Improving Structured Data Entry on Mobile Devices

Kerry Shih-Ping Chang[1], Brad A. Myers[1], Gene M. Cahill[2], Soumya Simanta[2], Edwin Morris[2], Grace Lewis[2]
[1]Human-Computer Interaction Institute, [2]Software Engineering Institute
Carnegie Mellon University
{ kerrychang, bam }@cs.cmu.edu, { gmcahill, ssimanta, ejm, glewis }@sei.cmu.edu

ABSTRACT
Structure makes data more useful, but also makes data entry more cumbersome. Studies have found that this is especially true on mobile devices, as mobile users often reject structured personal information management tools because the structure is too restrictive and makes entering data slower. To overcome these problems, we introduce a new data entry technique that lets users create customized structured data in an unstructured manner. We use a novel notepad-like editing interface with built-in data detectors that allow users to specify structured data implicitly and reuse the structures when desired. To minimize the amount of typing, it provides intelligent, context-sensitive autocomplete suggestions using personal and public databases that contain candidate information to be entered. We implemented these mechanisms in an example application called Listpad. Our evaluation shows that people using Listpad create customized structured data 16% faster than using a conventional mobile database tool. The speed further increases to 42% when the fields can be autocompleted.

Author Keywords
Personal information management (PIM); data entry; structured data; mobile devices; autocomplete

ACM Classification Keywords
H5.2 [Information interfaces and presentation]: User Interfaces – Input devices and strategies.

General Terms
Human Factors; Design.

INTRODUCTION
Various mobile personal information management (PIM) applications have been developed to support people's need to record and consume information while they are on the go. One of the key factors that affects whether users can and will use a PIM application is how easily, quickly and accurately they can *enter* data [4,6,10,11]. The most intuitive way, suggested by prior research, is simply entering data as an unstructured note [4,6,11]. Unstructured PIM applications (e.g., Memo, Notes) let people enter any text they want, and the simple "open and type" interaction is often preferred by mobile users who are in a hurry and can only devote limited attention to the devices [6,18].

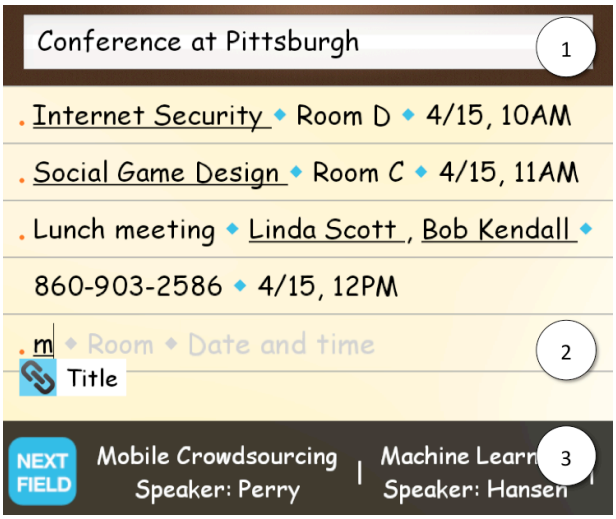

Figure 1. The Listpad Interface. The user is creating a list of to-dos for a conference, where (1) is the title box, (2) is the editing interface, with a label for the type and name of the field currently being edited, and a gray hint text suggesting a possible structure for the current item, and (3) Listpad's autocomplete suggestions are to the right of the "Next Field" button, showing possible values for the current field.

In contrast, structured PIM applications (e.g., Contacts, Calendar) provide a set of predefined fields for users to fill in when creating a new item. This structure increases the usefulness of the data and is highly utilized in these applications to support searches, visualization of the data in calendar and map views, and data sharing. However, entering structured data is much less flexible than entering an unstructured note. Structured applications force users to enter data in predefined formats. They also restrict the kinds of data that can go into the application by their predefined fields, while prior studies have found that a significant amount of what people want to store does not fit into these fields [4]. Most structured PIM tools have limited customizability, and even if customizable, require users to tediously go through multiple screens to select types and names for each field (e.g., see [9,14,20]). These downsides drive users back to entering unstructured data into the "notes" field in these applications or just into a separate plaintext note [6],

UIST'13, October 8–11, 2013, St. Andrews, United Kingdom.

http://dx.doi.org/10.1145/2501988.2502043

but then the data loses all the advantages of structure. The ease and flexibility of unstructured text input motivates us to rethink the design of interfaces for creating structured data on mobile devices. We show that we can bring the benefits of unstructured input to make creating structures and entering structured data more fluid on mobile devices.

Another barrier to effective use of both structured and unstructured PIM applications on mobile devices is the difficulty of typing on today's touchscreen keyboards, which can be slow and error-prone due to the small key size and the lack of tactile feedback [18]. Minimizing the amount of typing can thus be beneficial to users in their interactions with mobile applications. Most smartphone input systems have a built-in dictionary that provides word suggestions for users as they type, or dictionary-based input techniques like ShapeWriter [21] and Swype [19]. This occasionally proves useful, but much of the data entered in PIM applications is likely to not be in the dictionary (e.g., proper names, addresses, etc.). Similarly, speech recognition systems do not work well for entering data into PIM applications because speaking this kind of data is often awkward and error-prone.

An insight that motivates our design is that when users are entering a value, they are often giving the system more information about the data than just the text. For example, the data typed in a "location" field in a Calendar is likely to be some place's name. The data a user types often already exists in some local or online database. For example, the entry in the "location" field is likely to be some place in the current city and could be looked up using a web service such as Google Places. Similarly, the data in a "person" field is likely to be found in that user's contact book or on Facebook or LinkedIn. Another insight is that data is often highly inter-dependent, so that after entering one value, the next value entered might also be about the same thing, such as the phone number of a location. Because entering structured data inevitably increases users' cognitive load compared with entering an unstructured note (even simply reading the structure requires more attention from the user), we hypothesized that we could compensate users by using these local and online public databases to provide *autocomplete suggestions* based on the structure to save users some of the effort in typing.

To demonstrate our approach to addressing these issues, we developed Listpad, a new PIM application for Android that allows users to create customized structured data in an unstructured, notepad-like interface, where users can add structure to the text as they are entering the data (see Fig. 1). By "structure," we refer to both the format of individual fields (like a time or place or plain-text), and also the specific sequence of fields in a record (such as that a record has a time, a name, then a location). Our goal is to design an interface that places minimum restrictions on how users can enter data, so that it is as quick to use as a note application, while still providing the ability to let users easily create and reuse customized structure for data. Furthermore, Listpad eases data entry by augmenting the built-in autocomplete mechanism with a rich collection of local and online data based on the type of the field and the user's context.

In summary, this paper makes the following contributions:

- A novel data entry technique that lets users create and reuse flexible customized structures and enter structured data in an unstructured, notepad-like interface.
- The novel use of public and personal databases as sources to provide autocomplete suggestions, which are then used to provide rich hyperlinking among data items and subsequent contextual autocompletes.
- An instantiation of these techniques in an Android application called Listpad, which was shown in a within-subjects evaluation to be faster compared to a mobile notepad application (AK Notepad [1]) and a conventional mobile database application (Memento [14]). People were 16% faster using Listpad to create customized structured data in the phone compared to using Memento. In addition, with the help of autocomplete, this difference increases to 42% faster, and Listpad was even able to outperform the notepad application by 18% when autocomplete can help with data entry.

SCENARIO

Here we describe a scenario where Jim, a graduate student, is attending a conference in Pittsburgh and uses Listpad to record a list of his activities. We use this scenario to give an overview of how someone could use Listpad. Later we will explain our system in detail.

A day before the conference, Jim wants to make a to-do list for the conference, such as interesting talks he wants to attend, on his phone. Jim notices that the conference provides a mobile app, but the app does not let him add any personal items like his lunch meeting or make any annotations. So he uses Listpad, which provides fully flexible data entry. Jim first downloads the conference programs in an Excel file and imports it to Listpad's database. Then he starts a new list in Listpad. Jim begins by entering some talks he wants to attend. Jim sets Listpad to use the conference program he just imported as the source of autocomplete suggestions. Now as he types, he sees that Listpad searches the programs and suggests talks that match what he has entered. Fig. 1 at 3 shows that just after typing a single letter, Jim finds the full name of the talk among the suggestions. Jim quickly autocompletes the field, and Listpad further suggests other information about the talk, such as the time, location and speaker. Again, Jim uses the suggestions and enters the talk's time and room. Jim enters a few more talks and his lunch meeting and saves the list. By default Listpad displays data in a list view (Fig. 5 at 1). Jim chooses to display his list in a day view (Fig. 5 at 4), where he can view the talks by their times of the day in a calendar. Later while listening to a talk, Jim thinks of a cool

idea. He opens his list in Listpad, and quickly adds a new field to the talk and types in the idea.

During the conference, Jim talks to his friends and finds that several universities have parties at night. He wants to quickly write down the times and locations of the parties. Jim opens the same list in Listpad and starts adding the party information. When he is about to enter where the party is held, he links Listpad to Google Places, an online location database. Now as he types, Listpad uses Google Places as the source to provide autocomplete suggestions (Fig. 3 at left). The places returned are sorted by their distances from his current location. Again, just after a few taps Jim finds the place he wants among the suggestions, and using the autocomplete he quickly fills in the name, address and phone number of each place. Later in the day, Jim opens Listpad and views his list in a map view (Fig. 5 at 2), which plots the restaurants on a map using their locations.

RELATED WORK

The design of our data entry technique is inspired by prior studies on how people record information in real life. Research has found that a large amount of information that people record is in the format of a *list*, which consists of multiple items with similar structure. Examples are to-do lists, desired items, login/password information, etc. [2,4,13]. Therefore, Listpad particularly focuses on supporting mobile entry of lists of this kind of information. The need for structured data has been well addressed in [11], as the authors argue that structured data is the "*lingua franca*" of many daily applications from music players to productivity applications, all of which rely on specific data structures to provide their services. Most complaints about structured PIM tools are due to two reasons. First is that these tools are often too rigid and lack customizability as to the kinds of data that can be stored. Studies have found that much personal information is unable to fit into the existing structures provided by applications [4]. In order for the data to be storable, the user sometimes must pick an arbitrary field in which to put the data, even though the description of the field does not match the data [6]. As a result, a key feature we designed in our mechanism is to let users define customized structures to fit their needs. Also, studies reported that people naturally tend to record information tersely, often using abbreviations or partial sentences [2,13], whereas most structured tools require rigid and full entry of data or force users to enter data through standard GUI widgets.

Another problem identified in the literature is that entering structured data requires more cognitive load for the users because they have to read extra GUI elements [11] and enter the information into the rigid, structured format, which users may want to avoid if they need to enter data in a hurry [6]. Therefore, Listpad lets users type in data in their preferred formats and uses flexible data detectors [12,17] to extract the values. For example, Listpad recognizes date information in different formats, such as "Sept. 19" or "9/19", and does not force users to enter dates through date input widgets. Listpad also lets users link self-defined fields to local or online databases which are then used for autocomplete suggestions. Many mobile systems and computer text editors (such as Emacs) provide autocomplete using dictionary words or the users' typing history. Listpad's autocomplete is different in that its autocomplete suggestions come from data sources that are relevant to the user-defined structure, and can be from on-line web services. For example, the autocomplete suggestions for a "book" field could be book names from Google Books.

The idea of letting users enter structured data using unstructured text has been explored in several prior systems for desktop computers. Inky [15] is a command line tool for entering data quickly into well-known structured desktop PIM applications such as Email or Calendar. The interface provides graphical feedback to suggest what field the data will go into. Listpad focuses on a different problem, which is to facilitate creating *customized* structured data instead of entering data into existing structures in other applications. Jourknow [11] lets users enter plaintext notes and automatically adds context-related information to the note, such as when and where it is created. Similar to Inky, Jourknow can extract structure from plaintext and use it as input for well-known structured PIM tools based on predefined grammars, such as creating a calendar event by typing "Meet David at 3pm". Again, our work is different from Jorknow as we focus on the creation of customized structured data. Jourknow also allows users to create customized structured data using Notation3 syntax [3], which uses many extra punctuation characters. We feel that such syntax is not intuitive to learn and also is not suitable to use on mobile devices because it would require much extra typing and frequent switching between the letter and symbol keyboards.

Mobile database applications are designed for creating customized structured data on mobile devices. Examples are commercial tools such as Memento [14], Tap Forms [20] and HanDBase [9]. All of these use a traditional form-based interface and a similar procedure to set up the database as is used on desktop computers: they require the user to first specify the type and name of each field in the database, and then use entirely different screens to enter the data items. Users can add or delete a field after the database is created, but the changes will affect all the items in the database. In contrast, Listpad lets users do all these operations in the same editor and uses data detectors to help identify the type and name of a field, so that users do not necessarily have to specify everything explicitly. Listpad also allows users to flexibly add or delete any field in an individual item without affecting other items. Most of the mobile database applications allow users to build links to cross-reference to other entries in its database. Selecting a desired entry requires the user to go through multiple form interfaces. In Listpad, we make such linking more fluid and extend the idea to include connecting to online web services. To make the interface more integrated, we use these entries as autocomplete suggestions for the user while typing in the data.

Despite having many shortcomings, mobile database applications are apparently successful commercial products. For example, Memento, which we used for our user study, has between 100,000 to 500,000 installs on Google Play, showing that there is a demand for entering and viewing structured data on mobile devices.

LISTPAD

In Listpad, we define a *list* as a collection of *items*. Each item has one or more customizable *fields*. Whereas most lists have a fairly uniform structure, this is not a requirement – in fact every item could have a different structure. In the scenario, Jim's list has items of three different structures: one for presentations having the fields *time*, *talk*, and *room*, the lunch meeting with *event*, *people*, *phone number*, and *time*, and some items about parties with *time*, *place*, *address* and *phone*. All lists are created and edited in a notepad-like interface (Fig. 1 and Fig. 2). The title of the list can be added or modified through the textbox at the top. Below this textbox are all items in the list. Each item starts with an orange bullet, which is automatically displayed when the user enters a new line. Within an item, fields are separated by the blue diamond symbols, which are inserted using the blue "Next Field" button (Fig. 3 at 1).

Just as if it was a plaintext editor like Notepad, the user can edit the text freely, for example to delete a diamond symbol or a new line anywhere using the regular delete key, or moving the cursor and adding a diamond or new line anywhere. This would be useful to split a string into two fields, or delete a return to merge two items into one. We designed this feature to make creating structured data a very flexible process. The user can always type in unstructured data if in a hurry, or the user might copy and paste unstructured data from the web or elsewhere. Later, the user can add structure to the data by inserting the blue diamonds and then viewing and editing the list in the structured layout.

The decision to use a special character (the blue diamond) as the separator for fields was made to avoid the use of more common characters that often appear *in* data. For example, many data types use commas internally, such as names ("Olsen, Jr.") addresses ("3rd Street, Apt. 3"), and dates ("Oct. 7, 2012"). Rather than requiring users to disambiguate and correct possibly erroneous parsing or use unfamiliar special characters (as in [11]), we decided to add an easy-to-hit and unambiguous button to separate fields.

Listpad uses a label under the text that the cursor is at to show the type and name of the field (Fig. 2 at 1). The label icon represents the field type, and the label text is the field name. The default type and name are "text" and "Field *n*", where *n* is the number of the field in an item (Fig. 2 at 4). However Listpad will detect and change the type and name based on the value that the user types (see next section). Alternatively, the user can tap on the label to bring up a dialog box, where the field type can be changed through a dropdown list and the field name can be edited (Fig. 2 at 2).

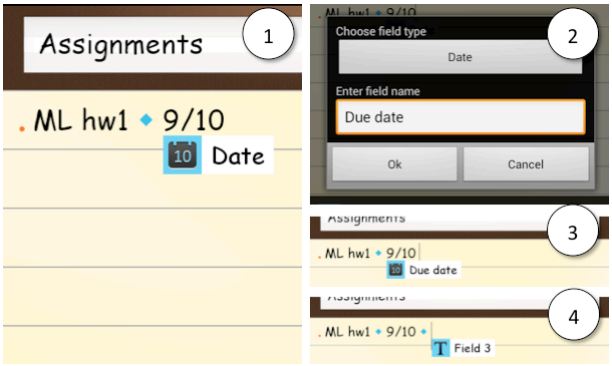

Figure 2. (1) The user is editing a list that has one item with two fields. The label shows the type (the calendar icon) and name (the label text) of the second field, both being "Date", because Listpad detects that the field value ("9/10") is in a date format. The user could tap on the label and bring up (2) the dialog box to change the field type and name manually. (3) The label updates after the user changes the field name to "Due date". (4) The user presses on the "Next Field" button to start a new field. The label moves to the new location and shows the default field type "Text" and name "Field 3".

When the user moves the cursor to another field by either tapping on it or inserting a diamond symbol to start a new field, the label moves to the new location and shows the new current field type and name (Fig. 2 at 3 and 4).

Data Detectors

Listpad currently supports 10 different data types: *text*, *number*, *phone*, *date*, *time*, *date and time*, *address*, *email*, *website*, and *link*. Except for "link" (see the next section) and text, for the rest of the eight data types Listpad provides corresponding data detectors to try to recognize the type of the current field value to see if it matches any of the predefined formats. When the system detects a type format other than a basic string, it updates the field type and name on the label in real time. Note that this use of data detectors is quite different from current smartphones or prior research [12,17], where the type is used to provide actions when tapping on the data in a plaintext note. In Listpad, the inferred types are persistent, and are used to define the structure for fields. (Tapping on the fields can also still invoke actions in Listpad, as discussed below.)

The default field name for a field is the name of the type (e.g., Fig. 2 at 1). Sometimes the default names are sufficient. For example, in our scenario when Jim enters the time of a talk, he just uses the default field name "date and time". If the user manually assigns a type to the field using the dialog box, the data detectors will no longer run. If the value that the user enters disagrees with the manually-specified type format, Listpad assumes that the user wants to consider the new value to be of the specified type, and simply allows it. For example, there might be a phone number format that Listpad currently does not recognize, such as an international number. This is an example of a design decision in the direction of flexibility and ease of entry. Of

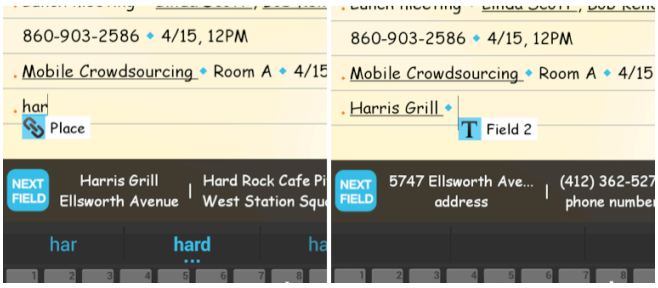

Figure 3. (Left) After the user sets the field to link to Google Places, the default field name becomes "Place", and search results from Google Places are shown as autocomplete suggestions. (Right) When the user starts a new field, Listpad shows options for the user to import more information about the selected place from the linked database.

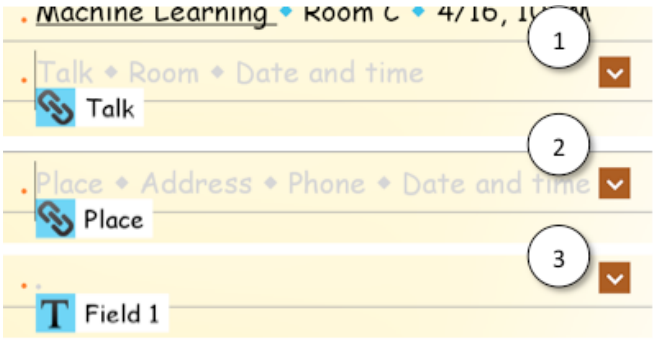

Figure 4. (1) When the user starts a new item, Listpad uses gray text to suggest a possible structure for it. (2) The user can tap on the brown button to switch to a different structure, or (3) choose a blank structure to start define a new item.

course, *uses* of that item will be affected if Listpad does not understand the format. For example, unrecognized dates and times will not be shown in the calendar view. But some uses still work. For example, tapping on a phone number will always open the dialing application and attempt to dial that number, even if Listpad does not recognize the phone number format. In the future we plan to make the data detectors customizable by the users.

Autocomplete Suggestions

In addition to data detectors, another process that constantly runs in the background while the user types is the autocomplete engine that generates suggestions. Android has built-in autocomplete suggestions using a dictionary that comes with the default keyboard (Fig. 3 at left, the blue text). Listpad's autocomplete suggestions are placed right above Android's autocomplete suggestions and next to the "Next Field" button (Fig. 1 at 3 and Fig. 3), and can be scrolled horizontally left and right to see more suggestions. The user can choose from either list. Listpad's autocomplete suggestions come in the following categories.

Built-in Keywords

Listpad has a set of built-in keywords that are used by the data detector to recognize certain types, such as date keywords ("January" ~ "December", "Monday" ~ "Friday", "/"), time keywords ("AM", "PM", ":"), and address keywords ("Road", "Street", "Avenue"…). These are added to the autocomplete list to facilitate entering these types of data. For example, "AM" will be shown if the field value seems to be in a 12-hour clock format. Considering the fact that mobile information needs are highly contextual, we add three more "context keywords" in the autocomplete word list, which are "today", "current time" and "current location". These words, when in the suggestion bar, are shown in blue to signify that they are not the actual text that will be inserted into the field.

Values of Fields in the Same List

The same fields in items in the list may reuse values entered into that field in previous items. For example, when entering an assignment list, the value in the *course* field may be repeated in multiple items because a course can have multiple assignments. Listpad searches for fields in other items that have the same field type and name as the current field, and includes their values as autocomplete suggestions.

Local and Online Databases

As mentioned earlier, in Listpad we explore the possibility of using local personal and online databases based on our observation that data is often highly interdependent. The idea is that if the data is already somewhere in another database, then the user should be able to simply reference the value instead of retyping it. Unlike the other autocomplete engines already discussed that look for possible suggestions independent of the field types, the autocomplete engine for local and online databases is only activated when the user has set the field type to "link" to a particular data source. This design decision was made for two reasons: first, sending a request to the databases on every keystroke would use up system resources because it requires Internet access. Second, searching through multiple on-line databases using the incomplete string that the user enters might return many irrelevant results. Asking the user to pick from these results may cause more cognitive load than just typing in the value. Therefore, we thought it was an appropriate tradeoff to require the user to help Listpad narrow down the search space to a particular kind of data so that Listpad can present accurate suggestions.

To set up the link to a local or online database, the user first sets the field type to "link" using the dialog box shown in Fig. 2 at 2, and then selects the desired source name from a list of available sources in a popup menu. After the user picks a source, the default field name changes to match the source (e.g., Fig. 3 at left). Then when the user types in the field, the field value is sent as the query string to the connected data source every time that the user enters a character, and the returned search results are shown as autocomplete suggestions.

Listpad uses an extensible architecture that allows new data sources to be added, often without programming [5]. Some example sources we have used with Listpad include two local – Android's People (the contact app) and Listpad's

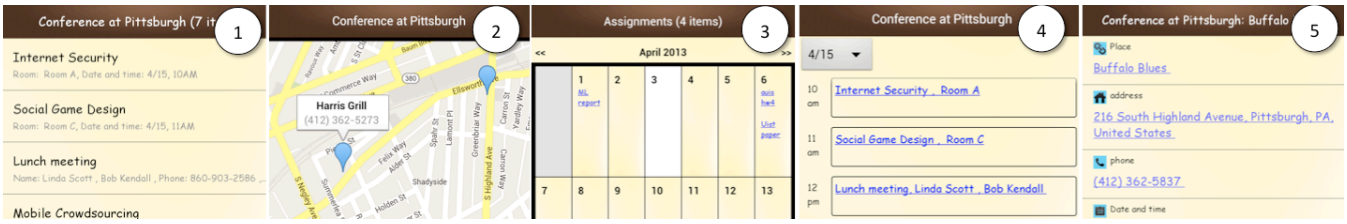

Figure 5. Listpad supports viewing a list in (1) a list view, (2) a map view, (3) a calendar month view, (4) a calendar day view, and a note view (not shown here, which displays a list as what it looks like in the editor). Clicking on an item entry in (1), (3), (4) or a marker in (2) opens (5) a detail view that shows all fields in the item, including the field types and names.

own databases - and five online databases – Google Places, Rotten Tomatoes (a movie database), Last.fm (a music database), Google Books, and Wine Searcher. The users can add new sources for autocomplete to Listpad in two ways. First, the user can import Excel files to Listpad's database, as Jim does in the scenario. Second, if the source is a remote online database, Listpad's architecture allows it to be added to the system as a plug-in. Furthermore, Listpad provides a web configuration tool, called Spinel, which allows end-users to create these plug-ins for Listpad without writing any code. The design of this architecture is explained in detail elsewhere [5]. Here, we will only briefly introduce it in the implementation section below.

After the user selects an autocomplete suggestion, Listpad not only inserts the full string for the user, but also provides autocomplete options for *other* fields of the selected item when the user moves to the next field (Figure 3 at right). If the user taps on a suggestion, not just the field value but also the field type and name will be autocompleted. Besides helping the user quickly enter other related information of an item, these autocompletes also serve another purpose: autocomplete has become a popular way for users to *explore* an information space and therefore serves as implicit queries [16]. For example, if a user has just entered "Bob" in the first field of an item, Listpad will then provide in its autocomplete menu the other information that it knows about Bob as possible values for other fields, which could help users understand what information is available and also helps them ensure that they have the right person without having to leave Listpad.

Entering a New Item and Reuse of Existing Formats

The user can press the "return" key to start entering a new item. Gray text appears on the new line to suggest a possible structure of this new item, based on the existing items in the same list (Fig. 4 at 1). The user can tap on the brown arrow button to cycle among the different structures that have been used for other items in this list (Fig. 4 at 2), and the last choice is a blank structure (Fig. 4 at 3) to start to define a new structure.

Viewing a List, Sorting and Searching

In addition to helping with the *entry* of the data through autocompletions, the structure of the data also enables Listpad to provide multiple views and sorting of the items, and actions when a user taps on items. Listpad provides:

1) A list view, where each item is displayed in a list layout showing its first field as the title and the rest of the fields in a line of small gray text under the title (Fig. 5 at 1). The user can sort the items in alphabetical order, number order (if the items have *number* fields), time order (if the items have *date* or *time* fields), and distances from the current location (if the items have *address* fields).

2) A note view, which shows the list just as in the editor. We provide this view in case the user only wants to enter unstructured notes. This makes Listpad fully capable of being used as simply a traditional notepad application.

3) A map view, where each item with an *address* field is plotted on a map showing its first and second non-address fields as a popup when the marker is tapped (Fig. 5 at 2).

4) A month view, where each item with a *date* or *date and time* field is displayed in a month calendar (Fig. 5 at 3), showing its first non-date field as the title.

5) A day view, which is similar to the month view, but for a single day. It displays items with a *time* or *date and time* field in a day calendar (Fig. 5 at 4).

In the list, map, month and day view, tapping on an item or a marker opens a detail view (Fig. 5 at 5) that shows all fields in the item, including the field types and names, in a list layout. In this view, Listpad provides actions on various types of data. For example, phone numbers are linked to the dialing application and addresses are linked to Maps. A link field is linked to its source item. For example, Place names from Google Places are linked to their Google Places webpage. Person names from the People contacts list are linked to their record in People.

Finally, Listpad currently supports free text search through all items. We are adding structured searching and filtering to the system, so that users will be able to search on individual fields or filter based on the structured value, such as only viewing places near me.

IMPLEMENTATION

Data structures

One of the challenges in implementing Listpad was to design an efficient database structure to store highly-dynamic, yet flexibly-structured data. We used db4o [7], an open source object-oriented database system, to host all of our data. In the database, each item in a list consists of a *value*

object and a *format* object. The value object holds an ArrayList of the field values in an item. Each value object has an ID that links it to a format object that has an ArrayList of types and names that specify the types and names of the fields in the value object. Every time a new item is created, the system searches in the database to see if there is any existing format object that is the same as the newly created format object. If so, the system links the newly created value object to the existing format object, then stores only the value object. If not, then the system links the newly created value object to the newly created format object, and stores both of them in the database. Listpad uses Lucene [8] to do indexing and searching of text.

Adding new online data sources

Listpad accesses all online data sources through their official web APIs. Listpad uses two types of APIs for each source. One is a "search" API that searches the database by the query string. The other is a "detail" API that retrieves the full record of an item given its ID returned in the search result. To enable new data sources (new APIs) to be added dynamically without recompiling the system, each data source is stored as a separate JSON file that describes the usage of its APIs, including the authentication parameters, the request URLs, and the paths to the desired fields from the returned models. We called this JSON file a data source "plug-in". All plug-ins are put in a designated folder on the mobile device. When Listpad starts, it reads all plug-in files inside that folder to determine what sources are available. Installing a new data source requires simply putting a new plug-in file into that folder on the phone.

Currently, this plug-in architecture supports data sources that provide REST web APIs and return JSON data. Because the plug-in itself is just a JSON file, anyone who is familiar with how web APIs work can create a new plug-in using any text editor. We also provide a web-based configuration tool that lets people create a new plug-in using a web-based GUI without programming. Full details are described elsewhere [5].

USER STUDY

We evaluated the design of Listpad using a within-subject study. A key innovation in Listpad is allowing users to add structure to data on the fly while typing in a notepad-like interface. Therefore, we thought it would be interesting to compare Listpad with two different extremes – a conventional mobile database application that requires the structure to be defined beforehand and uses standard form interfaces for all editing, and a plain-text notepad application in which users only enter unstructured data. We hypothesized that Listpad would be slower than the notepad when structure was being created, but the time could be compensated for when Listpad could autocomplete the data by linking to a relevant source using the structure. We selected the Memento Database [14] for the mobile database application and AK Notepad [1] for the notepad application because of their high ratings on the Google Play marketplace. All participants used the experimenter's phone for the study, which is a Samsung Galaxy Nexus. Participants used Android's default keyboard, with its autocomplete using the built-in dictionary, to enter the data in all three applications.

Tasks

We designed five tasks to assess different aspects of our prototype system. All tasks relate to entering structured data into the phone. The data was presented to participants in a table on paper. For Listpad and Memento, participants had to enter an exact item as given, with the same field name, type and value for each field in the same order. If the field structure did not exist, participants had to create the structure themselves. For AK Notepad, however, because there was no notion of items and fields in the notes, we asked participants to ignore the field type and field name and simply enter the field values in order, to use commas to separate each field, and to press "return" to start a new item. From our observations, this is a common format when people create a list of things in their own notes.

The first task was to add two items to an existing "Account and Password" list. For both the first and second tasks, all needed structure was already set up in the existing list and therefore participants did not need to create or modify any of the structure. In the first task, participants typed the values into the correct fields. We used this task as a warm-up task and to ensure that participants understood Listpad's way of specifying structured data. The second task was to add two items to an existing "My CDs" list. This task was similar to the first one, but this time, Listpad linked the fields to Last.fm to provide autocomplete suggestions. We hypothesized that participants would enter data faster with Listpad, since this time the data could be autocompleted. The third task was to create a new list named "My shopping list" that has two items in it. This task required participants to create the structure in Listpad and Memento and manually enter the values. We hypothesized that with Listpad, participants would spend less time creating customized structured data than with Memento. The fourth task was to create a new list named "Restaurants" that has two items. This task was similar to the third one, but using Listpad participants only needed to set up the first field as a link field to Google Places, and the rest of the fields could be autocompleted. We hypothesized that Listpad would be much quicker than Memento with the help of autocomplete for creating the structure. The fifth task was to modify the "Restaurants" list by adding two new fields to it without the help of Listpad's autocomplete. We hypothesized that Listpad would help participants be quicker when modifying an existing structure compared to Memento. For each task, we designed three sets of data for the participants to enter using the three different tools. For tasks one to four, each data set had 82~87 characters. For task five, each data set had 27 characters. We randomly assigned datasets to tool to guard against any differences. We found no significant dif-

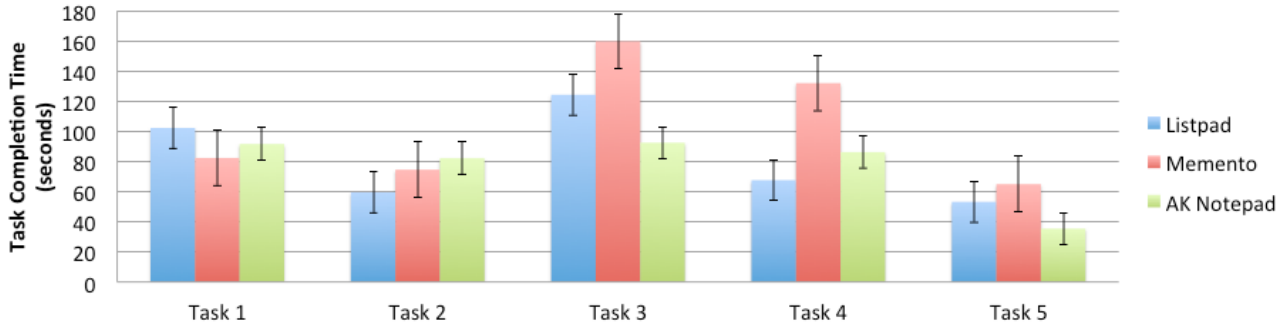

Figure 6. The average task completion time of Listpad, Memento and AK Notepad. Shorter bars are better.

ferences in the time that participants spent on each task entering different data sets, showing that we were successful at matching difficulty among the datasets.

Participants

Fifteen paid participants (ages 22-35) were recruited, all students or staff in our university. All participants used Android phones as their primary phones, so they were familiar with the Android UI in general. Participants rated their expertise with using computers and Android phones on a 7-point scale from "no experience" to "very experienced". The average rating for using computers for all participants was 6.67 and for Android was 5.6. None of the participants had used Memento or AK Notepad before. However, all participants had used other notepad applications, at least on regular computers, and they were all familiar with the form input interface because it is also used in other common Android applications such as People and Calendar.

Procedure

We randomized the order of the tools and the order of the data sets given to each participant. For each tool, the participants first received a short tutorial that covered how to create a list. We did not show participants the features in the tool that were irrelevant to the tasks, such as sorting and searching. The participants practiced with the tool as part of the tutorial. The data for the practice tasks was presented in the same table format as the real tasks, but the data structure was completely different. The whole tutorial took about 15 minutes for Listpad and Memento and 8 minutes for AK Notepad. After the tutorial, the participants began the first real task, as described above. The experimenter measured the time participants spent on each task. Participants were told to work quickly and carefully, and to enter the exact words that were on the sheets, except that they did not have to worry about upper and lower case. If the participants mistyped anything, they were asked to correct it. The correction time counted as part of the final task completion time. After finishing all five tasks, participants moved to the next tool, received the tutorial and did the tasks. After finishing all three tools, participants filled out a short survey to provide feedback. The study took between 60 and 75 minutes, depending on how fast the participant could type.

Results

Across all tasks, in addition to *performance* (time) we were evaluating the *usability* of the three tools by watching for any breakdowns and asking participants for their preferences. A paired-samples t-test was conducted to compare the time that participants spent on each task using different tools. The results are reported below and in Figure 6.

Task 1: Participants completed the task faster with Memento (M=88.47s, SD=19.78), than with AK Notepad (M=93.74s, SD=17.7) or Listpad (M=101.47s, SD=21.75). The difference between Memento and Listpad was significant ($t(14)=3.19$, $p<0.01$). We observed that some participants still appeared to be learning how to use Listpad during this first task. Because all participants were already familiar with entering data in a form interface, we suspect that the difference between Memento and Listpad on the first task was likely due to this learning effect. In all subsequent tasks, times in Listpad matched or beat Memento. The difference between Listpad and AK Notepad was not statistically significant.

Task 2: Participants spent significantly less time when using Listpad (M=56.59s, SD=9.6) than using Memento (M=72.3s, SD=20.31; $t(14)=5.55$, $p<0.01$) and AK Notepad (M=76.52, SD=22.63; $t(14)=5.31$, $p<0.01$). This result shows that participants understood how to use Listpad's autocomplete suggestions, and were able to use autocomplete to cut down their data entry time by an average of 24%. There was no significant difference between the Memento and AK Notepad times.

Task 3: Participants completed this task significantly faster using Listpad (M=131.62s, SD=37.58) than using Memento (M=156.14, SD=29.89; $t(14)=3.76$, $p<0.01$) by 16%. The result shows that the way Listpad lets people define structure while entering data could significantly shorten the time needed to create customized structured data. Both Listpad and Memento were slower than AK Notepad, where no structure is needed (M=98.82s, SD=25.47; $t(14)=4.89$, $p<0.01$; $t(14)=12.23$, $p<0.01$). The result confirms our hypothesis that specifying structure does require extra time.

Task 4: We found that Listpad's autocomplete significantly shortens the time participants needed to create and enter

structured data. Participants were 42% faster using Listpad (M=70.75s, SD=9.73) than using Memento (M=120.95s, SD=30.37; $t(14)=9.01$, $p<0.01$) and 18% faster than using AK Notepad (M=82.62, SD=20.04; $t(14)=3.04$, $p<0.01$). The difference between Notepad and Memento was also significant ($t(14)=10.08$, $p<0.01$). The result shows that participants with the help of autocomplete could create structure and enter data even faster than creating an unstructured note that only contains the values of the fields. Although in order to get the autocomplete suggestions with Listpad, participants had to set up the structure for the first field, the time later saved by the autocomplete on subsequent fields was considerably more than the time spent defining the first structure.

Task 5: Participants completed this task significantly faster using Listpad (M=55.32s, SD=11.81) than using Memento (M=65.27s, SD=13.3; $t(14)=3.75$, $p<0.01$) by 15%. Participants spent less time using Notepad (M=29.27, SD=10.9) than both Listpad ($t(14)=8.78$, $p<0.01$) and Memento ($t(14)=13.32$, $p<0.01$). The result is consistent with the result in Task 3: Listpad helps people modify structure significantly faster than Memento, but it is still faster to have no structure at all.

Subjective Results

In the survey we asked participants to rate the three tools on a 7-point scale from "very hard" to "very easy" in various scales, including ease of *learning* and *entering and editing data*. Participants rated AK Notepad as the most easy-to-learn tool. It received higher rating (M=6.13, SD=0.92) than both Memento (M=4.87, SD=1.3; $t(14)=5.10$, $p<0.01$) and Listpad (M=5.2, SD=1.42; $t(14)=2.43$, $p=0.03$). For ease of entering and editing data, participants rated Memento as the hardest-to-use tool. It received lower rating (M=4.73, SD=1.49) than both Listpad (M=5.93, SD=1.39; $t(14)=2.55$, $p=0.02$) and AK Notepad (M=5.47, SD=1.25; $t(14)=2.13$, $p=0.05$). Listpad was rated highest, but the difference between Listpad and AK Notepad is not significant.

Finally, we asked participants to rate how they liked the tools on a 7-point scale from "don't like it at all" to "like it a lot". Participants rated Listpad (M=6.21, SD=0.89) higher than both Memento (M=4.6, SD=1.18; $t(14)=4.46$, $p<0.01$) and AK Notepad (M=4.21, SD=1.37; $t(14)=4.16$, $p<0.01$).

DISCUSSION

The study confirms several interesting possible use cases for Listpad in real life. All participants reported Listpad's autocomplete as the feature they liked the most among the three tools. They described the autocomplete to be "fast", "saved a lot of typing", and that it "makes finding and entering data a lot easier". On average, Listpad received the highest rating in "the ease of entering and editing data". Although we told participants not to worry about the upper and lower case of the data, many participants still tried to have the letters correctly capitalized. One person insisted on entering all the upper and lower case letters correctly. He later wrote in the survey that the *correctness* of the data was more important to him than the speed of entering data, so he really liked the autocomplete.

The main usability issue we found with Listpad was the delay caused by the network connection when showing the suggestions from online databases. The delay was not a problem for most participants, as their average speed of entering data was still faster using autocomplete even with the delay. We did find one participant who could type so fast that he had to pause for a noticeable amount of time to wait for the suggestions to show up. However, that participant still liked the autocomplete feature and said it was "fun to hit a button and have the whole address just pop up."

10 out of the 15 participants reported that they had notes or lists of items on their own phone. Participants' notes covered a wide range of topics. The most common ones were to-do and shopping lists, followed by lists of ideas, expenses, movies, books, songs/albums, food, contact details, reminders, login/password info, discount cards, favorite quotes and recipes. We were excited to see these examples because many of them include public information and could be easily entered and put into structures by using autocomplete from available web services. For example, information in a shopping list could be autocompleted using the data from Amazon or grocery store web data sources.

Participants liked that in Listpad they could create structure as they were entering the data, which they said was easier and more intuitive than Memento, where they had to use separate interfaces for entering data and creating the field structures. All participants successfully utilized the data detectors in Listpad to set up the structure, which saved time compared to Memento where they had to manually assign both a type and name for every field. Participants expressed interests in using Listpad in real life and suggested practical features they would like, such as connecting Listpad to Google Drive and grouping lists with folders.

Limitations

Our study has several limitations. First, the study was run in a quiet lab setting, which was different from real life where people are often distracted or interrupted by the environment. However, we suspect that this would affect all three tools and therefore would not change our main study findings. Second, in the study, we asked the participants to enter the exact same words as given. This requirement was necessary in order to compare data entry times, but it also violated the reality that people sometimes use abbreviations when entering data on mobile phone. For the tasks where Listpad did not provide autocomplete (tasks 1, 3, and 5), we believe this again would equally affect all three tools and should not change the comparative findings. For the tasks where Listpad did provide autocomplete (tasks 2 and 4), the time differences between Listpad and the other tools may decrease if abbreviations were allowed. However, it would not change our main findings about autocomplete, which is

that it ensures that users can enter *correct* and *complete* data quickly. Finally, all our participants were fluent with typing on the touchscreen. For novice users who do not type on touchscreens well, using more GUI-like input widgets (such as a calendar widget) to enter some data may be faster and easier. We are considering adding a way to popup these kinds of widgets from Listpad's autocomplete bar.

CONCLUSIONS AND FUTURE WORK

We have presented a new data entry mechanism that facilitates creating customize structured data on mobile devices. It allows users to define structure while entering data using an unstructured, notepad-like interface, and uses data detectors to help determine the structure. The design was shown to be usable and more efficient than conventional mobile database applications and was preferred by experienced smartphone users. In addition, autocomplete using local and online data sources lets people create structured data even faster than entering unstructured notes in a note application.

Future work can be in many directions. First, we can extend the Listpad to allow storing multimedia data such as images and audio. Second, we can explore ways to automatically identify relevant local or online databases to use as autocomplete sources. One possibility is to use the list titles or tags. For example, lists having a "people" tag by default could link to Facebook's database. Third, we can allow syncing personal lists with their linked data sources after the lists are entered to keep them up to date. Using the conference scenario for example, if Listpad is linked to an online conference program database and one talk is moved to another room, the room information in user's personal list will automatically get updated. Finally, we can explore using Listpad-like data entry for conventional PIM applications like contacts. We feel that these interaction mechanisms are a starting point on the way to making many different kinds of data entry on mobile devices more efficient.

ACKNOWLEDGMENTS

We would like to thank Ruogu Kang for her help with the statistics for the paper. This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. This material has been approved for public release and unlimited distribution. DM-0000293

REFERENCES

1. AK Notepad on Android. https://aknotepad.com/
2. Bellotti, V., Dalal, B., Good, N., Flynn, P., Bobrow, D.G., and Ducheneaut, N. What a to-do: studies of task management towards the design of a personal task list manager. *Proc. CHI,* ACM (2004), pp. 735–742.
3. Berners-Lee, T. and Connolly, D. Notation3. http://www.w3.org/TeamSubmission/n3/
4. Bernstein, M., Van Kleek, M., Karger, D., and schraefel, m.c. Information scraps: How and why information eludes our personal information management tools. *ACM Trans. Inf. Syst. 26*, 4 (2008), 24:1–24:46.
5. Chang, K.S., Myers, B.A., Cahill, G.M., Simanta, S., Morris, E., and Lewis, G. A Plug-in Architecture for Connecting to New Data Sources on Mobile Devices. *Proc. VL/HCC,* IEEE (2013), to appear.
6. Dai, L., Lutters, W.G., and Bower, C. Why use memo for all?: restructuring mobile applications to support informal note taking. *CHI'05 extended abstracts* ACM (2005), pp. 1320–1323.
7. db4o. http://www.db4o.com/
8. Apache Lucene. http://lucene.apache.org/
9. HanDBase. http://www.ddhsoftware.com/handbase.html
10. Kalnikaité, V. and Whittaker, S. Software or wetware?: discovering when and why people use digital prosthetic memory. *Proc. CHI*, ACM (2007), pp. 71–80.
11. Van Kleek, M., Bernstein, M., Karger, D.R., and schraefel, mc. Gui --- phooey!: the case for text input. *Proc. UIST*, ACM (2007), pp. 193–202.
12. Lieberman, H., Nardi, B.A., and Wright, D. Grammex: defining grammars by example. *CHI'98 conference summary*, ACM (1998), pp. 11–12.
13. Lin, M., Lutters, W., and Kim, T. Understanding the micronote lifecycle: improving mobile support for informal note taking. *Proc. CHI*, ACM (2004), pp. 687–694.
14. Memento Database. http://mementodatabase.com/
15. Miller, R., Chou, V., and Bernstein, M. Inky: a sloppy command line for the web with rich visual feedback. *Proc. CHI*, ACM (2008), pp. 131-140.
16. Mooty, M., Faulring, A., Stylos, J., and Myers, B.A. Calcite: Completing Code Completion for Constructors Using Crowds. *Proc. VL/HCC*, IEEE (2010), pp. 15–22.
17. Nardi, B.A., Miller, J.R., and Wright, D.J. Collaborative, programmable intelligent agents. *Commun. ACM 41*, 3 (1998), pp. 96–104.
18. Oulasvirta, A., Tamminen, S., Roto, V., and Kuorelahti, J. Interaction in 4-second bursts: the fragmented nature of attentional resources in mobile HCI. *Proc. CHI*, ACM (2005), 919–928.
19. Swype. http://www.swype.com/
20. Tap Forms. http://www.tapforms.com/
21. Zhai, S. and Kristensson, P.O. The word-gesture keyboard: reimagining keyboard interaction. *Commun. ACM 55*, 9 (2012), 91–101.