Deep Diffs: Visually Exploring the History of a Document

Ross Shannon
Systems Research Group,
School of Computer Science and Informatics,
UCD Dublin, Ireland
ross.shannon@ucd.ie

Aaron Quigley
HITLab Australia,
School of Computing and Information Systems,
University of Tasmania, Australia
aaron.quigley@utas.edu.au

Paddy Nixon
Systems Research Group,
School of Computer Science and Informatics,
UCD Dublin, Ireland
paddy.nixon@ucd.ie

ABSTRACT

Software tools are used to compare multiple versions of a textual document to help a reader understand the evolution of that document over time. These tools generally support the comparison of only two versions of a document, requiring multiple comparisons to be made to derive a full history of the document across multiple versions. We present Deep Diffs, a novel visualisation technique that exposes the multiple layers of history of a document at once, directly in the text, highlighting areas that have changed over multiple successive versions, and drawing attention to passages that are new, potentially unpolished or contentious. These composite views facilitate the writing and editing process by assisting memory and encouraging the analysis of collaboratively-authored documents. We describe how this technique effectively supports common text editing tasks and heightens participants' understanding of the process in collaborative editing scenarios like wiki editing and paper writing.

Categories and Subject Descriptors

H.5.3 [**Information Interfaces And Presentation (e.g., HCI)**]: Group and Organization Interfaces—*asynchronous interaction, computer-supported cooperative work, web-based interaction*.

General Terms

Design, Human Factors.

Keywords

Text visualization, temporal visualization, wiki, versions, collaborative writing, editing, diffing, undo, history, Wikipedia.

1. INTRODUCTION

In collaborative editing scenarios like academic paper writing, software development, or online wiki-based composition like Wikipedia, editors frequently want to track changes made by other collaborators. Wikis and version control systems like Subversion store each version of a document as it is edited, allowing any editor to look back through the full evolution of the document.

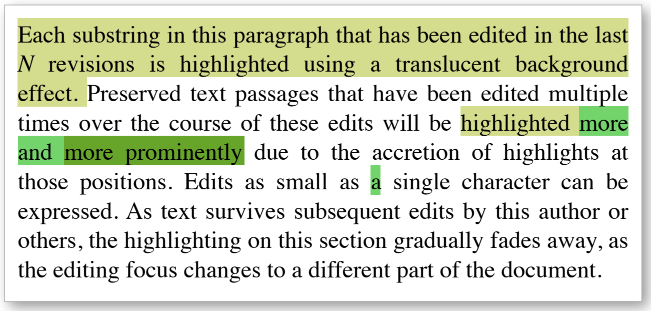
Tools exist for generating views of the difference between two text files. "Diffing" is the process of comparing two versions of a document and annotating the changes in a separate textual "diff" [5]. The standard way of presenting this comparison is to display both versions side-by-side and colour-code sections of text that has been deleted in red, and text that has been added since the last checkpoint in green.

Many editing tools and collaborative programming environments include this way of isolating the differences between your copy of a file and another version. See, for example, the Eclipse IDE. To access this information, the user must open history views or use external tools. While editing the file, there is often no contextual information of this sort available. Some rich text editors support a "track changes" feature which accumulates edits annotations alongside the text. This makes past editing operations explicit in the interface, and presents details about each edit in a GUI overlay until an author accepts them and dismisses the overlay elements.

Other specialised visualisation tools display an overview of the text using a heat map display of a document, giving metadata on a per-line basis [1]. These tools operate on a high level of abstraction, making them more useful for analysis rather than for supporting collaboration. To effectively support the writing and editing process, we can go down to the character level, and expose these levels of history directly in the composition interface.

2. DIFFING DEEPER

To take full advantage of the rich temporal information available, we will compute and simultaneously show the result of diffing multiple versions of a document together in a single view. We use the most recent version as a lens to look back through time, encoding history directly in the text. The following paragraph, which is presented as Figure 1, demonstrates the technique, with passages of text marked up visually with their historical context.



Each substring in this paragraph that has been edited in the last *N* revisions is highlighted using a translucent background effect. Preserved text passages that have been edited multiple times over the course of these edits will be highlighted more and more prominently due to the accretion of highlights at those positions. Edits as small as a single character can be expressed. As text survives subsequent edits by this author or others, the highlighting on this section gradually fades away, as the editing focus changes to a different part of the document.

Figure 1: A segment of text that has been processed with the Deep Diffs technique. Highlighted green areas have been edited recently. As a passage is edited more frequently by one or many editors, the highlighting strengthens in that area so that other author's attention is called to this part of the text.

This provides more information in-situ for collaborative writers or coders than the traditional approach, as text that has survived multiple rounds of peer review without being modified will gradually fade into the background, whereas contentious or unfinished passages or blocks will be highlighted until the co-authors are satisfied with them.

2.1. Editing Operations

Text is a simple one-dimensional datatype that can have a small set of fundamental operations applied to it:

- INSERTION—New text is added at some position in the text. All text after this position is moved forward by the length of the inserted text.
- DELETION—Pre-existing text is removed from this version of the document, shortening the text. All text after this point moves up towards the start of the document.
- REPLACEMENT—Text from the previous version of the document is replaced with new text. This operation can be performed by the user selecting the section of text they want to replace and typing over it. The user may see this all as one operation, but computer systems (and indeed diffing algorithms, since they are applied *post-hoc*) treat a replacement as two separate operations: a deletion followed by an insertion.

Figure 2 demonstrates these editing operations on a short passage of text. The Deep Diffs algorithm works by computing the difference between each of the last *N* revisions of the document. It then analyses these collections of modifications, looking for differences between two versions of a document that occur within the bounds of parts of the document that were modified in previous versions, allowing us to layer these annotations on top of each other when they intersect. Changes propagate forwards in time if the text that they affect is preserved. If any text is deleted between two versions, the annotations associated with those parts of the text are no longer shown.

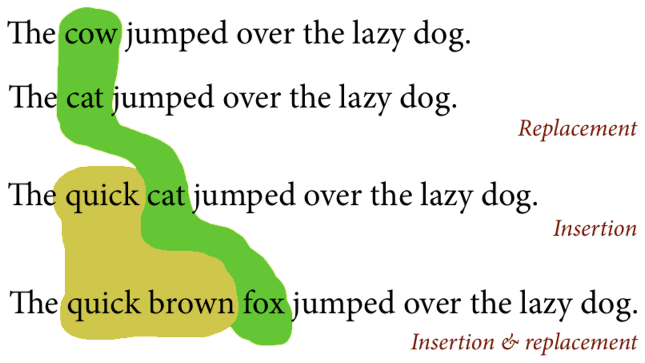

The cow jumped over the lazy dog.
The cat jumped over the lazy dog.
Replacement
The quick cat jumped over the lazy dog.
Insertion
The quick brown fox jumped over the lazy dog.
Insertion & replacement

Figure 2: A sample text undergoing replacement and insertion operations. The highlighted areas track segments of the text that are affected by edits in consecutive versions.

By composing layers of edits into this heat map-style view of editing activity underlaying a standard text view, many layers of history are exposed simultaneously. The view shows which areas of the text have been added most recently, edited most extensively, and those which have received little attention from editors. This visualisation can be incorporated into the normal text editing mode of an editor, not requiring any additional modes or perspectives on the text file to be created.

3. COLLABORATIVE TEXT EDITING

Deep Diffs is a technique designed to be applied *post-hoc* to any text document stored in a version control system, which might be integrated into a wiki, a source-code revision system or a content-management system. The technique works on any two or more versions of an unadorned text document, even if the document only exists as a file on two editors' separate machines. As it is applied in asynchronous editing scenarios, it does not require access to an edit history of a document, unlike the records stored in word processors like Microsoft Word or Apple Pages.

Many documents exist in online repositories where every previous version of the document is also available, such as Wikipedia for encyclopaedic text, and the multitude of programming code stored in online Subversion, CVS or git repositories. We will discuss relevant behaviours in collaborative text editing using Wikipedia as our example, though this approach is also applicable for other sources of text and program code.

3.1. Editing Behaviour on Wikipedia

Wikipedia is a collaborative encyclopaedia, containing over three million articles in the English version alone. Each version of each article is saved, with the diffs between them generated by the wiki system, so a reader can look back all the way to the first incarnation of any article.

Analyses of large-scale collaborative efforts like Wikipedia have found that there are a set of different roles that editors assume. Early reports suggested that there was a relatively small core set of users that perform the vast amount of editing throughout the encyclopaedia. In surveys conducted in 2005, it was found that 1400 people—which was 2% of the editor cadre at the time—had performed 73.4% of all the edits [4].

What was found in subsequent investigations however was that though this distribution of editing operations was accurate, if we instead look at authorship by counting the number of *characters added* to the article that survive peer review and persist to the current version of the document, it is in fact anonymous, generally unregistered users that arrive and add significant chunks of text to the article. The more frequent editors then encircle this new content, edit it for grammar and spelling, reword sections, add appropriate subheadings and otherwise format it to follow Wikipedia's guidelines and customs.

Thus any content added to an article will generally be followed by many smaller edits which improve the quality of the text. As Wikipedia's growth rate slows down over time [7], incremental improvements and refinement of newly-added text is a key task.

Tools have been developed for analysing editing activity on Wikipedia. For example, the work of Viégas et al. on visualising edit histories using history flow visualisation timelines can be used to show the authorship and age of each sentence in a Wikipedia article [2]. In this diffing scheme, text that replaces previous text assigns ownership of this complete sentence to the new editor, and editing a single character transfers ownership of an entire sentence. Tools like this offer rich views of trends within the document over time at a high level, but to support editorial decisions on the text itself will often require resolution at the word and character level.

Techniques like Deep Diffs, which build up indicators of activity over time, are more appropriate for non-adversarial editing scenarios—that is, situations where all editors are working together in the same direction, with little threat of malice or vandalism. Team-based editing tasks like paper writing and software engineering will fit this description.

Wikipedia article histories have an artefact endemic to its open design known as *reverting*. When a malicious or inexperienced editor makes a change to the article that is to its obvious detriment, such as adding nonsense text or deleting parts of the document recklessly, other editors will quickly revert those damaging edits and restore the article to its previous state. Malicious edits disrupt the history of the document with layers of noise, and are not interesting in the analysis of how text is edited and refined. When applying our technique to Wikipedia articles, it is necessary to pre-process the versions under study to remove vandalism and other deleterious edits. This can be done by analysing the edit history and excising edits marked as reverted.

3.2. Collaborative Editing Tasks

Wikipedia articles are "living documents", and are continually edited for clarity, accuracy and presentation. When editing a document collaboratively, there are common tasks that an editor will perform:

- ASSESS—to gain an overview of the amount and variety of editing activity in the document's recent history when reading a new article for the first time. Tools that support this task would enable a reader to asses the trustworthiness [9], stability and maturity of the text. Deep Diffs views present all modifications over a configurable time frame simultaneously. Viewers can see what parts of the document are new (large green areas), mature (text annotated with further deeper highlights indicating refinement), and stable (where annotations have faded away).

- MONITOR—keep track of recently-changed sections of an article of interest. This is particularly important in a collaborative editing environment when the user hasn't seen the new text that has been added by another editor. When new text is added (either through an insertion or a replacement), an editor may want to delete it, amend it for clarity by fixing spelling, grammar or formatting, or add additional content alongside it.

  We assume that text that has been added to a document recently and has not been seen by more than one editor is more likely to contain errors or omissions than text has been preserved over multiple edits. For this reason, the Deep Diffs technique highlights these new additions, and gradually fades old annotations as further edits are made.

- QUANTIFY CHANGE—to understand what proportion of a document has changed since the previous version that the editor saw. This will inform their understanding of how much text they need to review—whether most edits are small changes such as words being replaced, or whether there are significant new passages of text that need to be copy-edited.

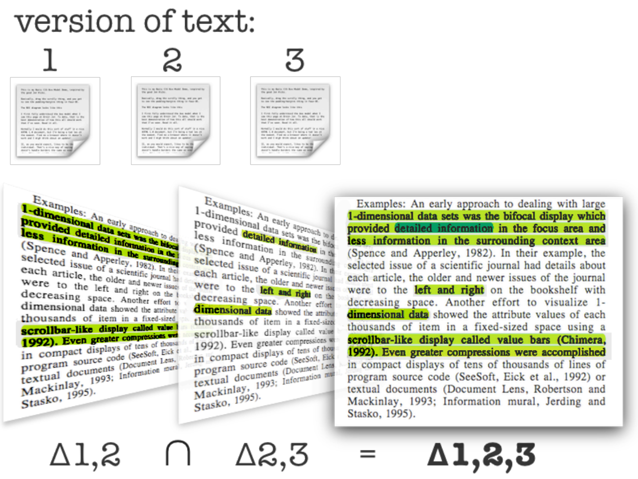
  Previous work has used the metaphor of "computational wear" to indicate where documents had been read and edited [8], with compact visualisations being drawn on top of scrollbars to indicate accrual of activity on the corresponding line. Resolution at the level of individual characters rather than lines or sentences is needed to effectively support this task.

  Traditional diff views remove much of the context of an edit, stripping away the surrounding text to a few lines. As Deep Diff annotations are added in-situ, users are given better location awareness of where in the document the edits are performed.

4. METHOD

The Deep Diff algorithm has been implemented in JavaScript for deployment on the client-side in a web browser. It works by taking any two or more versions of a text, and first generating a chain of diffs between each two versions in order. So a diff is computed between versions 1 and 2, and between 2 and 3, and so on. Each diff is a collection of various *insertion* and *deletion* records (remembering that a *replacement* is simply a deletion followed by an insertion at the same index in the document). The diffs between versions are generated by the diff-match-patch libraries [6] (the same set of algorithms used in Google Documents). These libraries provide character-level diffs for high-resolution comparisons.

The final view is then constructed by proceeding through the records in each diff from oldest to newest, and adding or amending annotations for each modification. Figure 3 shows the combination of multiple diffs. All insertions in the diff are annotated with *edit markers* at the start and end of each passage of added text (these are the highlighted sections of text in the images). A running tally of all extant edit markers is maintained. Each marker has a numerical start index and an end index which are used to place HTML open and closing tags in the final output.



Examples: An early approach to dealing with large 1-dimensional data sets was the bifocal display which provided detailed information in the focus area and less information in the surrounding context area (Spence and Apperley, 1982). In their example, the selected issue of a scientific journal had details about each article, the older and newer issues of the journal were to the left and right on the bookshelf with decreasing space. Another effort to visualize 1-dimensional data showed the attribute values of each thousands of item in a fixed-sized space using a scrollbar-like display called value bars (Chimera, 1992). Even greater compressions were accomplished in compact displays of tens of thousands of lines of program source code (SeeSoft, Eick et al., 1992) or textual documents (Document Lens, Robertson and Mackinlay, 1993; Information mural, Jerding and Stasko, 1995).

Figure 3: The composition of annotations between versions. Once pairwise diffs (or *deltas*) have been generated for each pair of versions, the corresponding annotations are added. The final diff of all three versions together contains relevant annotations from each constituent diff, as well as the depth information unique to this technique when two diff annotations are found to intersect. (The intensity of the highlighting has been increased to make these images suitable for monochrome printing).

The first layer of annotations between versions 1 and 2 is straightforward. After this first round of annotations, further annotations may require the locations of these existing edit markers to be moved, as the text has gotten longer or shorter in various places. For the remaining deltas, the algorithm takes the diff records in order, and applies them to the collection of edit markers from the previous step by:

- Shifting markers forwards in the text for new insertions that occur earlier in the document;
- Shifting markers backwards in the text if a deletion of some characters occurs earlier in the document;

- Expanding existing annotations by the length of the insertion if it occurs within the boundaries of this existing annotation;
- Contracting existing annotations by the length of a deletion if it occurs within the boundaries of this marker. If this leaves the marker with zero length—that is, the text that this marker originally surrounded is no longer present in the current version of the document in any form—the marker is deleted as this text has been replaced or removed outright.

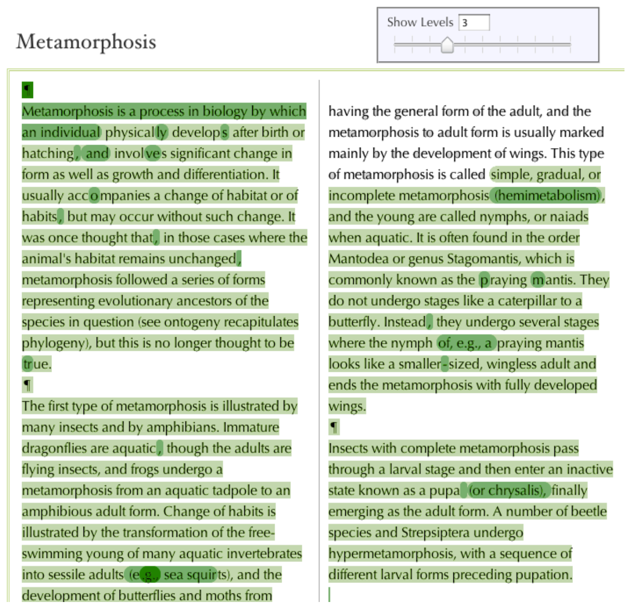
Each of these operations keep the edit markers at the correct positions in the text, so that the highlighting is robust. Once all the existing annotations have been modified as necessary for the current diff, new markers corresponding to each of the insertions in this diff are added to the marker collection. This process is repeated for as many diffs as are being combined.

Finally, HTML elements that are styled to add translucent backgrounds to text that they enclose are merged into the final text. To preserve the integrity of the marker indices into the text, this is achieved by converting the text into a character array and concatenating an opening `<ins>` tag at each marker's start point and a closing `</ins>` tag at each end point.

5. DISCUSSION

Though we have demonstrated this technique on free text in this article, it works just as well for structured content like HTML markup or programming code in any language. The diffing libraries can be instructed to ignore certain tokens, such as HTML elements which take the form `<element></element>`, which allows the technique to be applied not just to the underlying HTML markup, but to live webpages too.

A benefit of this client-side implementation is the opportunity to apply the highlighting dynamically, based on a slider control as seen in Figure 4. The diffs could also be pre-computed on the server side and similarly manipulated with client-side scripting.


Metamorphosis

Show Levels 3

¶
Metamorphosis is a process in biology by which an individual physically develops after birth or hatching, and involves significant change in form as well as growth and differentiation. It usually accompanies a change of habitat or of habits, but may occur without such change. It was once thought that, in those cases where the animal's habitat remains unchanged, metamorphosis followed a series of forms representing evolutionary ancestors of the species in question (see ontogeny recapitulates phylogeny), but this is no longer thought to be true.
¶
The first type of metamorphosis is illustrated by many insects and by amphibians. Immature dragonflies are aquatic, though the adults are flying insects, and frogs undergo a metamorphosis from an aquatic tadpole to an amphibious adult form. Change of habits is illustrated by the transformation of the free-swimming young of many aquatic invertebrates into sessile adults (e.g., sea squirts), and the development of butterflies and moths from having the general form of the adult, and the metamorphosis to adult form is usually marked mainly by the development of wings. This type of metamorphosis is called simple, gradual, or incomplete metamorphosis (hemimetabolism), and the young are called nymphs, or naiads when aquatic. It is often found in the order Mantodea or genus Stagomantis, which is commonly known as the praying mantis. They do not undergo stages like a caterpillar to a butterfly. Instead, they undergo several stages where the nymph of, e.g., a praying mantis looks like a smaller-sized, wingless adult and ends the metamorphosis with fully developed wings.
¶
Insects with complete metamorphosis pass through a larval stage and then enter an inactive state known as a pupa, (or chrysalis), finally emerging as the adult form. A number of beetle species and Strepsiptera undergo hypermetamorphosis, with a sequence of different larval forms preceding pupation.

Figure 4: A view of a complex document (the Wikipedia article on "Metamorphosis") with Deep Diff highlighting enabled. The reader can see up to three levels of the document's history, which can be modified using the slider at the top to expose further layers in real-time.

A downside to encoding history with colour is that it becomes unavailable as an encoding property to distinguish multiple authors, as it is used in other systems [1, 2, 3, 9], because the differently-coloured overlays would clash and become unreadable. Author information is made available in an overlay when the cursor is placed within an annotation.

The views purposefully don't expose passages of text that have been recently deleted. We assume that editors have good intentions and are working together towards a common goal. Records of any deletions are of course still available in the version-control system. Additional logic could be added to the system so that it manages displaced text more intelligently. Currently it sees moved text as unconnected deletion and insertion operations, with no conception of the semantics involved.

If similar techniques are built into a text editor which has access to the entire edit history of a document for one author or many, a different type of analysis is available [3]. Both analyses lead to interesting insights: performing diffs between "committed" versions allows us to explore editorial decisions that have been made, whereas a constant stream of editing operations allow us to explore the thought process of the author as they compose text.

6. CONCLUSIONS

We have presented Deep Diffs, a technique to support editing and writing tasks in asynchronous collaborative applications. This lightweight visualisation technique provides contextual information about the edit history of a document directly in the text, visually identifying text that has been newly-added to the document, and showing where other editor's corrections and refinements have been performed.

This technique supports common collaborative writing tasks like monitoring changes and analysing a document's history for insight into its composition. Highlighting is applied at the resolution of single characters, and at multiple levels, simultaneously showing modifications that occurred in separate versions, and distinguishing rewrites and refinements from new content.

7. REFERENCES

[1] Eick, S., Steffen, J., & Sumner Jr., E. (1992). Seesoft — A Tool for Visualizing Line Oriented Software Statistics. IEEE Transactions on Software Engineering, **18**(11), pp. 957–968.

[2] Viégas, F. B., Wattenberg, M., & Dave, K. (2004). Studying cooperation and conflict between authors with history flow visualizations. In Proc. CHI 2004 (pp. 575–582).

[3] Thimbleby, W. (2009). Drawing from Calculators, Swansea University. MSc thesis. See appendix section on "Recdit".

[4] Swartz, A. (2006). Who Writes Wikipedia? http://www.aaronsw.com/weblog/whowriteswikipedia

[5] Heckel, P. (1978). A technique for isolating differences between files. Commun. ACM, **21**(4), 264–268.

[6] Fraser, N. Diff Match and Patch libraries. Available at http://code.google.com/p/google-diff-match-patch/

[7] Suh, B.; Convertino, G.; Chi, E. H.; Pirolli, P. L. (2009) The singularity is not near: slowing growth of Wikipedia. In Proc. WikiSym 2009.

[8] Hill, W., Hollan, J., Wroblewski, D., & McCandless, T. (1992). Edit wear and read wear. In Proc. CHI 1992.

[9] Adler, B.T., and de Alfaro, L. (2007) A Content-Driven Reputation System for the Wikipedia. In Proc. WWW 2007.