ABSTRACT
The copy-paste command is a fundamental and widely used operation in daily computing. It is generally regarded as a simple task but the process can become tedious when frequent window switching is required to copy-paste across different documents. Auto-completion is another popular operation aimed at reducing users' typing effort. It contrasts to copy-paste by allowing for text completion without switching windows. However, the available content for completion is predefined. We introduce AutoComPaste, an enhanced autocompletion technique for cross-document copy-paste. AutoComPaste allows users to copy-paste different granularity of text from all opened documents without window switching. Our theoretical analysis and empirical study shows that AutoComPaste nicely complements traditional copy-paste techniques and outperforms the traditional copy-paste techniques when users have knowledge of the content to be copied.

Categories and Subject Descriptors: H.5.2 [Information Interfaces and Presentation]: User Interfaces—Interaction Styles

General Terms: Design, Experimentation

Keywords: Copy-Paste, Autocompletion, Windows management

1. INTRODUCTION
Copy-paste data across documents is a common task [15]. Example scenarios include the writing of a project progress bulletin, filling out a grant report, literature review, trip planning, etc. In such scenarios, a common practice is to open relevant documents without window switching. Our theoretical analysis and empirical study showed that AutoComPaste effectively pasting the text into the working document.

While the basic procedure for copy-paste (CP) is simple, the process is often complicated by window management tasks that can be time-consuming and distracting [4]. We propose AutoComPaste (ACP), an enhanced autocompletion technique to complement the traditional CP techniques for cross-document copy-paste. By building a dictionary containing different granularity of text from any opened document, ACP allows users to copy-paste text content across windows without leaving the working text editor. As in traditional autocompletion (AC) techniques, typing in the working document results in the set of entries in the dictionary that match the current prefix to be suggested to the user as a potential completion. The user can then choose one of the entries to complete and at what granularity (word, sentence, paragraph), effectively pasting the text into the working document.

ACP is an effective complementary technique to traditional CP techniques as it offers an alternative way of duplicating content in a cross-document setting. Our theoretical analysis and empirical study showed that ACP and CP techniques perform well in different scenarios. ACP has significant advantages over traditional CP methods when the prefix of the content to copy is known, but the location of the content to copy is unknown. In those cases, ACP eliminates the steps of switching windows, searching for the source text, and highlighting the source text (see Figure 1), thus reducing the time to complete content duplication as we demonstrate in a controlled experiment. In our qualitative study with a trip planning task, our participants found ACP effective and useful, and preferred ACP over the traditional CP method.

2. RELATED WORK
AutoComPaste is a copy-paste technique based on autocompletion. We survey previous work in both areas.

2.1 Copy-Paste (CP)
The basic procedure for cross-document CP typically relies on the same workflow pattern where highlighting is a key step: (1) select the source window (can also be a tab within the same application), (2) highlight the content to copy, (3) issue copy command, (4) select the target window, (5) place the cursor at the paste position, and (6) issue paste command, as illustrated in Figure 1.

Copy-Paste Techniques. Depending on the operating systems and applications, there are typically four interaction techniques to perform CP operations: keyboard, menus, drag-and-drop and the X Window technique. Keyboard shortcuts (Ctrl-C, Ctrl-V) and pop-up menus (typically placed under the Edit menu, or contextually invoked by right clicking) dissociate the six steps of the sequence, allowing for intermingling with other actions. Drag-and-drop, consisting of dragging the selected content and releasing the button at the destination, is a more continuous and direct CP technique as the copy and paste commands are implicitly performed. The fastest CP technique under multiple windows management is that of the X Windows System, where the copy command is automatically performed as the user highlights content [4].

Windows Management. All the CP techniques previously described require the user to perform additional window management operations when dealing with different applications. To facilitate CP between potentially overlapping windows, Chapuis et al. [4]
have proposed the Restack and Roll windows management techniques that aim to reduce the cost of switching between the source and the target document. Entity Quick Click [2] and Citrine [16] rely on structured identifiable entries such as addresses and telephone numbers and support fast simultaneous CP of multiple fields in a single operation. Other techniques, as found in [6] allow for retrieving previously selected, copied, or dragged content to enrich CP interactions in desktop environments. Although there has been some effort to improve CP techniques, almost all of them require the user to break her editing flow as she needs to perform the selection in the source window.

Ethnographic Studies. Stolee et al. have studied users behavior on performing general purpose cross-application CP activities in [15]. This piece of work, however, focuses on how data is transferred (i.e. clipboard patterns) more than the nature of the copied content, an important factor when considering autocompletion. Other ethnographical studies include [9] that focuses on code developers, and [8] but no results are reported.

2.2 Auto-Completion (AC)

AC is a widely adopted feature in many applications. It suggests a list of appropriate completions as the user writes to reduce typing effort and prevent misspelling. AC can also act as a retrieval tool (e.g., Apple’s Spotlight allows for retrieving a document or an application with little knowledge of its exact name) and can even play the role of a suggestion tool (e.g., Eclipse lists all matching methods of a class by only guessing a method’s prefix).

AC Techniques. AC systems are often restricted to single-word completion where the suggestions are a precomputed list of words. Examples include the bash shell, where the dictionary contains executable commands and file names; and text editors such as OpenOffice, Emacs, and Vim, where the dictionary contains words in the opened documents. Several AC techniques supporting multi-word completion have been proposed for richer completion capabilities. The most widely used is probably Google Suggest, where completion candidates are taken from popular search queries. Other examples include phrase prediction [11], and sentence completion [7]. These techniques, however, are limited to a small set of frequently used phrases or sentences.

Context-based Completion. Most of the AC techniques rely on a dictionary built from text input history. Suggestions are picked in a global database that contains content from previous documents that are hence not necessarily relevant to the current user’s context. A notable exception is the predictive text input method [10] where candidate words are selected based on the context of the text composition task. Other related work includes Remembrance Agent that exploits the context of the currently edited document as to suggest related files [13].

Other AC Techniques. Holger and Igmar proposed an AC search that queries based on the possible completion of the search, and returns the results with the highest hits instantaneously [1]. AC has also been extended so as to tolerate erroneous input [5] and list out the possible completion of users’ queries if they mistype.

In summary, there has been extensive work in both CP and AC techniques. Although both are useful, as far as we know there are no technique similar to AutoComPaste allowing for replicating existing content in an auto-completion manner.
3. PRELIMINARY STUDY

Designing the ACP technique requires an understanding of how CP is typically performed. Although the study of Stolee et al. [15] provides most of the answers, the nature of the content that is copied, which is crucial when considering using autocompletion, remains a question. Therefore, we conducted a study to capture this missing information while performing cross-application CP.

3.1 Procedure

22 participants (9 female, 13 male, aged 21-27, mean 23.14) took part in the 2-week study. All are university students in Computer Science or Computer Engineering. Each participant was rewarded 1% course credit after completing the study.

We developed a logging mechanism that collects CP activities running on the Windows XP/Vista/7 OS. Participants were asked to install the logger on their primary computer for a period of 14 days. The logger was automatically turned on without any extra operation from the user, and therefore was constantly running on the background. Logs were periodically sent to our server.

For each CP event, the logger logs its type (copy or paste), the host window and application, the time-stamp, and the content copied. We also record the time distance to the nearest typing event when it applies (i.e. duration between a CP event and the latest typing event performed before, and the earliest typing event performed after).

For each text object copied, we log its content by masking alphabetic characters and numerical digits to protect the user’s privacy (e.g. “joe12@gmail.com” is stored as “xxx00@xxxxxxx.xxx”). Punctuation and whitespace are preserved to retain structural information such as the number of words, sentences, and paragraphs.

3.2 Results

A total of 34.1 MB of text logs were collected. Among the 8168 events, 3481 (43%) were copies and 4687 (57%) were pastes. A similar distribution was observed in [15].

Windows management. We found that 83% of the time, users have 6-20 concurrently opened windows (average 12) when performing CP. Moreover, among the 4687 pastes, cross-document CP happened more often (2672 times, 57%) than within-document CP (2015 times, 43%). This finding concurs with previous work (only 35% of the CP events were within-document in [15]), making a strong case for the importance of cross-document CP techniques.

Units of text copied. The amount of text that is copied is important information when considering autocompletion. But, such information has not been reported in previous controlled studies. To understand the granularity of text content copied (groups of words, sentences, paragraphs). We empirically categorized the copy events into phrases (groups of 8 or less words), single sentences (groups of 8 or more words ending with a period), multiple sentences (at least one sentence without a newline), and paragraphs (one or more paragraphs, each ending with a newline).

Surprisingly, while CP of phrases is common (39%), CP of one or more sentences (33%) and paragraphs (28%) are also frequent. This finding suggests that a CP technique based on AC should support different units of text.

Working context. Stolee et al. [15] found that the word processors were the most popular type of application while performing CP events. We pushed the analysis a step further by analyzing the time interval between CP events and typing in order to identify if CP occurs when editing documents. Empirically taking 30 seconds as a threshold, we found that 42% of all the copy events were performed after a typing event, and 54% of paste events were followed by a typing event. These results show that CP often occurs together with text editing.

4. DESIGN GUIDELINES

Based on prior related work and the results of our preliminary study, we believe that the users can benefit from an enhanced CP technique that addresses the following design guidelines:

G1 Minimize window management operations (e.g. [4]): window management operations can significantly interfere with the primary CP task, therefore breaking the user’s working flow. An enhanced CP technique should minimize such distractions.

G2 Facilitate content selection: retrieving and acquiring the content of interest within the source document can be affected by a potential visual search to retrieve the text to copy and potential errors due to the required explicit selection when highlighting (e.g. unwanted or missing content in the selection, or unintended actions, such as clicking on a hyper-link while selecting, etc.). A technique that aids content retrieval (e.g. [6]) while allowing dynamic adjustments of the selection at different granularities could benefit the users.

G3 Reduce visual search distraction for source text: the two guidelines above could partially be addressed by reducing the amount of distracting information presented to the user. Ideally, a CP technique should take into account of the working context that is relevant to the current editing task [13] by prioritizing or even strictly narrowing down the access to the only relevant content, therefore facilitating the visual search.

5. AUTOCOMPASTE

ACP is a hybrid technique combining the copy-paste and autocompletion approaches to facilitate cross-document text duplication, following the design guidelines of Section 4. As a CP technique, ACP allows for the duplication of content as traditionally performed through selection within a source document. However, it behaves like an AC technique by dynamically suggesting possible completions to the currently entered word as the user types, based on the content from all the source documents available in the workspace. Figure 1 shows the general workflow of ACP.

By using autocompletion, ACP inherently addresses (G1) since it spares the trouble of leaving the working document while editing. In this section, we first describe the building of the ACP dictionary and how it is tied to working context question (G3), then we detail the user interface of our ACP prototype to support dynamic adjustment of the selection text at different granularities (G2).

5.1 Building up the dictionary

An ACP-enabled environment stores all the text from the current working context of the user (e.g. web pages from the browser, documents from word processors, PDF files, etc.) in a database. Text content is broken down into sentences and paragraphs using a background process that grabs and indexes content from every newly opened window. In our implementation, the working context is defined as being all the currently opened documents in their entirety (including documents in minimized windows and tabs). Entries corresponding to a document are removed from the database when the document’s belonging window (or tab) is closed.

Restraining the dictionary to the only opened documents is the direct AC equivalent to the traditional CP environment (where explicit highlighting is required) as for the data available. Since the database only considers documents in the immediate context that the user can dynamically adjust, risks for AC choices irrelevant to the task is limited (G3). However, the database is still prone to contain irrelevant information as the user keeps opened documents that are not directly related to the task. Alternative designs for defining the dictionary to address this issue are discussed in section 10.
5.2 User Interface

Figure 1 depicts the general workflow of ACP. We detail each step in order in the following.

Background process for autocompletion. ACP acts as a traditional autocompletor that constantly keeps track of the user’s keystrokes and identifies matches between the prefix the user types and the text in the database. Whenever ACP detects a match between the typed prefix and at least one entry in the dictionary, a drop-down list pops up close to the caret, showing the potential sentence candidates for completion. To limit distraction that may be caused by frequent appearances of the popup, the list appears only if the number of candidates does not exceed \( n \) entries at the time. Currently, when more than 10 matches are found, we consider the prefix not to be specific enough and therefore not trigger the drop-down list to prevent the user wasting time browsing a long list. Other alternatives are discussed in Section 10.

Browse the candidates. The user can browse the list using the arrow keys, and access more details before validation: a tooltip showing the complete sentence of the selected entry and the document it belongs to is triggered so as to provide the user with potentially useful contextual information on the source of the entry while in an in-place and unobtrusive manner.

Validate selection. At any time, the user can decide to continue typing, thereby ignoring or refining the suggested completions as with traditional AC. To paste the selected entry, she can press Enter. The typed prefix is then replaced by the complete sentence and the selection tailing interface takes place.

Adjust the content to copy. One of the main specific characteristics of ACP is its tailing mechanism. After selecting a completion, the user is offered the possibility of tailing more content from the document the copied sentence belongs to. The sentence that follows the copied text is automatically added after the copied sentence so as to give a preview of what additional content is available. The user can then dynamically adjust the content to paste at will, by adding or removing word by word (< and > keys), full sentences (← and → keys) or whole paragraphs (↑ and ↓ keys). An instruction widget is also displayed to help the user keep track she entered the ACP edition mode.

The user can press Enter to paste the additional content if satisfied with the text to copy, or ignore the tailing option by continuing to type. Both cases result in leaving the ACP edition mode.

ACP therefore enriches the traditional AC technique by allowing the user to interactively explore and extend the completion (G2). The in-place interface facilitates access to content in the immediate working context (G3) while sparing the user the trouble of performing tedious window management operations to retrieve and highlight the content of interest (G1) before getting back to the edited document.

6. THEORETICAL ANALYSIS

Cross-document CP may seem like a simple task overall. However, different scenarios exist in practice. The operation steps involved can differ significantly according to the user’s knowledge of the content to copy and the working context. This theoretical analysis aims to understand how ACP compares with traditional CP.

6.1 Scenarios classification

Based on the knowledge we gained through our preliminary study and literature review, we have identified four main factors that can affect users’ performance while cross-document CP. Related to the user’s knowledge about the content to copy, the actual content \( (F_a) \) and its location \( (F_l) \) have an impact on the time required to the user to acquire the text to copy. Other conditions, such as the current working context, including the visibility of the text to copy \( (F_v) \) and the user’s activity before copy and after paste \( (F_p) \) can also affect the user’s performance.

Figure 3 summarizes how the four different factors interact with each other to form a matrix. The two factors \( (F_c, F_l) \) regarding the user’s knowledge of the text to copy form the high level categories \( (C_{A-P}) \). Each category is further divided into four cells based on visibility \( (F_v) \) and pre-copy activity \( (F_p) \). In total, there are 16 copy-paste scenarios that we label \( (S_1 \ldots 16) \) in the rest of the paper.

6.2 Time Cost Analysis

Depending on the scenario, the user may need to perform more or fewer operations to duplicate content. The required operations depend on the aforementioned factors and incur time cost. In addition to the time cost incurred by the base case scenario where no extra operation is required, there exist other operations that can slow down the user in her task: context-switching, when the user has to change her context to perform the task; homing, when the user has to switch her device; window management, when the user has to access a different window; and visual search, when the user has to visually search on the screen for the required content.

Figure 2 surveys a synthesis of time implications of the different scenario for both CP and ACP. For simplification, the following analysis considers ideal conditions for the base case of each technique. It is important to mention, however, that the browsing of the autocompletion list of candidates may incur additional time, especially in the case of false positive. We discuss these limitations and design directions to address these issues in Section 10. Currently, the maximum number of items in the list is limited to 10.

CP Time Cost. The user’s knowledge of the exact location of the text to copy significantly affects traditional CP performance. When location is unknown, visual search is unavoidable and markedly increases the overall CP time. The visibility of the content is also important. Content hidden in an occluded window requires additional window management and visual search. Moreover, pre-copy activities such as typing also affect overall performances due to context-switching and homing time, but likely not as much as window management and visual search. In contrast, the knowledge of the prefix does not affect the user’s performance at all.

ACP Time Cost. ACP is highly dependent on the user’s knowledge of the prefix of the text to copy. If the prefix is known, the user can avoid window management, visual search and highlighting, which represents significant performance benefits. If the prefix is unknown, visual search for its location will also be required. Location knowledge and visibility are also important for AutoComPaste, but only when the prefix is unknown, so that the user can quickly identify the text to copy and type its prefix. Pre-copy typing will reduce homing time, but may have a minor influence as compared to other factors.
7. QUANTITATIVE EXPERIMENT

To measure the actual performance in \((S_{1-16})\) and to validate our analysis, we conducted an experiment, comparing ACP with the CP technique used in X Window Systems (XWin).

7.1 Experimental setting

12 university students (7 male, 5 female, aged 22-28, mean 24) participated in this study. All were familiar with word processors, common CP functions, and AC elements found in other software.

The experiment was conducted on two desktop computers running Windows Vista, equipped with 20-inch LCD monitor at screen resolution of 1024\(\times\)768, a mouse and a keyboard.

7.2 Task and Stimuli

All participants were asked to copy from a source document the answer of a question and paste it into a destination document for the two XWin and ACP techniques, under the 16 scenarios described in Section 6.1, that we simulate by varying the four prefix, location, visibility and pre-activity control conditions.

We collected 860 questions and answers from Answers.com and split them into 86 ten-question articles. Sentences, paragraphs, and common phrases for the articles were indexed in a database. We randomly selected 6 articles to present to the participant in each trial.

Prefix Knowledge. To simulate the known prefix condition, the participants were asked to read the text to copy before starting the trial. Simulating the condition when participants only have partial knowledge of the text, but do not know or remember its prefix is more challenging. In real life, it often happens that we need to copy the definition of a particular word or the answer of a specific question that we know what it is only when encounter it, but do not know its exact prefix. We replicate such a scenario by collecting definitions about engineering terms and asking the users to find the definition for a particular question. For instance, participants were asked to find the definition of “Sonic”; the text to be copied is “Pertaining to sound.”

Location Knowledge. When users know the location of a piece of text, they can find it immediately. To simulate that effect, we highlight the target text in yellow. If the highlighted text is occluded, the window’s name is also provided in the stimulus so that participants know where to find the window. Location unknown condition is simulated by simply removing the highlight.

Visibility. We enforce text visibility by bringing the containing window to the foremost position. On the contrary, invisible text is occluded by one or more windows.

Pre-copy Activities. We control the pre-copy activity as follow: (i) isolated CP – the stimulus only contains instructions about the text to copy, and (ii) typing something before CP – the stimulus also contains instructions to first type a short word.
Text Unit. In addition to the above factors, another factor of interest is whether the text to copy is a phrase, sentence, or paragraph. To access the difference, we include text unit as a control condition. For each of the 16 scenarios, participants complete 2 phrases, 2 sentences, and 2 paragraphs which are randomly selected without replacement from the data store.

Dependant Variables. Dependent variables are accuracy (ratio of successful trials to total trials) and completion time (the interval between clicking the start trial button and the completion of pasting the text to copy). Time incurred by additional typing task is not part of response time.

7.3 Procedure

Before the experiment, we first collected the participant’s demographic information, followed by a short practice session to let her familiarize with each of the two techniques. Once the experiment started, for each trial, the screen showed a blank window with the only textual stimuli displayed on the top.

Participants were told to read the instruction before clicking the “start trial” button, upon which, six overlapping windows with textual content and a text-editing window for typing and pasting were loaded. We used six windows to balance between simulating real workspace environments with multiple windows opened (average is 12, see Sec. 3.2) and to keep the complexity of the experiment manageable while not favouring our technique. Figure 4 shows a screenshot from the experiment during a trial.

Upon completion of the task, the system checked if the typed text (if it applies) and the copied content matches the original content. In the case of a mismatch, an error was counted and the participant was prompted to redo this trial; otherwise, the participant proceeded to the next trial or took a break. Once all trials were finished, the participant was asked to fill out a post-experiment questionnaire about her overall experience on the techniques and conditions. The whole experiment took about 90 minutes.

Experimental Design. We used a within-subject factorial design with five counter-balanced factors. Our design counts a total of 2304 trials: 12 subjects × 2 techniques (XWin, ACP) × 2 types of content knowledge (known, unknown) × 2 types of location knowledge (known, unknown) × 2 types of visibility (visible, invisible) × 2 types of pre-copy activity (isolated, typing) × 6 trials of 3 different units of text (2 phrases + 2 sentences + 2 paragraphs).

7.4 Empirical vs. Theoretical

Table 1 summarizes the response time and accuracy results for each scenario for both techniques. Pairwise t-tests were performed on every scenario pair between XWin and ACP. Differences (third column) that are significant (p < .05) are shown in bold.

Base Case Comparison. No significant difference was found between techniques in (S1). Since the homing time is small (about 0.4s [3]), this result indicates that both techniques’ base case requires about 3s, suggesting that CP and ACP’s base case (i.e. ideal conditions) are comparable.

Category-by-Category Comparison. ACP is significantly faster than XWin in (C12) by 2-3 times in performance, but for scenarios in (C13) and (C14), XWin outperforms ACP. Although the absolute difference in time performance for (C12) is larger than that of (C13), the relative difference in terms of percentage is similar. While the analysis only discloses the difference between homing times, in reality, the cost includes additional context switching time. This result largely matches our estimates from the theoretical analysis (see last column of Figure 2).

Usage Recommendation. The 4 two-way interactions related to method reveal important insights – when prefix is known, ACP is either comparable with (in S11) and (S12) or faster than XWin (in S13 and S14). If the location is unknown, ACP results in 2-3 times performance benefit.

When prefix is unknown, XWin is significantly faster than ACP in the scenarios (S2, S3, S13, S14, S15, S16). XWin is also faster in (S14), but not statistically significant. XWin is recommended in this case. In addition, the interaction effects on technique × location, technique × pre-copy activity, and technique × text unit show that ACP is advantageous when the location is unknown, interleaving with typing, and to copy phrases while XWin is better for isolated, known location copy-pastes, as shown by Table 1. However, these effects are secondary as compared with prefix.

8. QUALITATIVE EXPERIMENT

The qualitative experiment above have demonstrated how our ACP prototype fare with the state-of-the-art CP technique across different experimental conditions. To further understand natural users’ behavior when using an ACP-enabled environment with a more realistic task and evaluate users’ acceptance of the technique, we also conducted a qualitative evaluation.
We chose to evaluate ACP using trip planning as the task. Trip planning is an activity commonly carried out by travellers before visiting new places. For Internet savvy users, it typically involves identifying relevant travel resources via Web research, then collecting and compiling useful information from these resources into a document. During this process, many CP activities are likely be carried out, offering an ideal opportunity to run our study.

We are interested in investigating the following questions:

Q1 Is ACP useful in this task, and what are the usage patterns?
Q2 How users feel about ACP after the task?
Q3 What are the practical problems and issues with ACP?

8.1 Experimental setting

6 participants (3 female, 3 male; aged 22-25, mean 23.8) were recruited from the University community for the study. All are students familiar with computer technology and received $16 USD for completing the study.

The study was conducted on a Dell desktop computer running Windows 7, equipped with 19 inch LCD monitor display at a screen resolution of 1024 × 768. The 10 travel-related webpages were loaded on Google Chrome browser (v10). The arrangement of these documents and the text editor was determined by the participant.

8.2 Procedure

After filling out a pre-study questionnaire collecting background information, participants were instructed for the practice session. Participants were asked to open 3 Wikipedia pages and take notes in an ACP-capable text editor using a number of supplied keywords. ACP naturally triggered, but we left to the participants to discover the technique on their own (at this stage, no mention of ACP was made at all). After 10 minutes of exploration, participants were asked to study a 2-page manual about how to use ACP and answer some questions to test their understanding of the technique.

Once participants were familiar with ACP, they were asked to plan a 5-day trip in a text document for visiting a North American city by gathering relevant information from 10 given webpages. Participants were asked to include at least one outdoor activity, one indoor activity, and one restaurant for each day of the trip. A sample trip planning document for Paris was provided as guidance. This step investigates the usage of ACP in a realistic task setting and assesses how users perform copy-paste when given the choice of the technique to use. No time limit was imposed.

We collected overall comments and feedback about the techniques and the study through a post-study questionnaire. The entire study was conducted in one-sitting, in 75 to 130 minutes (including break) depending on participants.

8.3 Results

Q1. Users found ACP useful in this task. Despite the explicit instruction that traditional CP technique can be used, all participants primarily used ACP during the study. One participant even used ACP exclusively, while the rest occasionally used traditional CP techniques in addition to ACP.

Participants stated that using ACP feels more comfortable while typing, as “it allows me to stay focused on my current task” without the need of switching windows. However, most of them noticed that ACP sometimes behaves differently from their expectations; in which case they switched to traditional CP. For instance, participants tend to expect a drop-down list after typing keywords such as “art”, “restaurant”, “hotel”, etc. Since the latter appeared frequently in the documents (more than 10 times), ACP will not bring up the suggestion list. Participants switched back to traditional CP in such situations, thinking of a dysfunction of the program.

Q2. Participants were asked their opinions concerning usefulness, ease of use, ease of learning, satisfaction, and how likely they will recommend ACP to their friends on a 5-point Likert scale ranging from strongly disagree to strongly agree. The results of the post-study questionnaire are summarized in Figure 5.

Overall, we found the majority of the participants liked ACP and found it useful for the trip planning task, except one participant, who was negative and unsatisfied with the technique. We further elaborate his case when answering the third question.

Q3. Testing ACP on a realistic task helped us point at potential issues of ACP that warrant future improvement or investigation. As mentioned earlier, one issue is when ACP is expected to show the suggestion list. The current implementation brings up the suggestion list when there are 10 matches or less. This threshold seems to work well most of the time; it does not, however, always match participants’ expectations.

Furthermore, we found that the proficiency of English language and typing skills also affect user experience. One participant experienced tremendous problems with spelling. Hence, many of the expected CP events were not triggered due to misspelling. To this end, the participant found ACP less useful as compared to the traditional CP, and rated it negatively.

While the preference and usage situation differ among users, most of the participants reported they like ACP and would like to integrate ACP into their text editor as a complement to CP.

9. DISCUSSION

Overall, the results of the studies we conducted on our prototype are encouraging for the ACP technique. In some cases, ACP is faster than traditional CP, and the technique has been preferred by our participants. A longitudinal study with a refined implementation of ACP is however necessary to evaluate the adoption and usefulness of the technique on regular activity. In particular, we learnt from our studies that future ACP implementations should address the following issues:

Appropriate mapping and consistency. To limit potential distraction and frustration due to ill-timed pop up of the suggestion list, we decided not to show the list when too many matches. This however interferes with the mapping and consistency design principles as described in [12]. Participants can be disoriented as they coped with the violation of their expectations. Future implementation of ACP can benefit from a more adaptive algorithm to better fit users’ needs. An explicit feedback when too many matches are found would also help users understand better the behaviour.

Support fuzzy typing. Another serious issue of the current implementation of ACP is in its incapability of supporting fuzzy typing. ACP also requires to know the exact prefix of a word in order to trigger the suggestions. A possible solution will be to use a fuzzy dictionary so that relevant entries can be suggested even when misspelled. However, here again there is a tradeoff to be found between the frustration due to non-useful popups and the frustration due to the list that fails to trigger.
10. DESIGN ALTERNATIVES

While our current implementation of ACP chose one particular design, many other alternatives exist. We discuss those options along three dimensions that characterize AC techniques: (1) a dictionary of the possible completions, (2) a ranking function for proposing the best matches, and (3) an interactive visualization mechanism to present and browse the different suggestions. Some of these alternatives raise interesting UI design questions and point to future research directions of ACP.

10.1 Dictionary

ACP builds its dictionary based on all opened documents on the users' desktop only. The indexing scope, from which the dictionary is built, trades off between distracting the users (too many irrelevant matches) and helping the users (finding the right AC match). Speier et al. [14] have shown that one does not want to be more distracted than helped. By building a dictionary based only on currently opened documents, ACP reduces the space of possibilities and provides current-context relevant information.

Other alternatives include building a dictionary based on the current document (e.g., emacs de-abbreviation), history in the tool (e.g., OpenOffice), a predefined set (e.g., APIs in IDE), all user's documents (e.g., Remembrance Agent [13]), and to an extreme, all popular phrases and words on the Internet (e.g., Google Scribe).

One can also imagine a customizable dictionary, with the appropriate interface, the user would be allowed to add/discard documents to be taken into account associated to a specific editable document. Users can pin documents to keep the text from these documents in the dictionary even after closing them.

Another possible design is to build a dictionary based on recently opened documents. The set of recently opened documents can be limited based on the number of documents or based on time. The recency could provide the current-context relevant information without users having to explicitly keep the documents opened.

10.2 Ranking

The user is usually provided with a list of possible completions, which are typically ordered based on “good hits suggestions” (see [13]), or frequency (e.g., Google). There is no notion of frequency or good hit in our prototype – the list is ordered from the most recently added documents first, and linear order of the text.

Besides ranking by hit and frequency, one can organize the results based on the documents that the autocompletion matches appear in. Per-document ranking can be done, e.g., documents in opened windows are ranked higher than minimized windows.

10.3 Interaction

In our current design, the ACP feature is always on. One can easily use an alternative that allows user to turn on and off ACP (e.g., in Google Scribe) or invoke ACP with a combination of keystrokes (e.g., Ctrl-P in Vim) to limit distraction.

ACP currently displays the matches only if no more than five matches are found. Ideally, this threshold should be dynamically determined, perhaps depending on the number of opened documents and the frequency of use of autocompletion from the suggested list. Another option is to leave it to the user to decide on the threshold manually, and allow for an on demand drill-down of the list even when too many matches are found.

It would be helpful to show more contextual information too, in addition to the document that the suggested text is taken from, such as whether the window corresponding to the document is currently opened, how many additional sentences or paragraphs are available. A small counter showing the number of matching suggestions as the user types would also be helpful.

11. CONCLUSION

ACP is a complementary technique to perform copy-paste using autocompletion. Derived from traditional autocompletion mechanisms, ACP builds a dictionary using opened documents from different applications and provides a unique trailing mechanism to allow users to adjust the unit of text to be replicated.

The design of our prototype of ACP relies on design principles derived from studies of CP behavior, which reveals CP is mostly performed on sentences and paragraphs. We tested the prototype on a quantitative study, showing that ACP can be faster than traditional CP technique in a number of cases; and a qualitative study on a realistic task that revealed a positive welcome of the technique from participants, indicating that the technique is helpful and easy to use.

Our future plan is to explore some of the design alternatives of ACP. One important issue is to fine tune the triggering threshold of the suggestion list, to avoid distracting the users without sacrificing any convenience to the users. We also plan to deploy an improved ACP in real life use and further study the ecological validity of the technique on a longitudinal study.

12. REFERENCES