Interactive Mobile App Navigation with Uncertain or Under-specified Natural Language Commands

Andrea Burns 1  Deniz Arsan 2  Sanjna Agrawal 1  Ranjitha Kumar 2
Kate Saenko 1,3  Bryan A. Plummer 1

1Boston University, MA  2University of Illinois at Urbana-Champaign, IL
3MIT-IBM Watson AI Lab, MA
{aburns4, sanjna, saenko, bplum}@bu.edu

Abstract

We introduce Mobile app Tasks with Iterative Feedback (MoTIF), a new dataset where the goal is to complete a natural language query in a mobile app. Current datasets for related tasks in interactive question answering, visual common sense reasoning, and question-answer plausibility prediction do not support research in resolving ambiguous natural language requests or operating in diverse digital domains. As a result, they fail to capture complexities of real question answering or interactive tasks. In contrast, MoTIF contains natural language requests that are not satisfiable, the first such work to investigate this issue for interactive vision-language tasks. MoTIF also contains follow up questions for ambiguous queries to enable research on task uncertainty resolution. We introduce task feasibility prediction and propose an initial model which obtains an F1 score of 61.1. We next benchmark task automation with our dataset and find adaptations of prior work perform poorly due to our realistic language requests, obtaining an accuracy of only 20.2% when mapping commands to grounded actions. We analyze performance and gain insight for future work that may bridge the gap between current model ability and what is needed for successful use in application.

1. Introduction

Vision-language tasks often require high-level reasoning skills like counting, comparison, and common sense knowledge to relate visual and language data [6,10,11,13,34]. The goal of these tasks has been to employ reliable and robust vision-language reasoning, but prior work has failed to create AI agents that can interact with humans naturally and handle realistic use cases. For example, vision-language models fail to recognize when a visual question cannot be answered given the scene being viewed. Instead, these models provide visually unrelated, yet plausible answers, like answering ‘white’ to the question ‘what color is the remote control?’ when no remote is present in the input image [23].

This is particularly dangerous for users that are limited in their ability to determine if an answer is trustworthy, either physically or situationally, e.g., users that are low-vision or driving. While prior work has explored question relevance for text-only [10] and visual question answering [27], they have focused on the extreme case of language being completely unrelated to the visual input it is paired with. VizWiz [13] introduced a visual question answering dataset for images taken by people that are blind, resulting in questions which may not be answerable from the captured im-
Table 1. Comparison of MoTIF to existing datasets. For language comparison, we consider the number of natural language commands, command granularity, existence of feasibility annotations, and if there is user interaction or dialogue. For visual comparison, we consider the number of environments and views (we report the median number of views, i.e., the number of visited states during a demonstration), if the rendered screen is captured (i.e., app or website screenshot), and what type of actions it contains (click, swipe, or type).

The number of unanswerable train samples had at least one annotator which deemed it unanswerable due to distinct visual failures like image blur or flash. These works have been necessary to study particular or more obvious cases of infeasible language requests. However, they do not include natural language ambiguity or infeasibility that arises due to lack of familiarity with a new environment.

In addition to the dearth of research on more nuanced task feasibility, all prior work studies static image-text pairs, and not interactive visual environments where an AI agent may require navigation and state change to complete a request. Vision language navigation (VLN) datasets and models have not tackled such problems, and often require step-by-step user instruction (i.e., low task granularity) [3, 29].

To study nuanced task feasibility in interactive visual environments and capture a high impact real world use case, we study natural language requests for mobile apps. Automating mobile app tasks and capturing realistic task feasibility is a step toward empowering users of all ability levels to engage with mobile apps with ease. Current assistive technology lacks agency and flexibility [33], as screen readers are primarily used for browsing and information consumption, while interactive tools like virtual assistants (e.g., Siri, Alexa) have limited, structured commands.

We propose Mobile App Tasks with Iterative Feedback (MoTIF), the largest dataset designed to support interactive methods for completing natural language tasks in mobile apps. MoTIF has 4.7k samples thus far for apps across fifteen Android categories. We acknowledge MoTIF is not on the scale of huge pretraining datasets used for other vision-language tasks (e.g., Alt-Text [16], JFT-300M [30]), as it is very expensive and time consuming to collect. MoTIF is nonetheless useful to the research community as it is extremely realistic and can be used to evaluate how existing methods actually solve language-based digital tasks.

MoTIF will also allow for new research toward predicting task ambiguity that would arise in real use cases of these AI systems. Our infeasible tasks are always relevant to the Android app category they are paired with, making them challenging due to their inherent relevance to the visual environment. We propose a baseline model for task feasibility prediction and confirm exploration is necessary, which is notably different from prior work like VizWiz [13]. Surprisingly, we find that prior representation learning approaches specific to the mobile app domain do not result in the best performance for our feasibility classifier. We also evaluate off the shelf and trained from scratch models for automating MoTIF’s app tasks and examine performance failures.

MoTIF subsumes several datasets and topics: web task automation [12, 17, 20, 24, 28], VLN [3, 6, 11], task relevance prediction [10, 13, 27], app design [7, 8, 21]; see Table 1 for comparison. Prior work in automating web tasks [24, 28] limit user interaction to a single screen, unlike MoTIF which contains task demos with a median of 8 visited screens (# views in Table 1). PIXELHELP [19] was recently introduced as a small mobile app dataset, but most commands are device specific. i.e., the commands refer to the phone itself, such as ‘in the top control menu click the battery saver,’ and are not in-app commands like those in Figure 1. This is a key distinction, as PIXELHELP more closely mimics already existing virtual assistant abilities. MoTIF also contains clicking, typing and swiping actions, whereas PIXELHELP only contains clicking.

MoTIF provides greater linguistic complexity for interactive tasks with over 6.1k free form natural language commands across 125 Android apps. A sample includes the natural language command (i.e., task), app view hierarchy, app screen, and action coordinates for each time step, as shown in Figure 1. MoTIF uniquely includes binary feasibility annotations for each task, subclass annotations for why tasks are infeasible, and follow up questions. We present results for the portion of MoTIF collected thus far\(^1\), which is sufficient to understand pitfalls of prior work.

We summarize our contributions below:

\(^1\)Code and data are released for public use [https://github.com/aburns4/MoTIF](https://github.com/aburns4/MoTIF)
We introduce Mobile app Tasks with Iterative Feedback (MoTIF), a new dataset with free form natural language commands for interactive goals in mobile apps. It has natural language tasks for the most visual environments to date and is the only to include the app screen in its annotations. MoTIF uniquely has multiple types of interactions including clicking, swiping and typing actions, and a subset of infeasible tasks.

A new vision-language task: mobile app task feasibility classification, along with subclass annotations on why tasks are infeasible and follow up questions for research toward resolving task uncertainty via dialogue.

Benchmarks for feasibility classification and task automation with MoTIF. A thorough feature exploration is performed to understand the role of vision and language in the mobile app domain. We compare several methods on mobile app task automation and provide a thorough analysis with insights for future work.

2. MoTIF Dataset

For a mobile app task dataset, we need natural language tasks for apps and their corresponding demonstration. Figure 1 illustrates MoTIF tasks like ‘Open settings and change temperature unit to C.’ For each command, we collect expert demos of attempts to complete the request. At each time step we capture the app screen, the app backend view hierarchy, what type of action is taken, and where the action occurred. The rendered app screen can be useful features for downstream tasks, and reduce reliance on access to the app backend. Demonstration sequences can be used to learn to predict a set of grounded actions to complete a task.

2.1. Data Collection

We select 125 apps for MoTIF across fifteen Google Play Store categories (see the Supplementary for a complete list). Ten apps with (1) at least 50k downloads and (2) a rating higher than or equal to 4.0 out of a five point scale were chosen for each category. Section 2.1.1 contains the process by which we collect natural language commands. Section 2.1.2 discusses how we pair commands written for one mobile app with other applications where the task may be relevant, to ensure some task uncertainty in our dataset. Lastly, Section 2.1.3 details how we collect interactive user sessions with our annotators to either demonstrate the task, to specify that it is not feasible, or to indicate it is ambiguous and ask a clarifying question. The Supplementary includes annotator demographics, payment, and collection interface details.

2.1.1 Natural Language Commands

To collect natural language tasks, we instruct workers to write commands as if they are asking the app to perform the task for them. Annotators can explore the app before deciding on their list of tasks. We ask them to write functional or navigational tasks, and not request something requiring text comprehension such as summarizing an article. We do not structure the written tasks or require a template when workers submit their list of commands. This results in natural language queries that mimic real users, unlike automatically generated tasks from prior work, which all hold the same templated form (e.g., MiniWoB [28] creates versions of ‘Click on x’ by varying x). We also do not prescribe a specific number of tasks to be written for each app.

2.1.2 Task-Application Pairing

When collecting natural language tasks, annotators write a list of tasks for an app they can explore first. Once we have tasks for every app, we introduce additional task uncertainty for the demonstration stage by collecting demos for both the original (app, task) list, as well as tasks paired with apps they were not originally written for. We create new (app,
Specifically, we cluster the tasks within an Android app category to create reasonable uncertainty, e.g., cluster tasks written for music and audio apps. Clustering is performed using the mean FastText embedding [5] of the language command. We begin by collecting demos for five tasks per app, and consequently perform K-Means with $K = 5$. Figure 3 shows an example of a category clustered task which was deemed infeasible by annotators. The task ‘Open settings and clear search history’ was paired with the music app Spotify, which is a sensible request given that the primary function of the app is to search for music. However, no search history can be found under settings, only the option to ‘delete cache,’ and follow up questions are asked in response to the original request.

The clusters are visualized with T-SNE [31] and manually inspected (see Supplementary for the visualizations). We color the data points by app class instead of cluster; this reveals if an app has natural language tasks that are significantly different from others within the same category. If an app’s tasks are isolated from other clusters, we retain app-specific pairings, i.e., the (app, task) pairs for tasks specifically written for the given app. This resulted in 40 apps being paired only with their app-specific tasks. If two apps’ tasks appear closely clustered, we group them and manually select five to be used for both. A total of 17 apps’ tasks were gathered in this way. This creates a middle ground between the app-specific tasks and the category-clustered tasks.

2.1.3 Task Demos and Feasibility Annotations

Once the natural language tasks are paired with apps, we instruct annotators to demonstrate the task in the given app. We provide a website interface connected to physical Android phones for crowd workers to interact with, as well as anonymized login credentials so that no personally identifiable information is collected. They are instructed to record their demonstration after they have logged in (we consider logging in a separate task). After attempting to complete the task, they are brought to a post-survey where they provide details on whether or not the task was successfully completed. We therefore have demonstrations of actions taken both in successful and unsuccessful episodes, which may provide interesting insight toward how to reason about whether a task is or is not feasible, and why.

2.2. Dataset Analysis

2.2.1 Natural Language Commands

We collected over 6.1k natural language tasks across 125 Android apps. The vocabulary size was 3,763 after removing non-alphanumeric characters, or 3,658 after stop words are removed. The average number of tasks submitted per app is 56, with average length being 5.6 words. The minimum task length is one, consisting of single action tasks like ‘refresh’ or ‘login,’ with the longest being 44 words. Average task length does not vary significantly by category, with a range of 1.5 words. Word cloud visualizations and additional examples and statistics are in the Supplementary.

2.2.2 Feasibility Annotations

We start by collecting five tasks per app, and gather at least five demos per (app, task) pair. We collect several demos of each (app, task) pair to reach a majority feasibility label and to capture different attempts of the same task, as some tasks can be completed in multiple ways. Of the resulting 490 tasks, 143 are deemed infeasible by at least five crowd
workers. Annotators have above 80% feasibility agreement for over 75% of the (app, task) pairs; agreement is defined as the ratio of the majority voted feasibility label over all votes. See the Supplementary for an agreement histogram.

However, the tasks considered infeasible do not always correlate to mismatched (app, task) pairs, i.e., some app-specific tasks are deemed infeasible during demonstration. This confirms the need to capture task uncertainty, as someone familiar with an app can still pose requests that are either not possible, ambiguous, or app state dependent. Of the infeasible tasks, 23 (16.8%) are from app-specific pairs. E.g., the request to ‘Click shuttle and station’ originally written for the Nasa app was labeled infeasible because the app has changing interactive features. Thus app changes and dynamic features also motivate studying infeasible requests, as a task that was once feasible may not always be.

Table 2 provides feasibility statistics: the demos row shows the number of task demos and the F/U Qs row shows the number of follow up questions per feasibility category. There are three options for annotators to choose from: the action cannot be completed in the app, the action is unclear or under-specified, and the task seems to be possible, but they cannot figure out how to perform it or other tasks need to be completed first. These map to Table 2’s irrelevant (I), unclear (U), and premature (P) columns. If a crowd worker cannot complete the task, they are prompted to ask a follow up question. We instruct them to write the question(s) such that if they had the answer, they may now be able to complete the original action or perform an alternative task for the user. We collect these annotations to enable AI agents to respond with actionable feedback for requests that are classified as infeasible instead of failing in unexpected ways.

2.2.3 Task Demonstrations

The average time spent performing a task demo is 60.6 seconds. The average time spent between app categories varies at most by one minute. The number of views visited during a demo can be considered the number of screens visited or the number of actions taken to complete the task. The median number of views visited over all tasks is 8. If we further break this down by feasibility, we have a median of 7 and 14 views for feasible and infeasible tasks, respectively. This is reasonable, as annotators may explore more to confidently determine if a task is actually not possible.

3. Task Feasibility

We create a pipeline to reflect real world use of mobile app task automation in which we first reason about the input request. We train a feasibility classifier over the train split of MoTIF. After predicting feasibility on MoTIF’s test set, task automation is performed for the subset of tasks predicted feasible by our classifier (see Section 4). We describe the

| # | Feasible | Infeasible | Total |
|---|---------|-----------|-------|
| Demos | 3,337 | 911 | 159 | 300 | 4,707 |
| F/U Qs | 93 | 253 | 136 | 164 | 646 |

Table 2. Task demo breakdown for task feasibility and follow up questions. Annotator can state the action cannot be completed in the app (I), the action is unclear or under-specified (U), or the task seems to be possible, but they cannot figure out how to perform it or other tasks need to be completed first (P).

3.1. Experiments

Given the app states visited during a demonstration and its associated natural language command, the purpose of task feasibility prediction is to classify if the command can be completed or not. To determine feasibility, we expect a model needs to learn the most relevant state for the requested task and whether the functionality needed to complete it is present. Natural language reasoning about what requests result in the same task is also needed, as there are many ways to ask for the same thing, and users may request something synonymous with text present in the application (as opposed to an exact match). Our results provide an initial upper bound on performance, as input task demos can be considered the ground truth exploration needed to determine feasibility, as opposed to a learned agent’s exploration.

We propose a Multi-Layer Perceptron (MLP) baseline with two hidden layers that outputs a binary feasibility prediction. Each MLP is trained for 50 epochs with cross entropy using Stochastic Gradient Descent with a learning rate of 1e-2. The natural language command is always input to the classifier, and we ablate additional input features (app view hierarchy, app screen, or both) and how to aggregate the demonstration sequence (average, concatenation, or LSTM [15]). Because prior work in mobile app and website tasks has consistently neglected visual input, we intentionally investigate using language or visual only inputs to understand what features are most useful for our task.

We encode the task command and view hierarchy elements per step with mean pooled features. Specifically, we try FastText [4] and CLIP [26] (Visual Transformer [9]). The view hierarchy contains components such as element text (ET), IDs (ID) and class labels (CLS); we use the best combination in Table 3, ablations are in the Supplementary. We also try Screen2Vec [18], a semantic embedding of the view hierarchy, which represents the view hierarchy with a GUI, text, and layout embedder. To obtain visual features of the app screen, we extract ResNet152 [14] features for

5
the standard ten crops of each app image and CLIP representations of each app image. We also include icon features obtained from the embedding layer of a CNN trained for the downstream task of icon classification by Liu et al. [21].

For feasibility classification, we report the average F1 score over ten randomized runs, with ‘infeasible’ as the positive class, as we care more about correctly classifying tasks that are infeasible, than misclassifying feasible tasks.

3.2. Results

Our best task feasibility classifier (Table 3(c)) achieves an F1 score of 61.1 when CLIP embeds the task, view hierarchy, and app screen. This is still fairly low, and feature ablations demonstrate room to improve both the language and visual representations. While CLIP has shown significant performance gains in other vision-language tasks, it is somewhat surprising that domain-specific embeddings (e.g., Screen2Vec) are not as competitive. The combination of view hierarchy and app screen features does not largely outperform the app screen CLIP results (and does worse with LSTM aggregation), suggesting a need for a better vision-language encoding which can pull features together from different modalities such as the view hierarchy.

Table 3(a) compares methods of encoding the view hierarchy. Using CLIP for view hierarchy elements results in notably better performance than FastText, albeit less significant when input demos are aggregated with an LSTM. Our final view hierarchy embedding comes from Screen2Vec. Despite being trained on mobile app data, it performs worse than CLIP and on par with FastText. Screen2Vec may not capture enough low level view hierarchy information to predict feasibility, and methods trained on huge data, even if from another domain, are more powerful.

In Table 3(b) we ablate over visual features of the app screen. While icon representations are trained on images from the same domain as MoTIF, they are significantly less effective than ResNet and CLIP. The F1 score nearly drops to zero when the average icon feature is used, illustrating that the average icon does not carry useful information for feasibility classification. Icon features may be too low-level or require improved aggregation methods.

When comparing demonstration aggregation methods (averaging vs. concatenating vs. LSTM), there is a trend that concatenating time steps is the best method, suggesting a sequential representation of demonstration time steps is needed. However, when the best representations for the view hierarchy and app screen are combined in Table 3(c), averaging manages to outperform the LSTM performance.

In future work we hope to learn hierarchical representations in order to encode global information such as that of Screen2Vec as well as local information from icon embeddings. Taking advantage of the tree structure from the view hierarchy via Transformers or Graph Neural Networks may help learn structured app features. Additionally, all current approaches do not take into account any notion of app ‘affordance,’ i.e., which UI elements are most actionable.

4. Task Automation

Given an app and a natural language command, our goal is to interact with the app and output the sequence of commands that would complete the task. We provide task automation results on two versions of MoTIF to better understand its challenges. We evaluate the test set’s tasks and their full demo sequences (Table 4) as well as a simplified subset, MoTIF-S, which ‘shortcuts’ tasks to a single action (Table 5). We sliced samples to the final action and manually verified which tasks were reasonable to do this with.

4.1. Experiments

**Seq2Seq** [29] We first adapt Seq2Seq to train and evaluate MoTIF; Seq2Seq was proposed for ALFRED, a dataset for actionable natural language requests in home environments. The VLN setting is challenging as there is actual interaction with an environment at test time, and an incorrect prediction at time step \( t \) will change the rest of the exploration (unlike Seq2Act which we describe next, which evaluates prediction on the ground truth sequence). Seq2Seq predicts an action class and location at each time step using the attended language task, visual features of the current state, the last predicted action, and the hidden state of an LSTM which takes the former three as input. The test-time environment and model changes are described in the Supplementary.

**Seq2Act** [19] Seq2Act is used as a baseline to evaluate both MoTIF-S and the full sequence MoTIF test sets. Seq2Act approaches mobile app tasks in two stages: (action, phrase) tuple extraction and action grounding. It trains a Transformer [32] for each stage. The first predicts a substring,
Table 4. Mobile app task automation on MoTIF. We evaluate the Seq2Seq navigation model trained from scratch and the double transformer grounding model Seq2Act off the shelf.

| Model       | Action | Ground | Action + Ground |
|-------------|--------|--------|-----------------|
| Seq2Seq [29] |        |        |                 |
| Complete    | 22.3   | 0.6    | 0.6             |
| Partial     | 66.6   | 3.7    | 3.7             |
| Seq2Act [19] |        |        |                 |
| Complete    | 56.5   | 30.1   | 20.2            |
| Partial     | 59.6   | 34.5   | 24.6            |

i.e., span, of the input task and an associated action. The second grounds the predicted (action, phrase) tuple amongst the actionable UI elements. We describe the model and input features used in greater detail in the Supplementary.

MAP [19] We also use the methods by Pasupat et al. to evaluate MoTIF-S, which were trained to map single click commands to elements on website homepages. The first model (MAPE) embeds the command and webpage elements using Glove [25]. The second model (MAPA) builds an alignment matrix between the input command and web elements via multiple convolutions and max pooling. In both, a linear scoring function learns to maximize the log likelihood of the correct website element given the concatenated embeddings. Model variants with N (e.g., MAPEN) refer to neighbor web elements being included as additional input.

Metrics We report accuracy for MoTIF-S, which is classification accuracy over the UI elements. Table 5 also compares when the full, original task language was input vs. the shortcut version which slices the language to the final task. E.g., for the original task ‘Go to lullabies and select Chants of Tibet’ requested in a sleep cycle app, we can input this full instruction or shortcut it to ‘select Chants of Tibet’ which only requires reasoning at the current time step.

When evaluating the full sequence MoTIF, complete and partial accuracy are reported per [19]: the complete score is 1 if two sequences have the same length and the same prediction at each step, else 0. The partial score is the fraction of predicted steps that match the ground-truth sequence. We report complete and partial accuracy for action prediction (Action), action grounding (Ground), or both action and grounding jointly.

4.2. Results

Seq2Act achieves the highest performance for all metrics except partial sequence action prediction (Table 4 Seq2Seq achieves 66.6% accuracy compared to 59.6% of Seq2Act). Seq2Seq is trained from scratch on MoTIF and performs the much more challenging task of screen localization, where a point in the app screen is predicted at each time step, unlike Seq2Act which predicts over the leaf UI elements. The VLN setup also requires environment interaction and may not recover from a wrong prediction. As Seq2Seq has lower results, we defer its discussion to the Supplementary.

Comparing the columns of Table 4, Seq2Act’s action prediction is much better than its grounding, which is sensible as it only predicts actions over four classes (click, type, swipe, and EOS), much fewer than 28, the average number of leaf UI objects grounded over in MoTIF-S. Compounding errors result in the lowest accuracy when evaluating Action + Ground. Seq2Act was originally evaluated with PIXELHELP [19] and achieved 70.6% complete Action + Ground accuracy. It is clear PIXELHELP’s click-only Gmail, Google Photos, Google Chrome, and device tasks do not reflect the model’s ability on a greater variety of apps or tasks, nor on swiping or typing actions.

We look to further understand what makes MoTIF challenging for Seq2Act with MoTIF-S. While performance is significantly higher, the results in Table 5 demonstrate that MoTIF’s natural language task inputs and mobile app environments are difficult even when simplified to single-action tasks. The MAP models’ transfer performance is notably lower than Seq2Act due to domain shift from websites, unlike Seq2Act which was trained on mobile app data. However, we can still gain insight from MAP results: including spatial context via neighboring elements only helps in some cases (e.g., Table 5 MAPEN 21.1% vs. MAPE 17.4% Full). This provides more evidence that hierarchical vision-language features may help learn context and better capture neighboring elements, as these text-only features fall short.

Looking at misclassifications, we find one culprit to be Seq2Act overly relying on matching input task text to view hierarchy text. In Figure 4, we show Seq2Act’s text matching tendency, which can result in failure. For example, Seq2Act predicts the UI element with the word ‘my’ in it for the input command ‘go to my profile.’ These results, in addition to the high visual performance from the feasibility classifier, verifies the need for visual input to correct model bias to match input task text directly to the view hierarchy.

Both methods perform worse when the natural language input is not sliced to the substring of the final action. Thus, most of the poor performance originates from the (action,
Still, task feasibility classification results are low at only 8.6% of test samples were incorrectly predicted as feasible and have zero accuracy in the task automation portion of our pipeline. If we only consider the samples correctly predicted as feasible, complete Action + Ground performance would be nearly 4% higher (20.2% in Table 4 vs. 16.8% in Table 6). Most of the poor complete pipeline performance originates from the task completion failures.

Table 6. Task completion on MoTIF with the full pipeline. The subset of tasks predicted as feasible are input to Seq2Act [19].

| Complete Pipeline | Action | Ground | Action + Ground |
|-------------------|--------|--------|-----------------|
| $C_{feas} + \text{Seq2Act}$ |        |        |                 |
| Complete          | 47.0   | 25.0   | 16.8            |
| Partial           | 49.6   | 28.7   | 20.5            |

We lastly discuss the ethical impact this work may have. The collection of MoTIF was designed to ensure no personally identifiable information is captured. But, in downstream use of mobile app task automation, user privacy and security is a concern. People who use assistive technology (e.g., people who are blind) already expose sensitive information to other humans to receive help [1, 2]. To mitigate potential harm from automating app tasks, deployment of our research can be limited to apps which do not require logging in information; these apps are less likely to include name, address, or payment data. MoTIF does not have tasks which require payment, and we can blacklist payment related tasks to prevent fraud and other undesired outcomes. We also acknowledge that improving task feasibility prediction can have more distant negative outcomes, as improved vision-language reasoning could be used for surveillance.

6. Conclusion

We introduced Mobile app Tasks with Iterative Feedback (MoTIF), a new dataset that contains natural language commands for tasks in mobile apps which may not be feasible. MoTIF is the first dataset to capture task uncertainty for interactive visual environments and contains greater lin-
guistic and visual diversity than prior work, allowing for more research toward robust vision-language methods. We introduce the new task of feasibility prediction in interactive visual app environments and evaluate prior methods for automating mobile app tasks. Results verify that MoTIF poses new vision-language challenges, and that the vision-language community can make use of more realistic data to evaluate and improve upon current methods.

Acknowledgements

This work is funded in part by DARPA, the NSF, and the Google Ph.D. Fellowship.

References

[1] Tousif Ahmed, Roberto Hoyle, Kay Connelly, David Candall, and Apu Kapadia. Privacy concerns and behaviors of people with visual impairments. In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, CHI ’15, page 3523–3532, New York, NY, USA, 2015. Association for Computing Machinery. 8

[2] Taslima Akter, Bryan Dosono, Tousif Ahmed, Apu Kapadia, and Bryan C. Semaan. “I am uncomfortable sharing what I can’t see”: Privacy concerns of the visually impaired with camera based assistive applications. In USENIX Security Symposium, 2020. 8

[3] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian D. Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018. 2

[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics, 5:135–146, 2017. 5

[5] Alexis Conneau, Guillaume Lample, Marc’Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation with parallel data. In International Conference on Learning Representations (ICLR), 2018. 4

[6] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embracing Question Answering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018. 1, 2

[7] Biplab Deka, Zifeng Huang, and Ranjitha Kumar. Erica: Interaction mining mobile apps. In 29th Annual Symposium on User Interface Software and Technology (UIST), 2017. 2

[8] Biplab Deka, Zifeng Huang, and Ranjitha Kumar. Erica: Interaction mining mobile apps. In 29th Annual Symposium on User Interface Software and Technology (UIST), 2016. 2

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. 5

[10] Rachel Gardner, Maya Varma, Clare Zhu, and R. Krishna. Determining question-answer plausibility in crowdsourced datasets using multi-task learning. In W-NUT@EMNLP, 2020. 1, 2

[11] Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. IQA: Visual question answering in interactive environments. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 4089–4098, 2018. 1, 2

[12] Izzeddin Gur, Ulrich Rueckert, Aleksandra Faust, and Dilek Hakkani-Tür. Learning to navigate the web. In International Conference on Learning Representations (ICLR), 2019. 2

[13] Danna Gurari, Qing Li, Abigale J. Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P. Bigham. Vizwiz grand challenge: Answering visual questions from blind people. In Conference on Computer Vision and Pattern Recognition (CVPR), 2018. 1, 2

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016. 5

[15] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural Comput., 9(8):1735–1780, Nov. 1997. 5

[16] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yunsuhan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision, 2021. 2

[17] Sheng Jia, Jamie Kiros, and Jimmy Ba. Dom-q-net: Grounded rl on structured language. In International Conference on Learning Representations (ICLR), 2019. 2

[18] Toby Jia-Jun Li, Lindsay Popowski, Tom M. Mitchell, and Brad A. Myers. Screen2vec: Semantic embedding of gui screens and gui components. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI ‘21, 2021. 5

[19] Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldrige. 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, July 2020. Association for Computational Linguistics. 2, 6, 7, 8

[20] Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. Reinforcement learning on web interfaces using workflow-guided exploration. In International Conference on Learning Representations (ICLR), 2018. 2

[21] Thomas F. Liu, Mark Craft, Jason Situ, Ersin Yumer, Radomir Mech, and Ranjitha Kumar. Learning design semantics for mobile apps. In 31st Annual Symposium on User Interface Software and Technology (UIST), 2018. 2

[22] M. Luft. Least squares quantization in pcm. In IEEE Transactions on Information Theory, 1982. 4

[23] Daniela Massiceti, Puneet K. Dokania, N. Siddharth, and Philip H. S. Torr. Visual dialogue without vision or dialogue. CoRR, abs/1812.06417, 2018. 1
[24] Panupong Pasupat, Tian-Shun Jiang, Evan Zheran Liu, Kelvin Guu, and Percy Liang. Mapping natural language commands to web elements. In Empirical Methods in Natural Language Processing (EMNLP), 2018. 2, 7

[25] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, 2014. 7

[26] 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. Learning transferable visual models from natural language supervision. CoRR, abs/2103.00020, 2021. 5

[27] Arijit Ray, Gordon Christie, Mohit Bansal, Dhruv Batra, and Devi Parikh. Question relevance in vqa: Identifying non-visual and false-premise questions, 2016. 1, 2

[28] Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of bits: An open-domain platform for web-based agents. In 34th International Conference on Machine Learning (ICML), 2015. 2, 3

[29] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020. 2, 6, 7

[30] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. CoRR, abs/1707.02968, 2017. 2

[31] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. Journal of Machine Learning Research, 9:2579–2605, 2008. 4

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Conference on Neural Information Processing Systems (NeurIPS), 2017. 6

[33] Alexandra Vtyurina, Adam Fournier, Meredith Ringel Morris, Leah Findlater, and Ryen W. White. Bridging screen readers and voice assistants for enhanced eyes-free web search. In International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS), 2019. 2

[34] Chi Zhang, Feng Gao, Baoxiong Jia, Yixin Zhu, and Song-Chun Zhu. Raven: A dataset for relational and analogical visual reasoning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019. 1
Supplementary Material

We provide additional data collection and analysis for MoTIF along with extra experiment ablations.

7. MoTIF Collection

For data collection, we use UpWork\(^2\) as our crowd sourcing platform and hired 34 people to collect our dataset. Of the annotators, 21 identified as female and 13 identified as male. The median age of the annotators was 23.5 years old. Annotators were from 18 different states in the U.S. and had a range of education from a high school diploma to a master’s degree (2 have high school degrees, 24 have bachelor’s degrees, and 8 have master’s degrees).

Annotators were selected on UpWork if their profile skills listed data entry. As the initial iteration of MoTIF is in English, we also required annotators be fluent in English, but did not require them to be native speakers. We posted separate job listings for the task writing (base rate $15/hr) and task demonstration (base rate $10/hr) portions of the data collection, and as a result had independent annotators for these stages. Annotators hired for the task writing were not informed of our interest in potentially ambiguous or infeasible tasks.

For the annotators hired for task demonstration, we additionally required them to personally have experience with Android devices so that there was no additional noise introduced from people unfamiliar with Android apps. We created anonymized log in information for annotators so that no personally identifiable information was collected. An example of the interface used by the workers is in Section 7.1.

7.1. Data Collection Interface

We provide an example of what our data collection interface looks like for annotators while they explore an Android app and perform a task demonstration in Figure 6. Annotators are given the natural language task to attempt within the Android app in the ‘Your Task’ section on the right side of the interface. Below, we provide anonymized email login and password details for them to use if needed. The left hand side of the collection interface displays the phone screen from a physical Android device which is remotely connected to our collection website.

7.2. Natural Language Task Statistics

We provide additional statistics on the natural language tasks in MoTIF in Figure 9. In Figure 9a we plot a histogram over the word frequency of the task vocabulary and Figure 9b shows a histogram over the task length (i.e., how many words a task consists of) frequency across all collected natural language tasks. Both reflect a long tail distribution, which is common for word frequency, and follows Zipf’s Law. For task length, the distribution is skewed towards shorter length tasks (nearly all collected tasks have fewer than ten words), which aligns with MoTIF’s tasks mostly capturing high level instruction instead of step-by-step subtask instruction.

7.3. Annotator Feasibility Agreement

We define annotator feasibility labeling agreement as the fraction of the number of votes for the majority voted label (max($C_{yes}, C_{no}$)) over all votes ($C_{yes} + C_{no}$) for an (app, task) pair in MoTIF, where $C_{yes}$ is the count of votes for feasible and $C_{no}$ is the count of votes for infeasible. In Figure 5, we bin different degrees of annotator agreement and plot each bin’s counts over all (app, task) pairs with demonstrations in MoTIF. The minimum agreement is 50% and maximal agreement is 100%. The majority of our (app, task) pairs have annotation agreement between 90-100%, with 296 out of (app, task) pairs falling in this maximal bin.

7.4. Word Cloud Visualization

We include a word cloud illustration over all tasks in MoTIF in Figure 7. The larger the word in the word cloud, the more often it occurs in MoTIF’s collected tasks. As we compute the word cloud over all tasks (which span fifteen different Android app categories) we can see the largest words are those that are action or instruction oriented words, like ‘click,’ ‘search,’ or ‘show.’ In Figure 8, we show word clouds for tasks per Android app category.

While there are some common words with high frequency across all app categories (like the action oriented words largest in Figure 7), there are other words illustrated that reflect each app category and functionality specific to

\(^2\)https://www.upwork.com/
that topic. For example, in the Education word cloud in the top left of Figure 7, we see words ‘lesson,’ ‘math,’ and ‘history.’ In contrast, the Shopping category shows words like ‘deal,’ ‘search,’ and ‘cart’ with high frequency.

The word cloud visualizations also show the density of words for each Android app category’s collected tasks. The Food & Drink, Productivity, and Music & Audio app categories have the smallest vocabularies, with less densely populated word clouds. This reflects there being lower diversity in the kinds of requests asked by people for these app categories. On the other hand, Maps & Navigation, Weather, and Travel are examples of Android app categories with larger task vocabularies. This can reflect greater diversity in app requests collected, which may be due to the diversity of functionality in these app categories, or the fact that these apps can have highly specific requests (like searching for one location’s weather out of the nearly unlimited locations one could request).

7.5. App Category Clustering Visualizations

We provide the K-Means T-SNE cluster visualizations used in the (task, app) pairing process for each category of app tasks. These clusters decide whether an app’s tasks are kept app-specific, paired to one or two other apps, or category clustered. We zoom into the cluster visualization for the Weather Android app category in Figure 10. On the left, we see the cluster output for K-Means on the average task embedding (using FastText representations) for the tasks written for weather apps. On the right we show the exact same clustering, but now color the points (the points are written tasks) by which app they come from, i.e., which app they were originally written for. In the lower left corner of the cluster visualization we can see an isolated cluster formed from tasks written for the com.droid27.transparentclockweather. As a result, its tasks are left to be app-specific, while all other apps have (app,
Figure 8. Word cloud visualization of MoTIF language tasks per Android app category. There are fifteen total categories: Education, Dating, Communication, Food & Drink, Entertainment, Lifestyle, Maps & Navigation, News & Magazine, Music & Audio, Shopping, Productivity, Social, Tools, Weather, and Travel. The larger the word is illustrated, the more often it occurs.

7.6. MoTIF Examples

We include more example (app, task) pairs from MoTIF. Figure 12 and 13 show samples for infeasible and feasible tasks, respectively.
7.7. Application List

We include lists of all Android apps we collect demonstrations for in Tables 7-9. In addition to app package name, we provide the corresponding Google Play Store Category and how that particular app’s tasks were paired (app-specific, paired, or category-clustered). The apps selected for MoTIF were across fifteen app categories: lifestyle, communication, dating, food and drink, maps and navigation, news and magazines, productivity, shopping, social, travel, weather, tools, music and audio, entertainment, and education. For privacy, we do not intend to collect any demonstrations of natural language tasks within dating apps, and will not be releasing any of the raw data collected when annotators decided on a list of natural language tasks for dating apps in the first stage of collection. We simply include dating apps as one Android category to see what kinds of tasks people would consider being automated in this setting. We will share the resulting natural language tasks, but no captured screen or view hierarchy data.
Figure 11. T-SNE visualization of K-Means clusters for each Android Google Play Store Category. The visualizations are colored with the originating app label (and not cluster label). These visualizations are used to inspect which apps should retain their app-specific tasks during the demonstration stage.
Figure 12. Example tasks from MoTIF deemed infeasible by annotators. We show the input (app, task) pair for task demonstration, the resulting task demo (which captures the rendered screen, app view hierarchy, and action localization), and the feasibility annotations and follow up questions posed by annotators.
Figure 13. Example tasks from MoTIF deemed feasible by annotators. We show the input (app, task) pair for task demonstration, the resulting task demo (which captures the rendered screen, app view hierarchy, and action localization), and the feasibility annotations and follow up questions posed by annotators.
| Google Play Store Category | App Name                          | (app, task) Pair Method |
|----------------------------|----------------------------------|-------------------------|
| Education                  | com.ted.android                   | app-specific            |
|                            | gov.nasa                          | app-specific            |
|                            | example.matharithmetics           | paired                  |
|                            | org.khanacademy.android           | app-specific            |
|                            | com.duolingo                      | app-specific            |
|                            | com.quizlet.quizletandroid        | app-specific            |
|                            | com.remind101                     | N/A                     |
|                            | org.coursera.android              | N/A                     |
|                            | com.microblink.photomath          | paired                  |
| Entertainment              | com.megogo.application            | app-specific            |
|                            | com.app.emotes.dances.fortnite    | app-specific            |
|                            | com.scannerradio                  | app-specific            |
|                            | com.google.android.youtube        | app-specific            |
|                            | com.zombodroid.MemeGenerator      | app-specific            |
|                            | tv.pluto.android                  | app-specific            |
|                            | com.tubitv                        | app-specific            |
|                            | com.imdb.mobile                   | app-specific            |
|                            | com.eventbrite.attendee           | app-specific            |
| Dating                     | com.wilddec.dating.meet4u         | category-clustered      |
|                            | com.once.android                  | category-clustered      |
|                            | emotion.onekm                     | category-clustered      |
|                            | ru.fotostrana.sweetmeet           | category-clustered      |
|                            | com.mason.wooplus                 | category-clustered      |
|                            | com.hitwe.android                 | category-clustered      |
| Communication             | com.google.android.gm             | app-specific            |
|                            | com.sec.android.app.sbrowser      | paired                  |
|                            | com.facebook.orca                 | N/A                     |
|                            | com.whatsapp                      | N/A                     |
|                            | org.mozilla.firefox                | paired                  |
|                            | com.skype.raider                  | N/A                     |
| Food & Drinks              | com.joelapenna.foursquared        | app-specific            |
|                            | com.yum.pizzahut                  | app-specific            |
|                            | com.chickfila.cfaflagship         | app-specific            |
|                            | com.dominospizza                  | paired                  |
|                            | in.swiggy.android                 | app-specific            |
|                            | com.opentable                     | app-specific            |
|                            | com.starbucks.mobilecard          | app-specific            |
|                            | vivino.web.app                    | app-specific            |
| Lifestyle                  | com.hm.goe                        | app-specific            |
|                            | com.adpog.diary                   | app-specific            |
|                            | com.aboutjsp.thedaybefore         | app-specific            |
|                            | info.androidz.horoscope            | N/A                     |
|                            | ru.mail.horo.android              | paired                  |
|                            | com.urbandroid.sleep              | app-specific            |
|                            | com.hundred.qibla                 | app-specific            |

Table 7. A list of applications used in MoTIF, their Google Play Store Category, and how their submitted natural language tasks were grouped with applications in the (task, app) pairing stage. N/A refers to apps which has technical difficulties during the demonstration stage and we are working to resolve.
| Google Play Store Category | App Name                                                                 | (app, task) Pair Method |
|----------------------------|--------------------------------------------------------------------------|-------------------------|
| Maps & Navigation          | com.tranzmate                                                            | category-clustered      |
|                            | com.mapfactor.navigator                                                  | category-clustered      |
|                            | com.thetrainline                                                        | category-clustered      |
|                            | com.citymapper.app.release                                              | app-specific            |
|                            | com.prime.studio.app.route.finder.map                                   | category-clustered      |
|                            | com.waze                                                                 | category-clustered      |
|                            | com.nyctrans.it                                                         | category-clustered      |
|                            | com.radio.fmradio                                                       | app-specific            |
|                            | deezer.android.app                                                      | app-specific            |
|                            | com.spotify.music                                                       | category-clustered      |
|                            | com.pandora.android                                                     | category-clustered      |
|                            | com.springwalk.mediaconverter                                           | category-clustered      |
|                            | com.google.android.music                                                | category-clustered      |
|                            | com.clearchannel.iheartradio.controller                                 | category-clustered      |
|                            | com.melodis.midimiMusicIdentifier.freemium                             | category-clustered      |
| Music & Audio              | fm.castbox.audiobook.radio.podcast                                      | category-clustered      |
|                            | com.ss.android.article.master                                           | N/A                     |
|                            | com.opera.app.news                                                      | category-clustered      |
|                            | bbc.mobile.news.ww                                                       | category-clustered      |
|                            | com.quora.android                                                       | N/A                     |
|                            | com.google.android.apps.magazines                                       | category-clustered      |
|                            | com.reddit.frontpage                                                    | app-specific            |
|                            | com.sony.nfx.app.sfrc                                                   | category-clustered      |
| News & Magazines           | com.amazon.mShop.android.shopping                                      | app-specific            |
|                            | com.abtnprojects.ambatana                                               | category-clustered      |
|                            | com.contextlogic.wish                                                   | category-clustered      |
|                            | com.joom                                                                | category-clustered      |
|                            | com.ebay.mobile                                                         | category-clustered      |
|                            | com.walmart.android                                                     | category-clustered      |
|                            | club.fromfactory                                                        | app-specific            |
|                            | com.zzkko                                                               | app-specific            |
|                            | com.groupon                                                             | category-clustered      |
| Shopping                   | com.wps.moffice_eng                                                     | category-clustered      |
|                            | com.google.android.apps.docs.editors.sheets                              | category-clustered      |
|                            | com.google.android.apps.docs                                            | N/A                     |
|                            | com.microsoft.office.outlook                                             | category-clustered      |
|                            | com.google.android.calendar                                              | category-clustered      |
|                            | com.google.android.apps.docs.editors.slides                              | category-clustered      |
|                            | com.dropbox.android                                                     | N/A                     |

Table 8. A list of applications used in MoTIF, their Google Play Store Category, and how their submitted natural language tasks were grouped with applications in the (task, app) pairing stage. N/A refers to apps which has technical difficulties during the demonstration stage and we are working to resolve.
| Google Play Store Category | App Name                                      | (app, task) Pair Method |
|-----------------------------|----------------------------------------------|------------------------|
| Tools                       | com.lenovo.anyshare.gps                      | app-specific           |
|                             | com.antivirus                               | paired                 |
|                             | com.google.android.calculator                | paired                 |
|                             | com.miui.calculator                         | paired                 |
|                             | com.google.android.apps.translate            | app-specific           |
|                             | com.avast.android.mobilesecurity             | paired                 |
| Travel                      | com.kayak.android                            | paired                 |
|                             | com.tripadvisor.tripadvisor                  | paired                 |
|                             | com.trivago                                  | paired                 |
|                             | com.google.android.apps.maps                 | paired                 |
|                             | com.yelp.android                             | app-specific           |
|                             | com.booking                                 | N/A                    |
|                             | com.google.earth                            | paired                 |
|                             | com.mapswithme.maps.pro                     | app-specific           |
|                             | com.google.android.street                   | paired                 |
|                             | com.yellowpages.android.ypmobile            | app-specific           |
| Weather                     | com.gau.go.launcherex.govidget.weatherwidget | N/A                    |
|                             | com.devexpert.weather                       | category-clustered     |
|                             | com.chanel.weather.forecast.accu            | category-clustered     |
|                             | com.weather.Weather                         | category-clustered     |
|                             | com.droid27.transparentclockweather         | app-specific           |
|                             | aplicacion.tiempo                           | category-clustered     |
|                             | com.accuweather.android                     | category-clustered     |
|                             | com.windyty.android                         | category-clustered     |
|                             | com.handmark.expressweather                 | category-clustered     |
| Social                      | com.zhiliaoapp.musically                    | category-clustered     |
|                             | com.pinterest                               | category-clustered     |
|                             | com.instagram.android                       | category-clustered     |
|                             | com.facebook.katana                         | category-clustered     |
|                             | com.ssigggle.production                     | app-specific           |
|                             | com.snapchat.android                        | app-specific           |
|                             | com.ss.android.ugc.boom                     | category-clustered     |
|                             | com.lazygeniouz.saveit                      | category-clustered     |

Table 9. A list of applications used in MoTIF, their Google Play Store Category, and how their submitted natural language tasks were grouped with applications in the (task, app) pairing stage. N/A refers to apps which has technical difficulties during the demonstration stage and we are working to resolve.
8. Experiments

We include additional experimental results and model details. For task feasibility experiments, we include the additional view hierarchy ablations in Table 10. We then add details about the input features and model architecture of the Seq2Seq and Seq2Act models we adapt and use to benchmark MoTIF for app task automation in Section 8.2.

8.1. Task Feasibility Ablations

In Table 10(a), we have additional rows in which we ablate which view hierarchy elements are included as input features to our feasibility classifier. The view hierarchy of an Android app contains several types of elements: text (ET), identity (ID), and class (CLS) elements. We ablate using one or multiple of these element types and find that on average across demonstration aggregation type, the (ET + ID) input results in the best performance. Consequently, we keep it for our best results in the main text.

8.2. Task Automation Benchmarks

For the mobile app task automation experiments, we use Seq2Seq and Seq2Act to benchmark MoTIF and evaluate the quality of prior work for more realistic input tasks. We describe in greater detail each method and provide additional discussion of Seq2Seq results, which were notably lower than Seq2Act.

8.2.1 Seq2Seq Model

We use Seq2Seq to train and evaluate MoTIF. The Seq2Seq model was proposed as a VLN model for the ALFRED dataset, which contains actionable language requests for tasks in household environments. The original Seq2Seq model predicts an action and binary mask at each time step. The binary mask should isolate the household object on which the action is performed. The language task is attended to at each time step, which is used as input with the visual features of the current state, the previous predicted action, and the previous hidden state of the bidirectional LSTM which takes the former three as input.

These four features are then input to a fully connected layer with a Softmax that outputs an action prediction and a deconvolutional network for mask prediction. The input language task is attended to with the previous hidden state from the LSTM decoder and is weighted with an additional fully connected layer. We replace the deconvolutional mask prediction network for three fully connected layers that predict a point in the app screen and minimize the mean squared error to the ground truth action location in the app screen. Action prediction is trained via cross entropy.

To train this modified version of Seq2Seq, we use ResNet18 features from the last convolutional layer as done in the original paper, which is needed for meaningful localization on the app screen. In addition to the features used in the original Seq2Seq model, we also include CLIP representations of the screen state at each time step.

8.2.2 Test-time Evaluation of Seq2Seq

We build an offline version of each Android app environment to approximate a complete state-action space graph at test time. We merge demonstrations we’ve collected across all samples, in addition to separate exploration demonstrations we obtain from instructing annotators to navigate the app as much as possible to find (an approximation of) all states within the app.

The nodes in this state-action space graph are unique ‘views’ of an application, i.e., a particular screen within a demonstration. Nodes are connected by edges which represent the transition between any pair of screens. A transition between app screens is defined by the action type and the location of the action taken at the current screen state. Action types include click, type, or swipe and the action location is defined as coordinates (where does it occur on the screen).

8.2.3 Seq2Seq Results

The Seq2Seq model resulted in significantly worse performance than Seq2Act. The performance drops off heavily when evaluating grounding performance. We consider a predicted point valid if it falls within the bounding box of

| C\textsubscript{feas} Input Features | Demo Aggregation |
|-----------------------------------|-----------------|
|                                  | Avg | Cat | LSTM |
| **Random**                       |     |     | 20.1 |
| **(a) View Hierarchy**           |     |     |      |
| **FastText**                     |     |     |      |
| ET                                | 22.8| 44.3| 37.0 |
| ET + ID                          | 16.7| 43.6| 34.1 |
| ET + ID + CLS                    | 19.7| 39.6| 36.2 |
| **CLIP**                         |     |     |      |
| ET                                | 27.0| 48.4| 35.9 |
| ET + ID                          | 28.0| 50.9| 36.2 |
| ET + ID + CLS                    | 29.6| 49.2| 35.2 |
| **Screen2Vec**                   |     |     |      |
|                                  | 25.9| 33.7| 36.0 |
| **(b) App Screen**               |     |     |      |
| ResNet                            | 31.3| 41.9| 35.9 |
| Icons                             | 0.4 | 40.0| 15.2 |
| CLIP                              | 44.7| 58.2| 42.8 |
| **(c) Best Combination**         |     |     |      |
| CLIP (Screen + ET + ID)           | 44.8| 61.1| 40.9 |

Table 10. Task feasibility F1 score using our MLP. We ablate input features and how demo sequences are aggregated. The random baseline predicts a feasibility label given the train set distribution.
the ground truth UI element associated with the action. To investigate if augmenting MoTIF with additional data could help performance (since we train Seq2Seq from scratch), we use RicoSCA, the same synthetic data used by Seq2Act for their grounding model. Since we do not want to train on step-by-step instructions, we slice their dataset to single step tasks instead. While we are interested in high level goals like those of MoTIF, the hope was that we could improve the localization portion of the task with the augmented data.

We report results in Table 11 and find that grounding performance does not change. While additional model ablations (such as varying the weights of the action and grounding losses or architectural changes) may improve performance, it is clear that predicting action localization over the continuous space of the app screen is a more challenging task. Thus, predicting over the UI leaf elements as Seq2Act does is more constrained and would be the best formulation of the problem. It is important to note that while predicting action localization in a continuous space results in poor performance, this does not necessarily mean that visual screen information can not be utilized to aid task automation.

### 8.2.4 Seq2Act Model

As described in the main text, Seq2Act models mobile app task automation in two stages: action-phrase tuple extraction, and action-phrase grounding. Both stages are modeled with Transformers. All Transformers used have 6 layers and are trained with cross entropy losses.

The first model predicts spans that correspond to the action phrase tuples needed to complete the full input instruction. A span indexes into the input instruction string and the action phrase tuple is defined as [action type, action location, action input]. Thus, each extracted tuple consists of three spans: the span referring to the action type, the span referring to the UI element the action occurs on, and the span referring to the input text argument if the action class is typing.

This model has an encoder-decoder architecture: the encoder embeds the input instruction’s text tokens and the decoder computes a query vector for each span representation given the previously decoded spans. A span representation for indices $b : d$ with $1 \leq b \leq d \leq N$ for an N-token input instruction is defined as the sum of encoder outputs $h_b$ through $h_d$. There is a decoder query for each element of the action tuple per time step: an action type query, a UI object query, an input text query. Next, a similarity score is computed between the span representations and the corresponding decoder query. A Softmax is applied and the span that results in highest score is used as the prediction for that time step.

The action grounding model then takes each extracted phrase from the first stage as input and predicts an associated action type and action location (which UI element it is performed on). Actions are predicted given the encoder embedding of the predicted action span using a Multi-Layer Perceptron. To localize the action, UI elements from the current state’s screen are first embedded. UI inputs to the Transformer encoder are represented as content (UI name and type) and positional features (screen coordinates, preorder and the postorder traversal index in the view hierarchy tree). A Softmax is applied over the similarity between the predicted object span encoding from the phrase tuple extraction model and the latent object representation output from the grounding encoder. Before the similarity (i.e., dot product) is computed, the grounding encoder UI representation is also passed through an MLP. The max scoring UI element becomes the predicted grounding for that time step.

| Model      | Action | Ground | Action + Ground |
|------------|--------|--------|-----------------|
| Seq2Seq    |        |        |                 |
| Complete   | 19.4   | 0.3    | 0.3             |
| Partial    | 72.2   | 4.0    | 3.9             |

Table 11. Mobile app task automation on MoTIF. We evaluate the Seq2Seq navigation model trained from scratch with MoTIF and additional RicoSCA data.