Flexible Diff-ing In A Collaborative Writing System

Christine M. Neuwirth
Ravinder Chandhok
David S. Kaufer
Paul Erion
James Morris
Dale Miller

Carnegie Mellon University
Pittsburgh, PA 15213
prep-project+@andrew.cmu.edu

ABSTRACT

An important activity in collaborative writing is communicating about changes to texts. This paper reports on a software system, *flexible diff*, that finds and reports differences ("diffs") between versions of texts. The system is flexible, allowing users to control several aspects of its operation including what changes are reported and how they are shown when they are reported. We argue that such flexibility is necessary to support users' different social and cognitive needs.

KEYWORDS

Text comparison, collaborative writing, flexible differencing

INTRODUCTION

In the course of a collaboration, co-authors often make changes to each other's documents. A principal difficulty co-authors face is coping with those changes, especially understanding why the other person made them. For example, in a study of eight writers' production of an insurance company's two-page annual report, Cross [3] observed that each writer "omitted, added, highlighted or modified" the text to agree with his or her preconceptions, with unexplained changes causing "considerable frustration" for other writers (p. 193).

This paper describes an approach to communicating about changes to documents. At the heart of the approach is a flexible text differencing system, *"flexible diff,"* that allows collaborative writers to tailor difference reports to their various social and cognitive needs. The program is imbedded in the PREP Editor [13; 14; 2], a writing environment being developed to study and support collaborative writing processes. The PREP Editor allows difference reports to be annotated with such things as explanations of changes.

ISSUES IN THE DESIGN OF A FLEXIBLE DIFF-ING SYSTEM

The technical question in the design of a difference finding program is how to take an original text and a revision, and to produce--in a time and space efficient way--an edit script (or difference report) that describes changes between the two versions. From a time/space efficiency standpoint, the "optimal" difference report is in the form of a list that describes the minimal number of editing changes required to transform the original text into the revision. This technical question has received considerable research attention [7; 10; 11] and is not the focus here. The focus in this paper is on the following human interface design questions and our approach to answering them:

- What changes should be reported?
- How should changes that are reported be pinpointed?
- What should the user interface to the change report be like?

In the course of answering these questions, we argue that, from a human factors standpoint, an optimal time/space difference report is not necessarily optimal.

What Changes Should Be Reported?

The most fundamental design question we addressed was "What changes should be reported?" One might think that the answer is obvious: "All of them." This answer, however, is not unconditionally correct. Nachbar [12] argues compellingly that reporting all changes is inappropriate for some tasks, such as comparing the output of floating point calculations (where roundoff errors result in spurious differences) and comparing program source code (where differences in white space and comments have no effect on the operation of compiled code). In response to these task demands, Nachbar's *spiff* (spiffy diff) provides a flexible diff-ing interface; for example, a user can instruct *spiff* to ignore differences between floating point numbers that are below a user-settable threshold, a feature that allows users to compare the outputs of a statistical package running on two different machines approximately 200 times faster than when using *diff* [7] to generate the report.

We identified three aspects of collaborative writing tasks that require developing a flexible difference finding algorithm:

- Different roles within a writing group;
- Changes in the needs of collaborative writing groups over time;
- Different types of collaborative writing groups.

Different roles within a writing group
A writing group is usually composed of different people fulfilling several different social roles. While a co-author may not be interested in seeing every change that a trusted co-author made, he or she may want to check every change from a less trustworthy editor. For example, Cross [3] observed that a secretary acting as editor changed "crises that could not be overcome" to "crises," because of a belief that crises, by definition, could not be overcome.

Changes in the needs of collaborative writing groups over time
A writing group's needs for detecting changes may be different at different times during the lifetime of a project. At the beginning of a project, co-authors may be changing a draft so radically that it would not be useful to see changes at a detailed level. In those cases, it is less likely that a co-author would want to detect what was inserted, but he or she may want to answer the question, "Where did my 'good bit' go?" At later stages of the project, for example, when responding to reviewer's comments, it may be useful to look at each change a co-author made.

Different types of collaborative writing groups
Different groups often need very different ways to detect changes. For some tasks, writers want to see every last change that was made from the previous version--for example, lawyers doing contract work. If, for example, the reader is a lawyer whose task it is to find all the changes that may have legal implications, then it certainly makes sense to report all of them. To illustrate, Figure 1.a. depicts the original of a legal document; Figure 1.b., a revised version, and Figure 1.c, a change report with all changes shown.[1]

[1]Strike-through corresponds to deleted material; underline corresponds to inserted material. Example based on material from [15].

But in another type of writing group, writers might actually be distracted by seeing every last change because the task might be quite different. If the reader is a co-author, for example, the tasks might be to read the new draft for meaning and, at the same time, to build a representation of the problems that remain in the paper. Such a reader may not be interested in all the changes, many of which draw attention to problems in the previous draft that are no longer of interest because they have been solved.

a. Original	The Pennsylvania law also establishes a time limit within which injuries or occupational diseases must be reported to an employer. Under this provision, compensation will be paid based on the date of the injury only if the injury is reported to your employer within 21 days of its occurrence. After 21 days, compensation will begin based on when the report is made. The law bars compensation for any injury or disease reported more than 120 days after the occurrence.
b. Revised	The Pennsylvania law establishes a time limit within which injuries or work-related illnesses must be reported to an employer. In order to receive full compensation, that is, compensation based on the date of the injury or the onset of illness, you must report it to your employer within 21 days of its occurrence. If you report it between the 22nd and 120th day, you will receive partial compensation, that is, compensation based on the date you file the claim. If you report it more than 120 days after the injury or onset of illness, you will receive no compensation.
c. All changes reported	The Pennsylvania law ~~also~~ establishes a time limit within which injuries or ~~occupational diseases~~ work-related illnesses must be reported to an employer. ~~Under this provision,~~ In order to receive full compensation, that is, compensation ~~will be paid~~ based on the date of the injury ~~only if the injury is reported to~~ or the onset of illness, you must report it to your employer within 21 days of its occurrence. ~~After 21 days, compensation will begin based on when the report is made. The law bars compensation for any injury or disease reported~~ If you report it between the 22nd and 120th day, you will receive partial compensation, that is, compensation based on the date you file the claim. If you report it more than 120 days after the ~~occurrence.~~ injury or onset of illness, you will receive no compensation.

Figure 1. An example revision and difference report.

Instead, the reader may prefer to see only some changes--for example, sentences or paragraphs that have been deleted, on the grounds that destruction of "hard won" sentences or paragraphs is an interesting event (see Figure 2.c). Indeed, a difference report of all the changes to the document might induce such a reader to focus on task-irrelevant information at the expense of building meaning and a representation of problems remaining (Figure 2.b.).

To understand how a change report might interfere with the tasks of building meaning and a representation of remaining problems, it is useful to look at what role attention plays in the process of reading. Most reading models assume that attention is required in order to derive meaning from texts and that the amount of attention available is limited, so attention must be allocated [15]. When two or more tasks exceed attentional capacity, the tasks cannot be performed simultaneously and readers engage in "attention switching," first allocating attention to one task, then the other. For example, beginning readers have difficulty decoding words and building meaning simultaneously, and must often switch attention from one task to the other; however, attention switching comes at a cost: It is time consuming, puts a heavy demand on short-term memory and tends to interfere with recall [8].

a. Original	Finally, and somewhat speculatively, we wondered if more experienced teachers would use electronic communication modes differently than less experienced teachers. New computer technologies do not, in and of themselves, create educational improvements; instead, they create opportunities for improvements.
b. All changes reported	Finally, and somewhat speculatively, we ~~wondered if more~~ hypothesized the following about individual teachers: More experienced teachers ~~would~~ and their students will use electronic communication modes ~~differently~~ more than less experienced teachers. ~~New computer technologies do not, in and of themselves, create educational improvements; instead, they create opportunities for improvements.~~
c. Sentence level changes reported	Finally, and somewhat speculatively, we hypothesized the following about individual teachers: More experienced teachers and their students will use electronic communication modes more than less experienced teachers. ~~New computer technologies do not, in and of themselves, create educational improvements; instead, they create opportunities for improvements.~~

Figure 2. All changes reported vs. sentence level changes.

Based on our initial experience with difference reports,[2] it is our hypothesis that reading for meaning, building a representation of remaining problems and detecting differences are tasks that require attention-switching for many readers. Indeed, the report of a change seems to invite the reader to focus on the change.[3] But an "invitation" can distract the reader from more appropriate tasks. Lesser and Erman [9] define "distraction" as the degree to which an agent's focus can be shifted. Positive distraction shifts an agent to tasks that are more useful; negative distraction shifts the agent to tasks that are redundant or diversionary. The distraction in a difference report might be negative, leading the reader to switch tasks and focus on the changes to the detriment of constructing the gist of the new material. Moreover, a reader with limited metacognitive capacity may fail to detect distraction or may recover from the problem only after considerable misspent time.[4]

This framework allows us to give a general answer to the question, "What changes should be reported?" In general, only those changes should be reported that will reduce negative distraction and increase positive distraction. Since this depends, as we have argued, on the reader and the task, a differencing program for collaborative writing needs to be flexible, to allow readers to specify what changes to ignore.

Like *spiff* [12], *flexible diff* achieves flexibility by allowing users to specify whether to ignore changes in whitespace (spaces, tabs, and newlines) or to treat it as any other non-alphanumeric character (i.e., to parse each whitespace character into its own token). In both systems the goal is to avoid the introduction of spurious and semantically meaningless entries in the difference report. For numerical values, *spiff* goes one step further and allows numerical differences within a threshold "epsilon" to be ignored. For text values, however, there is a question of how to measure differences. To address this issue, we have extended the *spiff* paradigm to text by introducing a heuristic for measuring differences. The user can instruct the program to ignore differences that are smaller than a specified grain size, with the choices for grain size being word, phrase, sentence, or paragraph.

[2]Users have been members of the PREP Editor project group, their collaborators, and first-year students in a selected section of a college writing course.

[3] This observation is consistent with the work of Gibson who observes that events--such as the notification of a change--have "affordances," that is, they demand or invite certain behaviors.

[4] With time and practice, frequent readers of change reports may become skilled at "decoding" them and be able to process them at the same time as reading for meaning and representing remaining problems; however, such a development is not inevitable. For example, it is generally **not** possible for proofreaders to become so skilled at finding errors that they are able to build meaning at the same time as they proofread (cf. [6]).

Differences are measured by the following heuristic: differences between two units are ignored if some percentage of their parts are equal. Setting the value to anything except 100% places the burden of difference detection on the reader.[5] For example, a co-author interested in not losing any "hard won" sentences might specify a grain size of a sentence, and a 70% change threshold (Figure 2.c). While we have not had any experience with lawyers using *flexible diff*, the appropriate setting for examining revisions of contracts might be 100% (Figure 1.c.). To complement our notion of "change threshold," we are experimenting with notations for showing relative amounts of differences in the interface. In other words, if some of the actual differences are suppressed from the report via the change threshold, the interface might still (through change bars, color or shading) indicate that there are changes to be explored. Paramount to the effectiveness of such a visual notation is its unobtrusiveness--it must *not* draw undue attention to the suppressed changes, or it will subvert the purpose of the change threshold completely.

Another aspect of the question *"What changes should be reported?"* concerns the type of events that should be reported. In particular, a "move" can be defined as a deletion followed by an insertion of the same material elsewhere. Our initial experience with difference reports, however, suggests that readers appreciate a move being signalled explicitly, because it reduces their search task. We developed a simple technique that searches for insertions and deletions of identical text and remaps them to a move operation. The user can control a parameter, "minimum length for a move," in units of the current grain size, that controls the minimum length of an insertion/deletion before it is interpreted as a move, on the grounds that multiple occurrences of short strings such as "the" and so forth should not be interpreted as moves. Based on our initial experience, it appears to be useful to allow larger units of text, such as sentences and phrases, to be treated as moves even though they have some changes within them, but this is not yet implemented.

How Should Changes That Are Reported Be Pinpointed?

Our experience with differencing tools for texts suggests that several factors--the number, density, and complexity of changes--should condition how changes in a text are pinpointed. Above a certain threshold of change intensity, writers seem to prefer a report that shows the entire region of text as changed rather than one that pinpoints each change piecemeal. To illustrate, Figure 3.a. depicts an original text and Figure 3.b., a revision. Figure 3.c. is a change report with the changes pinpointed at the word level; Figure 3.d. pinpoints the changes by concatenating replacements that are closer together than a user-controlled number of words away. Notice that the change report in Figure 3.d. is no longer strictly accurate: The word "brown" is reported as having been deleted and inserted, even though it was constant across the two versions. Yet many readers prefer the second version, reporting that it increases the readability.

In response to this observation, *flexible diff* defines several parameters that allow users to control how changes are pinpointed. The first parameter is "coarseness" of pinpointing: character, word, phrase, sentence, or paragraph. For example, if the user sets the coarseness parameter for how the change should be pinpointed to the sentence level (and the grain size parameter for what changes are to be reported to the word level), then an insertion of a word in a sentence will be reported, but it will be reported as the whole sentence having been deleted and inserted.

a. Original text	It was cold. The quick brown fox jumps over the lazy dog. Her bowl is over there, by the car.
b. Revised text	It was morning. It was cold. The slow brown dog jumps over the lazy cat. Her bowl is over there, by the truck.
c. Fine pinpointing	It was morning. It was cold. The ~~quick~~ slow brown ~~fox~~ dog jumps over the lazy ~~dog~~ cat. Her bowl is over there, by the ~~car~~ truck.
d. Medium pinpointing	It was morning. It was cold. The ~~quick brown fox~~ slow brown dog jumps over the lazy ~~dog~~ cat. Her bowl is over there, by the ~~car~~ truck.
e. Coarse pinpointing	It was morning. It was cold. ~~The quick brown fox jumps over the lazy dog.~~ The slow brown dog jumps over the lazy cat. Her bowl is over there, by the ~~car~~ truck.

Figure 3. Pinpointing changes.

In addition to grain size, the user can control how precisely replacements (insertion/deletion pairs) are pinpointed. Three parameters are defined: "maximum distance to look for commonalities," "maximum percent of differences," and "maximum distance to concatenate." The first two parameters control the recursive top-down application of the differencing process which is applied, for example, at the phrase level before it is applied at the word level: If two phrases are very different in length ("maximum distance to look for commonalities") and have a great many differences ("maximum percent of differences"),[6] a replacement is reported at the phrase

[5]It is important to note that this is not to say that a co-author might not notice a change below the sentence level; simply that the change report would not point it out.

[6]The number of differences is computed by the first phase of the differencing algorithm. Our experiences are that the first phase of the algorithm (which yields the number of changes but not their exact values) generally executes in less than one-half the time required for a complete comparison.

level on the grounds that it is less worthwhile to examine them for commonalities, many of which may be cryptic and fortuitous (e.g., they may contain function words in common such as "the," "of," etc.); otherwise the differencing algorithm is continued and the replacements are pinpointed more exactly. Figure 3.e. depicts a change report that results when the parameters are set so that the algorithm does not recurse to the word level for the phrases "The quick brown fox jumps over the lazy dog" and "The slow brown dog jumps over the lazy cat."

The third parameter, "maximum distance to concatenate," controls a bottom-up concatenation process. As the list of differences (or edit script) is being built by the differencing process, it is examined for replacements that occur within the range of "maximum distance to concatenate" on the grounds that concatenating replacements that are adjacent or relatively close together increases the readability of the report. Figure 3.d. depicts a change report that results when the "maximum distance to concatenate" parameter is set so that "~~quick~~ slow brown ~~fox~~ dog" (cf. Figure 3.c.) is concatenated to "~~quick brown fox~~ slow brown dog." Depending on how the user sets the parameters controlling the top-down and bottom-up processes, they will produce identical or different results for any given example.

Incidentally, the parameters described in this section allow users to trade-off accuracy of pinpointing changes and execution speed because the speed of the differencing algorithm is proportional to the number of input tokens and to the number of differences between the two versions. Thus, users have an additional motivation to select parameters that are appropriate to their cognitive and social needs. In particular, users are motivated to initially generate an initial coarse description of changes (which is faster) and then to look selectively at the details of particular changes (which is more accurate). Our implementation actually uses heuristics like this to improve overall performance.

What Should The User Interface To The Change Report Be Like?

We have implemented an interface for *flexible diff* in the PREP Editor, a writing environment that supports side-by-side columns of text, with horizontal alignment that enables "at-a-glance" viewing of large numbers of annotations and related texts. The comparison interface produces its report in a new column, with the differences linked to the original column for easy, side-by-side evaluation. To illustrate, Figure 4 depicts four columns: an original draft, its revision, the comparison report and an evaluation column. The evaluation column in Figure 4 consists of annotations to the comparison report that a co-author produced in order to explain some of the changes or to solicit advice about them.

This interface has two important features. The first is horizontal alignment of the changes to the point of difference in the text. There is evidence that readers accustomed to horizontal writing read faster in the horizontal direction than in the vertical (*cf.* [17], p. 137). Thus, side-by-side, horizontal alignment of the changes to original and revised documents is consonant with what we know about the cognitive processing demands of the task of seeing what changed. The following excerpt from a "think-aloud" protocol of a reviewer looking at an author's original text, the reviewer's comments on that text, the author's revised text, and a change report is typical of those we have observed when changes are aligned horizontally and illustrates that users are able to understand changes quickly:

> *...the object-oriented developers encountered fewer of the classic and costly surprises.* So he didn't actually change anything really here. Well all right--he changed a little, but it wasn't exactly what I wanted him to do.[7]

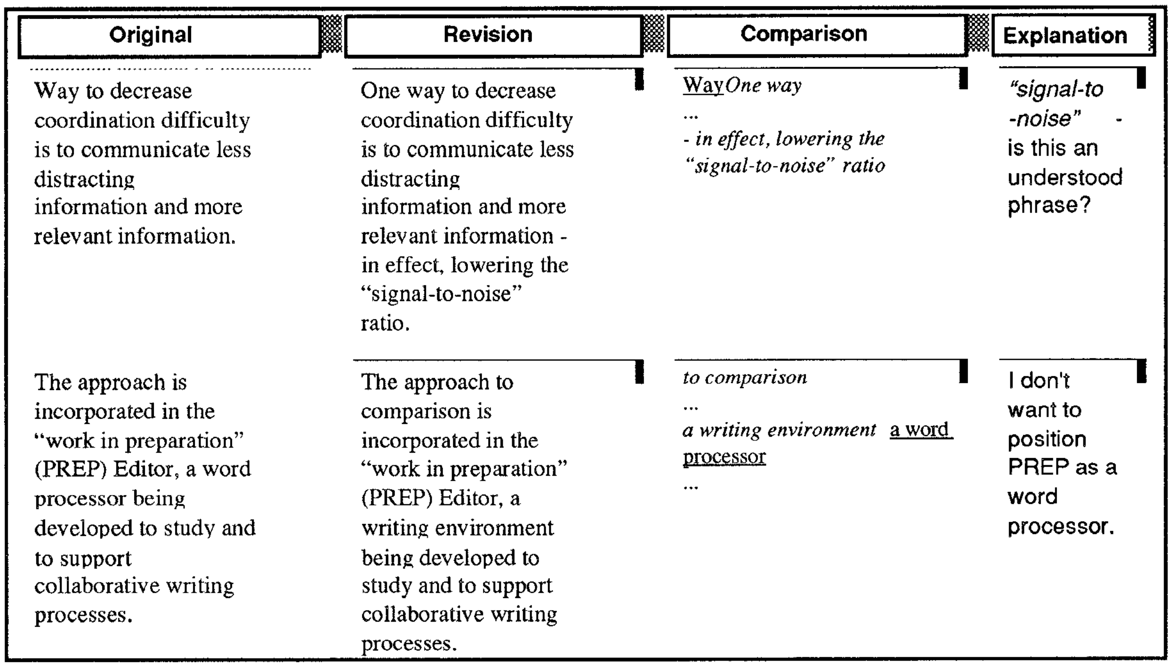
Because the comparison exactly links the changes to the source text, it is possible to provide a mechanism to accept or reject each change easily (as in InterNote [1]), though we have not yet implemented this.

The second important feature of the interface is the ability of users to annotate changes with explanations of the change or questions to co-authors. Our group's experience with this feature suggests that reviewers will annotate changes selectively in order to draw their co-authors' attention to changes they want to discuss or explain. Likewise, a co-author can ask a reviewer to explain a change. Our hypothesis is that the ability to annotate changes will greatly alleviate writers' frustrations with unexplained changes that Cross [3] observed.

The column layout just described requires a significant amount of screen real estate. In order to be usable with smaller screens, PREP provides flexibility in the comparison interface by allowing the user to display the text with the changes interspersed between the unchanged text (a common format used in commercial differencing tools). In this format, users can also annotate the changes.

Besides the interface to the change report, there is an interface to the parameters (described above) that control how the differencing process operates. While the parameters we have described allow enormous flexibility in our comparison tool, we also recognize that most users will not care to change the defaults. We chose the default values after testing the algorithm against widely varying input texts, including texts jointly written among ourselves in PREP (this paper was co-authored in PREP).

[7] In a "think-aloud" protocol, subjects are asked to think-aloud--to say whatever it is they are thinking--and are video-taped or audio-taped. In the transcription italics indicate that the subject is reading, and each "-" indicates a pause of 1 second duration. The author and reviewer, who were subjects in an experiment evaluating aspects of the PREP Editor, were working on an actual paper intended for publication.



Original	Revision	Comparison	Explanation
Way to decrease coordination difficulty is to communicate less distracting information and more relevant information.	One way to decrease coordination difficulty is to communicate less distracting information and more relevant information - in effect, lowering the "signal-to-noise" ratio.	Way*One way* ... *- in effect, lowering the "signal-to-noise" ratio*	"*signal-to -noise*" - is this an understood phrase?
The approach is incorporated in the "work in preparation" (PREP) Editor, a word processor being developed to study and to support collaborative writing processes.	The approach to comparison is incorporated in the "work in preparation" (PREP) Editor, a writing environment being developed to study and to support collaborative writing processes.	*to comparison* ... *a writing environment* a word processor ...	I don't want to position PREP as a word processor.

Figure 4. Actual PREP Editor column interface showing *flexible diff.* Note that the current implementation of PREP shows insertions as *italic* text, and deletions as underlined text.

It is essential to note that we do not claim to have discovered the mappings between our heuristics, the parameter settings and output that is optimal for a reader trying to make sense of a report of differences across versions of text. We only claim to have created a tool where these settings and their effect on interpretability can be systematically explored. How the heuristics should be applied (and perhaps the proper defaults based on different task scenarios) is something we regard as important future research.

Currently, comparisons are generated only upon request from the user. We plan to experiment with heuristics for automatically generating comparison reports, depending on role relationships among writers. For example, if the annotated draft is from a co-author, then display changes upon request; if from a reviewer, then display all changes automatically. Apart from generating the comparison before returning the revision (which we, as co-authors, currently sometimes do), the revision's author has little control over how the comparison might be done. As this information might lead to a more productive exchange, we plan to experiment with adding "comparison settings" information to revisions that would serve as hints from the co-author to anyone who would generate a difference report. For example, as a teacher, I might wish to hide/suppress character level changes (e.g., spelling corrections) so that my student's attention would be directed towards the more meaningful, higher-level changes.

OPERATION

Initially, *flexible diff* does a lexical analysis of the input streams based on the settings the user has specified. During this tokenizing process, any white space is distinguished and usually ignored. The resultant streams are then fed into the differencing process for generation of the edit script. Note that even this simple abstraction of separating the actual characters of the input stream from the sequence comparison gives the engine considerable flexibility in terms of generating reports based on different grain sizes[8].

The Differencing Process

The differencing algorithm is modelled after the Myers [11] O(ND) longest common subsequence/shortest edit script algorithm.

In *flexible diff*, the basic Myers algorithm is applied with a hierarchical decomposition strategy that exploits the grain size (character, word, phrase, sentence, paragraph) of comparison reports. For the sake of illustration, assume that the user has set the grain size to the word level (the default setting). The hierarchical application of the Myers algorithm can be described by the following psuedo-code:

```
1.Compare text-A and text-B at the phrase level.
2. For all string replacements in the edit script Do
    If the original has little in common with its replacement
    Then report the whole string as changed
    Else compute & report diffs for the two strings at the
    word level.
```

After building a list of differences at a higher level (Step 1), several factors are examined to determine whether to

[8]In our implementation, a tokenizer is a subclass of a generic "tokenizer" C++ class, so adding different tokenizers is an easy task--ones for sound and graphics would be possible.

diff recursively the corresponding strings at the next level (Step 2). The decision as to whether strings have "a lot" or "a little" in common are controlled by the parameters "maximum distance to look for commonalities" and "maximum percent of differences" described above.

Finally, concatenations of the edit script, controlled by the parameter "maximum distance to concatenate," are done as the edit script is produced. This concatenation reduces co-located edit operations to one larger operation.

Performance

The Myers algorithm performs especially well when differences are small (i.e., sequences are similar) and is consequently fast in an application in which users adjust grain size to fit the stage in the writing process.

RELATED RESEARCH

Several researchers have argued recently that computer-support for cooperative work requires flexibility, that is, the ability to tailor the behavior of collaborative systems both to differences between groups and to differences among group members [4; 5]. We are especially indebted to Dewan and Choudhary's [4] work on flexible coupling. Their thoughts about the collaborative negotiation that might occur and parameters for such negotiation have influenced our entire approach.

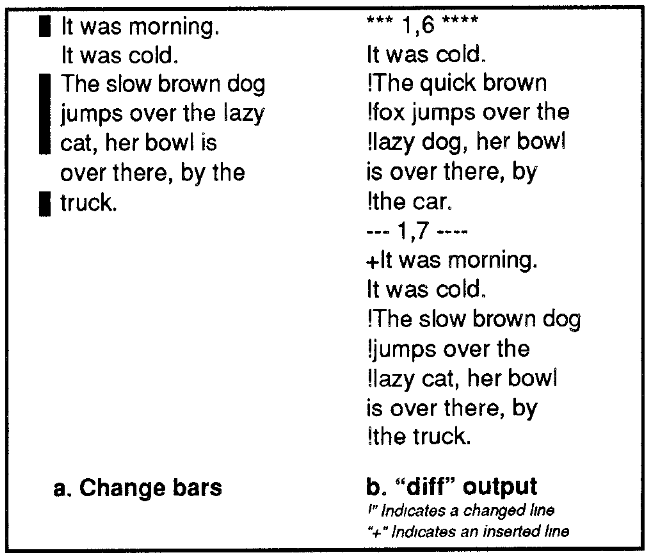
The technical problem of efficient and optimal change detection has been explored in detail by other researchers, most notably UNIX™ *diff* [7] and Miller and Myers [10; 11]. We have not tried to improve on the efficiency and compactness of differencing algorithms themselves, but instead on the readability of the report. Our implementation based on their techniques is able to process realistically sized texts, from 10 to 40 pages and up to 15,000 words, in a user-perceived "reasonable" amount of time.

The notion of flexible diff-ing in *spiff* [12] is closest to our research in its application of domain knowledge to reduce spurious difference reporting. Our work extends this idea by defining parameters we argue are appropriate to the domain of collaborative writing. Some of the settings for the parameters allow users to trade accuracy for efficiency. Others allow for tradeoffs between correctness of the difference report versus cognitive distraction due to unimportant information.

Surprisingly, little work has been done on how best to display difference reports to user. Existing tools pay little attention to the presentation of the comparison information. In the case of *diff*, the output is simply a list of the changed lines with an indication of whether the particular line was replaced, inserted, or deleted (Figure 5.b). Likewise, editors that include "change bars" indicate which lines have changed and do not report the actual site of the revisions (Figure 5.a). Note that while "diff" gives information such as which lines were inserted, deleted, or changed, the change bar interface does not show that detail. With change bars, the location of the change is less precisely pinpointed and the type of change is not specified. Commercial word processors and comparison tools produce output based on word-level differences, such as shown in Figure 3.c. None of these interfaces support commenting on change reports in order to explain changes.

FUTURE WORK AND CONCLUSIONS

The notion of flexible differencing is clearly related to versioning and concurrency control. As techniques for dealing with inconsistencies improve, there may be less need for strict concurrency control, since users could resolve conflicts that have arisen more easily. We have plans to explore the application of *flexible diff* in this context.



```
It was morning.            *** 1,6 ****
It was cold.               It was cold.
The slow brown dog         !The quick brown
jumps over the lazy        !fox jumps over the
cat, her bowl is           !lazy dog, her bowl
over there, by the         is over there, by
truck.                     !the car.
                           --- 1,7 ----
                           +It was morning.
                           It was cold.
                           !The slow brown dog
                           !jumps over the
                           !lazy cat, her bowl
                           is over there, by
                           !the truck.
```

a. Change bars **b. "diff" output**

!" Indicates a changed line
"+" Indicates an inserted line

Figure 5. Typical difference reports in existing systems.

Our approach can be improved in many ways. It will probably be useful to exploit semantic information such as a knowledge of function words (e.g., "and," "the" "of" etc.). Likewise, it might be useful to exploit the semantic and pragmatic knowledge of users themselves. For example, a facility might be useful that would allow a co-author to select a region of text to draw to his or her co-author's attention in a change report, regardless of grain size or change threshold. Also, full support of collaborative writing groups may require supporting difference finding and reporting for objects other than text. Our observations of a group of physicists working collaboratively on a paper suggest that we may need to explore diff-ing pictures.

While we have designed *flexible diff* specifically for PREP and the context of collaborative writing, many aspects of it apply to any collaboration system.

In collaborative writing systems of the future, users will want to automatically recover changes that they made themselves at an earlier time or that are the result of work by another writer. In such a world, good comparison reports of versions will be essential. We have argued a good comparison report depends not only on a tool's efficiency in detecting change but also on the tool's ability to be tailored to different contextual assumptions of change and to a user's cognitive capacity to process information about change in various social contexts. In this paper, we have reported some progress toward the development of such a tool that, we have argued, will support a wide variety of tasks and texts. It remains to be seen which parameters make the biggest difference to which editing contexts. That determination awaits future empirical work. Such a tool, however, allows for a systematic study that varies number, density and complexity of change and could help us discover more about how each factor affects the task definition processes and goals of collaborative writers.

ACKNOWLEDGEMENTS

The work reported here have been supported by the National Science Foundation (grant number IRI-8902891) and by a grant from External Research at Apple Computer, Inc. Special acknowledgement is due to Paul Erion who was chiefly responsible for our implementation of the diff-ing. Thanks to David Banks for consultation on parameters for string similarity measures and to Davida Charney for comments on an earlier draft of this paper.

REFERENCES

1. Catlin, T., Bush, P., and Yankelovich, N. InterNote: Extending a hypermedia framework to support annotative collaboration. In *Hypertext'89 Proceedings* (Pittsburgh, PA, Nov. 5-9). ACM, N.Y., 1989, pp. 365-378.

2. Cavalier, T., Chandhok, R., Morris, J., Kaufer, D., and Neuwirth, C.M. A visual design for collaborative work: Columns for commenting and annotation. In *Proceedings of the Twenty-fourth Hawaii International Conference on System Sciences (HICSS-24)* (Kalua Kona, Hawaii, Jan. 8-11), J.F. Nunamaker, Jr., Ed. IEEE Press, Washington, D.C., 1991, pp. 729-738.

3. Cross, G. A. A Bakhtinian exploration of factors affecting the collaborative writing of an executive letter of an annual report. *Research in the Teaching of English 24*, 2 (May, 1990), 173-203.

4. Dewan, P., and Choudhary, R. Flexible user interface coupling in a collaborative system. In *Proceedings of the CHI'91 Conference* (New Orleans, April 27-May 2). ACM, N.Y., 1991, pp. 41-49.

5. Greenberg, S. Personalizable groupware: Accommodating individual roles and group differences. In *Proceedings of the Second European Conference on Computer-Supported Cooperative Work* (Amsterdam, The Netherlands, Sep. 25-27), L. Bannon, M. Robinson, and K. Schmidt, Eds. Computer Sciences Company, Slough, UK, 1991, pp. 17-31.

6. Haber, R.N., and Schindler, R.M. Error in proofreading: Evidence of syntactic control of letter processing? *Journal of Experimental Psychology: Human Perception and Performance 7*, (1981), 573-579.

7. Hunt, J.W., and McIlroy, M.D. *An Algorithm for Differential File Comparison*, Bell Laboratories, N.J., Computing Science Technical Report No. 41, 1975.

8. Kahneman, D. *Attention and Effort.* Prentice-Hall, Englewood Cliffs, N.J., 1973.

9. Lesser, V.R., and Erman, L.D. Distributed interpretation: A model and experiment. *IEEE Transactions on Computers C-29 12* (Dec., 1980), 1144-1163.

10. Miller, W., and Myers, E.W. A file comparison program. *Software-Practice and Experience 15*, 11 (1985), 1025-1040.

11. Myers, E.W. An O(ND) difference algorithm and its variations. *Algorithmica 1* (1986), 251-266.

12. Nachbar, D. Spiff--A Program for Making Controlled Approximate Comparisons of Files. In *Proceedings of the Summer 1988 USENIX Conference* (San Francisco, CA, June 21-24). USENIX Association, Berkeley, CA, 1988, pp. 73-84.

13. Neuwirth, C.M., and Kaufer, D.S. The role of external representations in the writing process: Implications for the design of hypertext-based writing tools. In *Hypertext '89 Proceedings* (Pittsburgh, PA, Nov. 5-8). ACM, N.Y., 1989, pp. 319-342.

14. Neuwirth, C.M., Kaufer, D.S., Chandhok, R., Morris, J.H. Issues in the design of computer support for co-authoring and commenting. In *Proceedings of CSCW'90 Conference on Collaborative Work* (Los Angeles, CA, Oct. 7-10). ACM, N.Y.,1990, pp. 183-195.

15. Samuels, S.J., and Kamil, M.L. Models of the reading process. In *Handbook of Reading Research*, D. P. Pearson, Ed. Longman Inc., N.Y., 1984, pp. 185-224.

17. Taylor, I., and Taylor, M. M.*The Psychology of Reading*. Academic Press, N.Y., 1983.