MessageOnTap: A Suggestive Interface to Facilitate Messaging-related Tasks

Fanglin Chen
Human-Computer Interaction Institute
Carnegie Mellon University
Pittsburgh, PA, USA
fanglin@cmu.edu

Kewei Xia
Electronic Information School
Wuhan University
Wuhan, China
xiakewei@whu.edu.cn

Karan Dhabalia
Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA, USA
kdhabali@andrew.cmu.edu

Jason I. Hong
Human-Computer Interaction Institute
Carnegie Mellon University
Pittsburgh, PA, USA
jasonh@cs.cmu.edu

ABSTRACT

Text messages are sometimes prompts that lead to information related tasks, e.g. checking one's schedule, creating reminders, or sharing content. We introduce *MessageOnTap*, a suggestive interface for smartphones that uses the text in a conversation to suggest task shortcuts that can streamline likely next actions. When activated, MessageOnTap uses word embeddings to rank relevant external apps, and parameterizes associated task shortcuts using key phrases mentioned in the conversation, such as times, persons, or events. MessageOnTap also tailors the auto-complete dictionary based on text in the conversation, to streamline any text input. We first conducted a month-long study of messaging behaviors (N=22) that informed our design. We then conducted a lab study to evaluate the effectiveness of MessageOnTap's suggestive interface, and found that participants can complete tasks 3.1x faster with MessageOnTap than their typical task flow.

CCS CONCEPTS

• **Information systems** → *Information systems applications*; *Mobile information processing systems.*

CHI 2019, May 4–9, 2019, Glasgow, Scotland Uk

ACM ISBN 978-1-4503-5970-2/19/05…$15.00
https://doi.org/10.1145/3290605.3300805

KEYWORDS

Messaging/Communication, Information Seeking & Search; Contextual Computing; Personal Data/Tracking; User Experience Design; Text/Speech/Language; Productivity

ACM Reference Format:
Fanglin Chen, Kewei Xia, Karan Dhabalia, and Jason I. Hong. 2019. MessageOnTap: A Suggestive Interface to Facilitate Messaging-related Tasks. In *CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019), May 4–9, 2019, Glasgow, Scotland Uk*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3290605.3300805

1 INTRODUCTION

Messaging apps are among the most popular apps for smartphone users. Recent reports from WhatsApp and Facebook Messenger [24] indicate that both apps have more than one billion monthly active users worldwide, with more than 100 billion messages sent each day. As of 2017, U.S. smartphone users spent most of their time on instant messaging apps [60], with an average of over 10 minutes across all users [61].

Messaging apps can be thought of as a nexus for receiving and handling many kinds of tasks related to external apps, such as updating schedules in the `Calendar`, or look up phone numbers in the `Contacts`. Users usually have to switch back and forth between apps to complete tasks, wasting a disproportionate amount of attention. For example, receiving the message *"when is the party tonight?"* might lead a person to go to their smartphone home screen, look for the `Calendar` app, navigate to the right day, look up the information, switch back to the messaging app, and then reply with the time. This overall workflow of handling message-related tasks is tedious and inefficient and hampers user productivity and ongoing conversations.

We are interested in streamlining the user workflow in completing such *message-related tasks*. In this paper, we define message-related tasks as tasks initiated by information

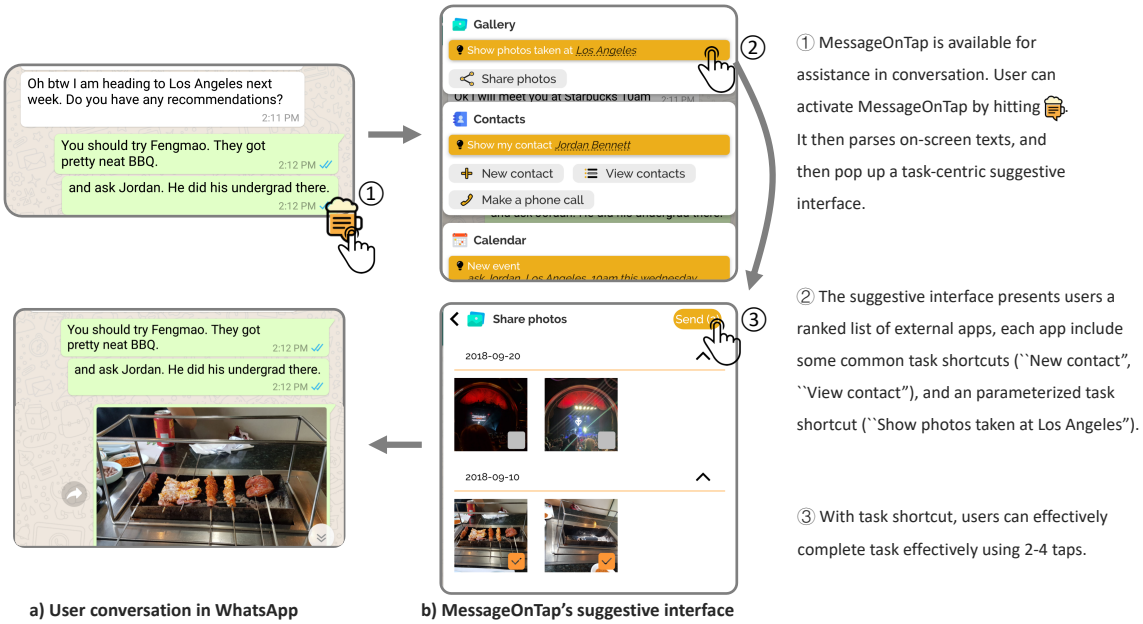

Figure 1: MessageOnTap (right) is a conversation-aware suggestive interface. Users can get access to external apps to complete related tasks. Once activated, MessageOnTap presents a task-centric interface to suggest users conversation-relevant apps, some common in-app task shortcuts (e.g., "View contacts") and parameterized task shortcuts based on the mentioned key phrases (e.g., "Show photos taken at Los Angeles").

needs expressed by some messages in ongoing conversations, which require users to access other apps to complete the majority of the task. There is currently little support for handling such tasks. In particular, intelligent features are typically not oriented towards understanding conversations and connecting to relevant task-oriented apps. Search-integrated soft keyboards make it easier for users to access contents on the web [29] or stored in the phone [35] whenever users start to type. However, due to a lack of conversation awareness, these general-purpose keyboards require additional user efforts in composing search queries, even if related contents are already mentioned in conversation. There are also screen-aware intelligent features in mobile OS [28, 50], offering on-demand suggestions related to on-screen contents. However, these suggestions are not tailored to conversations, and most of the suggestions are mostly designed to bring up publicly-available contents, such as celebrities, restaurants, and song name, rather than often-mentioned personal contents in conversation, such as contacts, schedules, documents, and images.

Offloading complex task workflow to language-perceptive interfaces is a promising approach to efficiently handle tasks. For example, conversational agents [22, 30, 36] can be triggered by user utterances to activate underlying in-app actions of relevance, mitigating user costs in looking for the right app and the right in-app page for tasks. This might still require users to reiterate the mentioned contents in conversation. Explicit user voice commands can also limit the use of such services in public spaces [8]. The now defunct M suggestions feature [25] in Messenger aimed to predict users' next logical tasks based on conversation understanding, which is a conversation-aware intelligent agent that requires minimal user efforts to get things done. While it shares the same broad goal as us, it tends to map the diversity of user conversations into a narrower space of suggestions and has an intrusive interaction style. For example, it pops up "start plan" whenever it detects any text related to time. This mismatch incurred many negative user reviews, such as it is *actively in the way* [17] and *irrelevant* [26].

To better study the relationship between user conversations and subsequent tasks, as well as the costs messaging app users currently have in handling these message-related tasks, we conducted an *in situ* study with 22 users over 4 weeks, collecting their messages as well as UI actions on their smartphones. In our analysis of this data, we found two connections between conversations and their subsequent tasks: 1) In task-related conversations, people often implicitly express their intent to use external apps, for example, talking about "going to sleep" before switching to an alarm app to set up an alarm for tomorrow. 2) Certain key phrases in conversations (e.g. person names, time expressions, event names, etc.) can be used in the subsequent tasks to retrieve a piece of information or filling in details.

Based on these findings, we developed MessageOnTap (Figure 1), a unified and suggestive interface designed to streamline the process of handling message-related tasks, by bridging the gap between conversations and external apps. Our current implementation works as a user-activated overlay on top of popular messaging apps, such as Messenger and WhatsApp. Once activated, MessageOnTap scans messages on the screen and then offers *task shortcuts* to relevant external apps. If MessageOnTap can correctly suggest useful task shortcuts, then it can greatly streamline the process of seeking the right app, looking for the right in-app functionality, and doing text entry related to the task. Our current implementation of MessageOnTap supports task shortcuts for commonly used apps, including *Photo Gallery*, *Documents*, *Calendar*, *Reminders*, and *Contacts*.

Just like most existing dashboards in messaging apps, MessageOnTap organizes the task-related functionalities by apps. The novelty is that it surfaces relevant task shortcuts directly to users (e.g., "share photos" is under Gallery), and parameterize task shortcuts with extracted key phrases from a conversation. The apps are also ranked based on their relevance to an ongoing conversation (see Figure 1b). Keyphrases mentioned in conversations can be also used to streamline text inputs during the task. For example, if a user chooses to create a new calendar event through a MessageOnTap task shortcut, the text auto-complete will include key phrases extracted from the text, so users would have minimal text entry costs.

We offer the following contributions to HCI: (1) We present a 22-user study that characterizes the workflow and costs of people in completing message-related tasks, providing evidence of the semantic relevance between conversations and their subsequent tasks. (2) We introduce the design of MessageOnTap, which is a suggestive interface to reliably provide conversation-relevant task shortcuts from instant messages. (3) We demonstrate an approach to enable messaging app dashboards to be more conversation-aware and better connected to task-oriented apps.

2 MOBILE MESSAGING STUDY

We conducted an *in situ* study of mobile messaging behaviors to understand *i*) what kinds of costs there are in handling message-related tasks, and *ii*) how to reduce these costs. The study participants were college students at a large university in North America and were recruited during Fall 2017.

22 students (mean age 24.8, 11 females) were recruited through flyers posted in campus and participated in our experiment for 2-4 weeks. Data were collected for an average of 26.8 days (SD=9.5). To log a variety of app events, all participants were required to use Android smartphones as their primary device. Our recruited participants all had at least one messaging app installed on their phone prior to the study. At the beginning of the study, participants were asked to install a custom data collector app that would allow us to collect smartphone usage data. They were instructed to turn on the Accessibility permission setting on their phone, with warnings that this enables sharing UI actions, including on-screen texts. Our data collector, operating as an Accessibility Service, continuously log the screen content after user-initiated UI events (such as touch and text input events) and uploads the collected screen content to our secured server whenever there is a WiFi connection on phone. By *screen content*, we mean the GUI tree data structure and related view components, as offered by the Accessibility API. At the end of the study, we uninstalled the collector from their phones and compensated participants with $70 cash.

Methods

The collected continuous series of screen contents allow us to identify when users are in a particular messaging app, as well as specific user actions in the subsequent apps that users switched to. To facilitate analysis, we extracted all text from the screen content (e.g., text fields and content descriptions of `Button`, `EditText` and `TextView`).

Our analysis is comprised of two steps. We first annotated each individual message text based on their topics, to understand what topics led to tasks. Then, we performed a cross-app analysis on message-related tasks, to map out the relationship between a conversation and subsequent switches to an external app. Our message topic annotation followed a general inductive approach [63]. Two of the authors read through 200 randomly selected messages and worked together to form an initial codebook. We then had three annotators (undergrad researchers) code the corpus over 3 rounds, updating the codebook between rounds. Every message was coded by at least 2 annotators. Codes for the third round were considered final. While annotators could assign multiple codes per message, we required that each message at least be categorized as either *conversational* or *informational*. This is based on previous work on understanding user intents from natural corpus [19, 49]. One of the authors served as a tie-breaker to ensure all messages were assigned at least one code. After the topic annotation process, annotators annotated the app transitions between conversations with at least one *informational* message and the on-screen tasks of subsequent apps, to determine whether or not an app transition was indeed a message-related task. Finally, all message-related tasks were identified using majority votes.

Results

In the 4 weeks of our study, our participants used 260 different Android apps. There were 10 messaging apps used, including Google Hangouts, GroupMe, Line, Slack, WeChat,

WhatsApp, Messenger, as well as built-in SMS apps (e.g., Android Messages, Samsung Messages, Sony Ericsson Messages). After excluding messages in Spanish and Chinese, there were 19015 messages (7796 were sent by participants). On average, participants spent 28.3 minutes in their messaging apps, with 17.0 received messages (Min=3, Q1=4, Med=9, Q3=7, Max=18, SD=4.34) and 11.8 sent messages (Min=1, Q1=5, Med=11, Q3=25.5, Max=33, SD=10.6) per day. 78.6% messages were short messages with less than 7 words.

Finding 1: Message-related tasks are not common, but are slow to complete. In our topic analysis and cross-app task analysis, we found that the majority of messages were sent for conversational purposes and were not associated with any tasks or external apps (53.9% of messages, e.g. *"You put me in panic mode now"*, "*Such a great initiative :p*").

Informational messages were seen slightly less often in a conversation (46.1% of messages). However, informational messages were likely to be associated with tasks in external apps. Participants informed each other about their availability to communicate (2.54% , "*Yeah sorry just got out of a meeting*"), which might prompt users to place phone calls; coordinated upcoming events (13.7%, *"How is 16th?"*), which might prompt users to access calendar schedules; shared personal content (12.7%, e.g. contact information, photos and URLs), which might prompt users to access other external apps. There were also informational messages that were primarily sent to convey current status and were not associated with external apps (17.1%, e.g., "*leaving home in 2 mins*").

Examining informational messages more closely, we found participants completed 706 message-related tasks in total. Although 1.19 message-related tasks per day on average is not a lot (Min=Q1=0, Med=1, Q3=2, Max=6, SD=1.28), it took users an average of 82.2s to complete a message-related task, with a very high standard deviation (Min=15, Q1=40.2, Med=181.3, Q3=200, Max=319, SD=93.7). The most frequently used app category for message-related tasks is email (35.7%), followed by web-related content (16.2%) such as search, social networking and news, navigation (10.1%), calendar and to-do list (9.8%), contacts (8.2%), finance-related (6.7%), weather (4.7%), photo gallery (1.8%), and documents (1.0%).

Finding 2: The time to complete a message-related task can be broken down into several common stages. We further categorized message-related tasks into *retrieval tasks* which involve looking up content (e.g., checking schedules or sharing photos), and *creation tasks* which involve filling in details for a data item (e.g., ordering coffee, adding a new reminder, updating details for a local contact). Figure 2 shows our model describing the stages involved in handling these tasks.

Users start with *App Seeking*, looking for the desired external app. Then, for many retrieval tasks, users do *Data Selection*, which involves navigating or searching for the desired data, for example, to view the schedule for a day or to look up someone in one's contact list to place a call. For creation tasks, users engage in *Data Entry* to enter in details for a new data item (such as an upcoming reservation or a calendar event) or update an existing data item (such as updating a person's contact information). Upon completion of the task, users might go back to their original conversation for a follow-up *Reply* related to the task. Our cross-app analysis identified various costs across these stages, in terms of time to complete. In the *App Seeking* stage, it took participants an average of 6.19s to locate an app (Min=1.30, Q1=3.50, Med=10.01, Q3=10.50, Max=40.0). Users can access external apps faster if they are frequently used, but 30.7% of the time the task-related apps were not in listed in the app suggestions in the Android default launcher.

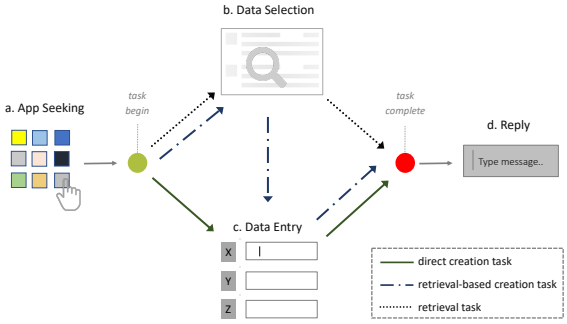

Figure 2: Our stage-based model for message-related tasks. Using the messages shown in Figure 3b as an example, the user switches out of the messaging app to look for the `Calendar` app in the *App Seeking* stage; scrolls down to view the next week and select Tuesday in the *Data Selection* stage; fills in the event name and selects the event time in the *Data Entry* stage; and finally returns to the original conversation to inform that the task is complete in the *Reply* stage.

Once in the right app, users had to traverse an average of 2.4 screens (SD=0.51) to get to the desired data, taking an average of 5.24s (Min=4.30s, Q1=4.82, Med=6.17, Q3=6.50, Max=9.5s). For example, to view a schedule for the desired day, a user would need to manually navigate to the desired date out of a monthly or weekly calendar. If a search bar is provided in the app, to book a restaurant, a user would search the restaurant for reservations, instead of navigating a long list of nearby restaurants. It on average took 15.5s (Min=4.30s, Q1=4.12, Med=4.5, Q3=10.3, Max=120.1s, SD=30.8) for users in data entry stage. Sometimes this includes the time spent for users to copy items from messages and manually paste them to related input fields in the task-related app (Mean=3.27s, Min=2.16s, Q1=2.3, Med=2.8s, Q3=4.1, Max=4.5s).

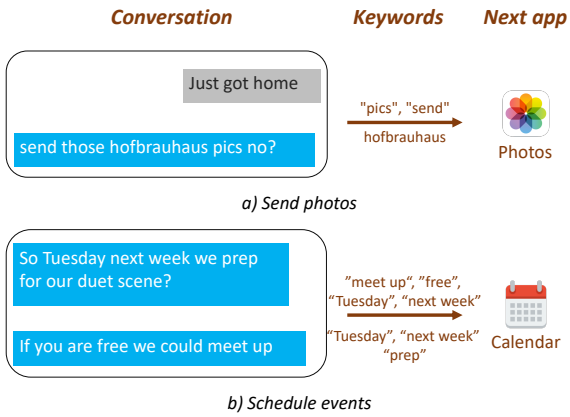

Figure 3: Messages can be related to subsequent external apps in two ways: *(i)* Certain keywords (top of the arrows) are related to functionality offered by another app, indicating an intent to use that app. In a) the keywords "pics" and "send" indicate a request for photos, suggesting a likelihood of going to a Photo Gallery app next; *(ii)* Some keywords (under the arrows) might be directly used in a subsequent task. In b) the user manually navigated to next Tuesday, and typed "prep" while creating a new event in the Calendar.

Finally, 8.5% message-related tasks conclude with a reply back into the original conversation, which took on average 6.45s.(Min=1.9, SD=3.11, Med=6.5s, Q1=3.3, Q3=10.1, Max=11.4s). The costs here include switching back to the messaging app and composing a reply, which might involve manually copying the data items from the external app and pasting them to the right text box.

Finding 3: The text of a conversation is often semantically relevant to the apps and text used for App Seeking, Data Selection, and Data Entry. A closer examination of conversation messages and their subsequent tasks reveals that in task-related conversations, users sometimes implicitly suggest their intent to use another app (see Figure 3 for some example messages in our dataset). We manually annotated key phrases in the collected message dataset, and found that 32.2% of the collected messages contained one or more key phrases[1], including time (11.6%), URLs (1.9%), phone numbers (1.4%), email addresses (0.8%), and common named entities such as a person's name (3.8%), event name (6.1%), or place name (11.8%). 71.1% of message-related tasks contained at least one of these key phrases. In these cases, we found that participants either explicitly copied the mentioned key phrase into their task-related app, or manually typed in the text, as shown in Figures 3. This led to our insight that text input in subsequent tasks can be simplified by connecting these key phrases to external apps.

[1] 5.2% of messages contain more than one key phrases.

3 BACKGROUND AND RELATED WORK

Our work touches on several areas including mobile phone usage understanding, app recommendation, and conversation-aware intelligence. More generally, MessageOnTap's design is deeply influenced prior principles of designing context-aware systems. We now summarize the related work in these respective domains.

Understanding Usages of Messaging Apps

Back in 2011, Church *et al.* 's study in mobile web access patterns [7] found that on mobile devices, some user information needs are sparked by conversations, followed by search activities in the browser. Since this study, we have seen declining mobile browser use at large [4, 40], with most of the web traffic being shifted from browsers to a wide variety of installed apps on users' phones. In keeping with this trend, it is important to understand what *tasks*, in what *apps*, are prompted by what kinds of *conversations*. This requires a systematic study of conversation and its relationship to task-oriented mobile apps. To solicit conversations at scale, prior studies either crowd-source screenshots that are hand-picked by users [4, 39], or collect conversation history from customized messaging apps [5]. Neither of these approaches, however, can be easily extended to capture various kinds of subsequent user actions resulted from conversations. Instead, we collect objective *in situ* phone usage data to continuously sample detailed in-app contents in time-series. This also allows us to perform a deeper level analysis of smartphone users than existing work [6, 21, 43]. By examining the detailed on-screen contents of the foreground-running apps of users, we can quantify the time spent on each user action, and conduct content analysis to investigate relevance between conversations and subsequent tasks.

Connecting Conversation with Apps

As messaging apps such as iMessage, Messenger, and WeChat gradually grow from communication tools to platforms for app integration [23, 27, 37], the messaging app dashboards have evolved as the next app launcher on phone [10], with tens of apps being squeezed into much smaller screen space. It is therefore imperative to redesign these app-enriched dashboards to help users efficiently get access to their intended apps. There are predictive algorithms that leverage temporal [58], spatial [62] and frequency-based information [3] for app recommendation. MessageOnTap contributes a conversation-based app prediction model which connects implicit user intents to mobile app functionality. While prior HCI systems also map natural language to underlying functionality, in MessageOnTap, we specifically explore the feasibility of mapping human-to-human conversations to app

functionalities, instead of human-to-machine queries to system commands [1, 11, 12, 14]. To represent app functionalities, we leverage the publicly-available app descriptions, which has been shown effective in-app suggestion tasks based on individual short tweets [51].

Conversation-aware Intelligence

Automatic suggestions of short text reply (e.g, "sounds good", "will do") for incoming messages [38, 65] can greatly improve user efficiency in text entry, and there are also specific language-based models to provide reply suggestions of photos [41] or documents [64]. Yet it is not well understood how to apply such individual language models to facilitate diverse kinds of tasks each mobile app has to offer. In MessageOnTap, we demonstrate extractions of various kinds of key phrases (e.g., person, location, time, etc.) can be used to support matching task shortcuts in commonly used apps, such as `Photo Gallery`, `Documents`, `Calendar`, `Reminders`, and `Contacts`.

Hyperlinked texts (e.g., "I can be reached at `804-222-1111`") are commonly seen in text-rich apps such as IM or email apps, offering users direct deep links to certain apps. However, they are limited to highly-structured texts (e.g, phone numbers, dates), and each hyperlink can facilitate tasks on one piece of text only (e.g, "create contact" and "add to existing contact" for a hyperlinked phone number). MessageOnTap supports a broader range of key phrases like names of place, event, and person. It also demonstrates an affordance to organize task suggestions for a number of apps, which can be generated by key phrases scattered across different messages in the conversation. All these cannot be well supported in the form of hyperlinks.

MessageOnTap also draws on insights from tools designed to provide general-purpose suggestions to on-screen contents [13, 28, 50]. All of these systems use Accessibility APIs to collect on-screen app contents for semantic (e.g., entity types) extractions, entity linking, and content recommendation. While sharing the same underlying technique to collect texts from screen, MessageOnTap specifically provides suggestions tailored to conversation dynamics, where 1) the required task parameters are often scattered across different messages, and 2) there is ambiguity in user-expressed intents and it is often uncertain what would be the next step for users to take in the task.

Getting the Right Design

An old and ongoing debate in HCI research highlights the tension between direct control by users versus automated actions done on users' behalf by intelligent agents [59]. For system innovations that facilitate message-related tasks, direct manipulation systems [23, 27, 29, 35, 37] grant users more control, but require additional user costs in composing queries to retrieve content [29, 35], or navigating to the intended apps [23, 27, 37]. Automated services [25, 41], on the other hand, tend to map the diversity of user conversations into a disproportionately small space of suggestions, generating many false positives. The resulted intrusiveness hurts the usability of such an intelligent agent in general [17, 26]. As Schilit *et al.* [55] point out for context-aware communication systems, automating either sensing or action tend to do reasonably well, but systems that do both tend to fail, largely due to mismatches between what is being sensed what is actually taking place in social interactions.

HCI researchers have also looked at how to combine the best of both worlds with mixed-initiative systems [20], offering principles as to when an intelligent system should proactively take action versus when a user should. We intend to seek a better middle-ground between direct manipulation and automated service for MessageOnTap's design. After multiple early design iterations, we finally homed in on a design that always keeps users in the interaction loop. Users have to explicitly activate MessageOnTap, just like how they normally interact with the existing dashboards in messaging apps. Task shortcuts are filled in with the most likely inputs, ordered by the relevance of their parent apps. From an interaction perspective, this kind of suggestive interface streamlines the process of handling message-related tasks without sacrificing control.

4 THE MESSAGEONTAP SYSTEM

The analysis of message-related tasks, coupled with lessons from past work, helped establish our vision for MessageOnTap. We want MessageOnTap to leverage content from existing conversations to suggest task shortcuts that can streamline interactions with external apps, thus minimizing the costs of seeking the right app, navigating to the right screen, entering text, and typing text for relevant replies. MessageOnTap presents a unified dashboard for users to access external apps as well as common in-app task shortcuts (see Figure 1). It also surfaces the most relevant parameterized task shortcut for each app. Other features include data filters and text auto-complete (see Figure 4). Behind the scene are two core components: an app ranking algorithm that uses the current conversation to rank external apps (see Figure 5e), and a shortcut selection pipeline which associates key phrases mentioned in a conversation to a set of shortcut templates supported by each app (see Figures 5a-d). The conversation understanding algorithms we use currently cannot easily run on a modern smartphone, so we do this kind of processing on a cloud server.

Suggestive UI features

App Drawer. When activated, MessageOnTap overlays an app drawer on top of the ongoing conversation (shown in

Figure 5f). The app drawer is a list of cards, each of which contains common in-app task shortcuts as well as at most one parameterized task shortcut for one app. App cards are ranked according to their relevance to the current conversation semantics. To improve suggestion relevance [55], we choose to display only one parameterized task shortcut per app /card. As user tests in our earlier prototypes suggest, displaying all valid parameterized task shortcuts at once would generally make users feel lost in selecting the right action, rather than focusing on finishing the task itself.

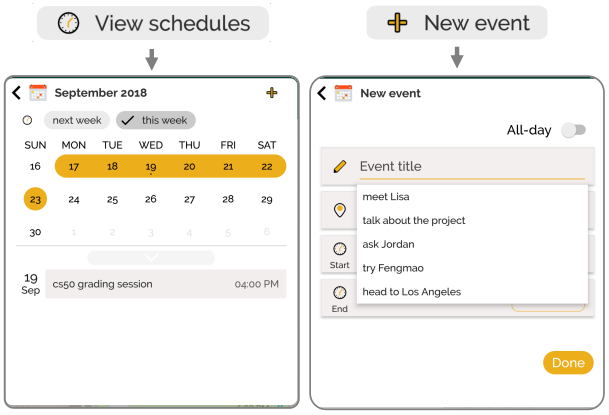

Figure 4: MessageOnTap uses extracted key phrases to facilitate data filters (left) and text auto-complete (right).

Auto-complete & Data filters. MessageOnTap also uses key phrases extracted from a conversation for auto-complete and data filter functionality (see Figure 4). Auto-complete streamlines *Data Entry* for *creation tasks* (e.g. a new reminder or new contact) by tailoring the text input auto-complete dictionary to include mentioned names, events, places, etc. Data filters help with *retrieval tasks* (e.g. viewing contacts or sharing images) by displaying the right content type and then easily filtering based on mentioned names, places, times, etc. To increase relevance, filters only show valid options. For example, a future time expression (e.g., next Sunday) is valid for the *Calendar* app but not for the *Photo Gallery* app. By introducing auto-complete and data filters, MessageOnTap can help users to mitigate costs when they enter/select conversation-relevant data, even if the provided parameterized task shortcut is not selected.

Selecting Relevant Shortcuts

Similar to the app shortcuts [2, 15] that users can access in the app launcher, MessageOnTap has task shortcuts for users to get quick access to in-app functionality without switching out of a conversation, minimizing the need to access pages that are not related to the task. To define task shortcuts, we first revisit the use cases of these apps in our *in situ* study, summarize the common purposes of using such apps, and seek functionality abstraction of these tasks. Some examples of these task shortcuts are listed in Figure 5c.

MessageOnTap introduces parameterized task shortcuts to minimize task costs in both *Data Selection* and *Data Entry*. To generate parameterized task shortcuts, we take the widely-used *slot-filling* approach. For each task shortcut of an app, we define one or more shortcut templates specifying the types of key phrases (i.e., *slots*) to look for, as well as the interfaces to display and the actions to execute. For example, the *Photo Gallery* app defines a template `Show my photos taken at <T_PERIOD>` that can be matched with times mentioned in conversations, as well as a code snippet that can display a user's photos for a given period of time.

As shown in Figure 5a, when selecting the most relevant task shortcut, an app might have multiple shortcuts matched, because 1) an app can have more than one task shortcut template, and 2) there might be several key phrases appearing in the conversation. We describe the steps MessageOnTap takes to rank the most relevant task shortcut below. On average, all these steps can be processed by the backend in about one second.

Step 1: Selecting task-related key phrases. MessageOnTap first uses Google Natural Language API [32] for part-of-speech tagging (noun, verb, etc) and named entity recognition (classifying texts into pre-defined categories such as person names, organizations, locations) on messages on the current screen, extracting key phrases such as event names, place names, person names, time expressions, phone numbers, and email addresses (see Figure 5a). MessageOnTap then puts these extracted key phrases into separate schemes, which are later used to match with task shortcuts (see Figure 5b).

For time expressions, we assume that different levels of time expressions (a precise time, a calendar day, or a longer period of time) are used in conjunction with each other [53], to suggest a more specific time. For example, in messages 1-3 shown in the conversation in Figure 5, "10 am" refers to the 10:00 am at the upcoming Wednesday (in reference to the time of conversation), when there is a calendar day "this Wednesday" in the vicinity to the message containing " 10 am". Also, for time expressions of the same level, a more recent time expression is considered to be more relevant. For example, scheduling often involves users to discuss different time options, while the last mentioned time always tends to be a time that is agreed by both ends of the conversation.

For event expressions, we use event entities detected by the named entity recognition algorithm. We also consider all verb phrases (e.g., "meet you", "talk about the project", etc.) as events, except those which are used as adjuncts. Coreference resolution is also performed if needed, e.g. converting "meet you" to "meet Lisa" if messaging with Lisa.

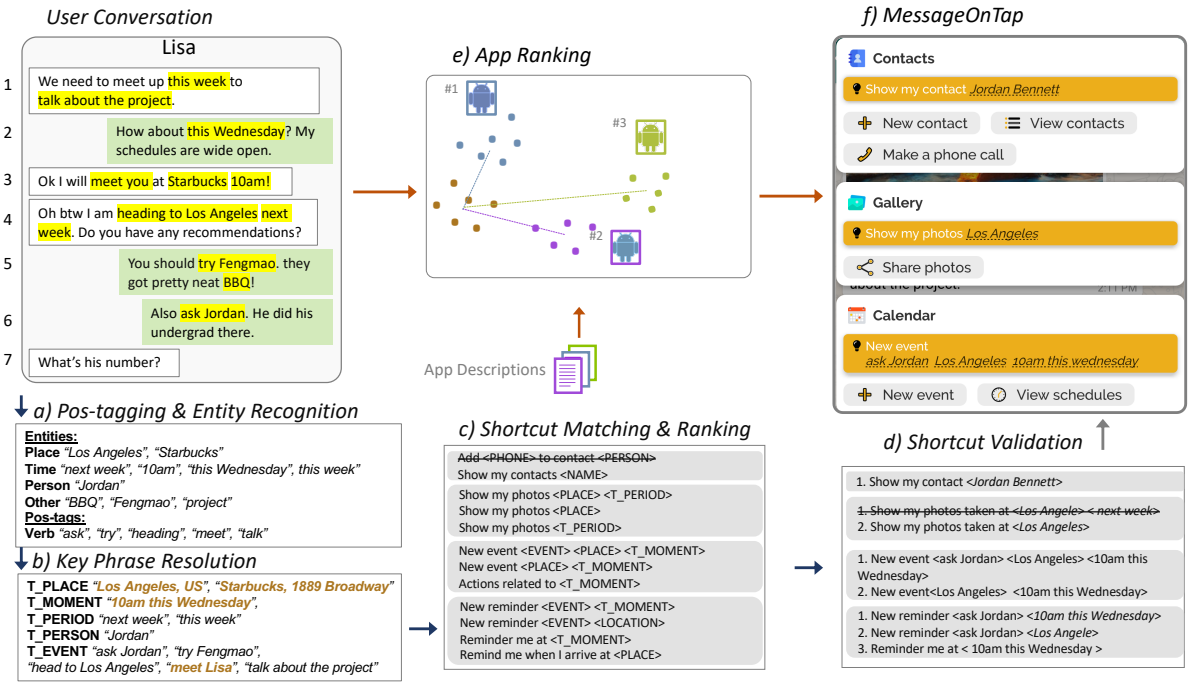

Figure 5: User messages are collected for two parallel analyses: *a-d*) Generating parameterized task shortcuts and selecting the most relevant one for each app, and *e*) ranking external apps according to their relevance to the conversation. The results of the analyses are then integrated to generate suggestive interfaces on the user's phone, constructing ranked cards of apps and the task shortcuts inside of each app card, as shown in f).

For place expressions, MessageOnTap retrieves auto-complete predictions of recognized location entities [34] based on the user's current location, and associates each place expression ("Starbucks") with a more specific physical location ("Starbucks at 1889 Broadway, New York, NY 10023", if the user is nearby). This physical location is used for task shortcuts if the task needs an exact street address for some services, for example, a location trigger for a reminder.

Step 2: Shortcut Matching & Ranking. The selected key phrases from Step 1 are used to match all defined shortcut templates. For example, all listed shortcut templates in Figure 5c) are matched except `Add <PHONE> to contact <NAME>`, because there is no phone number mentioned. Our heuristics for ranking all matched shortcuts are as follows. (1) *Specificity*: a task shortcut which has more matched parameters is more specific and can potentially mitigate more user costs, and so is ranked as more relevant. For example, `Show my photos taken at <PLACE> <T_PERIOD>` would bring users more specific photos to share, as referred in the conversation, therefore, should be ranked higher than `Show my photos taken at <PLACE>`. (2) *Recency*: A task shortcut which is matched using a more recent set of mentioned entities is considered to be more relevant. For example, in Figure 5, the user's conversation contains two topics: *i*) scheduling an upcoming meeting (messages 1-3) and *ii*) discussing Los Angeles (messages 4-7). There are key phrases mentioned in both topics, and thus two possible matches for the shortcut template `Remind me when I arrive <PLACE>`. Because "Starbucks" was mentioned in a more recent message, the task shortcut `Remind me when I arrive <Starbucks, 1889 Broadway>` is ranked higher than `Remind me when I arrive <Los Angeles>`.

Step 3: Validating shortcut parameters. MessageOnTap offloads the language processing in Step 1 and Step 2 to a cloud backend because the related entity recognition operations are computationally intensive. After Steps 1 & 2, the backend returns all ranked task shortcuts to the client for further validation, since not all task shortcuts are valid according to the in-app contents. For example, for a retrieval task shortcut `Show my photos taken at <Starbucks>, <10am this Wednesday>` would not be valid because 1) all photos are taken in the past, and 2) there are no photos taken at Starbucks for this particular user.

App Ranking Algorithm

In MessageOnTap, ranking relevant apps is framed as information retrieval (IR) problem, where the current conversation is the "query" to retrieve relevant app "documents". This approach is based on our messaging behavior study, which found that conversations in messaging apps often contained implicit intents to access external apps. To represent each app, MessageOnTap uses publicly-available app descriptions

from mobile app markets, in our case Google Play. Developers tend to present concise and accurate representations of their app functionality in app descriptions [51].

To retrieve app descriptions based on conversation queries, we perform a "semantic search" [16], using the implicit user intents of user conversation as the "query", which can address the typical challenge of *vocabulary mismatch* in IR tasks [56]. Specifically, MessageOnTap uses *word embeddings* to represent the semantic meaning of individual words, and measure the "relevance" between conversations and app descriptions using cosine similarity between the word embedding centroids of each individual app description and the conversation (see Figure 5e). Trained under the distributional hypothesis in which "*a word is characterized by the company it keeps*", word embeddings are effective vector representations of words, which has been shown to be effective in a wide range of semantic understanding tasks, such as sentiment analysis and topic modeling.

We trained our word embedding model on a corpus of one month's Reddit comments (from June 2018) using Word2Vec [48]. We felt this was a reasonable data set as it is a social site with a great deal of conversation and slang. We eliminated tokens appearing less than 5 times in the corpus, yielding a final model of 914027 words, each of which is mapped to a 100-dimensional word vector. To boost accuracy, before computing centroids, both the bags of words of conversations and app descriptions are re-weighted by inverse document frequency [66], and words in each message from the conversation are re-weighted by recency [45]. The intuition behind this is to consider less-frequent words in more recent messages are more representative to reveal user intents.

5 EVALUATION

Can MessageOnTap help users accomplish message-related tasks efficiently? What benefits and drawbacks does a unified suggestive interface provide over existing task workflow? We performed experiments to evaluate MessageOnTap's reliability of suggestions and ran a study to validate MessageOnTap's design, and compared it to existing message-related task workflow. Following the study, we asked participants about their impressions of the system.

To evaluate our algorithms, we used mean average precision (MAP) and mean reciprocal rank (MRR) as evaluation metrics. Precision is defined as the fraction of retrieved apps (or selected task shortcuts) that are relevant to the number of retrieved apps (or selected task shortcuts), and the average precision (AP) is computed over the precision values, limited to the top k retrieved apps (or selected task shortcuts). The mean average precision (MAP) is then aggregated over the conversation set for all users. The reciprocal rank of a conversation input's result is the inverse of the index of the first relevant app (or the selected task shortcut). In case none of the retrieved apps (task shortcuts) is relevant, the reciprocal rank is set to zero. The respective aggregation over the conversation set is the mean reciprocal rank (MRR).

Evaluating App Ranking

We first evaluated our app ranking algorithm using task-related conversations from our messaging study. We used 706 pairs of conversation snippets plus what subsequent apps were used. Excluding 18 user apps which did not have publicly available descriptions, and 21 apps of game-related categories, our evaluation dataset consists of 675 conversations and 221 different app descriptions. For our experiment, we limited the considered apps to the top k=5 ranked apps, which means we only consider the prediction correct if the actual app used is in the top 5 results of the ranking algorithm. This reflects the number of recommended apps seen in most home launchers (usage frequency based).

We compared our app ranking algorithm against two baselines: *i) Global Frequency*, which simply predicts the next app based on the number of times an app has been opened by a particular user [3]. *ii) TF-IDF*, a standard IR algorithm based on bag-of-words representations, re-weighted by inverse document frequency [54]. Our algorithm (MRR=.81, MAP=.76) consistently performs better than both Global Frequency (MRR=.36, MAP=.31) and TF-IDF (MRR=.41, MAP=.37). This result suggests that a semantic search-based approach is promising in suggesting next apps to users. TF-IDF was only able to match the right app when messages contained the exact words in app descriptions, which our embedding-based algorithm addresses. But our algorithm sometimes failed to infer user goals because it overly relies on literal semantics of conversations. For example, the retrieved apps for a conversation containing "where are you" were all location-related (e.g., Yelp, Maps, etc), but the user goal was actually to use the *Dialer* app, because of an implicit need to reach his contact immediately.

Evaluating Shortcut Selection

To evaluate the performance of selecting parameterized task shortcuts, we created a conversation dataset with 10 conversations each for *Photo Gallery*, *Documents*, *Calendar*, *Reminder*, and *Contacts*. We used the original conversations from our formative study to curate the dataset, but had to add two additional conversations for *Documents* due to a lack of samples. For all conversations, after anonymizing person names, we generated key phrases using the process described by Figure 5 a-b), and formed task shortcuts based on the selected key phrases. Due to a lack of users, we could not perform the per-app shortcut validation process using in-app content (as shown in Figure5d). Instead, we only use simple validation rules to generate reasonably valid task shortcuts. For example, we consider "New event: yesterday"

as invalid. There are in total 655 different task shortcuts generated for the dataset.

To obtain the actual set of task shortcuts which should be considered relevant, we further gathered 150 separate responses from MTurkers to collect the top-3 relevant task shortcuts for each app on each conversation, thus guaranteeing at least 3 responses for each conversation. The voting results were used as ground truth for evaluation. Our resulting dataset consists of 50 conversations over 655 different task shortcuts from 5 supported apps. We only consider the algorithm produces the accurate prediction if one of the selected task shortcuts is considered as relevant by the MTurkers. With an MRR score of .83 and MAP a score of .74, our algorithm performs best when a conversation contains time expressions. Conversations that only contain event or place names tend to perform worse because 1) it can only match shortcut templates that only look for location or event key phrases, and 2) the underlying named entity algorithms fail to extract the correct entities.

User Study

We conducted a preliminary in-lab user study to gauge the usability and effectiveness of MessageOnTap. Our study aimed to compare MessageOnTap to how a messaging app user would normally complete message-related tasks, as well as to evaluate the effectiveness of the shortcut selection algorithms running on personal contents of personal phones. Participants were asked to start conversations in their messaging apps in pairs (sitting by different tables, at the same room), covering two topics. Each topic was associated with one common task that is supported by MessageOnTap, and they were asked to only use MessageOnTap to complete their tasks. Task I is on sharing photos taken at a city previously visited by one participant in the pair, as recommended points of interest for the other participant. Task II is on coordinating an upcoming event a week from the present day. We measured the time it took participants to complete the task. Four pairs of participants participated in the study. Each pair had known each other for at least 6 months, and used a primary messaging app to communicate to each other (2 pairs used Messenger on Android, 1 used WhatsApp on Android, and 1 used iMessage on iPhone). They all used calendar apps on a daily basis to keep track of schedules. None had used MessageOnTap prior to the study.

User instruction. We explained the purpose of MessageOnTap as a task-centric interface to help with message-related tasks, but did not provide details about the provided suggestions, and what kinds of key phrases in conversation will be used to generate. The pair of iPhone users (P1-2) were given two Android phones installed with MessageOnTap and were asked using WhatsApp to communicate. Other three pairs of participants were asked to install MessageOnTap on their personal Android phones using their existing messaging apps. Afterward, users were asked to turn on Accessibility, to allow MessageOnTap to scan messages. For participants who were using personal phones, in each pair, we picked one participant who had previously taken geotagged photos in other cities as the recommender of a city, with the other participant looking for recommendations. P1's given Android phone had pictures taken in Los Angeles and pre-set upcoming schedules, and was selected as the recommender. In Task II, we selected the recommenders in Task I to propose an upcoming event, with the other selected as the one who would use MessageOnTap to check schedules. All participants were asked to add the scheduled event to their calendar using MessageOnTap.

Results. All participants completed the task successfully with MessageOnTap, with the completion time of 7.2s (SD=2.4), 3.1 times faster on average versus the corresponding tasks of calendar lookup and photo sharing in our messaging study. For Task I and II, each pair of participants invoked MessageOnTap 4 times (1 looking for photos, 1 checking schedules, and 2 adding an upcoming event) in total. It took an average 1.9s (SD=0.53) for MessageOnTap to generate the suggestive interface. In the 16 total invocations, MessageOnTap was able to generate the most relevant task shortcut and the most relevant app for participants 14 times. Specifically, MessageOnTap was able to detect and select many relevant key phrases, including "concert", "come to P4's housewarming party", "John's birthday" for <T_EVENT>, "next Tuesday", "Friday next week", "18th", "next week....Saturday" for <T_PERIOD>, "4 next Tuesday", "noon Friday next week", "530 18th", "6pm next week....Saturday" for <T_MOMENT>, and "SF", "Honolulu", "New Orleans" for <T_PLACE>. It failed to extract "career fair" (mentioned by P5) as <T_EVENT> and "Vegas" (mentioned by P1) as <T_PLACE> because the underlying named entity recognition did not recognize "Vegas" as an entity, and "career fair" was not followed by a verb.

In our post-survey, participants reported (in 7-point Likert scales) that they enjoyed the features provided by MessageOnTap (Mean=6, 95% CIs=[5.65, 6.35]), thought it improved their effectiveness in completing tasks on a mobile phone (Mean=5.25, 95% CIs=[5.79,6.70]), and wanted suggestive features like MessageOnTap for messaging apps on their phones (Mean=5.15,95% CIs=[5.76,6.89]).

In interviews following the study, participants mentioned aspects of MessageOnTap that they liked and disliked. All 8 participants noted the convenience of using MessageOnTap due to its relevant suggestions. P1 noted "*I really like that you don't have to type a lot to create a new event. All you need to do is click the top suggestion and it can fill in the rest.*". P6 said "*This is so much faster*". P5 and P6 didn't notice

MessageOnTap failed to recognize the mentioned event name "career fair". Because of the successful extraction of time expressions, both were able to leverage the task shortcut "New Event: 5:30 18th" to successfully create the event. They were also able to use the suggested auto-complete option "career" and "fair" while typing the event name. P6 even commented that "*It's so cool that it can have auto-complete even [though] I didn't type career fair before*". This supports the usefulness of tailoring auto-complete based on text in conversations.

Because MessageOnTap failed to recognize "Vegas" as a city name, P2 did not get a parameterized task shortcut for Task I. This led P2 to go to the "Share photos" task shortcut and scroll to the desired set of photos. After both tasks, P2 commented that he would expect MessageOnTap to detect the city name, and "*[give] me a suggestion to share photos of Vegas directly*". Three participants compared MessageOnTap with their existing messaging apps. P2 compared to iMessage and said MessageOnTap "*is more intelligent*" and "*integrated*", "*I know iMessage supports to connect to other apps, but not like this kind of integration.*" P5 said MessageOnTap "*has more useful features, instead of just stickers*" (compared to Messenger). Four participants pointed out the benefits of unifying suggestions. P7 said the actions are "*all in one place, so I don't have to look for it.*", and also commented on the advantage of confirmations for parameterized task shortcuts, "*it is good to ask for confirmation on the yellow suggestions, because I might need to change certain things, like the event name*".

We also asked participants what they found challenging about MessageOnTap. Two participants mentioned the *interaction style*. Although MessageOnTap's icon is draggable to avoid occlusion, P1 and P3 felt the beer-like icon is "*a bit annoying*" because of it "*blocked the texts*". When asked about how they might improve their experience, P3 noted "*will use it if it is part of the messaging app*". P2 noted that integrating search would further improve its usefulness. Specifically, "*search photos using Vegas*" in the Photo Gallery app.

6 DISCUSSION

Our evaluations suggest that MessageOnTap can reliably present users with quick shortcuts to streamline tasks prompted in real-world instant messaging conversations. This evidence does not, however, suggest that MessageOnTap is immediately ready for mainstream use. Better conversation modeling can help improve relevance in providing task suggestions, and developer integration efforts and end-user privacy concerns can prevent MessageOnTap framework being integrated as part of existing messaging app platforms to achieve task-oriented conversation-awareness for instant messaging. Here we discuss challenges to overcome in these aspects.

Conversation Understanding

Our text understanding algorithms can associate concepts in conversation ("hungry") to app descriptions ("restaurant"). However, as mentioned in Section 5, it cannot infer underlying intents through the literal meaning of the words. The shortcut template matching algorithm sometimes fails to recognize certain key phrases. It also cannot handle negation well. For example, "9-11 am" can be extracted from "I have class at 9-11 am", but the desired task suggestion is to look for schedules other than 9-11 am.

Advanced deep learning language model can improve conversation understanding. For example, contextualized word embeddings [9] can replace existing context-free word embeddings for better semantic representations, and app ranking performance; public knowledge graph [33] can be integrated into the key phrase extraction pipeline, so that phrases that are related to publicly-indexed entities can be recognized; existing template-based matching for parameterized task shortcuts can be replaced with recent RNN-based model [47] to achieve better performance in selecting relevant parameterized task shortcuts.

Finally, we note that suggesting tasks for human-to-human instant messaging conversations is still an under-explored area in both HCI and NLP community. Our future work is to generalize the understanding of IM-related task at a larger scale, and we plan to release a task-oriented conversation dataset to call attention to the problem we present in the current paper.

Privacy

The core of our text processing pipeline (Figure 5) is done in the cloud to reduce latency and battery impact. Despite the secured network, the sensitivity of conversation data may still raise user concerns [18]. There is ongoing research on deploying state-of-the-art deep learning models on mobile devices [42, 52], so we defer localizing language processing to future work.

Our app ranking algorithm is transparent to individual integrated apps, meaning ranking relevance of mobile apps to the conversation does not leak conversations to these mobile apps. However, other features such as auto-complete, and data filters would grant apps with access to the current conversation of users. Although this conversation access is constrained to what is being displayed on the screen rather than the whole conversation history, malicious apps can still take advantage of this information. We believe this is the most challenging factor preventing the adoption of MessageOnTap's concept as platform support for messaging apps.

Extensibility and Interoperability

Though our focus in this paper is more centered on improving end-users messaging efficiency, we touch on the platform aspects of existing messaging apps as well. We build MessageOnTap to be extensible. From an interaction design perspective, MessageOnTap can reduce user costs with even more apps are integrated, because relevant apps are generally displayed on top as a result of our app ranking algorithm. It can also easily work with a number of messaging apps if they follow recommended Accessibility guidelines in Android OS. Our custom code for message extraction only requires 20 lines of Java code for WhatsApp and just 15 lines for Facebook Messenger.

The present MessageOnTap prototype focuses on streamlining task workflow in 5 commonly used apps. Within the current framework, other task-oriented apps can be integrated to offer users task suggestions, by registering shortcut templates per task, as well as the corresponding task interfaces written in HTML. Both can be transmitted to MessageOnTap client runtime through interprocess communication supported by Android AIDL [31]. The amount of developer integration efforts are comparable to existing messaging platforms. To improve interoperability, we envision a future in which task-oriented apps can be integrated into the messaging app with no integration costs. Recent work in programming by demonstration [44] and reinforcement learning [46, 57] show that, in-app task workflow and schema of tasks can be extracted and automated in a way that is agonistic to both developers and end users, which can connect with our conversation-aware suggestion pipeline to offer a zero-integration messaging-app platform for task-oriented app integration. In general, we contribute to HCI by showing how to seamlessly connect conversations with external apps to streamline tasks. Our novel app ranking helps users navigate an overwhelming number of apps, and our shortcut matching reduces needing to repetitively type mentioned key phrases. Our approach can be used to advance task support for other domains, including conversational agents, chatbots, and deep linking.

7 CONCLUSION

Some messages prompt users to do information-related tasks. To better understand user pain points, we conducted a month-long study of messaging behaviors (N=22). Our analysis found that while message-related tasks were not common, they were rather long to complete. We also developed a stage-based model describing different steps in handling messages. Based on these findings, we designed and implemented MessageOnTap, a suggestive interface for smartphones that uses the text in a conversation to suggest task shortcuts to streamline likely next actions. In an evaluation of our app ranking algorithm, we found that it greatly outperformed two baselines. In a user study with 4 pairs of participants, we found that MessageOnTap was effective in handling their freeform messages and useful helping them complete tasks.

REFERENCES

[1] Eytan Adar, Mira Dontcheva, and Gierad Laput. 2014. CommandSpace: Modeling the Relationships between Tasks, Descriptions and Features. (2014).

[2] Apple. 2018. 3D Touch. https://apple.co/2ofbwLI.

[3] Ricardo Baeza-Yates, Di Jiang, Fabrizio Silvestri, and Beverly Harrison. 2015. Predicting the next app that you are going to use. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. ACM, 285–294.

[4] Frank R Bentley, S Tejaswi Peesapati, and Karen Church. 2016. I thought she would like to read it: Exploring Sharing Behaviors in the Context of Declining Mobile Web Use. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 1893–1903.

[5] Ying-Yu Chen, Frank Bentley, Christian Holz, and Cheng Xu. 2015. Sharing (and discussing) the moment: The conversations that occur around shared mobile media. In *Proceedings of the 17th international conference on human-computer interaction with mobile devices and services*. ACM, 264–273.

[6] Karen Church, Denzil Ferreira, Nikola Banovic, and Kent Lyons. 2015. Understanding the challenges of mobile phone usage data. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*. ACM, 504–514.

[7] Karen Church and Nuria Oliver. 2011. Understanding mobile web and mobile search use in today's dynamic mobile landscape. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*. ACM, 67–76.

[8] Benjamin R Cowan, Nadia Pantidi, David Coyle, Kellie Morrissey, Peter Clarke, Sara Al-Shehri, David Earley, and Natasha Bandeira. 2017. What can i help you with?: infrequent users' experiences of intelligent personal assistants. In *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services*. ACM, 43.

[9] 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* (2018).

[10] John Evans. 2018. 10 Useful iMessage Apps for Busy People. https://bit.ly/2QcWG2G.

[11] Ethan Fast, Binbin Chen, Julia Mendelsohn, Jonathan Bassen, and Michael S Bernstein. 2018. Iris: A Conversational Agent for Complex Tasks. (2018).

[12] Ethan Fast, William McGrath, Pranav Rajpurkar, and Michael S Bernstein. 2016. Augur: Mining human behaviors from fiction to power interactive systems. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 237–247.

[13] Earlence Fernandes, Oriana Riva, and Suman Nath. 2016. Appstract: on-the-fly app content semantics with better privacy. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*. ACM, 361–374.

[14] Adam Fourney, Richard Mann, and Michael Terry. 2011. Query-feature graphs: bridging user vocabulary and system functionality. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 207–216.

[15] Google Inc. 2018. App shortcuts. https://bit.ly/2BO6wTk.

[16] Ramanathan Guha, Rob McCool, and Eric Miller. 2003. Semantic search. In *Proceedings of the 12th international conference on World Wide Web*. ACM, 700–709.

[17] Harry Guiness. 2017. How To Turn Off Facebook Messenger's Annoying M Suggestions. https://bit.ly/2T95A2U.

[18] Robert Hackett. 2016. Everything You Need to Know About Google Allo's Privacy Backlash. http://fortune.com/2016/09/22/google-allo-nope/.

[19] F Maxwell Harper, Daniel Moy, and Joseph A Konstan. 2009. Facts or friends?: distinguishing informational and conversational questions in social Q&A sites. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 759–768.

[20] Eric Horvitz. 1999. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, 159–166.

[21] Ziniu Hu, Yun Ma, Qiaozhu Mei, and Jian Tang. 2017. Roaming across the Castle Tunnels: an Empirical Study of Inter-App Navigation Behaviors of Android Users. *arXiv preprint arXiv:1706.08274* (2017).

[22] Apple Inc. [n. d.].

[23] Apple Inc. 2018. iMessage Apps. https://developer.apple.com/imessage/.

[24] Facebook Inc. 2017. Fourth Quarter and Full Year 2017 Results Conference Call. https://s21.q4cdn.com/399680738/files/doc_financials/2017/Q4/Q4-17-Earnings-call-transcript.pdf.

[25] Facebook Inc. 2017. M Now Offers Suggestions to Make Your Messenger Experience More Useful, Seamless and Delightful. https://bit.ly/2o1M2PQ.

[26] Facebook Inc. 2017. Start Plan? No thanks. How do I disable this annoying unwanted planner feature. https://bit.ly/2EQBu17.

[27] Facebook Inc. 2018. Messenger Platform. https://developers.facebook.com/docs/messenger-platform.

[28] Google Inc. 2015. Find info about what's on your screen. https://bit.ly/2rYNgOC.

[29] Google Inc. 2015. Meet Gboard: Search, GIFs, emojis and more. Right from your keyboard. https://bit.ly/2gTvCXF.

[30] Google Inc. 2016. Google Allo - A smart messaging app. https://allo.google.com/.

[31] Google Inc. 2018. Android Interface Definition Language (AIDL). https://bit.ly/2CI8S8x.

[32] Google Inc. 2018. Cloud Natural Language. https://bit.ly/2auUyjq.

[33] Google Inc. 2018. Google Knowledge Graph Search API. https://bit.ly/2fVg2ss.

[34] Google Inc. 2018. Place Autocomplete. https://goo.gl/fcK9tD.

[35] Microsoft Inc. 2016. Microsoft Garage: Hub Keyboard - an app for Android and iOS. https://bit.ly/2LBGrLV.

[36] Samsung Inc. 2018. Meet Bixby - A Smarter Way To Use Your Phone. - Samsung. https://www.samsung.com/us/explore/bixby/.

[37] Tencent Inc. 2018. Experience Mini Programs. https://bit.ly/2QAXJd4.

[38] Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, László Lukács, Marina Ganea, Peter Young, et al. 2016. Smart Reply: Automated Response Suggestion for Email. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, Vol. 36. 495–503.

[39] Amy K Karlson, Shamsi T Iqbal, Brian Meyers, Gonzalo Ramos, Kathy Lee, and John C Tang. 2010. Mobile taskflow in context: a screenshot study of smartphone usage. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2009–2018.

[40] Simon Khalaf. 2014. Apps solidify leadership six years into the mobile revolution. *Flurry, http://www. flurry. com/bid/109749/Apps-Solidify-Leadership-Six-Years-into-the-Mobile-Revolution* (2014).

[41] Joon-Gyum Kim, Chia-Wei Wu, Alvin Chiang, JeongGil Ko, and Sung-Ju Lee. 2016. A picture is worth a thousand words: Improving mobile messaging with real-time autonomous image suggestion. In *Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications*. ACM, 51–56.

[42] Nicholas D Lane, Sourav Bhattacharya, Petko Georgiev, Claudio Forlivesi, Lei Jiao, Lorena Qendro, and Fahim Kawsar. 2016. Deepx: A software accelerator for low-power deep learning inference on mobile devices. In *Proceedings of the 15th International Conference on Information Processing in Sensor Networks*. IEEE Press, 23.

[43] Uichin Lee, Joonwon Lee, Minsam Ko, Changhun Lee, Yuhwan Kim, Subin Yang, Koji Yatani, Gahgene Gweon, Kyong-Mee Chung, and Junehwa Song. 2014. Hooked on smartphones: an exploratory study on smartphone overuse among college students. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*. ACM, 2327–2336.

[44] Toby Jia-Jun Li and Oriana Riva. 2018. KITE: Building conversational bots from mobile apps. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 96–109.

[45] Jia Hui Liang, Vijay Ganesh, Pascal Poupart, and Krzysztof Czarnecki. 2016. Exponential Recency Weighted Average Branching Heuristic for SAT Solvers.

[46] Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. 2018. Reinforcement Learning on Web Interfaces Using Workflow-Guided Exploration. *arXiv preprint arXiv:1802.08802* (2018).

[47] Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23, 3 (2015), 530–539.

[48] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[49] Meredith Ringel Morris, Jaime Teevan, and Katrina Panovich. 2010. What do people ask their social networks, and why?: a survey study of status message q&a behavior. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 1739–1748.

[50] Jordan Novett. 2015. Microsoft beats Google to the punch: Bing for Android update does what Now on Tap will do. https://bit.ly/2SMe6oT.

[51] Dae Hoon Park, Yi Fang, Mengwen Liu, and ChengXiang Zhai. 2016. Mobile app retrieval for social media users via inference of implicit intent in social media text. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 959–968.

[52] Sujith Ravi. 2017. Projectionnet: Learning efficient on-device deep networks using neural projections. *arXiv preprint arXiv:1708.00630* (2017).

[53] Xin Rong, Adam Fourney, Robin N Brewer, Meredith Ringel Morris, and Paul N Bennett. 2017. Managing uncertainty in time expressions for virtual assistants. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 568–579.

[54] Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management* 24, 5 (1988), 513–523.

[55] Bill N Schilit, David M Hilbert, and Jonathan Trevor. 2002. Context-aware communication. *IEEE Wireless Communications* 9, 5 (2002), 46–54.

[56] Hinrich Schütze. [n. d.]. *Introduction to information retrieval*. Vol. 39.

[57] Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. 2017. World of bits: An open-domain platform for web-based agents. In *International Conference on Machine Learning*. 3135–3144.

[58] Choonsung Shin, Jin-Hyuk Hong, and Anind K Dey. 2012. Understanding and prediction of mobile application usage for smart phones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM, 173–182.

[59] Ben Shneiderman and Pattie Maes. 1997. Direct manipulation vs. interface agents. *interactions* 4, 6 (1997), 42–61.

[60] Statista. 2018. Mobile app share of total digital time spent on selected content categories in the United States in June 2017. https://bit.ly/2BNemwO.

[61] Statista. 2019. Average time spent per day with mobile messaging apps by adults in the United States 2015 to 2019. https://bit.ly/2BLkaXD.

[62] Chang Tan, Qi Liu, Enhong Chen, and Hui Xiong. [n. d.]. Prediction for mobile application usage patterns.

[63] David R Thomas. 2006. A general inductive approach for analyzing qualitative evaluation data. *American journal of evaluation* 27, 2 (2006), 237–246.

[64] Christophe Van Gysel, Bhaskar Mitra, Matteo Venanzi, Roy Rosemarin, Grzegorz Kukla, Piotr Grudzien, and Nicola Cancedda. 2017. Reply with: Proactive recommendation of email attachments. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 327–336.

[65] Ning Ye, Ariel Fuxman, Vivek Ramavajjala, Sergey Nazarov, J Patrick McGregor, and Sujith Ravi. 2018. PhotoReply: Automatically Suggesting Conversational Responses to Photos. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1893–1899.

[66] Guoqing Zheng and Jamie Callan. 2015. Learning to reweight terms with distributed representations. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. ACM, 575–584.