rv you're dumb: Identifying Discarded Work in Wiki Article History

Michael D. Ekstrand
University of Minnesota
ekstrand@cs.umn.edu

John T. Riedl
University of Minnesota
riedl@cs.umn.edu

ABSTRACT

Wiki systems typically display article history as a linear sequence of revisions in chronological order. This representation hides deeper relationships among the revisions, such as which earlier revision provided most of the content for a later revision, or when a revision effectively reverses the changes made by a prior revision. These relationships are valuable in understanding what happened between editors in conflict over article content. We present methods for detecting when a revision discards the work of one or more other revisions, a means of visualizing these relationships in-line with existing history views, and a computational method for detecting discarded work. We show through a series of examples that these tools can aid mediators of wiki content disputes by making salient the structure of the ongoing conflict. Further, the computational tools provide a means of determining whether or not a revision has been accepted by the community of editors surrounding the article.

Keywords

wiki, visualization, article history, Wikipedia

1. INTRODUCTION

Wikis facilitate the collaborative development of web content by allowing open editing of the content. Wiki implementations typically maintain a history of all edits and make this history available to readers of the site. In the case of many wiki engines, including Mediawiki, a widely used wiki engine that powers Wikipedia, this history is displayed as a list of revisions in reverse chronological order with dates, editors, and edit summaries (see Figure 1).

History is used in a variety of ways in wiki communities. Wikipedia, with its large and diverse editing community, is subject to a substantial amount of conflict about how articles should be written. Wikipedia editors use history views to understand how an article has developed and how this conflict has played out in the article's history. In extreme cases, mediation and arbitration committees investigate disputes or behavior problems; the history of articles and their associated talk pages provide evidence regarding what has happened and enable the respective committees or other parties to make informed decisions.


Revision history of Chocolate
From Wikipedia, the free encyclopedia
View logs for this page
Browse history
From year (and earlier): From month (and earlier): all Go
(latest | earliest) View (newer 50) (older 50) (20 | 50 | 100 | 250 | 500)
For any version listed below, click on its date to view it. For more help, see Help:Page history and Help:Edit summary.
External tools: Revision history statistics · Revision history search · Page view statistics
(cur) = difference from current version, (prev) = difference from preceding version,
m = minor edit, → = section edit, ← = automatic edit summary
Compare selected versions
(cur) (prev) 22:16, 14 December 2007 KnowledgeOfSelf (talk | contribs) m (45,967 bytes) (Reverted edits by Sailormooncookie (talk) to last version by ClueBot)
(cur) (prev) 22:15, 14 December 2007 Sailormooncookie (talk | contribs) (45,998 bytes) (→Production)
(cur) (prev) 22:12, 14 December 2007 ClueBot (talk | contribs) (45,967 bytes) (Reverting possible vandalism by Special:Contributions/Sailormooncookie to version by Graham87. False positive? report it. Thanks, User:ClueBot. (121771) (Bot))
(cur) (prev) 22:12, 14 December 2007 Sailormooncookie (talk | contribs) (46,011 bytes) (→Production)
(cur) (prev) 13:11, 13 December 2007 Graham87 (talk | contribs) m (45,967 bytes) (→Production: sp)
(cur) (prev) 17:54, 11 December 2007 Fill (talk | contribs) (45,966 bytes) (→History: wikilink and copy edit)
(cur) (prev) 09:57, 11 December 2007 BorgQueen (talk | contribs) m (45,953 bytes) (Reverted edits by Shanahan88 (talk) to last version by ScAvenger lv)
(cur) (prev) 09:55, 11 December 2007 Shanahan88 (talk | contribs) (46,053 bytes)
(cur) (prev) 19:38, 4 December 2007 ScAvenger lv (talk | contribs) m (45,953 bytes) (lv: interwiki)
(cur) (prev) 08:40, 3 December 2007 Idioma-bot (talk | contribs) m (45,935 bytes) (robot Adding: bat-smg:Čėkuolads)

Figure 1: The Wikipedia article history view.

While the history of an article is a linear sequence of revisions (most wiki software does not facilitate forking and merging of branches in article development), this history does contain latent structure. As editors add, remove, and re-add words, reverse entire edits, and generally revise and restructure the article, new revisions are usually related to prior revisions in some way. Of particular interest to our work is when one change has the effect of reversing the work done by a prior change, having the net impact of discarding work done by the reversed change (and thus rejecting that change as a part of the article and its development). There are a variety of circumstances in which this can happen: vandalistic edits which deface the article in some way are typically removed quite quickly, editors can remove each others' work in the middle of disputes over article content, and good-faith edits perceived to be of low quality are often reversed to maintain article quality.

Detecting whether the work contributed by a revision was accepted or rejected by the community surrounding that article is useful for at least two reasons. First, if this information can be presented clearly, it has the potential to aid administrators, mediators, and others in determining how the flow of editing has played out in an article's history and how editors have been relating to each others' work in a dispute. Second, being able to detect whether a revision was accepted can aid in analysis of past history by determining

WikiSym '09, October 25-27, 2009, Orlando, Florida, U.S.A.

if a particular version of the article was considered by the editors to be representative of what form the article should ultimately take.

The remainder of our paper is structured as follows: after reviewing prior work on history relationship extraction and visualization, we present our initial method computing a tree from an article's history. We next present our visualization and use it to explore how the structuring algorithm behaves when applied to real-world wiki data (a selection of Wikipedia articles). We then describe and evaluate more sophisticated means of building history trees. Finally, we describe an approach to using history trees to computationally detect discarded work in a well-defined manner.

2. RELATED WORK

Sabel used a metric based on edit distance to structure used the revisions of an article into a history tree such that each revision is a child of the prior revision to which it is the most similar [12]. Aside from this paper and another by Alshattnawi et al. [2], we are not aware of much other work on formally building nonlinear structures of wiki article histories.

Viégas et al. developed a visualization of article history, dubbed *history flow*, which shows how content is added, modified, and relocated by various authors over time [14]. Their display exposes various patterns and shows clearly how particular revisions build on prior work in the article. It also exposes the underlying textual changes that are used by Sabel to structure history.

Kittur et al. further explored the relationships embedded in the article history by analyzing how editors related to each others' work [7]. They considered *reverts*, where one or more edits are undone by completely restoring the article to a prior state. By examining what who reverted who's work they were able to create maps depicting various factions in the editing community involved with the article. Brandes and Lerner [4] performed a similar analysis looking at what editors revised other editors' work, based on the order in which editors changed the page. Biuk-Aghai [3] presented a visualization of co-authorship relationships between articles, aiming to depict the relationships between articles based on the degree of co-authorship. Unlike Viégas and Sabel's systems, which consider individual events in an article's history, these analyses are all summaries aggregated over all the events in a time window.

All of these visualizations, however, are external visualizations; they are viewed in a separate program outside the wiki and have not been integrated into existing wiki interfaces. It seems likely that visualizations which can be implemented as extensions of existing interfaces will be better able to be deployed in live settings and provide value to wiki communities; to this end Suh et al. built a "dashboard" which augmented the Wikipedia interface with information about the authors who have contributed and recent activity level of the article [13]. Followup work [6, 10] found that such integrated displays effectively strengthen user perceptions of an article's reliability. As with the user relationship analysis mentioned previously, however, this is also an aggregate summary display. There has also been work on computing trustworthiness of articles [15] and reputation of users [1], with suggestions and later implementations of visualization strategies for this data.

The visualization aspect of our work draws from the techniques used by distributed version control system history tools such as `gitk` [8] to create an event-based visualization which can be integrated with existing wiki interfaces.

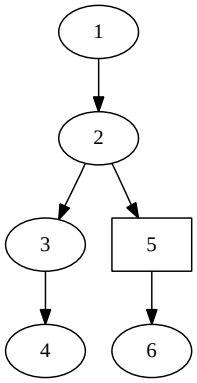

Figure 2: Tree built from revisions in "Chocolate".

3. BUILDING HISTORY TREES

In order to display and analyze revision relationships, we first structure the history of an article into a tree (called an *article history tree*). Using the first revision of the article as the root node, we add the rest of the revisions such that each revision is a child of the previous revision with one exception: if a revision is identical to some prior revision, we make it a child of that revision. If multiple prior revisions have the same text, the most recent one is used. This yields a deep binary tree with reverted-to revisions being the only nodes with two children.

Figure 2 shows an example of an article history tree, taken from a portion of the history of the article "Chocolate"[1]. After revision 1, three edits were made yielding revisions 2, 3, and 4. Revision 5 is a revert back to revision 2, and editing proceeds from that state to make revision 6.

As Kittur et al. did in [7], we detect reverts by computing the MD5 checksum of the UTF-8 encoding of each revision's text and comparing these checksums. Reducing text to a checksum enables prior history to be searched quickly with low space requirements as only 20-byte strings need to be stored and compared, and it is unlikely that two different revisions of the same article will have the same MD5 hash.

The tree can be built efficiently by processing revisions in chronological order and maintaining a hash table keyed with the checksums of the revisions that have been seen. If a wiki system stores revision checksums in its database tables, it can build the revision tree without needing to retrieve the full revision text.

4. VISUALIZING HISTORY TREES

In order for wiki editors to make use of structured history, it is necessary to display the relationships encoded in the tree in some manner. We have developed a history tree viewer to accomplish this, aiming to meet the following design goals:

1. Make it apparent which revisions were accepted as a basis for future development of the article and which were rejected.

[1]http://en.wikipedia.org/wiki/Chocolate

2. Clearly show when a revision restores an article identically to a prior state. This is the same as requirement 1 for revert-based history trees, but treating it separately enables the design to accommodate more general trees where the deep binary tree property no longer holds.

3. Indicate when edits were made by the same editor.

As a further goal, our visualization must integrate with an existing interface for viewing article history. This promotes ease of deployment, lowers barriers to adoption, and positions the tool to be able to directly improve social translucency.

`gitk` [8], a program for reviewing commit history in the `git` distributed version control system, inspired the design of the history tree viewer. It represents revisions with small circles, connecting each revision to its parent revision(s) with colored lines. In this way edits, forks, and merges are all apparent. History tree views have somewhat different requirements, as they do not need to display merges and there are many terminal branches, but the dots and lines alongside a linear log display is retained as the foundation of our visualization.

4.1 Layout

Figure 3 shows our interface embedded in a Wikipedia history view. Our visualization is specifically designed to not only be integrated into existing interfaces but to directly augment the traditional linear history view.

On the left side of the revision list we show the relationships graphically. Each revision is represented by a circle or triangle in that revision's row in the history. A normal edit is drawn as a circle with an arrow going to it from its parent revision. A revert is drawn as a downward-facing triangle connected to the revision it reverts to with a heavy line. This directly fulfills the second design goal; the revert lines also bypass rejected revisions, meeting the first goal.

The revision symbols are laid out in columns such that the most recent revision in the article's history will appear in the far left column. If a revision has more than one child, the additional children are placed in columns to the right.

4.2 Color

In order to satisfy our third design goal, showing when two edits were made by the same editor, we use color to distinguish edits made by different editors. This also enables the viewer to see which editors were involved in a particular set of revisions. There are two methods of mapping editors to colors that we considered.

The first is to construct an editor adjacency graph from the revisions, with editors as vertices and an edge between two editors whose revisions appear consecutively in the linear history or whose edits are connected in the history tree. This graph can then be colored. While optimal graph coloring is NP-complete, there are efficient algorithms for producing non-optimal colorings. A non-optimal coloring is also preferred for this case, as more distinct colors in use permit greater distinction between users. Graph coloring still has the disadvantage, however, of permitting two non-adjacent editors to have the same color, diminishing the ability of the user to quickly recognize edits by different editors.

The other method, which we use for our interface, is to assign distinct colors to distinct editors. This has the problem of requiring many colors, but the number required can be diminished by only assigning colors to active editors. We display all edits by anonymous editors in gray and all edits by editors making fewer than five edits in the article's lifetime in black. Each editor with at least five edits in the page history is assigned a distinct color. Figure 4, taken by computing the maximum number of colors required for any article in the main namespace in the January, 2008 Wikipedia dump, shows that this threshold allows us to distinctly identify a many users while avoiding the most substantial explosions in the number of colors required.

In the current implementation, editors are sometimes assigned very similar colors. It should be possible to combine the algorithms, using graph coloring to ensure that colors editors working in close relational proximity to each other have sufficiently different hues to permit easy distinction, but we have not yet attempted this.

To indicate who originally created a particular state of the text, the line connecting a revert to its parent is drawn in the color of the edit that originally created the text. This enables readers to not only see who first crafted a revision but who has reverted the article back to that state.

4.3 Interaction

Existing Mediawiki history views support some basic interactions to explore the history of an article. There are navigation links to change the number of entries per page and travel backwards and forwards in the article's life and support for comparing a revision with its parent, the current revision, or an arbitrary revision. Arbitrary revision comparisons are supported via two columns of radio buttons used to select the revisions to compare.

Our interface removes the radio buttons in favor of interactions supported by the graphical display which support the same tasks. Clicking on a revision's symbol brings up a menu (shown in Figure 3) that provides access to a diff against the previous revision and enables the revision to be "marked". Marking a revision causes it to be visibly indicated with a dotted circle and enables a further option in the menus for other revisions: comparing with the marked revision. This allows the user to compare any two revisions, replicating the functionality of the radio buttons in the original interface.

We also augment the navigation controls with links to go forward and backward by half the revisions-per-page count, overlapping with the current display. In cases where a revi-

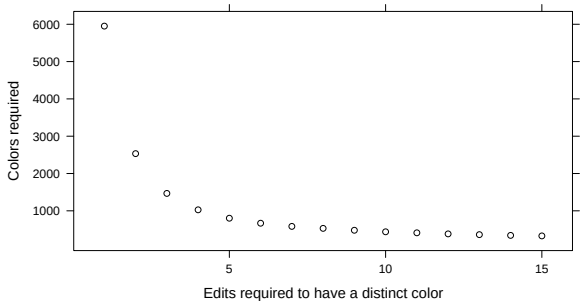

Figure 4: Maximum colors required for various user activity thresholds

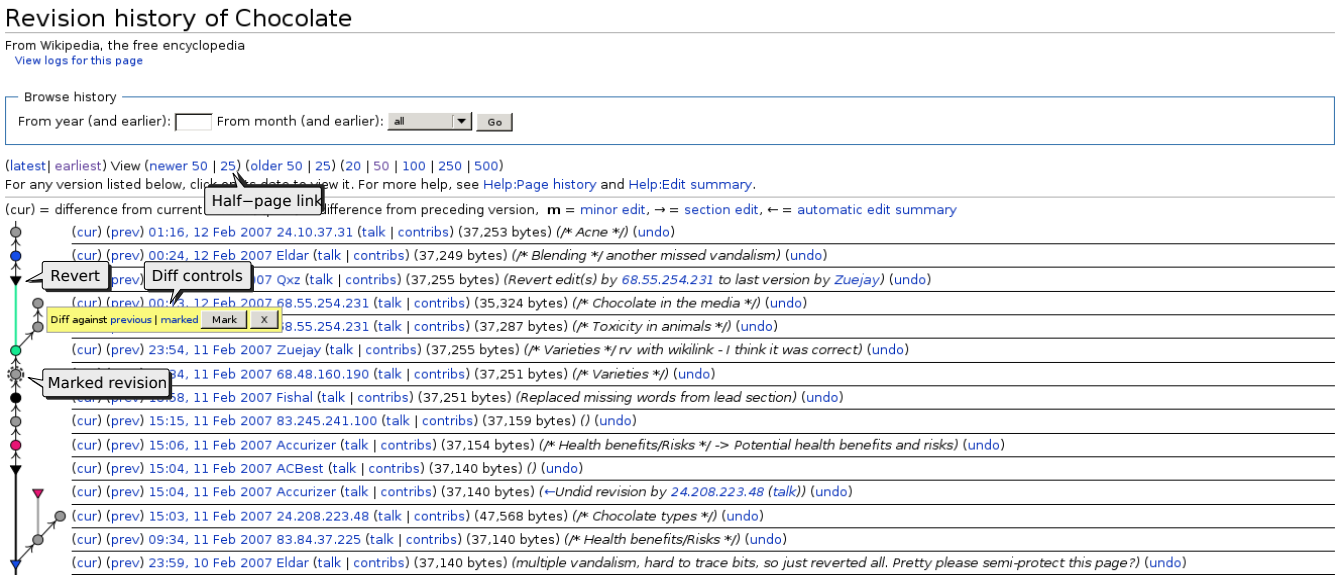

Figure 3: History view with a tree visualization. The top revert is revision 5 in Figure 2.

sion is connected to another revision not currently displayed, the user can use this link to view both revisions on one page.

4.4 Implementation

We implemented the visualization interface in JavaScript with jQuery for the Firefox web browser, using SVG to render the graphical display. For our prototype implementation, we use a server program written in OCaml to compute the revision tree and do the layout computations, providing this information to the JavaScript interface in response to an AJAX call. The visualization is embedded in a template page built from the standard Mediawiki revision history view; client-side JavaScript rewrites it to include the display after layout data is received from the server. Because of this design, the visualization can be easily implemented as an extension to existing wiki software (e.g. a Wikipedia gadget).

5. CASE STUDIES

To demonstrate the utility of our visualization, we present several case studies from Wikipedia articles that show how the history tree diagram facilitates understanding of article development. We selected the first two of these examples to demonstrate how our tool presents events in articles which have been considered in previous work on understanding revision history. They remaining examples were selected to demonstrate how our visualization displays various phenomena occurring elsewhere.

5.1 Chocolate

Viégas et al. highlighted a revert war that occurred early in the history of the "Chocolate" [14]. The dispute was over whether a short paragraph mentioning chocolate's rare use in surrealistic art should be included in the article. The conflict started when an anonymous editor removed this paragraph from the article. Another editor then reverted the article to re-insert the paragraph. This was followed by several repetitions, as the paragraph was repeatedly removed and re-inserted five times.

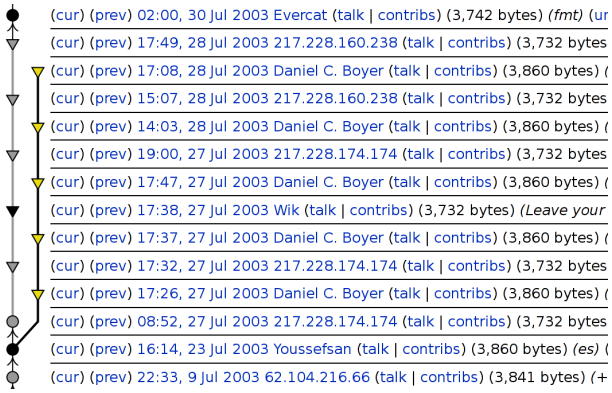

Figure 5: Revert war early in the history of "Chocolate".

The history flow visualization depicted this conflict clearly as a zig-zag pattern in the article's length. The history tree view, shown in Figure 5, shows this same event as parallel branches with alternating revert emblems. These indicate that the article oscillated between two states while the editors reverted each other. The visual language used is different from that in the history flow, but the fact that the article state oscillated remains apparent. Further, the information is displayed directly in the Wikipedia history view, facilitating easy access by administrators or other interested editors.

In this example, all three of our design goals are manifest. Our visualization shows the right sequence of edits discarded. The facts that these edits are all reverts back to a prior state, and they are likewise being reverted, is also made apparent. Finally, the coloring shows that all of the reverts to re-insert the edit were made by the same editor and that most of the removals were made by anonymous editors (the one non-anonymous removal was by a relatively inactive editor).

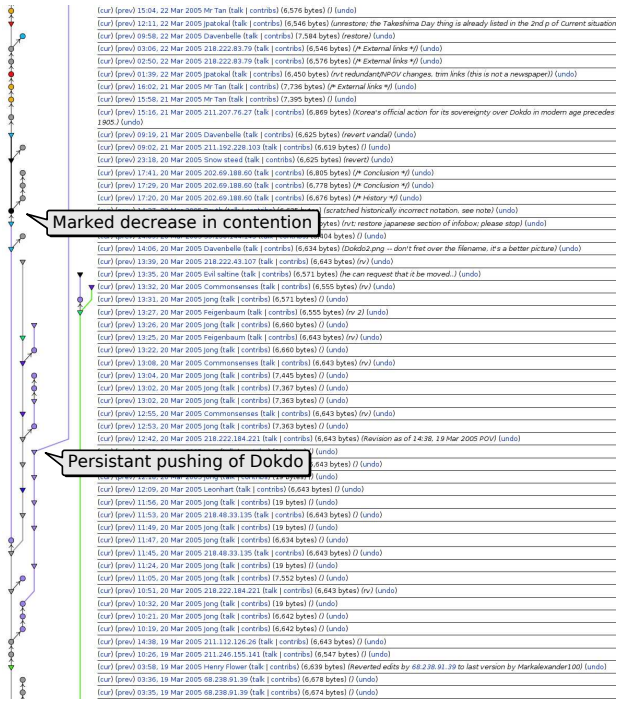

Figure 6: History of Liancourt Rocks in late March 2005.

Revert wars such as this are a common occurrence in Wikipedia articles, and are an easy pattern to identify in history tree views. Colored history trees also facilitate detection of violations of the three-revert rule (3RR), a Wikipedia policy forbidding editors from reverting the same page more than three times in a 24-hour period[2].

5.2 Liancourt Rocks

Kittur's conflict and faction analysis [7] found substantial revert activity and a clearly factioned editor community in the article on "Liancourt Rocks"[3] (then titled "Dokdo"), a mostly uninhabitable rocky island between Japan and Korea. Due to the dispute between the two countries over the ownership of the island, there has been disagreement amongst Wikipedia editors over the article.

The history tree view, shown in Figure 6, makes the significant revert activity evident in the lower half of the view. In the course of the period displayed the page was renamed from "Dokdo" (the Korean name for the island) to "Liancourt Rocks" (a sovereignty-neutral name); there was some contesting of this action, but the fighting settled into relative tranquility in the upper half of the view. There is one revert from a future revision back to the page redirecting to "Dokdo" after this point, but that attempted revert was itself rejected (this can be seen by the fact that it is not in the far left column).

History tree views cannot display editor factions as clearly as Kittur's or Brandes' visualizations [7, 4], but the coloring does show some of these relationships when a few colors

[2]http://en.wikipedia.org/wiki/Wikipedia:3RR
The three-revert rule was not yet in place when the Chocolate event occured.
[3]http://en.wikipedia.org/wiki/Liancourt_Rocks

Figure 7: 2006 Atlantic hurricane season

show up frequently in particular sides of revert battles. In Figure 6, this can be seen as the purple-colored editor (not visible in grayscale) persistently attempts to restore Korean naming to the article.

In general, periods of extensive multi-party or multi-version revert activity will result in wide views with many easily-identifiable revert edges. Colors can then be used to detect users and factions persistently furthering the war.

5.3 2006 Atlantic hurricane season

Figure 7 shows a short period of volatility in the development of the article "2006 Atlantic hurricane season"[4]. In this event, occurring on June 1, 2006, there was a disagreement over whether the 2006 hurricane season started at 00:00 EDT or UTC. Finally one of the editors removed the start time altogether; this was followed by an anonymous editor altering the article to refer to the start of the season in the past tense rather than in future; this change was reverted twice. Finally another editor made a pair of edits to fix formatting, resulting in a prior state of the article being restored. From the edit comments, it seems that the restoration of the old state was not a deliberate undo, but rather that the sequence of edits happened to exactly undo the right branch of the article's development. Again, our tool shows clearly the rejection of the spurious branch of development, and also demonstrates that an edit was reverted, along with its parents, by the same editor who made it. The restoration edit is also displayed as a revert, but it is impossible to distinguish between deliberate reverts and inadvertent returns of the article to a prior state.

5.4 WrestleMania III

Our last example comes from "WrestleMania III"[5]. This page was subject to an edit war listed in Wikipedia's list of so-called "Lamest Edit Wars"[6]. This list describes various disputes that have arisen over article content and were played out through edits to the article rather than through discussion on the article's talk page. The conflict in WrestleMania III that was listed as a Lamest Edit War was primarily over what attendance figure to report for the event: the official figure of 93,173 reported by World Wrestling Entertainment or the figure of 78,000 reported by an independent journalist some editors described as "anti-Vince McMahon".

[4]http://en.wikipedia.org/wiki/2006_Atlantic_hurricane_season
[5]http://en.wikipedia.org/wiki/Wikipedia:WrestleMania_III
[6]http://en.wikipedia.org/wiki/Wikipedia:Lamest_Edit_Wars

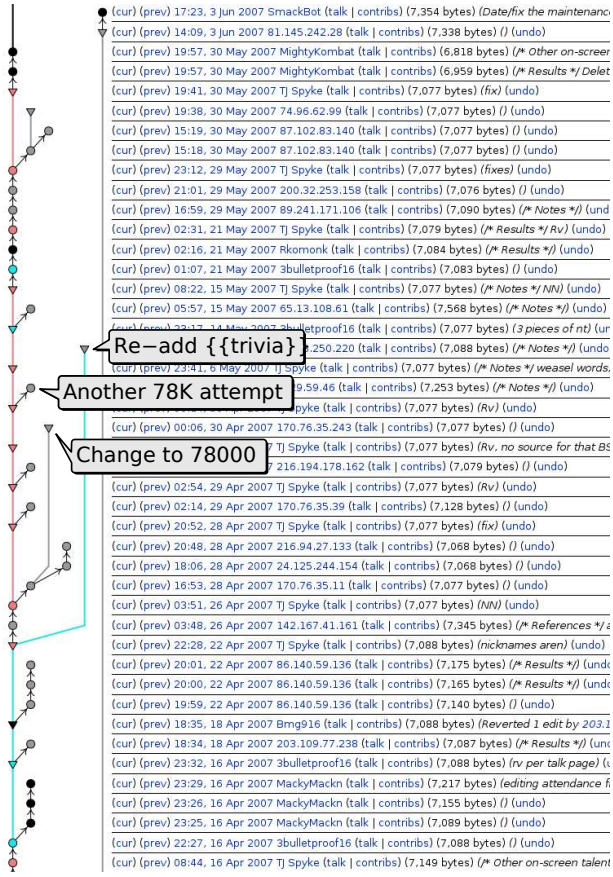

Figure 8: WrestleMania III

In the tree, we can see a variety of attempts to change the number. Some appear as revert wars, but a number are manifest as one-edit branches in the long revert branch as editors attempted to add the number with varying formatting (e.g. with and without a comma). The tree view shows these as different edit attempts, but the result is the same: a revert to the final conclusion of 93,173. The discarded work is clearly shown, as is the persistence of the article state representing the conclusion of the dispute. One particular editor was responsible for many of the reverts; this is visible in the color of the revision symbols.

The history tree also shows some side skirmishes visible in this view, such as adding and removing a {{trivia}} label in the article.

6. MORE ADVANCED STRUCTURING

While reverts are a useful means of detecting discarded work, there are cases they cannot detect. If an editor whishes to undo an edit and they click Mediawiki's "undo" link, this will result in a revert by default; if, however, they undo the work by hand-editing the text to reverse the changes, they may well create an new state that differs only slightly from some prior state of the article, such that it should effectively be considered the same state. Figure 9 shows an example of this happening; the left column shows the tree generated using reverts and the second and third columns show trees generated by our advanced algorithms described below. In this event, the top revision undoes the next three revisions,

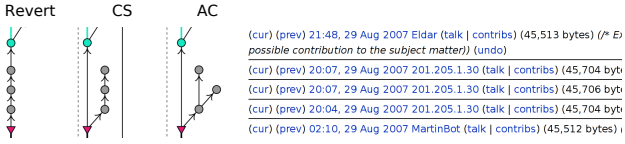

Figure 9: Inexact revert of three revisions in Chocolate. The left column is a revert-based structure, the middle cosine similarity, and the right adoption coefficient.

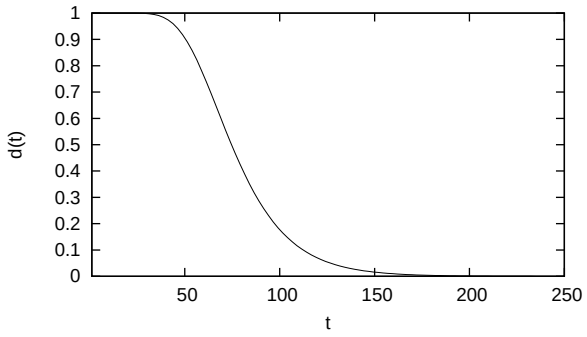

Figure 10: Plot of the decay function $d(t)$.

restoring the article to a state nearly identical to that of the last revision – they differ only in that the top revision has an extra blank line in the wiki markup. A similar problem arises if a later editor combines removal of previous content with another editing operation.

In order for the visualization to present a more complete and accurate picture of whether the changes made by a particular edit were retained by subsequent editors of the article, we would like to detect and display these more subtle relationships. To this end, we explore two different measures of similarity to identify discarded work. The general approach behind both is the same: if a revision is not identical to some prior state, find the most similar prior revision and assume that the user discarded all work between that revision and the present. This approach mirrors that used by Sabel [12]; we add explicit revert detection, recast the problem in terms of discarded work, and provide an additional similarity metric.

6.1 Cosine Similarity (CS)

A simple method of finding similar text is to compute cosine similarity between term vectors generated from the bag of words or tokens in the text. We employ this metric, building vectors from all words (sequences of alphanumeric characters) found in the article text. Our algorithm scans backwards from the revision in question to find the most similar revision.

In order to restrict how far back the program must search and to cause it to favor more recent revisions even if they are a slightly poorer match than a revision further back, we multiple the cosine similarity measure by a generalized logistic decay function $d(t)$:

$$d(t) = 1 - \frac{1}{(1 + e^{-0.05(t+35)})^{1/0.006}}$$

This decay function has the property of decaying slowly

for $0 \leq t \leq 50$ (the default span of a single page of revision history), then dropping off and approaching its asymptote near $t = 150$ (see Figure 10). Our development of this function was based on the assumption users are more likely to base their work on recent revisions from the default history view than older revisions.

6.2 Adoption Coefficient (AC)

Sabel proposed another similarity metric called the *adoption coefficient* [12]. The adoption coefficient between two texts is computed from the edit script required to produce one from the other (generated from LCS by tools such as diff). This metric is based on the intuition that the most similar prior revision is the one that the user reused the most of to get the current revision.

To compute the adoption coefficient between two revisions i and j, the text is first split into tokens (Sabel used "blocks"; we use sentences, which are equivalent to blocks in most cases) and the LCS of tokens between two revisions is computed. The adoption coefficient is then computed as follows (where l is the length of the LCS):[7]

$$a_{i,j} = 1 - \frac{\text{len}(j) + \text{len}(i) - 2l}{\text{len}(j) + \text{len}(i) - l}$$

The adoption coefficient (which falls in the range $[0, 1]$) is then used in the same way as the cosine similarity to find the most similar prior revision.

Computing the adoption coefficient requires significant processing time, as LCS algorithms are quadratic in the length of their inputs. The traditional dynamic programming solution to LCS is $\Theta(mn)$, where m and n are the lengths (in tokens) of the two texts to be compared. Hunt and McIlroy devised an algorithm that runs in $O(mn \log m)$ worst case time but fares far better in most common cases[5]. Myers proposed another algorithm that is efficient for highly similar sequences, running in $O(nd)$ (where d is the length of the shortest edit script) with tight array-based loops, but fares quite poorly for texts that differ greatly[9]. Our implementation, therefore, combines all three of these algorithms, using a heuristic based on Rabin fingerprints [11] to determine whether the text is "similar enough" to use Myers' algorithm and, if not, using Hunt and McIlroy's algorithm (or the dynamic programming approach on short sequences to take advantage of its tight inner loop and excellent constant time performance). The program also strips common prefixes and suffixes from the text to minimize the amount of text that must be compared by the quadratic algorithm.

6.3 Results

As is demonstrated with the example in Figure 9, there are cases in that the more sophisticated revision similarity computations enable the visualization to more accurately represent the relationship between revisions. There are cases, however, where they do not accurately reflect the underlying changes. Further, the two algorithms are alternately right and wrong in different cases, and there are difficulties intrinsic to their operation that can cause confusion.

While AC's intuitive basis is easy to understand and explain to users, it cannot respond accurately to textual rearrangements that leave the content of an article mostly unchanged, such as moving a paragraph or section to a different position in the article. Figure 11 shows one such case from the history of the now-deleted article "Adria (Stargate)" – Edit A substantially rearranged the text in the previous revision without editing it significantly. AC (the right column) sees a large edit between those two revisions and goes back several revisions to find one with a smaller edit distance. In doing this, it selected a parent revision that did not contain a major section of the article that Edit A moved but did not remove. CS (the middle column), since it is based only on word counts and does not consider relative order, accurately indicates that it is an incremental revision of the previous version.

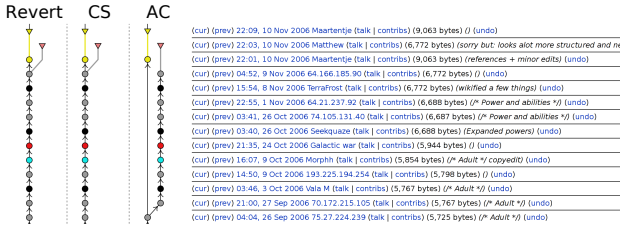

Figure 11: Portion of Adria history where CS correctly detects reversed work.

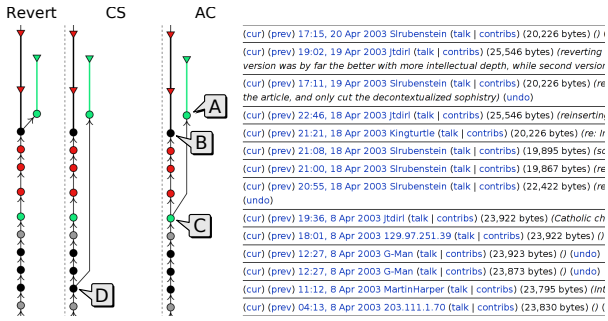

Figure 12: Portion of Abortion history where AC accurately reflects edit and CS selects the wrong parent.

Figure 12, however, shows a case in the history of "Abortion"[8] where the AC algorithm (in the right column) accurately shows that the editor creating the revision A was undoing a sequence of 4 edits ending at B without quite restoring the article to an identical prior state. The CS structuring algorithm indicated an attempt to roll back more work than was actually done (to D); this seems to be a result of the words changed between revisions D and C that have nothing to do with the removed changes between C and B affecting the term weights. Looking at the diffs between the revisions involved indicates that the AC structure (A returning the article to a state closest to C) more closely matches what the author intended to do to the article.

Another case that presents a more fundamental problem for advanced algorithms working with word-level tokens is demonstrated in Figure 13 (taken from "Brazil"[9]). In this case the most recent revision undid a change 5 revisions

[7]Our definition is formulated differently from that used by Sabel, but is mathematically equivalent and has the property of depending only on the length of the LCS rather than a complete edit script.

[8]http://en.wikipedia.org/wiki/Abortion

[9]http://en.wikipedia.org/wiki/Brazil

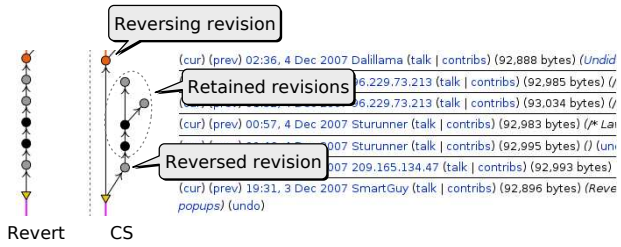

Figure 13: Exerpt from Brazil revision history where an old revision was undone.

back while leaving the subsequent changes intact. Since the change that was reversed changed more text than the ones extant, the CS structuring algorithm went back past the removed revision to find the parent revision. This results in a display that incorrectly shows all five edits as discarded. The AC algorithm displays it more reasonably, rendering the entire subtree as retained[10].

Finally, vandalism, especially vandalism consisting of replacing the entire page with some other text, causes notable problems for the advanced structuring algorithms. When encountering a vandalized revision, they will search back through the article's history and select as the parent revision some revision that doesn't bear any identifiable connection to the revision in question. Since vandalistic edits are not good-faith contributions to the development of the article, the assumption that the editor is bringing the article closer to a desired end state and that there may be some previous state that was close to their goals simply does not apply. Applying a vandalism detector prior to the structuring and considering vandalized revisions to be the child of the previous revision may reduce the impact of this particular problem.

In our experience, these problems arise frequently enough that the potential clarity gained by these more sophisticated algorithms is outweighed by the noise and confusion introduced by the bad structuring decisions they yield. Further, some of the problems seem to point to a fundamental incapacity of term vector cosine similarity and adoption coefficients to meaningfully analyze degenerate situations that arise in the wild in Wikipedia, rather than being merely artifacts of algorithmic details or implementation. This is particularly true of the problem of vandalism, as it renders false the basic assumption underlying using these metrics for tree structuring. The difficulties in detecting an old revision being redacted while leaving subsequent edits intact have a similarly deep impact on any algorithm dealing with the article in terms of small tokens; the sentence tokens render AC resilient in our example, but at the word level the reversal of the more substantial edit will likely outweigh the minor edits left in place for any algorithm.

6.4 Performance Characteristics

An additional difficulty with both the CS and AC algorithms is that they require the full text of each revision available to construct the history tree. They are also computationally intensive, as searching backwards through history causes the algorithm's worst case run time to be quadratic in the number of revisions in the article. The problem is further compounded for AC, as the underlying LCS algorithms themselves have quadratic worst-case behavior in the number of tokens in the revisions compared.

[10]When we run AC over alphanumeric tokens rather than sentences, it produces the same tree as CS.

Page	# revs	Basic	CS	AC
Adria (Stargate)	246	0.00	0.84	2.15
WrestleMania III	823	0.00	4.09	12.76
2006 hurricanes	2775	0.02	31.24	175.02
Liancourt Rocks	3825	0.03	63.67	561.63
Chocolate	4455	0.01	66.26	319.54
Abortion	8785	0.07	194.32	1467.23
Brazil	8929	0.05	260.03	2177.30

Table 1: Time required to to build history trees (seconds of user-space CPU time; 0.00 indicates times less than $1/100$ of a second).

Table 1 shows the time required to compute history trees for various pages. Time is reported in seconds of user-space CPU time required to construct the tree (not including the time needed to load the article's revision history). The article data was taken from the January 2008 full dump of English Wikipedia, and the timings were collected on a 2GHz quad-core Intel Xeon with 8GB of RAM[11]. To save memory, our program stores revision text and token sequences in hash tables that swap out to disk-based maps when they exceed a set number of entries; our threshold for these tests was 1000 entries per table (so pages with more than 1000 revisions were likely to use disk-based tables).

6.5 Summary

In light of the problems CS and AC have with various editing phenomena, we believe that they do not enhance the ability of the visualization to accurately and clearly represent the development of the article through its history. This, combined with the noticeable computational effort required to build trees with these algorithms and the relative efficiency of the revert-only algorithm lead us to conclude that the revert-only structuring algorithm is, at present, better for representing revision relationships.

7. COMPUTATIONAL METHODS

Revision history trees can also be used computationally to determine properties of revisions and articles.

7.1 k-acceptance

If we consider the revision tree to represent what work the community considered worth using as a basis for the continued development of the article and what work needed to be discarded, we can define a computable measure to determine community opinion of a revision. We say that a revision is *accepted* if the community used it as a step in the further development of the article rather than removing its contribution. In terms of our tree, a revision is accepted if it is on the path leading to some hypothetical "final" state of an article towards which the editors are moving the article. Merely checking if a revision has been reverted is not sufficient, as it may be that a later edit undoes the revert.

Since Wikipedia articles are never finished, and any analysis of history necessarily occurs at particular point in time,

[11]Our program is currently single-threaded, so it is only using one of the cores per process.

we limit the lookahead by defining k-acceptance. This enables us to formulate a well-defined notion of acceptance irrespective of work that has yet to happen.

To avoid saying revisions are unaccepted simply because there is insufficient history available after their creation, we declare k-acceptance to be undefined for certain revisions. This mitigates the effect of the temporal frontier on acceptance of revisions.

DEFINITION 1. *Let r be a revision, k a positive integer, and $\mathcal{H}$ a directed acyclic article history graph*[12]*. Then r is* k-accepted *in $\mathcal{H}$ if there is a path of length $k+1$ in the history graph starting from r.*

Whether r is k-accepted in $\mathcal{H}$ is undefined if there is some r' such that r' is connected to the last revision in $\mathcal{H}$ by a path of length k and r' was created before r.

The intuition underlying this definition is that k-acceptance reflects whether the editing community considered a revision worth building on for at least k further revisions.

7.2 Picking k

As k increases, k-acceptance is able to more closely approximate the acceptance of a revision with respect to the hypothetical final state of the article. The effect this has on acceptance of individual revisions is that some revisions which were accepted for some lower k will not be k-accepted. Increasing k will also cause k-acceptance to be undefined for a greater number of revisions. By Lemma 1, revisions which are rejected for some k will also be rejected for all greater values of k.

LEMMA 1. *Let r be a revision which is not k-accepted in $\mathcal{H}$. Then for every $j > k$, if j-acceptance is defined for r in $\mathcal{H}$, r is not j-accepted.*

PROOF. Suppose r is j-accepted. Then there is a path of length $j+1$ starting at r. There is a prefix of this path that has length $k+1$, and thus r is k-accepted, which is a contradiction. □

We note that increasing the amount of history available has the opposite effect: it will cause k-acceptance to be defined for more revisions, and may cause some previously unaccepted revisions to be accepted.

LEMMA 2. *Let r be k-accepted in $\mathcal{H}$. Let $\mathcal{H}'$ be obtained by adding additional revisions to the end of the history used to generate $\mathcal{H}$. Then r is k-accepted in $\mathcal{H}'$.*

PROOF. Adding revisions to a history tree will leave all edges in the original tree intact, and thus all paths. Therefore there is still a path of length $k+1$ starting from r in $\mathcal{H}'$. □

Since increasing k both causes k-acceptance to be more accurate and renders it undefined for more revisions, it is desirable to be able to use as small a k as possible while still detecting discarded work in order to maximize the usefulness of k-acceptance. To determine how low k can be while retaining most of the information value of k-acceptance when analyzing Wikipedia articles, we computed k-acceptance for

[12]While we only consider trees in this work, the definition of k-acceptance is equally applicable when history structures are permitted to be arbitrary DAGs.

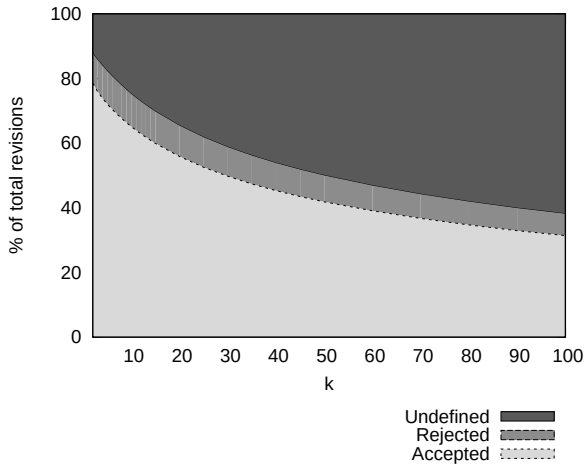

Figure 14: Classification of revisions as k increases.

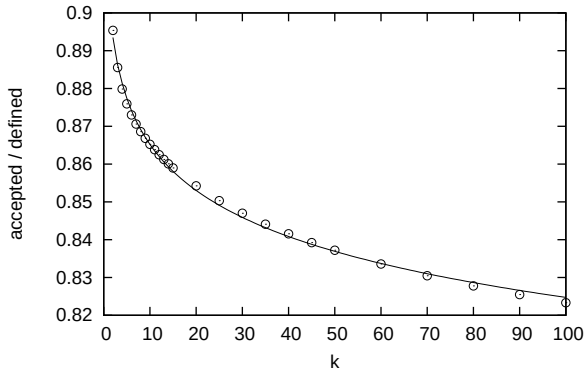

Figure 15: Fraction of articles accepted.

26 values of k in the range 2 to 100 on the revert-based history graph for each article in the Wikipedia main article namespace as of January 2008. Figure 14 shows how the classification of revisions changes as k increases, and Figure 15 shows how the fraction of articles for which k-acceptance is defined are k-accepted decreases as k increases. From this data, it seems that 15 is a reasonable value for k for practical use, as it takes into account the steepest change in percent accepted while allowing k-acceptance to be defined for a good fraction of revisions.

7.3 Applications

k-acceptance can be used to select a revision to analyze corresponding to a particular point in time. If a revision is desired corresponding to some event (e.g. a talk page edit or newscast), it may be that the last revision before that time was damaged or vandalized and that a human editor would revert it before working further with the article. It is difficult, however, to programmatically determine whether the revision being examined would be reverted. k-acceptance allows us to look into the future and see what editors looking at the revision *did* do and base our decision as to what revision to consider on that[13].

[13]Naturally, more accurate history trees will provide more accurate results for this purpose.

k-acceptance can also be used to quantify wasted or discarded work in an article over a period of time by computing *k-accepted revisions per revision* (k-ARpR) for an article over a time period. This metric captures how much work done over a period of time was discarded by the community verus how much was kept; a perfect score of 1.0 means that no work in the period was discarded. Computing k-ARpR over successive windows in the history of an article could provide a rolling metric of how much work has been wasted (frequently due to conflict) at various points in the article's history.

8. CONCLUSIONS AND FUTURE WORK

We have presented a means of structuring the history of a wiki article as a tree that represents the relationships among the article's revisions. We looked at a visualization of these relationships that can be integrated in-line with existing history views. Through case studies we examined the benefits of this tree in making salient key properties of article history, especially during times of high conflict. This structure and display should be useful in understanding the history of any Wiki page, although some types of pages (such as Mediawiki talk pages) will derive less benefit due to different editing patterns yielding less meaningful relationships.

Our visualization can be implemented as a gadget or other add-on for existing wiki software such as the Mediawiki engine behind Wikipedia. We believe it has potential to facilitate easier understanding of history for administrators, mediators, editors, and curious Wikipedia readers. Future work may involve incorporating even richer sets of data into the visualization, such as the experience of the editors involved or the age of the words in the article.

We explored three algorithms for constructing the article history trees. The basic algorithm only considers direct revert actions. This algorithm misses more subtle relationships. The more advanced algorithms are better at detecting these subtle relationships, but are misled by common Wikipedia editing patterns such as moving a paragraph within an article. Our experience suggests that these errors are fundamental to algorithms of this class. Future work should explore more sophisticated algorithms that are designed for the types of word manipulation that are common in editing natural language documents. Developing algorithms that are sufficiently sensitive and yet computationally tractable will be a challenge.

9. ACKNOWLEDGEMENTS

We gratefully acknowledge our research lab, GroupLens Research, and in particular Aaron Halfaker and Tony Lam for their help and support in this work.

This work is funded by the National Science Foundation grants IIS 05-34420 and IIS 08-08692.

10. REFERENCES

[1] B. T. Adler and L. de Alfaro. A content-driven reputation system for the Wikipedia. In *Proceedings of the 16th international conference on World Wide Web*, pages 261–270, Banff, Alberta, Canada, 2007. ACM.

[2] S. Alshattnawi, G. Canals, and P. Molli. *A Nonlinear Representation of Page History in P2P Wiki System*, volume 286 of *IFIP International Federation for Information Processing*, pages 151–160. Springer, Boston, 2008.

[3] R. P. Biuk-Aghai. Visualizing co-authorship networks in online Wikipedia. In *Communications and Information Technologies, 2006. ISCIT '06. International Symposium on*, pages 737–742, 2006.

[4] U. Brandes and J. Lerner. Visual analysis of controversy in user-generated encyclopedias. *Information Visualization*, 7(1):34–48, 2008.

[5] J. W. Hunt and M. D. McIlroy. An algorithm for differential file comparison. Technical Report 41, Bell Laboratories Computing Science, 1976.

[6] A. Kittur, B. Suh, and E. H. Chi. Can you ever trust a wiki?: Impacting perceived trustworthiness in Wikipedia. In *Proceedings of the ACM 2008 Conference on Computer Supported Cooperative Work*, pages 477–480, San Diego, CA, USA, 2008. ACM.

[7] A. Kittur, B. Suh, B. A. Pendleton, and E. H. Chi. He says, she says: Conflict and coordination in Wikipedia. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 453–462, San Jose, California, USA, 2007. ACM.

[8] P. Mackerras. gitk - the git repository browser.

[9] E. W. Myers. An O(ND) difference algorithm and its variations. *ALGORITHMICA*, 1:251—266, 1986.

[10] P. Pirolli, E. Wollny, and B. Suh. So you know you're getting the best possible information: A tool that increases Wikipedia credibility. In *Proceeding of the Twenty-Seventh Annual SIGCHI Conference on Human Factors in Computing Systems (to appear)*, Apr. 2009.

[11] M. O. Rabin. Fingerprinting by random polynomials. Technical Report TR-15-81, Center for Research in Computing Technology, Harvard University, 1981.

[12] M. Sabel. Structuring wiki revision history. In *Proceedings of the 2007 International Symposium on Wikis*, pages 125–130, Montreal, Quebec, Canada, 2007. ACM.

[13] B. Suh, E. H. Chi, A. Kittur, and B. A. Pendleton. Lifting the veil: Improving accountability and social transparency in Wikipedia with WikiDashboard. In *Proceeding of the Twenty-Sixth Annual SIGCHI Conference on Human Factors in Computing Systems*, pages 1037–1040, Florence, Italy, 2008. ACM.

[14] F. B. Viégas, M. Wattenberg, and K. Dave. Studying cooperation and conflict between authors with history flow visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 575–582, Vienna, Austria, 2004. ACM.

[15] H. Zeng, M. A. Alhossaini, L. Ding, R. Fikes, and D. L. McGuinness. Computing trust from revision history. In *Proceedings of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services*, pages 1–1, Markham, Ontario, Canada, 2006. ACM.