Figure 2: Distribution of request types in the user study. References using labels i.e. text within or around entity are most common, followed by repeating full text. For entities like addresses and URLs, repeating full text may be cumbersome, hence references are more common.

Biten et al. (2019) could utilise grounding to text, but doesn't contain labels for this. Lastly, none of these works cater to task-oriented dialog for text on screen, which is our primary focus.

3 User Study

To understand how users would make requests about entities on the screen, we conduct a user study on Pollfish (pol). We use 4 screens for each *actionable text category* (phone numbers, emails, addresses, and URLs), and each screen has 3 instances of the category (eg. 4 screens with 3 phone numbers each). A total of 300 participants are selected from across US, balanced for VA usage, gender and English as first or other language. Overall, 4800 typed requests were collected.

The responses were reviewed by two researchers. Using heuristics, three common types of requests surfaced: 1. Full Text: "call 1-866-902-7144" 2. Labels: using text other than full entity text "directions to the one in Portland" 3. Ordinals: "send the third email address". The data shows a heavy preference towards the first two (Fig. 2). Intuitively, when browsing information, the eyes are often scanning for a topic of interest. For instance, "I need to call support", explaining the label based requests. Our hypothesis for the high use of full text is that they didn't want to rely on VA's ability to understand the context. Within references, using the text in or around the entity is common and the position is used sometimes. Note that our study was performed on a limited set of users and for a limited set of screens, but we uncover interesting patterns on how users might request actions on screen texts. It is important to keep in mind that speaking full texts could be cumbersome, unnatural and have speech recognition errors, especially for entities like URLs.

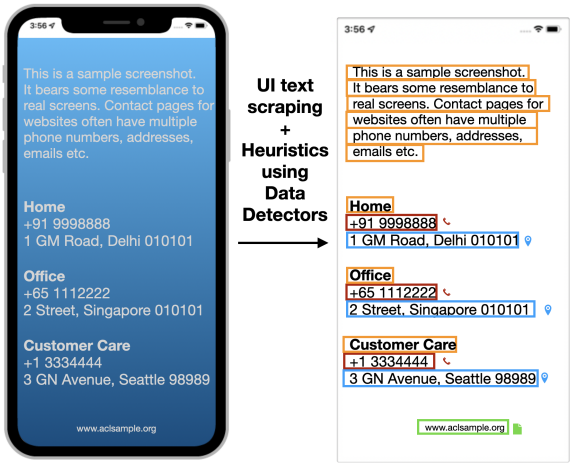


Figure 3: Screen processing is done by upstream systems using data detectors to get entity categories. We get texts with their location, and texts classified as phone, email, address, date/time, URL that form the candidate entities. Example entity: [*text:* +91 9998888, *location:* [0.04, 0.36, 0.4, 0.03], *category:* Phone Number]. These are the inputs to our grounding system.

4 Task and Dataset

Given a screen S (with OCR texts t), text entities $e_1, ..., e_k$ and a request r, the task is to select the entity(entities) $e \in e_1, ..., e_k$ being referred to in r.

We collect ScreenRef, a collection of requests to Voice Assistants with references to actionable text categories on screen (phone number, address, email address, URL, date/time). Due to privacy concerns on sharing a dataset with extracted phone numbers/emails from web pages, we are unable to share the dataset but we discuss the collection protocol in detail (see samples in Fig. 4 and annotation guidelines in Appendix A). We collect full requests, not just reference phrases, since words outside of the explicit reference phrase may hint at the targeted entity. For instance, *call this* has the reference *this* which is ambiguous, however the request can be understood as referring to a phone number. 20 annotators are recruited for our data collections.

We first started with a simple collection protocol. After extracting entities of our interest using data detectors for a list of web pages, we show a web page screen with one highlighted entity and ask graders to provide a request referring to that entity. This would get us a dataset of requests referring to screen entities and their referred entity. However, this ran into major issues. First, annotators would often miss other similar entities on the screen and

Descriptive Data Sample

Request	Send a message to the bottom number
Entities	Phone number [0.04, 0.36, 0.4, 0.03] '+91 9998888'
	Phone number [0.04, 0.59, 0.4, 0.03] '+8 111 2222'
	Phone number [0.04, 0.76, 0.4, 0.03] '+1 333 4444'
Target	+1 333 4444
Screen texts	This is a sample screen [location] | Customer Care [location] ...

Category-level Data Sample

Request	Call this number
Entities	Phone Number | Email Address | Physical Address | URL | Date Time
Targets	Phone Number

Figure 4: Samples from ScreenRef. Descriptive data is collected using screens and has entities and texts from the screen. Category-level data is collected without screens and has an entity pool containing one dummy entity from each of our scoped categories.

provide requests which are ambiguous, eg. "call this number", when there is more than 1 number on the screen and only 1 of them is highlighted, thereby resulting in an incorrect sample for the resolution task. Second, there were a large number of duplicate requests (>40%). This may happen due to several screens may have one entity and thereby annotators may use simple references. Other issues included the need of screens with entities to collect any data, lack of representation of different reference types within the collected data and lack of awareness of ambiguous requests.

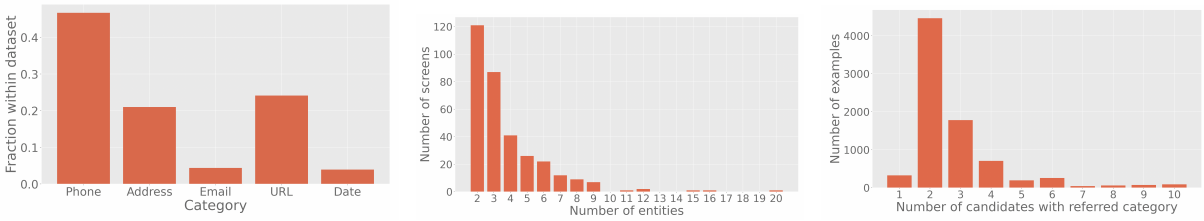
The quality and efficiency concerns led us to develop a new protocol in the form of descriptive and category level data collections. Within descriptive collection, we use a similar screen based collection technique. However we restrict to screens with more than one instance of a category in order to collect challenging and diverse requests (similar to visual grounding datasets like RefCOCO). Alongside the target entity, we highlight all entities of that category to reduce chances of erroneous ambiguous requests. Within category level collection, we do not use screens and the focus is on unique diverse requests with simple references. This split addresses the issues described above leading to more efficient collection and better quality datasets.

Descriptive Data Collection. Though we aim to support references on all apps on phone, this collection is carried out with web pages due to their varied layouts and ease of access. For a list of top visited web pages, we extract texts by UI scraping and get text categories using data detectors. This is similar to running object detectors in open scenes. In order to get challenging references, we only keep screens with more than one entity from a category (eg. 2 URLs).

One entity on the screen is highlighted as target and users are asked to provide requests for that (Fig. 7b). Guidelines provided in A.2. For quality check, we run a verification to confirm the requests are unambiguous: three independent annotators are shown the screen and the collected request and asked which entity from the screen is the request referring to. Samples where at least 2/3 annotators agree are kept, leading to ∼6% data drop.

Category-level Data Collection. This collection targets *simple references* for a category ("phone number - Call that number", "URL - Open it"). During screen mining, we observe that a lot of the screens have only one entity from a category. In these cases, users may prefer succinct simple references instead of descriptive ones. Note that these references do not use the screen layout. Hence, we design this collection independent of screens. This gives a simpler collection scheme that allows us to scale to new categories and/or locales more quickly, with reduced time and cost.

We show a category and ask annotators to give requests, assuming that entity is on their screen. The collection is carried out on shared spreadsheets, one sheet per category (Fig. 7a) in order to avoid duplicate requests across annotators. Annotators are given automatic instant feedback by COUNT_UNIQUE to encourage variations. Through pilot annotation projects, we recognize several constraints to ensure that the uniqueness is not from spurious modifications, which are also added to the guidelines. Detailed grading guidelines provided in A.1. For verification, 3 independent annotators are shown a request and asked to mark *all* categories it could refer to. This also gives annotated multi-label samples i.e. requests that are category ambiguous: "take me there" could be referring to a *URL* or an *address*. Requests with majority agreement are kept. After the requests are collected, dummy entities, one of each scoped category, are added to each request to form a data sample. In a way, this makes the dataset more complete and challenging than real screens which may include only a subset of the entity categories. (Fig. 4).



(a) Requests across categories. (b) # of entities on screens. (c) # of entities with same category as gt.

Figure 5: Histogram of various factors in the Descriptive Data

5 Models

5.1 Heuristic-based Baseline

This is designed for quick prototyping and development without much training data. We define a set of hand-crafted rules using keywords from a subset of the training data. The rules are applied in sequence:

1. **Phrase-match.** Look for synonyms or verbs or apps in the request that indicate the target category (like 'number', 'call' indicate *phone number*, 'navigate', 'maps' indicate *address*).
2. **Location-match.** Regex match to find positional or ordinal reference in request, sort candidates by coordinates and pick the entity at the mentioned position.
3. **Label-match.** Locate the text on screen that has maximum match to the request using a set of string matching features like word overlap (after removing stopwords). Pick the entity closest to this text.
4. If none of the entities are selected above (like "Share this"), score all entities identically.

5.2 Screen Reference Resolver

We design SRR, a modular attention-based network for resolving references (Fig. 6). Inspired by MattNet (Yu et al., 2018), the model contains 3 modules, each of which use a subset of signals from entities, use soft attention to attend to relevant tokens of the request, and compute relevance scores for each entity with the request. We focus on two key dimensions crucial for deploying in an industrial setup- first, memory footprint; second, reusing the existing components in the pipeline.

	Category-level		Descriptive	
	Train	Test	Train	Test
Total requests	4137	486	7993	1082
Unique requests	4123	486	6520	957
Multilabel	934	126	0	0
Tokens per request	7.78	7.95	7.46	7.65
Tokens per reference	2.09	2.06	4.25	4.31
Screenshots	-	-	336	42

Table 1: Statistics for all requests in ScreenRef

We re-use the request token embeddings generated by the upstream embedder (like Bert (Devlin et al., 2018)) and the text categories recognized by upstream. The embedded request passes through the *weight compute block*, an MLP followed by softmax, that predicts weights for each module. A request like "call the top phone number" could give high weight to location and category modules, while "call the one in Palo Alto" could give higher weight to the text and category modules. Embedded tokens also go to the *module-specific embedder* where soft attention is applied on the token embeddings to get embeddings independently for the category and location modules. For "call the top phone number", category module could attend more to 'call' and 'phone number', while location to 'top'. Modules produce scores by fusing entity features with these embeddings. Module scores are combined using the module weights to get the final score for each entity. Specifically, the final score is $w_{cat} \times s_{cat} + w_{loc} \times s_{loc} + w_{text} \times s_{text}$. Let us understand the three modules.

Category module. Entity categories (phone number, URL etc) are embedded using the same embedder as the request. These are pre-computed for all categories. During runtime, given an entity and a request, the embedding for the entity category is loaded and matched with the request embedding from the *module specific embedder*. Both are passed through separate MLP blocks, followed by an inner-product to compute the matching score.

Location module. This takes in bounding boxes of the entity and of other entities of the same category (similar to (Yu et al., 2018)). Bounding boxes of entities $[x, y, w, h]$ are normalized by $K = max(I_{width}, I_{height})$, preserving the aspect

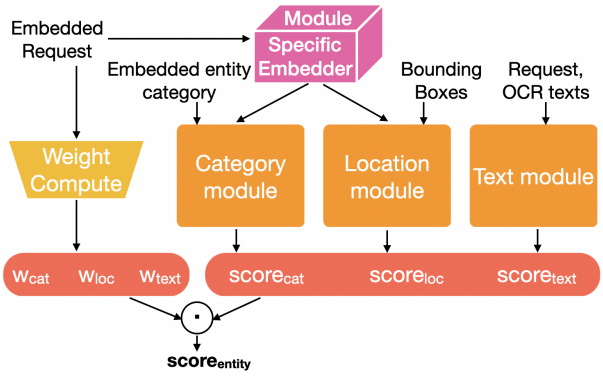


Figure 6: Architecture for the Screen Reference Resolver. It uses the embedded request, embedded entity category, location features and text matching features to predict a matching score for the entity and the request.

ratio and featurised as:

$$\left[\frac{x}{K}, \frac{y}{K}, \frac{x+w}{K}, \frac{y+h}{K}, \frac{w*h}{K^2}\right]$$

These features are concatenated and passed through an MLP. Embedding from the *module specific embedder* is passed through a separate MLP, and lastly an inner product gives the module score.

Text module. We do not embed the screen texts but instead use string matching features. This choice is made for three main reasons. First, we observe in the user study users typically use the text and not synonyms of the text present on screen when making references. Second, our entities of interest, like numbers, emails, URLs make little sense to embed due to their content and presence of OOVs. Third, screens can have a large number of texts. Embedding so many texts in run time could cause compute overhead. Hence, instead of embedding, we utilize the texts by designing simple features like: is the text fully contained in the request, word overlap after removing stopwords, digit overlap. Along with matching the request to the entity text, we also match with the entity's neighboring texts (sorted by distance). All features are concatenated and passed through an MLP to get the module score.

Since the Category-level data has multi-label instances (eg. 'take me there' could refer to a URL or an address), we use a threshold (a hyperparameter obtained from fine-tuning on val data as 0.7) to get final predictions. We add intermediate supervision on the module weights by annotating ~500 samples each for ordinal references (labelled for high weight to location module), references using visible text (text module), and simple reference (category module).

Modular nature offers memory efficiency, giving an option to skip running some modules which get very low weights for a request. It also provides flexibility for varied reference resolution use cases, eg. scenarios with only entity categories available. Note that SRR is only 1MB in size, does not need access to DOM or view hierarchy and hence can work on any screen, in fact any context where recognized texts are available, including documents.

6 Results

Experimental Setup. Data is split into train/val/test in 80/10/10 ratio. To avoid data leak in Descriptive data, we split the data by screens, thus all requests for a screen are in one set. Table 1 summarizes the overall statistics for the dataset. We randomly pick a negative sample for each positive sample and use binary cross entropy loss and Adam optimizer with an initial learning rate of 4×10^{-4}.

Metrics. We use two metrics to measure performance. First, `exact match accuracy` indicates whether the predicted entities, after applying the threshold, exactly match the true entities (if an additional entity crosses the threshold or one of the true entities doesn't, exact match is 0). Second, `top-1 error` indicates whether the entity with the highest score, regardless of threshold, is in the ground truth entities. This is useful as often only the top prediction is used by downstream.

Results. We summarize the results in Table 2.

Dataset	Model	Top-1 Err.	EM
Category-level	Heuristic	6.5	87.5
	SRR	**1.1**	**89.9**
	Cat. Oracle	0.0	100
	No text Oracle	0.0	100
Descriptive	Heuristic	25.0	74.2
	SRR	**14.2**	**78.7**
	Cat. Oracle	54.0	0.0
	No text Oracle	32.6	45.5

Table 2: Top-1 Error and Exact Match accuracy of various systems on ScreenRef. SRR reduces the relative top-1 error by 83% on category-level data and 43% on Descriptive data compared to the heuristic baseline. Category Oracle predicts all entities of the true category. Exact Match going from 100 to 0 and top-1 error from 0 to 55 between the two subsets shows how they differ by design. No text Oracle knows all simple and ordinal references but not the text values.

Modules in SRR			
Category	Location	Text	Top-1 Error
✓	✓	✓	**14.2**
×	✓	✓	31.2
✓	✓	×	33.7
✓	×	✓	35.3
×	✓	×	49.7
×	×	✓	51.5
✓	×	×	54.9

Table 3: Ablation Study results for the different modules in SRR namely category, location and text modules. Top-1 Error on the Descriptive data is reported. The observed loss in performance across all subsets underscores that all modules are critical for achieving high performance.

Observe that the performance on Category data is higher than on the Descriptive data, indicating the challenging nature of the latter. SRR reduces the relative top-1 error by 43% on Descriptive and 83% on Category-level data compared to the baseline. The oracles know the true category hence get perfect results on category-level data. Their low performance on Descriptive reflects the importance of all inputs, particularly screen texts. We carry out an ablation study on the model (Fig. 3). It shows that each attribute and thus each module is critical in understanding the references.

7 Conclusion

We explore a new user experience of executing actions on screen elements with Voice Assistants. To make interactions more natural, we explore the use of references. An important decision was what UI elements to support. We decided to use texts that are most commonly used for task oriented dialogue and, commonly present on phone screens and easy to classify. We collected a dataset of requests and proposed solutions to understand references. This is a step towards making Voice Assistants more context aware, but there is a lot more context. We hope that our work will motivate further research towards this goal, and towards semantic visual text referencing.

Limitations

Our work explores a dimension of context understanding by Voice Assistants but it is only a small step. Firstly, we only consider 5 categories, while screens have a myriad of other texts and visual content. We do not include image context into our reference understanding models. But users could use them when formulating references to texts near them. Using image captions or some pixels would improve coverage. Our system leverages entities extracted by upstream and hence is bounded by the performance of that. Also our model evaluates each entity separately while there may be benefit in considering the entire screen holistically.

Ethics Statement

This work aims at improving user experiences with voice assistants. By allowing users to refer to entities on screen, it reduces user friction and enables a smoother and more natural experience. No voice assistant usage log data was used and all requests were collected by recruited annotators.

Acknowledgements

We would like to thank Hadas, Lucia and Kyanh for their help with the data annotations, revisions and data quality check; Sachin and Dhivya for help with data planning; Melis and Junhan for support in modelling experiments; as well as Lin and Murat for general direction.

References

Pollfish. https://www.pollfish.com.

Ali Furkan Biten, Ruben Tito, Andres Mafla, Lluis Gomez, Marçal Rusiñol, Ernest Valveny, C. V. Jawahar, and Dimosthenis Karatzas. 2019. Scene text visual question answering.

Richard A. Bolt. 1980. "put-that-there": Voice and gesture at the graphics interface. In *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '80, page 262–270, New York, NY, USA. Association for Computing Machinery.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Heiko Drewes, Alexander De Luca, and Albrecht Schmidt. 2007. Eye-gaze interaction for mobile phones. In *Proceedings of the 4th International Conference on Mobile Technology, Applications, and Systems and the 1st International Symposium on Computer Human Interaction in Mobile Technology*, Mobility '07, page 364–371, New York, NY, USA. Association for Computing Machinery.

Yu-Chung Hsiao, Fedir Zubach, Maria Wang, et al. 2022. Screenqa: Large-scale question-answer pairs over mobile app screenshots. *arXiv preprint arXiv:2209.08199*.

Thomas E Hutchinson, K Preston White, Worthy N Martin, Kelly C Reichert, and Lisa A Frey. 1989. Human-computer interaction using eye-gaze input. *IEEE Transactions on systems, man, and cybernetics*, 19(6):1527–1534.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. 2014. Referitgame: Referring to objects in photographs of natural scenes. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 787–798.

Tae Soo Kim, DaEun Choi, Yoonseo Choi, and Juho Kim. 2022. Stylette: Styling the web with natural language. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22, New York, NY, USA. Association for Computing Machinery.

Gierad P Laput, Mira Dontcheva, Gregg Wilensky, Walter Chang, Aseem Agarwala, Jason Linder, and Eytan Adar. 2013. Pixeltone: A multimodal interface for image editing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2185–2194.

Gang Li and Yang Li. 2022. Spotlight: Mobile ui understanding using vision-language models with a focus. *arXiv preprint arXiv:2209.14927*.

Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldridge. 2020. Mapping natural language instructions to mobile UI action sequences. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8198–8210, Online. Association for Computational Linguistics.

Yang Li, Gang Li, Xin Zhou, Mostafa Dehghani, and Alexey Gritsenko. 2021. Vut: Versatile ui transformer for multi-modal multi-task user interface modeling. *arXiv preprint arXiv:2112.05692*.

Alice Ljungholm. Voice interaction vs screen interaction when controlling your music-system. In *CONFERENCE IN INTERACTION TECHNOLOGY AND DESIGN*, page 103.

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.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L Yuille, and Kevin Murphy. 2016. Generation and comprehension of unambiguous object descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 11–20.

Diako Mardanbegi and Dan Witzner Hansen. 2011. Mobile gaze-based screen interaction in 3d environments. In *Proceedings of the 1st conference on novel gaze-controlled applications*, pages 1–4.

Sven Mayer, Gierad Laput, and Chris Harrison. 2020. Enhancing mobile voice assistants with worldgaze. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–10, New York, NY, USA. Association for Computing Machinery.

Panupong Pasupat, Tian-Shun Jiang, Evan Liu, Kelvin Guu, and Percy Liang. 2018. Mapping natural language commands to web elements. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4970–4976, Brussels, Belgium. Association for Computational Linguistics.

Xuejian Rong, Chucai Yi, and Yingli Tian. 2017. Unambiguous text localization and retrieval for cluttered scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5494–5502.

Xuejian Rong, Chucai Yi, and Yingli Tian. 2019. Unambiguous scene text segmentation with referring expression comprehension. *IEEE Transactions on Image Processing*, 29:591–601.

Julia Rozanova, Deborah Ferreira, Krishna Dubba, Weiwei Cheng, Dell Zhang, and Andre Freitas. 2021. Grounding natural language instructions: Can large language models capture spatial information? *arXiv preprint arXiv:2109.08634*.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. 2019. Towards vqa models that can read. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8317–8326.

Alexandra Vtyurina, Adam Fourney, Meredith Ringel Morris, Leah Findlater, and Ryen W. White. 2019. Bridging screen readers and voice assistants for enhanced eyes-free web search. In *The World Wide Web Conference*, WWW '19, page 3590–3594, New York, NY, USA. Association for Computing Machinery.

Bryan Wang, Gang Li, and Yang Li. 2022. Enabling conversational interaction with mobile ui using large language models. *arXiv preprint arXiv:2209.08655*.

Nancy Xu, Sam Masling, Michael Du, Giovanni Campagna, Larry Heck, James Landay, and Monica Lam. 2021. Grounding open-domain instructions to automate web support tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1022–1032, Online. Association for Computational Linguistics.

Licheng Yu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, Mohit Bansal, and Tamara L Berg. 2018. Mattnet: Modular attention network for referring expression comprehension. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1307–1315.

Xiaoyi Zhang, Lilian de Greef, Amanda Swearngin, Samuel White, Kyle Murray, Lisa Yu, Qi Shan, Jeffrey Nichols, Jason Wu, Chris Fleizach, Aaron Everitt, and Jeffrey P Bigham. 2021. Screen recognition: Creating accessibility metadata for mobile applications from pixels. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, New York, NY, USA. Association for Computing Machinery.

A Appendix

A.1 Annotation Guidelines for Category Level Request Collection

In this project, you will be shown an entity category (phone number, url etc). Assume you see a particular instance of that entity on your screen. You have to come up with various requests you would say to a Voice Assistant to perform action on that. The main idea is to provide varied natural ways of interacting with that entity. The request should be one which holds valid when looking at different kinds of images containing that entity.

Consider you see that entity on the screen. **Do not** assume any other information about that entity, like what digits occur in the number or what place is the address for (note to readers - such references are the focus of the unambiguous request collection, hence skipped here).

- Take me to the California address - Incorrect
- Call the number ending in 99 - Incorrect
- Take me to COUNTRYNAME address - Incorrect
- Call COMPANYNAME number - Incorrect

1. You are encouraged to use varied request formulations with different ways of referring to the entity as well as carrying out different actions on that entity. Example - Phone number
    - place a call to that phone number
    - dial this number
    - add this to my contacts
    - remind me to call here at 5
    - send this to PERSON on text message
2. Constraints to follow -
    (a) Enter requests only in column 1 and do not change the values in column 2 in any way.
    (b) The number in the second column reflects the number of unique requests so far.
        i. When you enter a request and the number does not increase, this means that request is already present. CHANGE the request.
        ii. When you enter a request and the number in column 2 increases by 1, you have successfully entered a unique request. Move to the next row.
    (c) Do not make minor irrelevant variations. Replace proper nouns with uppercase tags like "PERSON", "COMPANYNAME", "DAY".
        i. Incorrect -
            - send this to Mom on text message
            - send this to Dad on text message
            - send this to John on text message
        ii. Correct -
            - send this to PERSON on text message
            - share the number with PERSON
    (d) Use **only lowercase letters** in the request, apart from the proper noun replacements with all uppercase tags (PERSON, COMPANYNAME etc). Use these only in a way that one can replace them with any name without knowing the actual screen. The request should hold valid for a variety of different screens containing phone numbers.
        i. Incorrect - Send this to PERSONNAME on text message - first letter should be small
        ii. Incorrect - copy PERSONNAME's number - assumes you see PERSONNAME
        iii. Correct - send this to PERSONNAME on text message
        iv. Correct - send PERSONNAME'S number to this number - here PERSONNAME can be any person in your contacts.

(e) No fullstops after the request. 1. Incorrect - call this. 2. Correct - call this

(f) No trailing or leading whitespaces should be added.

(g) Assume you see the entity in front of you. Target the request to ask a VA to *act on the entity type mentioned*. Do not just add the entity type in the request randomly. Do not assume anything more about what you see. Invalid requests -

- tell me the number that just called me - the request is not about a phone number you are seeing - "did I just receive a call from that number" is a valid request
- is PERSON's number in my missed calls - the request is not about a number you are seeing - "is this number in my missed calls" is valid
- get rid of COMPANYNAME's number - You may not be seeing a company name- get rid of their number is valid

(h) Use varied ways to refer to the entities. For instance, for 'phone number', You can use generic references like "this", "that" as well as phrases including "phone number", "contact number" etc.

- call that
- call this contact number
- call them
- call this number

(i) You need not explicitly use the phrase mentioning the entity type always, specially if the intent conveys that. Example - Email address

- draft a mail to this
- draft them a mail

(j) Use *varied ways of referring* to the entity

- generic phrases - this/that/it/them/.... etc
- specific phrases - email/email address/address/contact/... etc

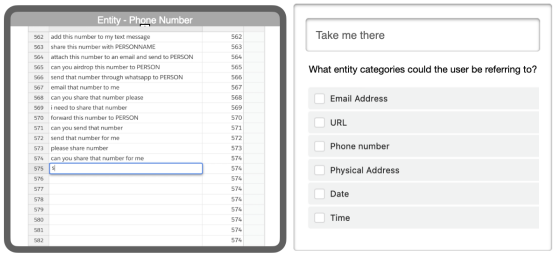
A.2 Annotation Guidelines for Unambiguous Request Collection

Overview The goal of this task is to generate a variety of requests for text in a screen. The requests should be queries or requests you would make to a voice assistant, based on the text. You will be shown a screen with a green bounding box around specific text. You will need to:

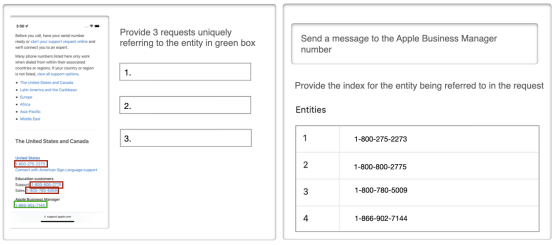
> Write three uniquely referential requests about the marked text for a voice assistant

A.2.1 Green vs. Red Boxes

Screens will contain green and red boxes. The green box contains the text for which you need to write the requests. The requests for the text within the green box need to uniquely identify it. Red boxes mark the texts that are similar to the text within the green box. For example, if an image has three phone numbers, the red box will capture the other two phone numbers. Do not write requests for the text within the red box. They are intended to serve as a guidance so that you don't miss them out and ensure you write uniquely referential requests for the text within the green box.



(a) **Category Level Data Collection**: First, annotators are asked to provide category level requests in a spread sheet (Left). Column2 of the sheet reflects the unique count so far, which encourages varied requests for a diverse dataset. We define constraints in the guidelines so that variations are not spurious changes. Second, annotators are asked to verify the collected requests to capture entity level ambiguity (Right). 3 annotators are asked to verify each collected request.



(b) **Unambiguous Data Collection**: First, annotators are shown a screen with multiple instances of a category (Left). One is highlighted in a green box, while others in red boxes as initially annotators tended to provide ambiguous references. An annotator provides 3 different requests with references. Second, the correctness of a request is verified by showing the screen and request (Right). 3 annotators are asked to mark the referred one.

Figure 7: Unambiguous and Category level Data collections protocols.

A.2.2 Request Guidelines

Imagine you are viewing that screen on your phone, and were to ask a voice assistant about that text you came across. What would you ask the voice assistant regarding the text that you could not gather just from looking at it? What additional actions or requests would you ask the voice assistant to execute in relation to the text that can be carried out on your mobile device?

Keep in mind the following:

- Unique: Each request will require a referring expression that uniquely identifies the detected text.
- Require a voice assistant's help: Requests should not be questions that a user can answer simply by looking at the text. Example: "Does this phone number contain 007 at the end" is invalid.
- Mix it up: Requests can be a mix of questions about the text or action commands to be executed on the text.
- Sound natural: Come up with requests that would sound natural, coming from a user. Verbally say the request out loud to ensure it sounds natural and not too long.
- Make sense for a user to request a VA: Think about whether the request would make sense for a user to request, based upon the text type/context of the screen, and what a user would usually do on a device with that information.

Uniquely Referential Use references that ensure the request uniquely identifies the marked text. All 3 requests for a particular text need to be uniquely referential. Use varied actions and request types. Do not use the same reference across the requests. Remember that the request needs to be uniquely referential, not just with other similar texts marked in red, but also with all content within the screen. Example **errors**:

- Too General:
  1. Call that
  2. Text it to John
  3. Save that to my notes
- Same references for all 3 requests:
  1. Call the third number
  2. Share the third number
  3. Copy the third number to my notes