
Entity Quick Click: Rapid Text Copying Based on Automatic Entity Extraction

Eric A. Bier

Palo Alto Research Center, Inc.
3333 Coyote Hill Road
Palo Alto, CA 94304 USA
bier@parc.com

Edward W. Ishak

Columbia University
Department of Computer Science
1214 Amsterdam Ave.
450 CS Bldg
New York, NY 10027 USA
ishak@cs.columbia.edu

Ed Chi

Palo Alto Research Center, Inc.
3333 Coyote Hill Road
Palo Alto, CA 94304 USA
echi@parc.com

Abstract

Retyping text phrases can be time consuming. As a result, techniques for copying text from one software application to another, such as copy-and-paste and drag-and-drop are now commonplace. However, even these techniques can be too slow in situations where many phrases need to be copied. In the special case where the phrases to be copied represent syntactically identifiable entities, such as person names, company names, telephone numbers, or street addresses, much faster phrase copying is possible. We describe *entity quick click*, an approach that reduces both the amount of cursor travel and the number of button presses needed to copy a phrase.

Keywords

Copying text, entity extraction, information extraction, text editing, sensemaking, copy-and-paste

ACM Classification Keywords

H.5.2. Information interfaces and presentation (e.g., HCI): User Interfaces – interaction styles. H.4.1. Information systems applications: Office Automation – word processing.

Copyright is held by the author/owner(s).

CHI 2006, April 22–27, 2006, Montréal, Québec, Canada.

ACM 1-59593-298-4/06/0004.

Introduction

Promising new interaction paradigms in document manipulation are resulting from incorporating pattern recognition and natural language technologies. Here we describe a way to enhance copy-and-paste operations with entity extraction technology. Copy-and-paste and drag-and-drop are ubiquitous methods for moving text from one computer application to another. Using these methods a computer user can transfer textual information without retyping it. This is generally an advantage as retyping can introduce errors, requires great user attention, and is slow. Modern graphical interfaces apply these two methods to move text between a broad range of software applications, including Web browsers, document editors, and spreadsheets. As a result, improvements to these techniques have the potential to improve the usability of many applications.

A small improvement in the speed of these techniques (the number of copy-and-paste operations that can be done per minute, for example) should have a small

quantitative effect on user productivity. Furthermore, a large increase in the speed of these techniques (e.g., a factor of 5 or more) could make a qualitative difference in user performance for some applications. Previous studies of the effects of system speed [7] have demonstrated this kind of qualitative effect across many time scales and applications. In the case of copy-and-paste, a large speed-up might enable applications, for example, in which the user extracts many phrases from a document in order to add them to a form, a database, or some other structured data format. We have discovered the need for more rapid copy-and-paste in our own research on tools that help users take structured notes while reading through many documents on a topic.

As commonly deployed copy-and-paste takes roughly eight steps. Using a pointing device, the user moves a cursor to the beginning of the text to be copied, initiates a selection command, moves the cursor to the end of the text to be copied, finalizes the selection, initiates a copy operation, moves the cursor to the

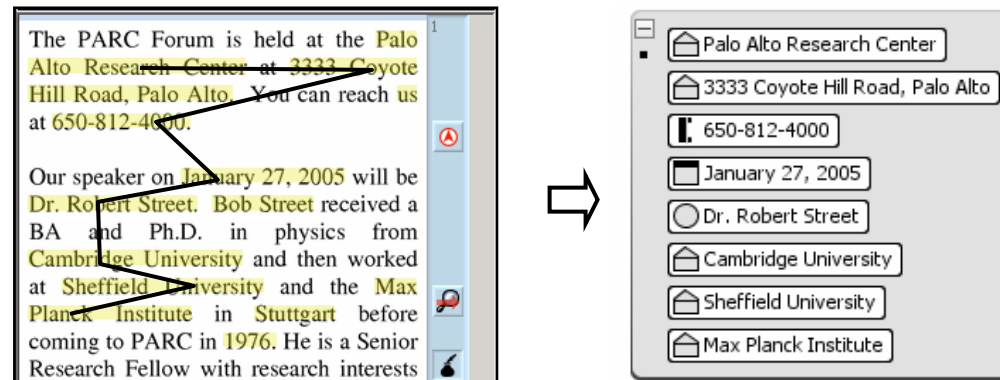


Figure 1. Each time the user clicks on a highlighted phrase in the document, the phrase is copied to a receiving application. The overlaid polyline shows one path that the cursor might take during the copying of the eight phrases shown.

place where the text is to go, initiates a cursor placement command, and initiates a paste command. Several steps here require moving the cursor to small on-screen targets and attending to rapidly changing feedback. In this paper, we describe a technique that takes as few as two steps per copy operation and provides larger target objects, which users can point to more rapidly, as shown by studies related to the two-dimensional extensions of Fitts' Law [4].

To get a larger speed-up, we sacrifice the generality of copy-and-paste in two ways. First, we assume that the user will be copying phrases in natural language text that describe an *entity* in a form that can be recognized by a computer algorithm, such as person names, city names, company names, telephone numbers, or street addresses. Second, we assume that the application receiving the phrases will keep track of where the copied phrases should be inserted; it may have a current insertion position, for example, or it may add the phrases to a list as they come in. In cases where one or both of these assumptions are false, our technique degrades gracefully to traditional copy-and-paste or drag-and-drop, as we will describe.

Single Word Quick Click

Before describing entity quick click, we describe a simplified version of it, *single word quick click*. A user of single word quick click identifies an application that is to receive the copied text fragments. To be concrete, let's assume this application is a tool that makes a list of all of the copied fragments. Next, the user opens a document containing some natural language text. The user holds down a modifier key (e.g., the Shift key) to activate quick click. Then, whenever the user wants to copy a word from the

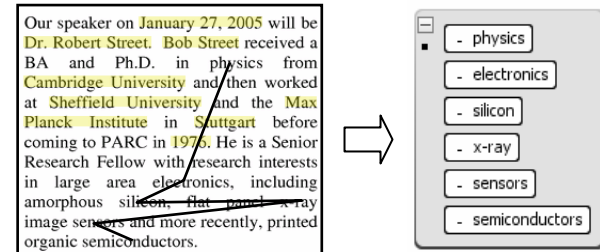


Figure 2. The user copies non-highlighted words by clicking on them. The overlaid polyline shows cursor movement.

document into the receiving application she moves the cursor over the word and clicks a button (e.g., the left mouse button). A copy of the word under the cursor is immediately added to the receiving application.

After the set-up is done (opening the two applications and depressing the Shift key), the cost of copying a word to the receiving application is the cost of moving the cursor and clicking a button. In this way, individual words can be copied rapidly, with one click per word. This is an improvement over applications that require a double-click to select a word or that require additional keystrokes to initiate copy, paste, or both.

Entity Quick Click

One limitation of single word quick click is that it only copies individual words. It cannot copy a longer phrase, a sentence, or a paragraph. We could add additional steps to allow the user to specify the beginning and end of the text string to be copied. Instead, we took an approach that preserves the one-click paradigm while allowing phrases to be copied.

The new idea is to run an algorithm over the text that finds phrases that the user is likely to want to copy as a

unit. For example, a user extracting information from news stories may be interested in the names of people and companies involved, phone numbers to call, or places where an event has taken place. Such phrases can be found using information extraction technology.

In our system, entity phrases are extracted for all documents before they are displayed to the user. If desired, these phrases can be highlighted (e.g., with a yellow background) as a way to draw the user's eye to important entities. Alternatively, the highlighting may be shown only when the user is about to use entity quick click to copy a phrase (e.g., when the Shift key is down and the cursor is positioned over the document).

Entity quick click then works like single word quick click when the user clicks on an unhighlighted word but has a new behavior when the user clicks on a highlighted phrase. In this new case, the indicated phrase is copied and sent to the receiving application. For example, **Figure 1** shows a document with some of its entity strings highlighted with a yellow background. If the user moves the cursor along the path indicated by the overlaid polyline, clicking at each vertex, eight strings are sent to the receiving application, which displays the copied phrases in a vertical list.

Copying arbitrary strings

In some cases, the user wants to copy a string that does not correspond exactly to a phrase found by the entity extractor. In these cases, entity quick click degrades to a procedure that is more like traditional copy-and-paste. If the string to be copied is a word, and it is not part of one of the automatically extracted phrases, the user clicks on it as for single word quick click and it is copied, as illustrated in **Figure 2**. If the

string to be copied is a phrase but does not correspond exactly to one of the highlighted phrases, the user copies it by first selecting it (using whatever selection mechanism is available in the tool displaying the source document) and then holding down the Shift key and clicking on any part of it. This works even if the word that is clicked is part of an entity phrase because the boundaries of the selected phrase are given precedence. In **Figure 3**, for example, the user selects the phrase "coming to PARC in 1976", which includes the extracted entity "1976". The user then Shift-clicks on the phrase to copy it to the receiving application.

Use with paste or drag-and-drop

As described so far, entity quick click assumes that the receiving application will automatically place the copied text strings. However, most modern commercial applications expect to receive copied text from an explicit paste or drag operation that specifies where the copied text should be placed. With some loss in efficiency, entity quick click can be used with these applications as well. For example, after the quick click is made on a desired phrase, that phrase can be placed in the standard operating system cut buffer and then

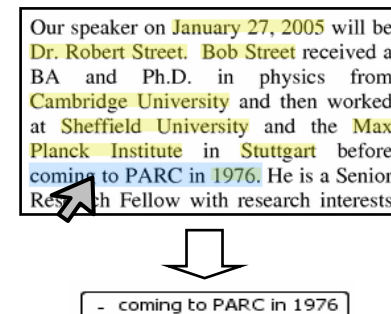


Figure 3. The user selects the phrase "coming to PARC in 1976" and then clicks on it to copy it as a whole.

pasted into an application in the traditional way. Alternatively, after selecting the desired phrase, the user could also drag-and-drop the phrase into another application in the traditional way.

Extensions

For some applications, having a single set of recognized phrases may be too limiting. For example, the user may wish to select person names, company names, and locations at one time, but may wish to select unrestricted noun phrases or some other text units at another time. Entity quick click can be extended to handle this case by providing multiple modes, one for each way of recognizing phrases.

We also allow the user to extend the set of automatically extracted phrases with a private dictionary of additional phrases that should be highlighted. One way to build this dictionary is to add to it all words or phrases that are copied using entity quick click but are not automatically extracted phrases.

Implementation

Entity quick click is written in Java. Users view documents in the CorpusView reading environment [1]. Entity extraction is performed using the ANNIE component of the University of Sheffield's GATE [3].

Applications and Testing

We are developing entity quick click as part of a suite of tools that help knowledge workers make sense of information that is drawn from a large collection of documents. Some important tasks in sensemaking include discovering the names of important entities in the topic being studied and how these entities are related to each other. One of the tools in our suite,

Entity Workspace, acts as an electronic notebook for collecting this information. Entity quick click is an important component of this application, allowing entity names to be gathered quickly so the knowledge worker can make rapid progress.

We are applying this approach to build sensemaking tools for intelligence analysts, patent attorneys, and technology analysts. Ongoing work on entity quick click includes testing it on knowledge workers performing real tasks and comparing its performance to related techniques for copying phrases.

Related work

The three main components of entity quick click are: automatic entity extraction from documents, efficient user interface for copying text strings, and automatic highlighting of important words and phrases. In all three areas, we are building on previous work.

A year ago, *IT Professional* estimated that there were 15 to 20 information extraction tools on the market [9]. Entity quick click can take advantage of the output of any of these tools to improve the speed of text copying (or selection) in the domains that these tools cover.

Efficient user interfaces for copying text strings have been around for decades. For example, in 1984, Teitelman described the design of the Tioga text editor in the Cedar programming environment [10]. As in entity quick click, the Tioga user would hold down a Shift key and select a text object. The selected text would then be copied to a previously-selected insertion position in the same or some other document. Entity quick click builds on this basic scheme by allowing an entire phrase to be selected in a single click.

Stylos et al. [8] developed Citrine, a system that extends traditional copy-and-paste by parsing the copied text and pasting the structured information, for example, into a form with many fields. However, this system often requires training before it works as desired. Entity quick click takes a different approach that allows the user to visualize the results of parsing by highlighting the extracted entities within the source text before copying. This eliminates pasting of undesired text and allows the user to choose a subset of the fields found.

A number of systems have used automatic highlighting of computed words or phrases to help readers find information in a document. The ScentHighlights technique [2] highlights both keywords that are related to a set of search terms and their surrounding sentences. XLibris [6] highlights phrases and sentences that are characteristic of a document when the user requests a skimming mode.

Finally, Miller created an editor, LAPIS, that uses pattern recognition approaches to identify repeated patterns during editing tasks [5]. In his system, edits to a single line of source code could be propagated to other lines with similar structures nearby. As in LAPIS, our tool finds text strings that match a pattern; however, our tool is used during reading rather than editing and focuses on copying the matching strings.

Acknowledgments

This research was funded in part by the Advanced Research and Development Activity ARIVA program (MDA904-03-C-0404).

References

- [1] Eric Bier, Lance Good, Kris Papat, and Alan Newberger. A document corpus browser for in-depth reading. *Proceedings of the 2004 Joint ACM/IEEE Conference on Digital Libraries (JCDL)*, 2004, 87-96.
- [2] Ed H. Chi, Lichan Hong, Michelle Gumbrecht, Stuart K. Card. ScentHighlights: highlighting conceptually-related sentences during reading. *Proceedings of IUI'05*, 2005, 272-274.
- [3] Hamish Cunningham. GATE, a general architecture for text Engineering. *Computers and the Humanities*, Volume 36, Issue 2, May 2002, pages 223–254.
- [4] I. Scott MacKenzie and William Buxton. Extending Fitts' law to two-dimensional tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1992, 219-226.
- [5] Robert C. Miller and Brad A. Myers. Multiple selections in smart text editing. *Proceedings of the 6th International Conference on Intelligent User Interfaces (IUI 2002)*, 2002, 103-110.
- [6] Bill N. Schilit, Morgan N. Price, Gene Golovchinsky. Digital library information appliances. *Digital Libraries '98*, 1998, 217-226.
- [7] Ben Shneiderman. Response time and display rate in human performance with computers. *ACM Computing Surveys* Volume 16, Issue 3 (Sep. 1984), 265-285.
- [8] Jeffrey Stylos, Brad A. Myers, Andrew Faulring. Citrine: providing intelligent copy-and-paste. *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST 2004)*, 2004, 185-188.
- [9] Sarah M. Taylor. Information extraction tools: deciphering human language. *IT Professional*, Volume 6, Number 6, 2004, 28-34.
- [10] Warren Teitelman. A tour through Cedar. *Proceedings of the 7th international Conference on Software Engineering*, 1984, 181-195.