Cluster-Based Find and Replace

by

Alisa Marie Marshall

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the requirements for the degree of

Masters of Engineering in Computer Science and Engineering

at the Massachusetts Institute of Technology

June 9, 2003

Author .....
Department of Electrical Engineering and Computer Science
May 21, 2003

Certified by .....
Robert C. Miller
Assistant Professor
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Cluster-Based Find and Replace

by

Alisa Marie Marshall

Submitted to the Department of Electrical Engineering and Computer Science
on May 21, 2003, in partial fulfillment of the
requirements for the degree of
Masters of Engineering in Computer Science and Engineering

Abstract

Current Find and Replace tools offer their users only two options when performing a find and replace: replace one match at a time while prompting the user to confirm each replace, or replace all matches at once without any user confirmation. Each of these options can be prone to errors. In this thesis, I investigated, implemented, and tested a third option when performing a find and replace: a cluster-based find and replace. Instead of replacing one match at a time or all matches at once, the user can choose to replace a cluster of similar matches. While I hypothesized that a cluster-based find and replace interface would make users complete a find and replace task faster and more accurately, preliminary user studies suggest that a cluster-based interface may improve speed.

Thesis Supervisor: Robert C. Miller
Title: Assistant Professor

Acknowledgments

I would like to thank my advisor, Robert Miller, for his immense support and assistance throughout my thesis.

I would like to thank my friends, Stefanie Tellex, Lin Wu, and Shastri Sandy for all the emotional support and encouragement and keeping me sane while I worked on this thesis.

I would like to thank Sara Su, Eugene Hsu, Sally Lee, Eric Chan, Vishy Venugopalan, May Zhou, Patrick Nichols, Matt Notowidigdo, Peter Luka, Glenn Eguchi, Elmar Eisemann, Rob Jagnow, and Vince Lee for being pilot users, paper prototype test subjects, and for helping debug code and usability problems.

I would like to thank my family whose emphasis on education and hard work made this thesis possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Repetitive text editing tasks, such as spell-checking and find and replace, are both tedious and error prone. While conventional interfaces for these tasks allow users to quickly and easily make changes throughout their documents, they also allow users to quickly and easily introduce errors throughout their documents.

A typical text editor offers its users only two choices when performing a find and replace: replace one match at a time while prompting the user to confirm each replace, or replace all matches at once without any user confirmation. However, when documents are long or contain a large number of matches, neither option is optimal.

Replacing one match at a time with confirmation can take a significant amount of time to do, and is prone to errors. If most of the matches are to be replaced, a user will often start to confirm a replace automatically and miss the few cases where they would have preferred to pay more attention to the match presented.

Replacing all matches at once, however, can lead to interesting side effects. Examples include:

- Occurrences of "stogard" instead of "standard" in the Danish users' guide for Windows for Workgroups 3.11. This error likely occurred when in an attempt to translate the English version to a Danish version, the translator performed a global search for the English word "and" and replaced all matches with the Danish equivalent "og" [7].

- Occurrence of "eLabourated" in the Wall Street Journal. This error was the result of running a global search for "labor" and replacing all matches with "Labour" in order to correct a previous error where the British Labour Party had been referred to as the British Labor Party [7].

- Occurances of "AmriCzar" in a Reuters news story. This error was the result of running a global search for "tsar" and replacing all matches with "Czar" [7].

The careful writing of search patterns can mitigate some of these problems, but this requires that the user either to know the document they are editing well enough to create a perfect pattern or to have a better grasp on regular expressions than most users have.

In this thesis I implemented a third possible method for performing a find and replace: a cluster-based method in which the user is instead presented with clusters of similar results within his/her search. Instead of replacing one match at a time or all matches at once, the user can select and replace a cluster of similar matches. Now when the user is performing a find and replace task, they can choose which clusters match the desired pattern and ignore the rest of the matches. This method allows a user to create simple, general patterns and then choose to replace a specific subset of the matches without needing to write a more complicated search pattern. The user is also able to quickly replace a subset of matches to their search query, with a lower risk of unintentionally replacing of any of the other matches to their search query.

For example, in Figure 1-1, all the single and double quote marks have been rendered as question marks. This incorrect rendering can happen when trying to copy text from one text viewer and paste it in another. In order to make it readable again, you would have to change all of the question marks (?) to the appropriate quote mark. Some of the question marks, however, are meant to be question marks. This paragraph contains eleven question marks: five are meant to be single quotes, five are meant to be double quotes, and one is actually meant to be a question mark. Current find and replace interfaces require the user to look at every instance of the question mark and figure out if they want to replace it with a double quote, single quote or

leave it alone. This requires eleven confirmations from the user. Alternatively, you could try to make a better pattern by first replacing all question marks that have a space in front or behind them with double quotes, then replace all other question marks with a single quote, and finally find all examples of a single quote in front of a double quote and replace them with a question mark and a double quote. That would require three replace all actions, and with no guarantee to have made perfect patterns.

Replacing all question marks with either double or single quotes at once would lead to either Figure 1-2 or Figure 1-3. Neither of these examples is correct. Replacing all question marks with double quotes leads to do"t and replacing all question marks with single quotes leads to quotes being surrounded by single quotes rather than double quotes.

> ?We hope to make the contents of our service courses more clearly known to other faculty,? said Miller. ?We?re always getting questions from faculty like: ?Do you really teach complex numbers in your courses?? because students claim they?ve never seen them. Whether the students have forgotten or just don?t recognize them in another context, we don?t know. It?s complicated because things often appear in slightly different language further downstream. [1]

Figure 1-1: A piece of text where every single and double quote has been rendered as a question mark as a result of copying from one test viewer and pasting in another.

> "We hope to make the contents of our service courses more clearly known to other faculty," said Miller. "We**"**re always getting questions from faculty like: "Do you really teach complex numbers in your courses**"**" because students claim they**"**ve never seen them. Whether the students have forgotten or just don**"**t recognize them in another context, we don**"**t know. It**"**s complicated because things often appear in slightly different language further downstream.

Figure 1-2: Result of replacing all question marks in Figure 1-1 with double quotes. Errors are shown in bold.

Given the paragraph in Figure 1-1 my cluster-based find and replace interface presents the user with four clusters: two clusters containing matches that should

'We hope to make the contents of our service courses more clearly known to other faculty,' said Miller. 'We're always getting questions from faculty like: 'Do you really teach complex numbers in your courses" because students claim they've never seen them. Whether the students have forgotten or just don't recognize them in another context, we don't know. It's complicated because things often appear in slightly different language further downstream.

Figure 1-3: Result of replacing all question marks in Figure 1-1 with single quotes. Errors are shown in bold.

be replaced with double quotes, and two clusters containing matches that should be replaced with single quotes (Figure 1-4). Because the example is very short, the match that should not be replaced is mixed into the cluster containing matches that should be replaced with double quotes. In a larger document, this error would not happen. The clustering algorithm depends on the features of the text surrounding the question marks to group the examples in to similar clusters.

There are four key differences and advantages between a cluster-based find and replace interface and current find and replace interfaces.

1. **Matches are grouped by similarity.** Grouping the matches allows the user to make one decision about a group of matches rather than a decision for each match in the group.

2. **Matches are not presented in document order.** A user can select and replace an entire group of similar matches located throughout the document without accidentally replacing any other matches.

3. **All matches in a single group can be replaced all at once.** Only one action is needed to replace multiple matches.

4. **Matches which cannot be grouped can be replaced one at a time.** A user can carefully go through matches that do not easily fit into any larger groups.

I have approached this problem in two parts. I designed an extension to LAPIS, a usable cluster-based find and replace interface and tried to make that user interface as

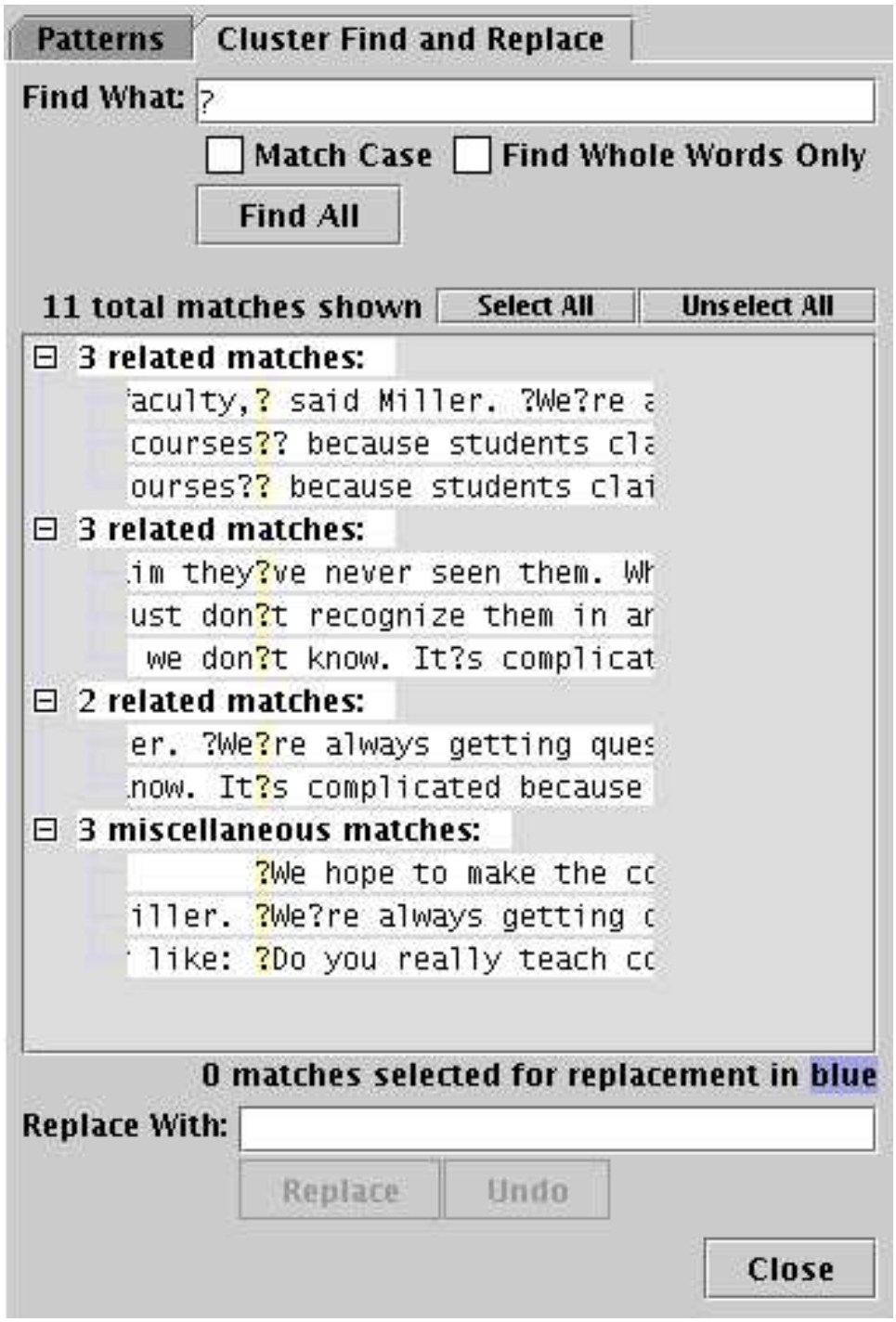


Figure 1-4: Screenshot of the cluster-based find and replace interface which presents the user with four clusters. The first cluster contains matches that should be replaced with double quotes, except for the second match which should be replaced with a question mark. This error occurs because the example is too short. The second and third clusters contain matches that should be replaced with single quotes. The fourth cluster contains matches that should be replaced with double quotes.

intuitive as possible so that people familiar with standard find and replace tools can seamlessly transition to a cluster-based find and replace. I then tested and compared the speed and accuracy of users performing find and replace tasks with a cluster-based interface and a conventional find and replace interface to discover whether or not it improved user performance.

I hypothesized that a cluster-based find and replace interface will allow users to complete find and replace tasks faster and more accurately than a standard find and replace interface. I found that a cluster-based interface did not help as much as I hoped it would.

Chapter 2 explains previous work done in find and replace interfaces and clustering.

Chapter 3 explains the process of designing, prototyping, and implementing the cluster-based find and replace interface.

Chapter 4 explains the testing of the cluster-based find and replace interface and the results of the user study.

Chapter 5 explains what was learned about building a cluster-based interface, future work, and the contributions to user interface research this thesis made.

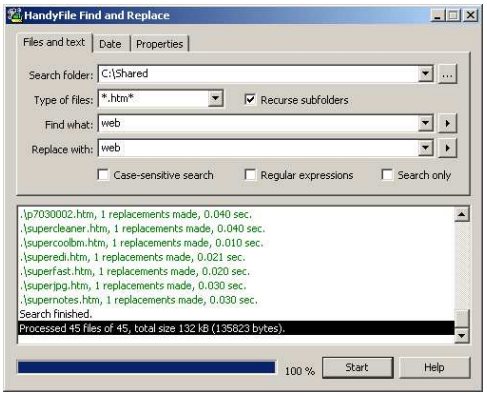
Chapter 2

Background

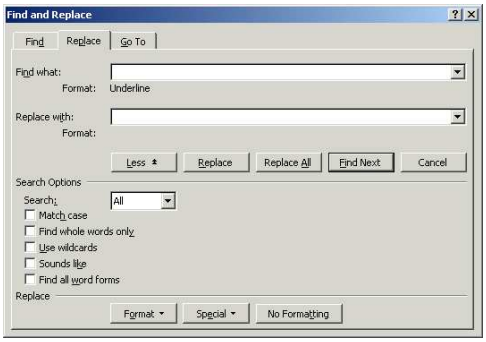
2.1 Find and Replace Programs

Most text editing programs (i.e. MS Word (Figure 2-1(b)), notepad, and emacs) contain a Find and replace tool. A quick search of downloads.com yields over 50 different find and replace programs. Examples include Advanced Find and Replace 1.5.0, HandyFile Find and Replace 1.2 (Figure 2-1(a)), and EFR (Extended Find and Replace) 1.0. All of these tools force users to come up with very specific patterns for their search queries themselves either through the use of regular expressions or by the use of check boxes which allow a user to specify restrictions such as "match case" or "match whole word only". None of these tools allow users to pick a subset of the matches to replace without picking each member of the subset individually.

Regular expressions can be difficult to create, especially with the use of wild cards that may find matches other than the ones intended. When used in a global context, this can lead to errors throughout an entire document. As one user of a find and replace product complained "A one-character typo resulted in hours of restore operations" [2]. Another user mentioned that regular expression matching forced the user "to stay strict with the code" and that it would be an improvement if a find and replace program had "some intelligence in choosing where and when to change" [3].



(a) Handy File Find and Replace



(b) MS Word find and replace interface

Figure 2-1: Examples of find and replace utilities (a) HandyFile Find and Replace (b) MS Word

2.2 Clustering

In machine learning, clustering is a form of unsupervised learning [4]. Unsupervised learning is the process of classifying a set of unlabeled data. This classification method allows one to skip the step of creating and labeling a set of data for future data to be compared against. A standard clustering algorithm takes a set of unclassified data, creates a feature vector for each piece of data, and then groups pieces of data together given their distance from each other in the feature space. The distance between samples in the same cluster should be smaller than the distance between samples in different clusters. Clustering requires that the features used are significant (i.e. a feature that distinguishes just one sample from all other samples is probably not useful).

In the find and replace world, a cluster is a set of matches to a search query that share the same features. Features in the find and replace world include text stylization (bold, italics), surrounding text, and position in the document (in a table or a list). For example, in a search query for the word "hello", a possible cluster of matches might include all examples of the word "hello" that appear in bulleted lines.

2.3 LAPIS

Cluster-based find and replace was implemented as a addition to LAPIS. LAPIS is a lightweight structure text-editing program and the basis for my work. The premise of this program is that users often deal with text that contains some structure they may wish to repetitively edit. Examples of structured text include: e-mails (To, From, and Reply-To fields), HTML code (tags that format the way you see a web page), and web pages (links, advertising blocks, and stylized text. Even this paper contains some structure (headings and the format of the bibliography). Lightweight structure is the combination of the models of various text structures (i.e. tables, HTML or code), a library of structure abstractions (such as phone numbers or words), and the algebra to combine these models and abstractions to create even more powerful abstractions [5]. LAPIS also allows the user to make multiple selections in a document, either manually, through pattern matching, or through inference (having the user highlight a few examples and inferring the type of selection that the user wants to make).

In the find and replace world, the ability to make multiple selections in a document and edit all of them simultaneuosly will be useful in performing a single action on a group of matches to a search query rather than performing an action for each match.

LAPIS uses outlier finding [6] to focus a user's attention on possibly incorrect inferences during multiple selection. In statistics, an outlier is a piece of data that does not look like most of the rest of the data, or a piece of data significantly different from most of the rest of the data. Miller's outlier finder takes the set of text regions that match the pattern being searched for and generates a list of features for each match. In LAPIS features include text stylizations (bold, italics, or font size), position in document (being in a table cell or a bulleted line), and word specific features such as capitalization style, and part of speech. The outlier finder then clusters the matches with identical features together and ranks the clusters based upon their Euclidean distance of the cluster from the median match in the feature space. Matches that are furthest away from the mean are considered to be the outliers of the matched set.

In the find and replace world, the ability to identify outliers will be useful in

identifying clusters of matches to search queries that may need more careful attention paid to them. The ability to apply features to text regions will be helpful in grouping text regions by similarity.

Chapter 3

User Interface

There were three phases in building the user interface: designing, prototyping and testing, and implementation. The designing and prototyping phases were intermixed. After a design was decided on, it would be tested with a small number of users and evaluated according to usability heuristics. Changes to the design would be made based upon the evaluation and the new design would be prototyped and evaluated again. Iterative design enabled rapid improvements in the design.

3.1 Design

The design phase starts off with brainstorming and ends with a preliminary design. During this phase general layout, component functionality, labeling, and component interactions are tentatively decided upon. In this phase designers attempt to iron out any ambiguity in functionality and labeling of the interface. The design phase also takes into account representation of information, feedback, and usability heuristics.

As an extension of current find and replace interfaces, Cluster-Based Find and Replace has the following design issues:

- Consistency with current find and replace interfaces: How can the cluster-based find and replace interface be made intuitive for users of current find and replace interfaces? What needs to remain the same? What can be changed?

- Cluster description: What's the best description of a cluster? Should a cluster be described by its features as a list? Or should clusters be described using examples? Should the user be allowed to specify features it wants the clusters formed around?

- Cluster selection: Should a user be able to select more than one cluster at a time? Should a user be able to select individual matches out of a cluster? How can a cluster be shown to be selected? How should clusters be displayed in the document?

- Cluster display: Should clusters be displayed in a hierarchical manner, such as a tree where users are first presented with large general clusters and can then progress through the tree to smaller, more specific clusters? Or should clusters be displayed in a flat manner such as a table where each row or column contains a cluster of matches?

Initial designs explored the possibilities of displaying clusters in tabular (showing the heads of all the clusters in one list, then showing the members of the cluster in another list when the user selected a cluster), tree (showing the heads of the clusters to the user then allowing the user to expand a cluster-head and view the members in the cluster), and user-driven categorization (give the user fields to split the matches by and make their own matches). I also explored how clusters could be selected, whether multiple selection should be allowed, how selections should be represented in the document being edited, how matches should be represented in the document.

Following the usability heuristics as laid forth by Nielson [8], I placed the buttons related to finding clusters in one area, and replacing clusters in another. There is very little color used in the interface except to call attention to certain areas of interest (ie the clusters).Button labels were carefully selected to convey to the user what would happen when they clicked on it. In order to provide exits, the close button is clearly marked and present at all times in the interface.

3.2 Paper Prototyping

Once a user interface has been tested against usability heuristics, it must be tested with actual users. In the prototyping phase of user interface building, the preliminary design is turned into a paper prototype. Paper prototyping is a process in which a user interface is laid out on paper, and tested on users [9]. The user can interact with the paper modules of the design by using a finger as a mouse and a pencil as a keyboard-like input, while a member of the design team imitates the responses of the program by moving and displaying the components of the paper prototype in response to the user's' actions. This method of prototyping allows a user interface design and function to be tested before committing anything to code. In this phase, ambiguity in functionality missed by the design phase will be discovered and the user interface modified as a result.

For this interface I used a hybrid paper and computer prototype. Users were presented with a paper version of the cluster based find and replace interface and interacted with it via a finger and a pencil. The document they were performing the find and replace task on was shown on a computer screen and all changes (replacements and text highlights upon cluster selection) they made to the document through the find and replace task were reflected on the screen. Users could also scroll through the document on the screen, but all editing had to be done through the paper prototype.

3.2.1 Initial Prototype

The initial paper prototype, shown in Figure 3-1 and Figure 3-2, had two modes which allowed users to switch between one-at-a-time unclustered replacement and a cluster-based interface. The Matches column gave the number of matches in the cluster, the Confidence Level was a bar graph showing how different or strange the cluster in comparison to the most normal cluster. The stranger the cluster, the shorter the bar, indicating a lower confidence in the cluster being a normal cluster. The description explained the similarities in the cluster. A user could indicate selection of a match

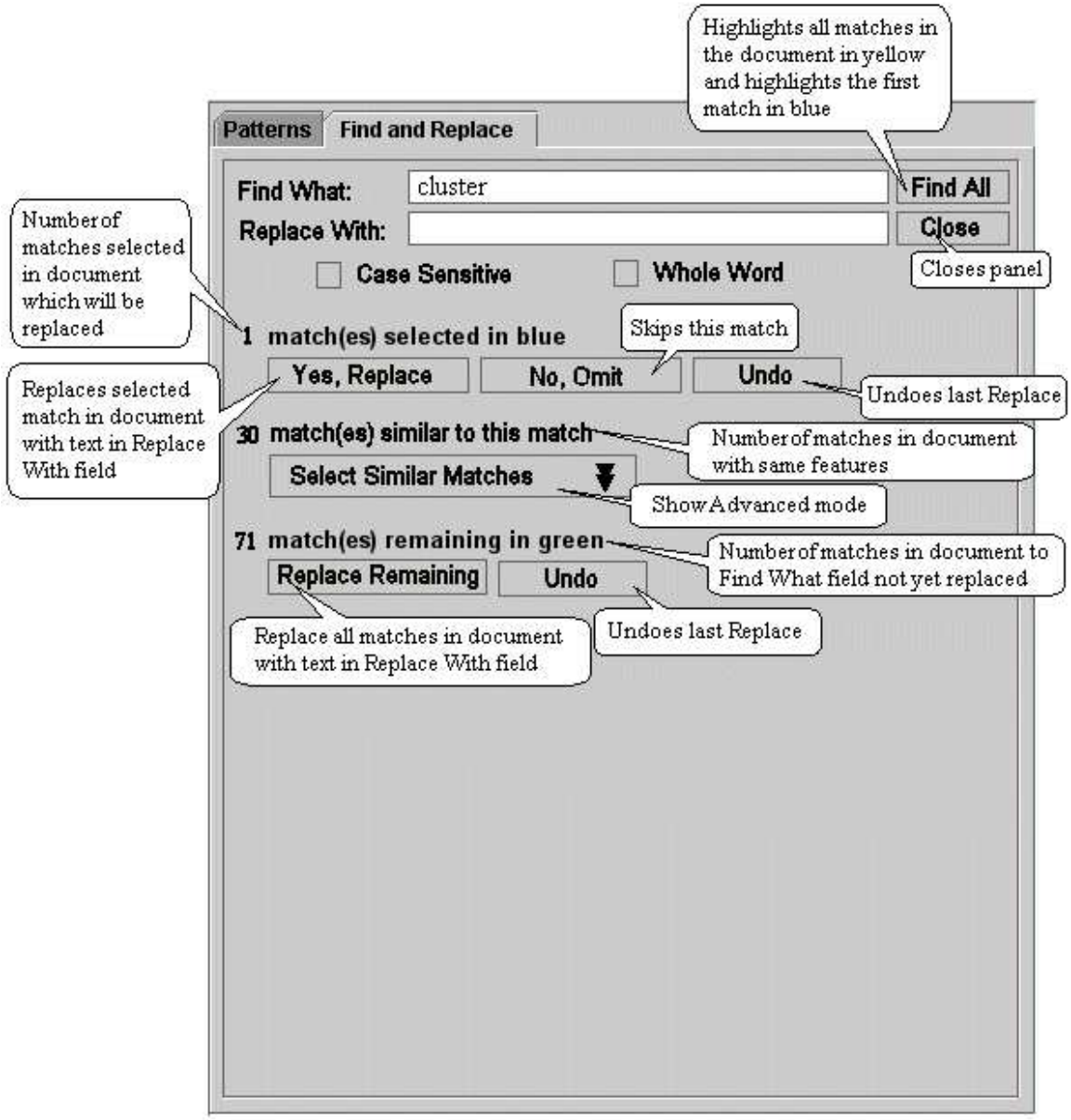


Figure 3-1: The simple mode of the original prototype was meant to emulate standard find and replace interfaces as much as possible while leading the user to become comfortable with the idea of a different find and replace interface.

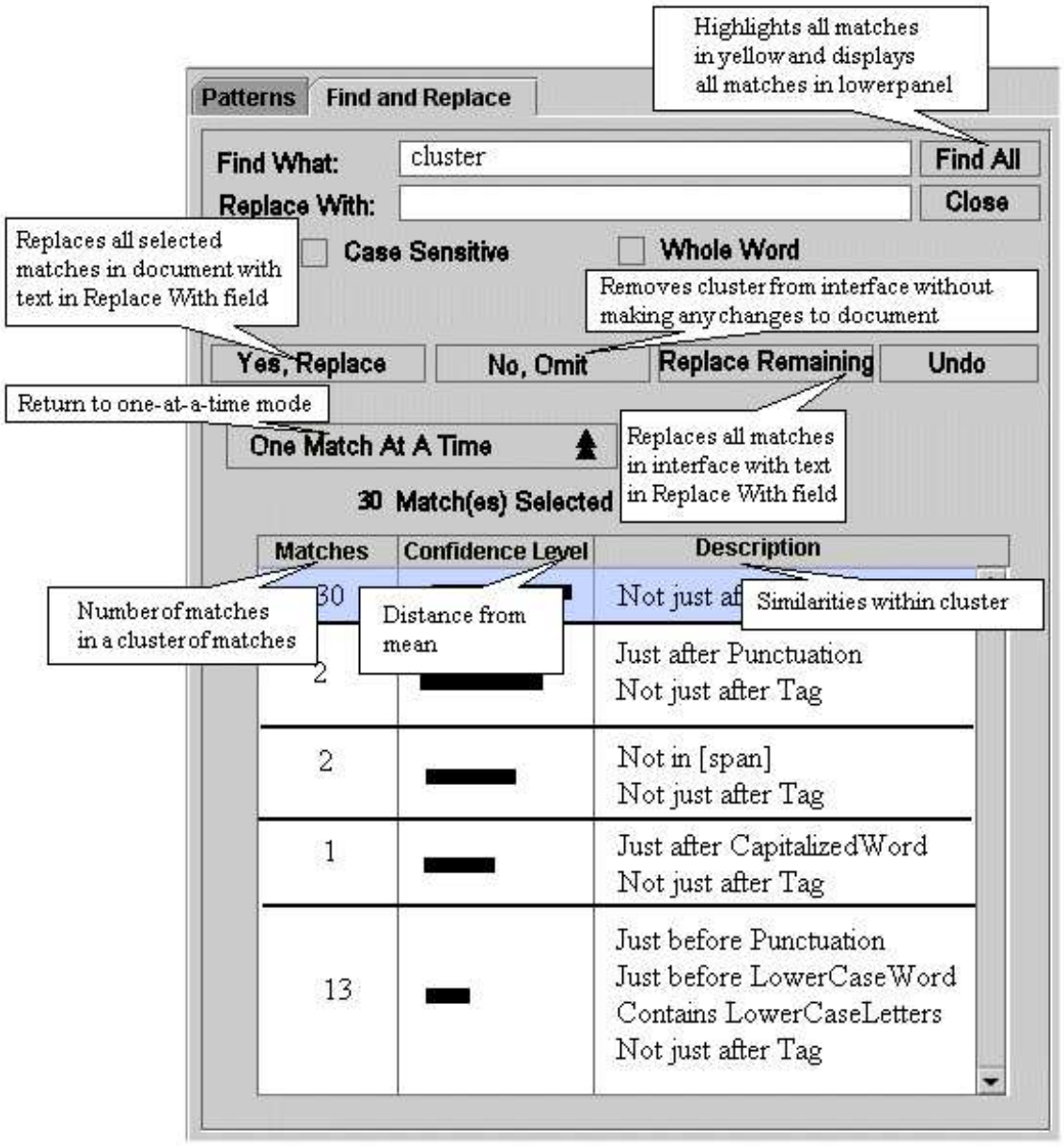


Figure 3-2: The advanced mode of the interface clusters the results to the search string and allows the user to replace a cluster of matches at a time.

or cluster by clicking on the row, at which point the cluster of matches would be highlighted in the document and the row itself would also be highlighted. Pushing "Yes, Replace" would replace the words highlighted in the document with the text in the Replace With field. Pushing "No, Omit" would make the row in the interface disappear while not making any changes in the document. "Replace Remaining" would replace all matches visible in the find and replace interface with the text in the "Replace With" field. A summary of these functions are shown in Figure 3-1 and Figure 3-2.

Testing

The paper prototyping was run on three subjects, each of who were asked to perform the following two tasks.

1. Shown a modified copy of the MIT finals schedule (Figure 3-3), change all examples of "M" to "Monday", but only where M actually represented Monday. When finished the final results should look similar to Figure 3-4.

SUBJECT	SUBJECT NAME	INSTRUCTOR(s)	LOCATION	DATE/TIME
10.32	Separation Processes	**M M**ohr	Johnson	**M** 12/16 **M**orning

Figure 3-3: Header Row and sample row from the MIT finals schedule shown to subject. (All M's are in boldface for emphasis. Table shown to subject did not use boldface.)

SUBJECT	SUBJECT NAME	INSTRUCTOR(s)	LOCATION	DATE/TIME
10.32	Separation Processes	**M M**ohr	Johnson	**Monday** 12/16 **M**orning

Figure 3-4: Header Row and sample row from the MIT finals schedule after correct changes were made. (Monday and all M's are in boldface for emphasis.)

2. Shown a copy of the proposal for this project, change all occurrences and forms of the word "cluster" to "group", but do not change the phrase "cluster-based" to "group-based". Attempt this task using one search pattern rather than

changing all matches of "clusters" to "groups", then changing all matches to "clustering" to "grouping", etc.

Paper prototyping showed that users did not find the confidence level useful. Most thought that it showed the same information as the Matches column. Descriptions were also found to be confusing. For example, many users did not understand what *in [p]* meant. There was also some concern about non-homogeneous clusters. Users did not trust that the individual clusters would be perfect enough and contain either only matches they wanted to replace or only matches they were not interested in replacing. As a result, users were frustrated by the fact that they could not select individual matches to replace.

During paper prototyping, most users did not attempt to use the Omit button. When questioned, users responded they didn't know what clicking it would do. Some felt that clicking Omit would cause the selected clusters to disappear from the document (similar to a delete).

There was also some confusion with the layout. After typing in a word to find, users were unable to quickly spot the button that would cause the matches to show up (Find All). Users were also observed to hit the Replace button before filling in the Replace With field. Some also would hit Replace before Find All and were confused as to why nothing happened (they hadn't selected any matches). Some of these problems were due to the reduced capabilities of a paper prototype, for example, buttons can't easily be made to look disabled, and the mouse cursor can't be changed to add feedback.

3.2.2 Second Iteration of Prototype

The second iteration of the interface, shown in Figure 3-5, removed the Confidence Level column, the Omit and Replace Remaining buttons, and the two level design which would allow users to switch between a one-at-a-time mode and a clustered view mode. Select All and Unselect All buttons were added. The label for the button that performed the replace action was changed to Replace, as I discovered that a

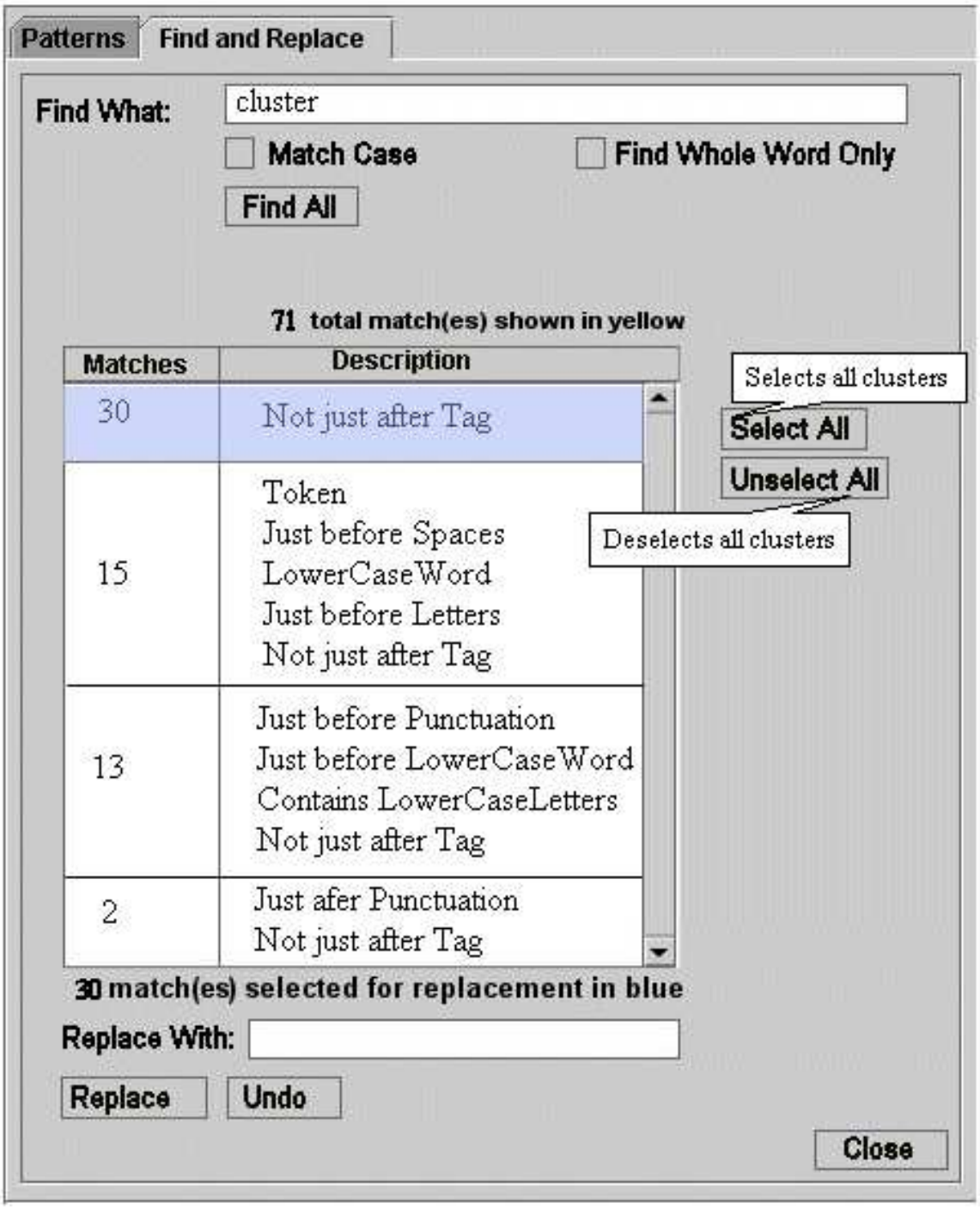


Figure 3-5: Second iteration of the paper prototype. This iteration investigated changing the layout to better match the user's work flow. Areas not labeled have the same function as previous iteration.

more descriptive button label did not help the users and took up too much screen space. The layout of the panel was also changed to better match the user's work flow. By putting the Find All button below the Find What field and the replace related actions below the displayed matches, users followed the components down the screen, could then select the appropriate matches and then do the appropriate replace actions. Descriptions of the clusters remained as they were in the first iteration of the prototype.

Testing

This iteration was prototyped on three users on the same tasks as the previous iteration. Paper prototyping this iteration showed that users did not miss the information regarding confidence level, or the ability to remove clusters. No one asked for these features. There was also less confusion as to what the next step should be. Users were able to quickly locate the Find All button and didn't attempt to click the Replace button before having found anything.

Users still found the descriptions confusing and intimidating. Those that did try to decipher them, appeared confused. Users were also observed to click on a cluster of matches in the interface and not notice that the matches were highlighted in the document. There was also some confusion as to why there wasn't just "one correct cluster". Some users felt that they should only need to select the one correct cluster.

A surprising thing noticed during prototyping was that users would try to select all the correct clusters before performing a replace, rather than replacing one or a few clusters at a time as we thought they would. It wasn't clear at the time whether or not this was an artifact of the paper prototyping. As it turned out, this behavior continued to appear in user studies of the implemented interface.

3.2.3 Third Iteration of Prototype

In the third iteration, shown in Figure 3-6, I replaced the description of the clusters with an excerpt of the text surrounding the match in the document, with the match

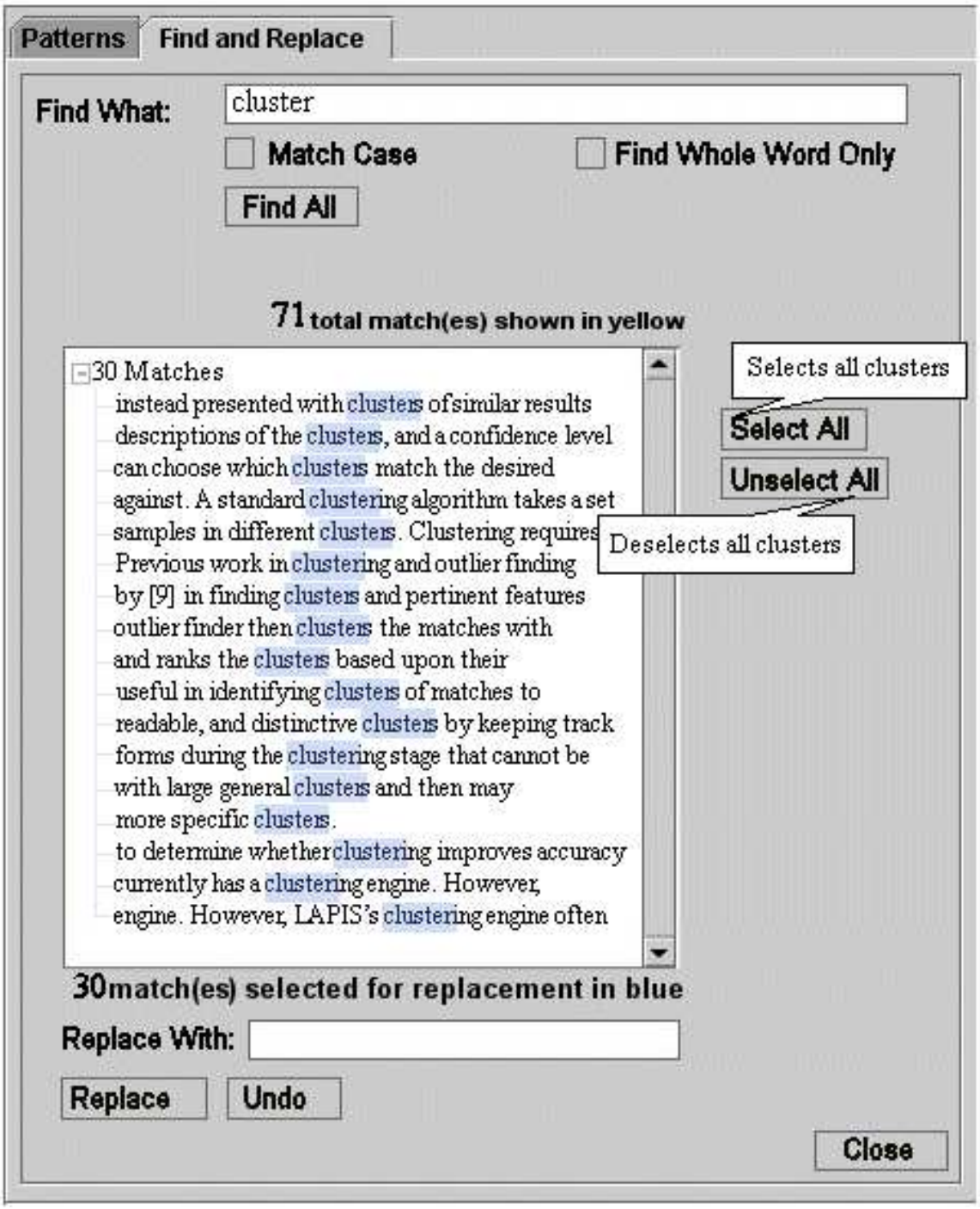


Figure 3-6: Third iteration of paper prototype. This iteration presented the matches in a tree structure grouped by similarity. Matches are now described by excerpts from the document.

itself highlighted in blue as it would appear if it were in the document and selected. The representation of the matches and descriptions were changed from a table representation to a tree representation. Clusters were ordered from largest cluster on the top to smallest cluster. Matches to the search string that were unclusterable were displayed at the end. Initially the entire tree was closed. To open a cluster, the user had to click on the + sign. Matches could be selected individually by clicking on its corresponding row, or as a cluster by selecting the top row of the cluster.

We also briefly tried a few users with a version of the Omit button under a different label (Reject, Exclude and Discard).

Testing

This iteration was prototyped on three users. Their first task was the same as the first task in previous iterations (replacing the letter M with Monday where appropriate). The second task was on the same document as the second tasks in previous iterations, but subjects were instead asked to replace all examples of the words "clusters" and "clustered" with "groups" and "grouped" respectively, but not to change the words "clustering" or "cluster".

Paper prototyping this iteration showed that users appreciated the ability to partially select a cluster in the case of heterogeneous clusters. Users appreciated the excerpts of the document. They found it made it easier to quickly scan the list and pick out which examples matched the pattern they were looking for. An interesting behavior was noticed, however. Users did not tend to select an entire cluster at a time. User instead would scan the list of matches and decide on each match if it was correct or not, very similarly to the way users would handle a standard find and replace interface.

When we first presented the collapsed tree, users complained that in order to perform the task, they were required to open up every cluster to scan it as they didn't know why a cluster was grouped the way it was. Presenting a completely expanded tree, though, led to the one-at-a-time selection for replace where the user would scan through the tree as though it were a list and chose whether or not to

select that row.

The Omit-type button was universally considered confusing, regardless of its label, as noted in the first iteration. Subsequent designs omit it entirely.

3.2.4 Fourth Iteration of Prototype

In this fourth iteration, shown in Figure 3-7, I changed to a two list format. The cluster headings were in the list labeled Matches. When users clicked on one of the clusters, the Description section would be populated with the individual matches belonging to the cluster. Users could chose to select all the matches in the cluster by clicking Select All, or select the matches individually per cluster if they desired. I hoped that this would encourage users to deal with whole clusters at a time rather than individual matches at a time.

The rest of the functionality of the interface remained the same.

Testing

This iteration was prototyped on two users on the same tasks as the previous iteration. Paper prototyping this iteration showed that users really wanted to be able to select all the correct matches to replace before performing a replace. Some users attempted to display all the matches at once so they could go through the entire set of matches at once.

I also asked users to compare this interface to the second iteration. Users preferred context descriptions of the matches over feature description of the clusters.

There remained some confusion as to why matches were clustered the way they were. Users were also not happy about needing to interact with each cluster on its own since they didn't know anything about the cluster from the description "30 matches".

3.3 Implementation

From the user feedback from the paper prototyping I decided to implement a tree representation similar to the third iteration (Figure 3-8).

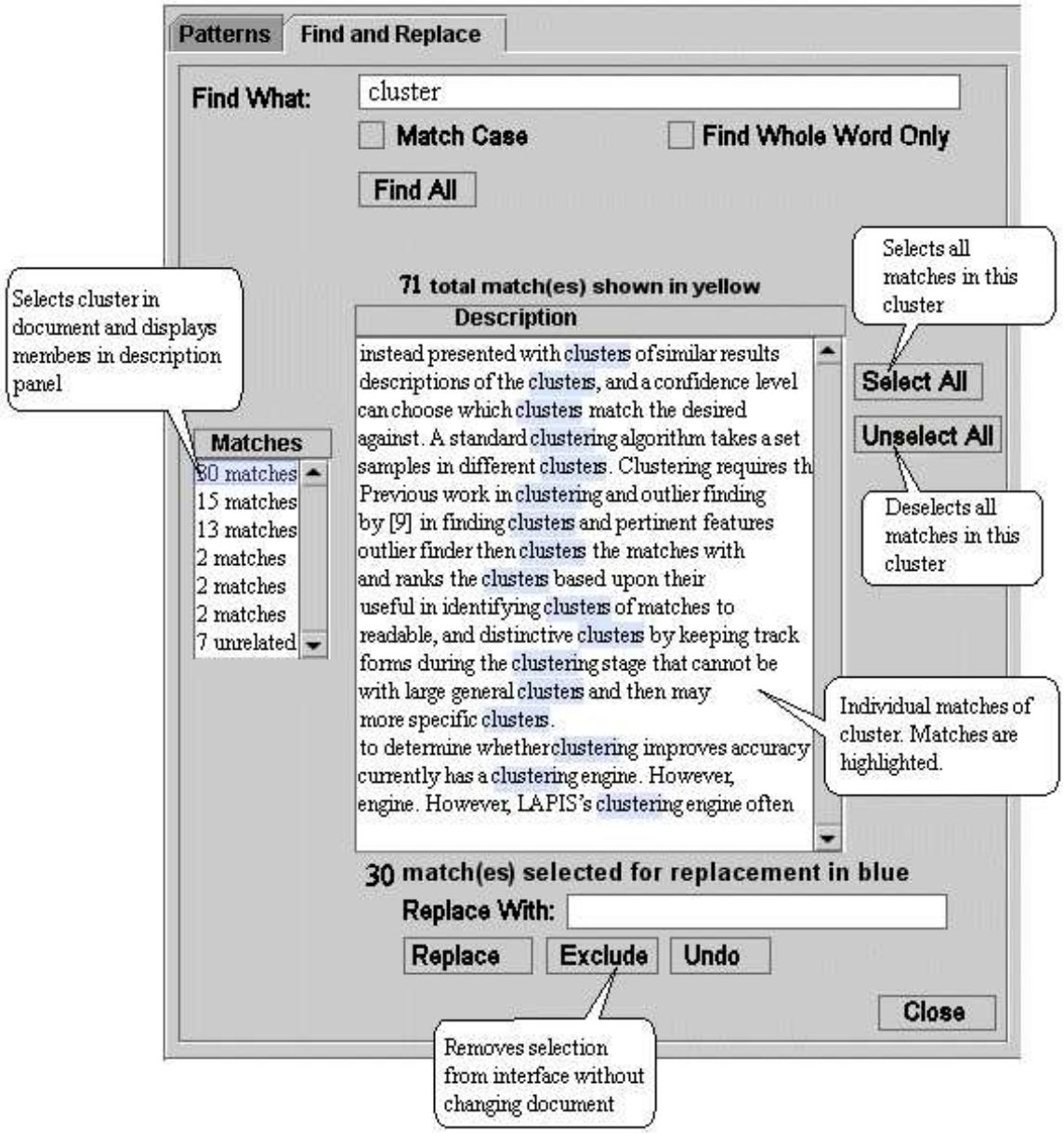


Figure 3-7: Fourth iteration of paper prototype. This iteration explored presenting the matches in a two-list structure where clusters were presented in the left list and the individual matches of the selected cluster were presented in the right list.

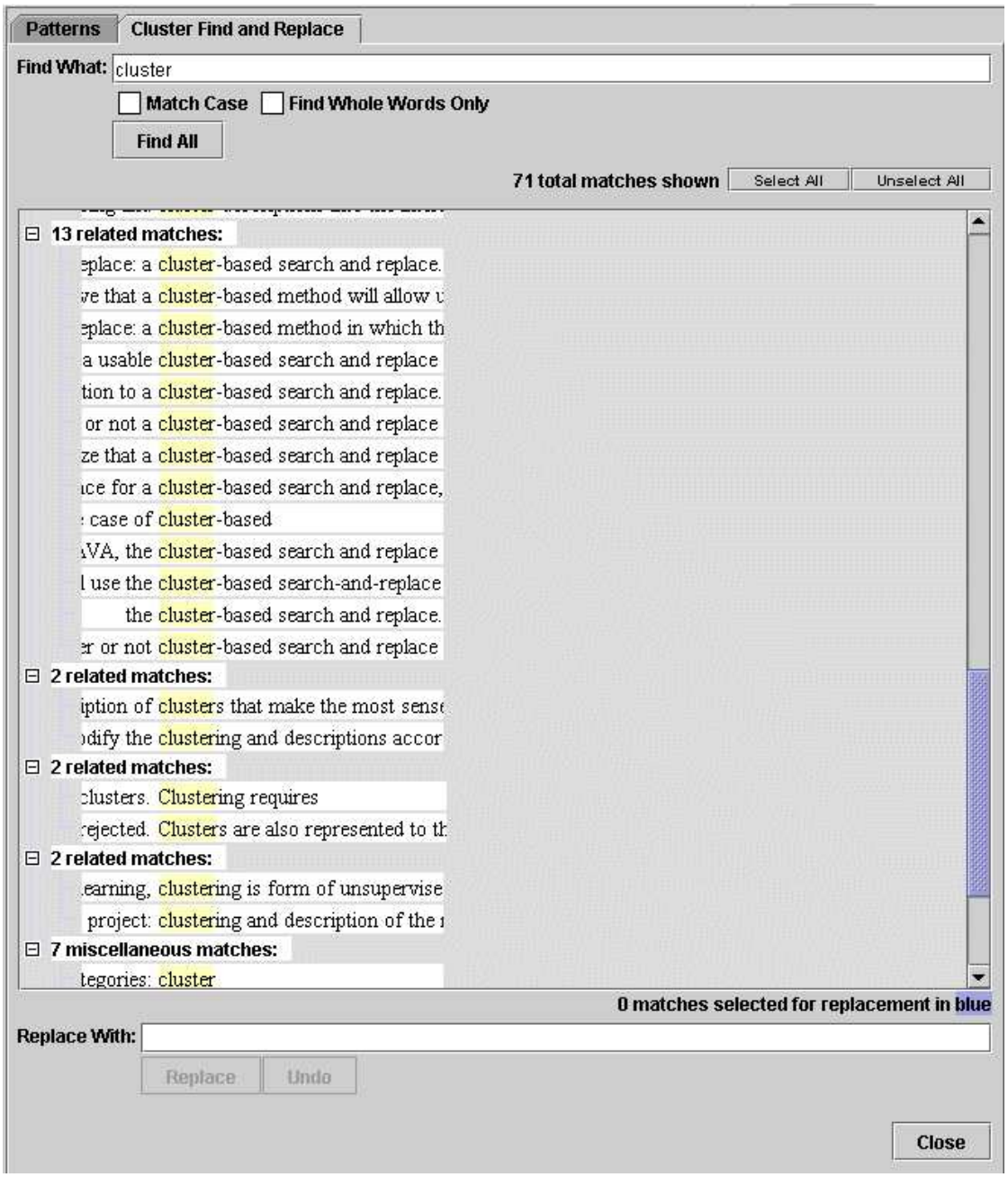


Figure 3-8: Screenshot of Tree Representation of clusters. Matches are clustered into groups by similarity and are ordered by the number of matches in each group.

When users are first presented with the interface, they see only the Find What field, check boxes for controlling case sensitivity and whole words search and the Find All button. The Find All button is disabled until something is typed in the Find What field. By only presenting the users with a few things they could do, this minimized the confusion as to what the next step should be as there were a limited number of options.

After typing a search string into the Find What field and either hitting the Enter key or the now enabled Find All button, users were presented with the matches to their search query grouped by similarity and sorted by size of the resulting groups.

When the user clicks on one of the matches in the find and replace interface, the corresponding match is highlighted in the document in blue and the selected row is highlighted in the interface. Until the user selects at least one match, the Replace button is disabled.

Match selection is toggle on click. If the user clicks on a match that is already selected, that match is deselected and vice versa. If the user clicks on the top level of the cluster, all matches in the cluster are highlighted in the document and the entire cluster is highlighted in the find and replace interface.

Originally, match selection followed standard Java conventions, where toggle on click was only active if the control key was held down. By holding down the shift key, users could select a contiguous collection of rows. Early user studies showed that not everyone was familiar with Java conventions for multiple selection. Some users were observed noticing that making a selection would clear the previous selection and then selecting and replacing each match one at a time. This set of actions frustrated some users as they found the process tedious.

Early user studies showed that users also had trouble realizing that nodes were selectable in the interface. This problem didn't appear in the paper prototype, perhaps because paper prototype users were not able to interact very much with the screen and were therefore forced to interact with the interface. In order to help users figure out that the nodes in the tree are selectable, feedback was added. When the user puts the mouse cursor over or near one of the nodes, the node is highlighted so as to

appear three dimensional.

In the tree representation, users can collapse clusters by clicking on the - and later expand them by clicking on the +. When the cluster is closed, the users are shown how many matches are in the cluster and an example of one of the matches. As mentioned before, paper prototyping showed that users were confused by the feature descriptions of the clusters. It was finally decided that the best description of a cluster was the cluster itself. Users are better able to form their own idea of the cluster than a program is able to create human readable or natural language descriptions of a cluster.

In the list representation, users can only look at one cluster at a time and select matches within that cluster. When users switch clusters, their previous selection is lost. When users look at a new cluster, all the matches are originally selected.

Users are kept aware of the total number of matches that are shown in the interface and the number of matches that have been selected for replacement through the labels above and below the match display section of the interface.

Early implementations of the interface also highlighted all the matches to the search query in the document in yellow as shown in Figure 3-9. This was later removed as some users were confused by the yellow highlighting and felt that a Replace would change all text highlighted in yellow. Other users were observed to go through the document scanning the yellow text and manually selecting the yellow text they wanted to replace. their selection to what they wanted.

The entire implementation of the cluster-based find and replace interface is approximately a 6,000 line addition to LAPIS and is written in JAVA.

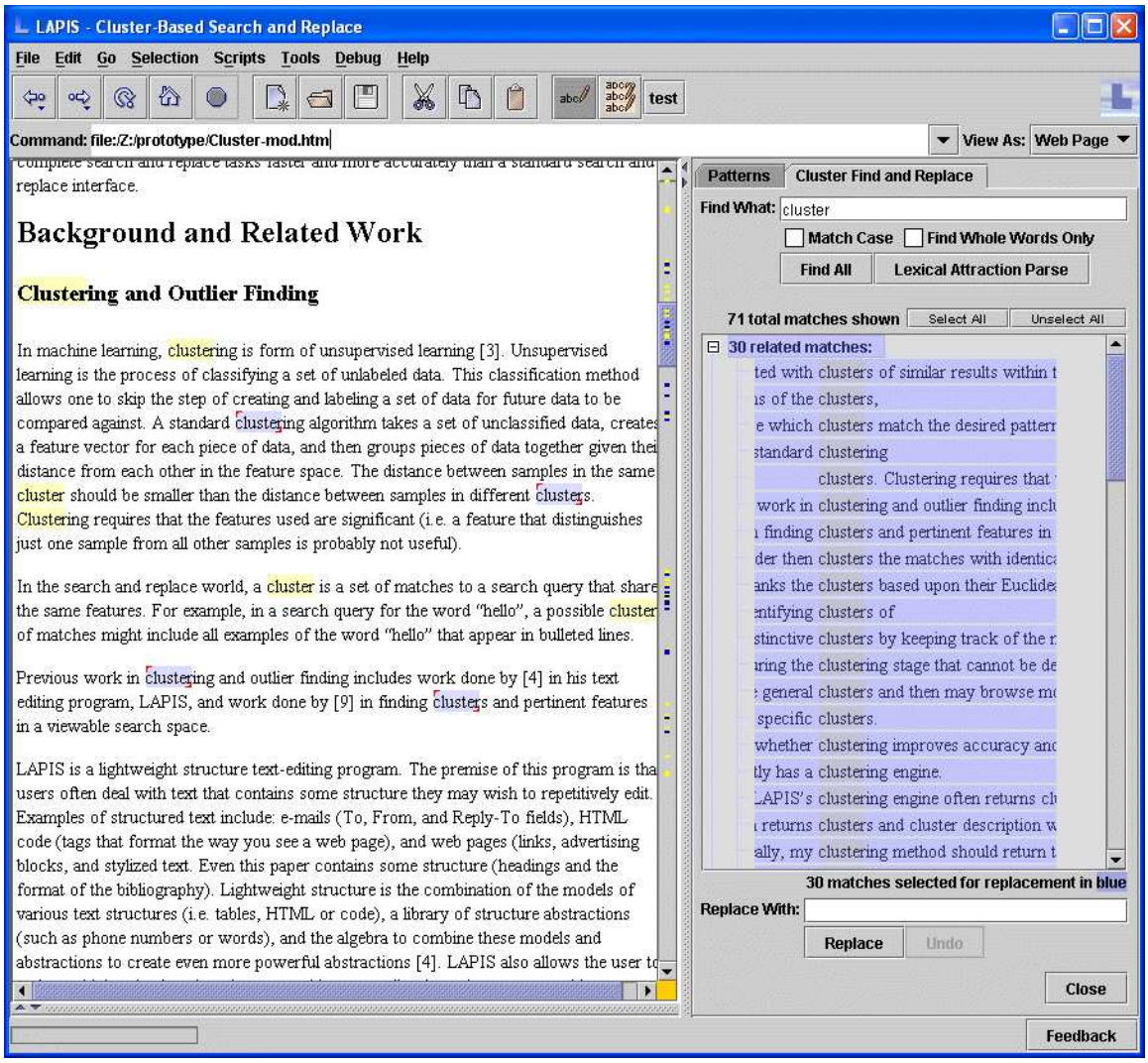


Figure 3-9: Screenshot of all matches highlighted in yellow while selected matches are highlighted in blue

Chapter 4

Testing

The goal of this study was to test whether or not a cluster-based find and replace interface made a difference in user accuracy or speed when performing a find and replace task. The cluster-based interface was compared to a standard find and replace interface and a list based interface. Comparing the cluster-based and standard find and replace interfaces showed if there was any value in clustering the results. Comparing the cluster-based and list find and replace interfaces showed whether the value in the cluster-based interface was the clustering that assisted the user or the fact the could see all the matches at once and in context.

4.1 Method

Participants were 7 males and 6 females ranging from 18 to mid-30's. Subjects were recruited through fliers posted around campus and the Brain and Cognitive Science experimental subjects list advertising a user interface study. Subjects were told that they would be paid $5 for each half hour they participated. Subjects were asked to have some experience with word processing.

Each subject completed three find and replace tasks, one on each of three different interfaces. The first two tasks were the same tasks as the first iteration of the paper prototype as described in section 3.2.1. For the third task subjects were shown a copy of a paper and told that they were planning on submitting it to a British

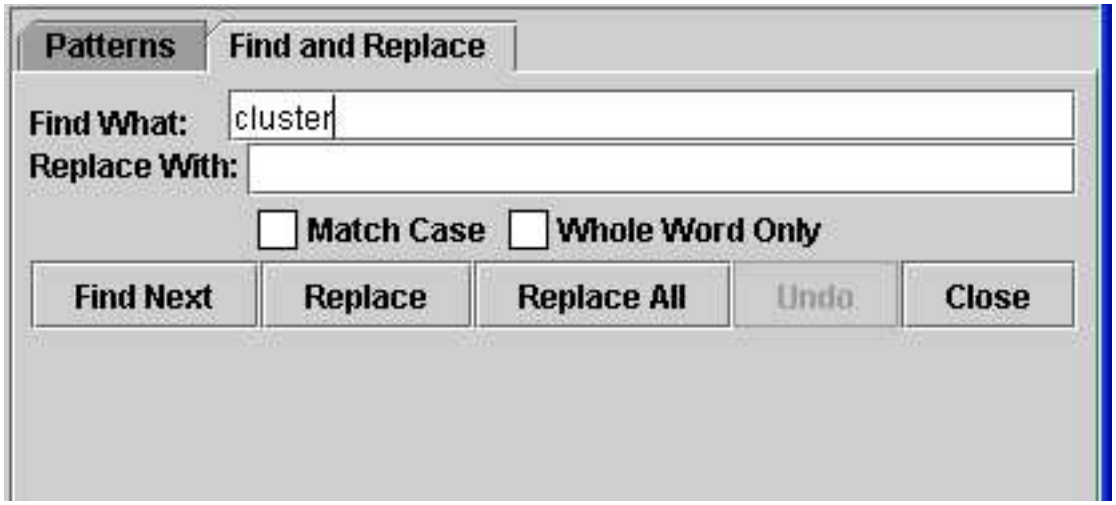


Figure 4-1: Standard Find and Replace interface

journal. However, since the paper was written in American English, they needed to make a few spelling changes. They were asked to change all forms of the word "labor" to "labour" but to not change "elaborate" to "elabourated" or "laboratory" to "labouratory". They were also asked to ignore any other words that may need changing, for example color or tire.

The following three interfaces were used:

1. A conventional interface similar to Microsoft Notepad, but implemented in LAPIS (Figure 4-1). This interface was a standard one-at-a-time with a prompt for user response at each match. Matches were presented in document order.

2. A unclustered interface (Figure 4-2) that showed all matches to the user's search query at once. Users then could select which matches to replace. Although multiple selection of matches was supported, the matches were not clustered. Instead matches were simply presented in document order.

3. The tree representation cluster-based find and replace interface described in section 3.3 and shown in Figure 3-8.

The order of interfaces and tasks varied from user to user and was predetermined so that approximately equal numbers of users were presented with a specific ordering of interfaces and tasks.

Figure 4-2: Screenshot of List representation of matches. Matches are shown in document order.

Before the experiment, subjects were given a brief pretest to assess their abilities with text-editing programs, features within some text-editing programs such as find and replace and regular expressions, and computer programing. Subjects were also asked about their find and replace history, how they had performed such tasks in the past, and how often they had performed the task in that manner.

Subjects were then presented with a paper describing the three tasks they were to complete. The description of each task contained examples showing how the document should look before and after making the appropriate changes. Subjects were told to carefully read and complete the tasks in the order indicated. At the completion of each task subjects were told to notify the experimenter so that the find and replace interface could be changed.

Subjects could ask for and receive clarification on tasks, but questions regarding the functionality of the interface components or of the "How do I ..." variety were answered with encouragement to try out what ever they thought made sense or to experiment.

After the subjects completed all three tasks to their own satisfaction, they were given a post test to judge their impressions of the three interfaces they had used. Subjects were asked how well they trusted the system to display useful sets of related matches, and whether they felt they would use a cluster-based find and replace in their own tasks if it was built into their favorite word processor.

All tasks on the computer were monitored through the use of screen capture software which captured all changes on the screen, including cursor movement and keystrokes.

4.2 Results

Number of errors were calculated by doing a strict comparison using the UNIX diff command between the subject completed document and a document where all the find and replace tasks were completed perfectly. Errors included, but were not limited to, capitalization errors, missing text, and added text.

	Interface			Task		
	Conventional	List	Clustered	M → Monday	cluster → group	labor → labour
Time (sec)	251 ($\sigma = 158$)	262 ($\sigma = 101$)	134 ($\sigma = 83$)	262 ($\sigma = 157$)	236 ($\sigma = 127$)	161 ($\sigma = 84$)
Error	60 ($\sigma = 118$)	9.8 ($\sigma = 10$)	44 ($\sigma = 104$)	351 ($\sigma = 260$)	15 ($\sigma = 10$)	12 ($\sigma = 7.9$)

Table 4.1: Mean and Standard Deviation (σ) of speed and accuracy of completed tasks by interface and task.

Time to complete tasks was calculated from the time the user made its first move toward starting the task (time of first movement of cursor), to time of last movement of the cursor.

Table 4.1 shows the basic performance statistics of subjects on the three interfaces and tasks tested. Four data points were left out of the speed calculations by interface (one each from conventional and list interfaces, and two from clustered interface) as they were enormous outliers (more than two standard deviations from mean). The corresponding outliers were also removed from the speed calculations by tasks.

For time taken to complete each task by interface, using a one-way ANOVA, the differences between interfaces were not significant with $p > 0.80$. However if you remove the enormous outliers (the four occurrences where the subject's completion time was more than two standard deviations from mean), using a one-way ANOVA, the results do become significant with $p < .05$. Users completed their tasks faster on the cluster-based interface than on the other two interfaces.

For number of errors in completed documents by interface, using a one-way ANOVA, the results were not significant with $p > 0.60$. None of the interfaces were significantly more accurate than the others.

A glance at the data appears to indicate that accuracy might be more related to the task. The M → Monday task contained all the enormous outliers and all but one of the perfect scores. A summary of the number of errors by task is shown in Table 4.1. However, these results using a one-way ANOVA were also not significant with $p > 0.06$.

Using a correlation test, there was strong correlation between the average amount of time to complete each task and the number of errors the subject made ($r = 0.73$).

The longer the subject took to complete the task, the more errors they made. Further investigation showed that these results may be the result of the user's performance on the standard find and replace interface as the correlation between performance time and accuracy for the list ($r = 0.18$) and clustered ($r = 0.16$) interfaces were not significant.

The cluster-based interface showed the fastest completion times and 2 of the slowest completion times. It also showed some of the worst error rates. The list interface showed the lowest average number of errors and also the highest number of error free completions (5).

4.2.1 Discussion

The results of the initial user study were not very promising in terms of accuracy. In terms of speed, however, the user study showed that users can complete tasks faster using a cluster-based interface over a conventional interface. Also, during the user study we discovered a number of issues with the interface that may have confused users. As a result, numerous changes were made to increase feedback to the user during a find and replace task. These changes include providing more feedback as to what was selectable in the interface such as nodes in the tree by highlighting the node when the user placed the mouse over it. Preserving the case of replaced text in the replacement text was added. For example if the user's find query returned "The" and was replaced by "some", the result in the document would be "Some".

Chapter 5

Conclusion

It is very difficult to describe clusters. Throughout the user studies, I discovered that users seldom found a description of a cluster useful. Often it made their progress slower as they had to read every description, process it, and check whether or not their translation of the description matched their own idea for the pattern they were trying to match. Some users would look through the descriptions for certain key terms they had in mind and be lost if they couldn't find them. Ultimately, it is difficult to predict how humans will classify things. Every person will have their own internal description of a pattern. Writing an algorithm that can predict exactly how that person would describe a group is almost impossible. The next best solution for creating a written description of a group is to show examples of the group. If people have different descriptions for the same patterns, the best thing would be to let them make up their own descriptions based on what they can perceive.

Representation of clusters can also be difficult. Without clear separations between clusters, users run the risk of unintentionally selecting more than one cluster to interact with. Other cluster-based interfaces will need to take this into account and make the divisions between clusters very clear so that users will notice when they have switched clusters.

As the results of the user study were not as promising as I had hoped, I have made some changes to the interface. I am in the process of starting a new user study, the results of which will not be available for this paper, but should be available in the

future.

5.1 Future Work

Another repetitive text-editing task that could benefit from a cluster-based interface might be spell-checking. Conventional spell-checkers require that each match be dealt with individually. A cluster-based approach might group the misspelled words by text stylization so that code fragments, equations, and quotes embedded in the document can quickly be identified and ignored, while words that are actually misspelled can be corrected appropriately.

Clusters might not be intuitive to users. In the user studies, if the user had access to the individual matches of the cluster, they seldom selected the entire cluster at once. Instead they would quickly select all the matches in the cluster one-at-a-time. Clustering's biggest advantage appears to be that users are less likely to accidentally replace a match they didn't intend to if it's located among a number of other matches they did mean to replace. Clustering the matches means that matches are presented so that the user can replace or select a lot of matches in a row, then not replace a lot of matches in a row. It would be interesting to test if users use clusters as separate groups or as a sorted list.

5.2 Contributions

My contributions to User Interface research are as follows:

- Exploration of design alternatives for a cluster-based find and replace.
- Evaluation and documentation of the results of these design alternatives on users.
- Implementation of a design for cluster-based find and replace in a tree representation.

- Comparison of a cluster-based find and replace interface with conventional find and replace interface which found that clustering may not help in task accuracy and speed, but more research is required.

Bibliography

[1] Denise Brehm. Web project aims to increase student fluency in mathematics. MIT News Office, May 14 2003.

[2] Downloads.com - user opinions for handyfile find and replace. [Online Document] http://download.com.com/3302-2068-9098569.html.

[3] Downloads.com - user opinions for search and replace 98. [Online Document] http://download.com.com/3302-2048-916215.html?ob=0&pn=2&fb=0.

[4] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. Wiley-Interscience, New York, NY, 1973.

[5] R.C. Miller. *Lightweight Structure in Text*. PhD thesis, Carnegie Mellon University, Computer Science Department, School of Computer Science, 2002.

[6] R.C. Miller and B.A. Myers. Outlier finding: Focusing user attention on possible errors. *ACM Symposium on User Interface Software and Technology (UIST 2001)*, pages pp 81–90, 2001.

[7] P. G. Neumann (moderator). Risks digest: Forum on risks to the public in computers and related systems. [Online document] http://catless.ncl.ac.uk/Risks/ v17 n57, v19 n12, v22 n71.

[8] J. Nielsen. *Usability Engineering*. AP Professional, 1933.

[9] M. Rettig. Prototyping for tiny fingers. *Communications of the ACM*, 37(4):21–27, 1994.