Hyperstructure Maintenance Costs in Large-scale Wikis

Philip Boulain
prb@ecs.soton.ac.uk

Nigel Shadbolt
nrs@ecs.soton.ac.uk

Nicholas Gibbins
nmg@ecs.soton.ac.uk

Intelligence, Agents, Multimedia Group
School of Electronics and Computer Science, University of Southampton
Southampton SO17 1BJ, United Kingdom

ABSTRACT

Wiki systems have developed over the past years as lightweight, community-editable, web-based hypertext systems. With the emergence of Semantic Wikis, these collections of interlinked documents have also gained a dual role as ad-hoc RDF [8] graphs. However, their roots lie at the limited hypertext capabilities of the World Wide Web [1]: embedded links, without support for composite objects or transclusion.

In this paper, we present experimental evidence that hyperstructure changes, as opposed to content changes, form a substantial proportion of editing effort on a large-scale wiki. The experiment is set in the wider context of a study of how the technologies developed during decades of hypertext research may be applied to improve management of wiki document structure and, with semantic wikis, knowledge structure.

Categories and Subject Descriptors

H.3.5 [**Information Systems**]: Information Storage and Retrieval—*Online Information Services*; H.5.3 [**Information Systems**]: Information Interfaces and Presentation—*Group and Organization Interfaces* ; H.5.4 [**Information Systems**]: Information Interfaces and Presentation—*Hypertext/Hypermedia*

General Terms

Experimentation

Keywords

Hypertext, Semantic Web, Wiki, Web Science

1. INTRODUCTION

This experiment forms part of a broader project looking into the potentially beneficial relationships between open hypermedia, the study of interconnected documents; Semantic Web, the study of interconnectable data; and 'wikis', web-based communal editing systems.

Hypermedia is a long-standing field of research into the ways in which documents can expand beyond the limitations of paper, generally in terms of greater cross-referencing and composition (reuse) capability. Bush's *As We May Think* [2] introduces the hypothetical early hypertext machine, the 'memex', and defines the "essential feature" of it as "the process of tying two items together". This *linking* between documents is the common feature of hypertext systems, upon which other improvements are built.

As well as simple binary (two endpoint) links, hypertext systems have been developed with features including n-ary links (multiple documents linked to multiple other documents), typed links (links which indicate something about *why* or *how* documents are related), generic links (links whose endpoints are determined by matching criteria of the document content, such as particular words), and composite documents, which are formed by combining a set of other, linked, documents. Open Hypermedia extends this with interoperation, both with other hypermedia systems and users, and with non-hypermedia resources. A key concept in open hypermedia is that of the *non-embedded* link—links (and anchors) which are held external to the documents they connect. These allow links to be made to immutable documents, and to be added and removed in sets, often termed 'linkbases'. One of the earliest projects attempting to implement globally-distributed hypertext was Xanadu [9], a distinctive feature of the design of which was *transclusion*: including (sections of) a document into another by reference.

In related work, we are currently investigating the relationship between an exemplar semantic wiki, Semantic MediaWiki [6], and open hypermedia systems, as defined by the Dexter Hypertext Reference Model [5]. Our preliminary results based on a formal description of Semantic MediaWiki in terms of the Dexter model suggest that such semantic wikis can be treated as simple open hypermedia systems. While details are beyond the scope of this paper, some basic parallels are evident: a wiki node is akin to a hypermedia document, and a semantic web resource. Semantic wikis generally treat typed inter-node links as RDF statements relating the nodes, and these links are embedded and binary in hypermedia terms. From this we can see a meaningful similarity between a graph of documents connected by typed links, and a graph of resources connected by RDF statements. We can also see that wikis do not have features covering more advanced hypermedia links: such as those which are not embedded, or have more than two endpoints.

This then suggests that semantic wikis stand to gain from techniques developed within hypermedia, but we must first judge if there is any substantial cost to be reduced. Hence we have performed an quantitative experiment on a large-scale public wiki system to measure the proportion of effort expended on hyperstructure-related activities, as opposed to editing the document content.

SWKM'2008: Workshop on Social Web and Knowledge Management @ WWW 2008, April 22, 2008, Beijing, China.
.

2. HYPOTHESIS

We carried out an experiment to estimate the proportion of effort expended maintaining the infrastructure around data, rather than the data itself, on a weak hypertext wiki system. We define a 'weak' hypertext system here as one whose feature set is limited to embedded, unidirectional, binary links, as with the World Wide Web. Our hypothesis is that the manual editing of link structure, of a type which richer hypertext features could automate, will show to be a significant overhead versus changes to the text content.

This experiment also seeks to partially recreate a related, informal experiment, discussed in an essay by Swartz [10].

3. DATASET

We chose English Wikipedia[1] as the experimental dataset, because it has both a considerably large and varied set of documents, and a complete history of the editing processes—performed by a wide range of Web users—between their first and current versions[2]. The wiki community keep the dataset fairly well inter-linked and categorised for cross-reference, but they do this via the cumulative efforts of a large body of part-time editors. As well as being statistically significant, demonstrating possible improvement of English Wikipedia is socially significant, as it is a widely-used and active resource.

It is important to stress the size of the English Wikipedia dataset. Wikipedia make available 'dumps' of their database in an ad-hoc XML format; because this study is interested in the progression of page contents across revisions, it was necessary to use the largest of these dumps, containing both page full-text and history (unfortunately, also non-encyclopædic pages, such as discussions and user pages). This dump is provided compressed using the highly space-efficient (although time-complex) bzip2 algorithm; even then, it is 84.6GB. The total size of the XML file is estimated to be in the region of two terabytes.

4. PROCEDURE

Figure 1 shows the simplified data flow of the processing of the dump performed for the experiment.

First, we trimmed down the dataset to just those pages which are encyclopædic articles, as these are the pages of greatest significance to the Wikipedia project's goals, and thus the most important to study. Otherwise, the dataset would include a lot of 'noise' in the form of discussion and user pages, which are likely to have different editing patterns, and be less connected to the hyperstructure. The most practical way to do this was to remove any page placed in a namespace. On English Wikipedia, this also has the effect of removing other page types, such as media and image descriptions, help pages copied from MetaWiki, front-page portal components, and templates. As this stage also required decompressing the data, it ran over the course of several days on a multi-processor server.

We took a random subset of the data for processing. Samples of 0.04% and 0.01% of pages (approximately: see the description of the subset tool below; actual page counts 14,215 and 3,589 respectively) were selected, yielding a compressed dataset which would fit on a CD-ROM, and could be processed in a reasonable timeframe. Further iterations of the experiment may study larger subsets of the data.

We performed categorisation on the revisions, into several edit types which would be automatically distinguished. In particular, a simple equality comparison between a revision, and the revision two edits previous, can detect the most common (anti-)abuse modification: the rollback, or revert (unfortunately, MediaWiki does not record such operations semantically). A sequence of reverts[3] is usually indicative of an 'edit war', where two users continually undo each-others changes in favour of their own. Page blanking was also easy to detect, but identifying more complicated forms of vandalism (e.g. misinformation, spam) was not feasible—if reliable, automatic detection were possible, they would not be present in the data, as Wikipedia could prevent such changes from being applied. Identifying abuse (and abuse management) of the simpler types is important, as otherwise they would appear as very large changes.

In order to detect changes in the text content, templates used, MediaWiki categories, and links from a page, it was necessary to attempt to parse the MediaWiki markup format. Such 'wikitext', as it is known, is not a formally defined language: there is no grammar for it, and it does not appear likely that an unambiguous grammar actually exists. MediaWiki does not have a parser in the same way as processing tools such as compilers and XML libraries; instead it just has a long and complicated set of text substitution procedures which convert parts of 'wikitext' into display-oriented HTML. These substitutions often interact in a ill-defined manner, generally resulting in either more special-case substitutions, or as being defined as a new, hybrid, feature, which editors then use. Because of these problems, and the lack of abstraction in MediaWiki's 'parser', as much as the programming language boundary, a 'scraping' parser was created which attempted to approximate partial processing of the wikitext format and return *mostly* correct results. This parser is a single-pass state machine (42 states) with a few additional side-effects. This yields excellent performance: testing showed that the time spent parsing is dominated by the time performing decompression.

To determine if an edit included a significant ('major') change to the text content, we required a difference metric between the plaintext of the revisions. This metric was then compared to a threshold to classify edits as being content changes or not (in particular, the imperfect parser generates 'noise' from some non-content changes, as it cannot correctly remove all the markup). The default threshold was chosen as 5%: sentences in the English language are generally around twenty words in length, so this considers anything up to changing one word in each sentence as non-major (minor). MediaWiki also allows registered users to explicitly state than an edit is minor; this flag was respected where present.

We chose an approximation of Levenshtein distance[7], as it is a simple measure of insertions, deletions, and substitutions, fitting the kind of edit operations performed on the wiki. However, the algorithm for computing Levenshtein itself was far too time-complex, even with aggressive optimisation, taking two minutes on a tiny test set of just a few thousand revisions of a single page (before trimming away the identical parts at either end of both strings to take advantage of edit locality, this took 45 minutes). The problem

[1]`http://en.wikipedia.org/`

[2]MediaWiki, unlike many wikis, never deletes old revisions of a page.

[3]e.g. `http://en.wikipedia.org/w/index.php?title=Anarchism&diff=next&oldid=320139`

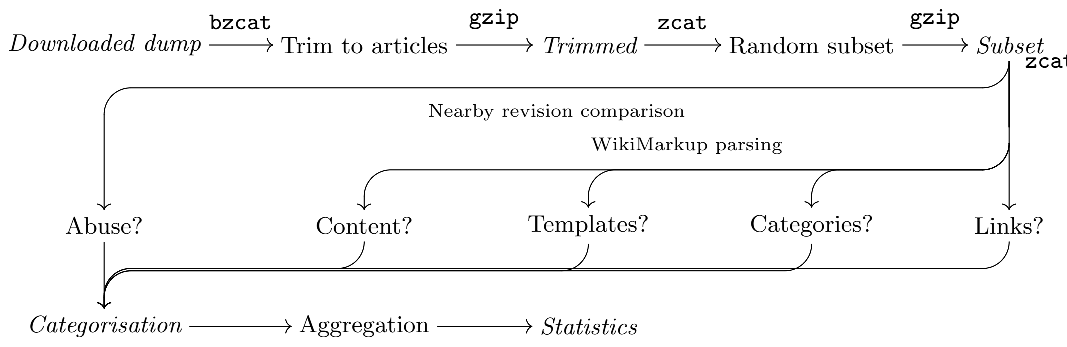

Figure 1: Data flow of Wikipedia experiment

was that the matrix-based approach is O($n \times m$), where n and m are the string lengths, in all cases: for n and m in the region of 16,000 characters, as found on many revisions, merely iterating through all 256 million matrix cells was prohibitively expensive.

Instead, we developed a new approach to computing such a distance, taking advantage of the domain-specific knowledge that the two strings being compared are likely very similar save for 'local' edits: the difference is likely to be a new paragraph, or a removed sentence, or some changed punctuation. Instead of efficient search within the space of editing operations, as Levenshtein, it is based on the idea of "sliding windows": a pass is made over both strings in parallel; when characters begin to differ, a look-back 'window' is opened between the point at which differences began, and continues until similarity is again found between these windows. At this point, the position through the strings resynchronises, the distance is increased by the offset required, and the windows are again 'closed'. When the end of either string is reached by the far edge of the window, the algorithm can terminate, as any remaining characters in the other string must be unmatched and thus add to the distance. As a result, the algorithm scales with regard to the shorter of the two strings, which is helpful when revisions may add whole paragraphs of new text to the end. To reduce inaccuracy in certain cases, the algorithm maintains a 'processed point' cursor, to avoid double-counting of overlapping insertions and deletions. Pseudocode is presented as algorithm 1, which works on a pair of string buffers, and `upstr.c` in the tool source contains a C implementation. This approach is still O($n \times m$) worst-case, but is O(n) (where n is the shorter string) for identical strings, and degrades smoothly as *contiguous* differences increase in size: instead of two minutes, the tiny test set was compared in a little over ten seconds.

Unfortunately, changes such as 'ABCF' to 'ADCDBCF' can return overestimates, as the localisation which explicitly prevents full lookback (and keeps computational cost below O(n^2)) causes the 'C' in 'BCF' to match with the 'C' in 'DCD': 'ADC' is considered a substitution of 'ABC' before the algorithm can realise that 'BC' is still intact in the string, and 'DCD' is merely an insertion. As a result, the later 'B' is considered an insertion, as it no longer matches anything, and the distance is overestimated by one. Synthetic tests showed this overestimation to be minor; tests against Levenshtein on a tiny subset of Wikipedia data (a node's first few hundred revisions, thus under heavy editing) show it to be larger, with errors in the tens, and a peak error of over two-hundred. The reason for such large errors is unclear, as the resynchronisation approach should also keep *error* localised, but it does not greatly affect the result for the purpose of minor/major determination: the majority of changes were correctly classified.

Identifying changes to links, etc. was significantly simpler, and merely required comparing the sets of links identified by the parser across two revisions. These categorisations yielded simple information on which kinds of changes were made by each revision, and removed much of the 'bulk' of the dataset (the revision texts); as a result, simple scripts could then handle the data to aggregate it into various groupings in memory, so as to produce graph data and statistics for analysis. Gnuplot[4] was used to plot the graph data into graphics as part of the build process for this paper.

We identified the following non-mutually-exclusive groupings to usefully categorise edits:

Revert Edit which simply undoes a previous edit.

Content Major (nontrivial) edit of the page content.

Minor Minor (trivial) edit of the page content.

Category Edit to the categories of a page.

List of Edit to a page which is an index to other pages.

Indexing Edit to categories or listings, possibly both.

Template Edit to the templates used by a page.

Page link Edit to an internal page link.

URL link Edit to a WWW URL link; usually external.

Links Edit to page or URL links.

Link only As 'links', but excluding major edits.

Hyperstructure Any hypermedia change: indexing, linking, or template.

We expand upon the definition and significance of these groups as needed in section 6.

[4]`http://www.gnuplot.info/`

Algorithm 1 'Sliding window' string distance metric

```
procedure STRING-DISTANCE($A, B$)
    $proc \leftarrow 0$    ▷ No. of chars. of string processed
    $procstr \leftarrow$ NEITHER    ▷ Last string aligned upon
    $dist \leftarrow 0$    ▷ Difference accumulator
    $nearA \leftarrow farA \leftarrow A$    ▷ Near and far pointers
    $nearB \leftarrow farB \leftarrow B$
    Let $endA$ be the beyond-last character of buffer $A$,
        and $endB$ beyond $B$
    procedure SCAN($near, far$)
        for $scan \leftarrow near$ to before $far$ do
            if Chars. at $scan$ and $far$ same then
                return $scan$
        return false
    repeat
        $synfarA \leftarrow$ SCAN($nearA, farA$)
        $synfarB \leftarrow$ SCAN($nearB, farB$)
        if $synfarA \vee synfarB$ then    ▷ Missed alignment
            if $synfarA$ is further into $A$ than $synfarB$
                is into $B$ then
                $farA \leftarrow synfarA$
            else
                $farB \leftarrow synfarB$
        else if $synfarA$ then
            $farA \leftarrow synfarA$
        else if $synfarB$ then
            $farB \leftarrow synfarB$
        if Chars. at $farA$ and $farB$ same then
            ▷ Aligned; calc. nears after proc. point
            $enA \leftarrow$ MIN($nearA, A + proc - 1$)
            $enB \leftarrow$ MIN($nearB, B + proc - 1$)
            ▷ Unaligned lengths
            $unA$ = positive dist. from $enA$ to $farA$
            $unB$ = positive dist. from $enB$ to $farB$
            procedure ALIGN($un, far, buffer, other$)
                $distance \leftarrow distance + un$
                $proc$ = $far$'s distance into $buffer$
                if $procstr = other$ then
                    $proc \leftarrow proc + 1$
                $procstr \leftarrow buffer$
            if $unA > unB$ then
                ALIGN($unA, farA, A, B$)
            else
                ALIGN($unB, farB, B, A$)
            if $farA = endA$ then    ▷ Ending
                $distance \leftarrow distance+$ distance between
                    $farB$ and $endB$
            else if $farA = endA$ then
                $distance \leftarrow distance+$ distance between
                    $farA$ and $endA$
            else    ▷ Advanced with closed window
                $nearA \leftarrow farA \leftarrow farA + 1$
                $nearB \leftarrow farB \leftarrow farB + 1$
                $proc \leftarrow proc + 1$
        else    ▷ Not aligned; widen windows
            if $farA \neq endA$ then
                $farA \leftarrow farA + 1$
            if $farB \neq endB$ then
                $farB \leftarrow farB + 1$
    until $farA = endA \vee farB = endB$
    return $dist$
```

5. TOOLS DEVELOPED

To process the sizable dataset, we created a set of small, robust, stream-based tools in C. Stream-based processing was a necessity, as manipulating the entire data in memory at once was simply infeasible; instead, the tools are intended to be combined arbitrarily using pipes. We used standard compression tools to de- and re-compress the data for storage on disk, else the verbosity of the XML format caused processing to be heavily I/O-bound.[5] The open source Libxml2[6] library was used to parse and regenerate the XML via its SAX interface. A selection of the more notable tools:

dumptitles Converts a MediaWiki XML dump (henceforth, "MWXML") into a plain, newline-separated, list of page titles. Useful for diagnostics, e.g. confirming that the random subset contains an appropriate range of pages.

discardnonart Reads in MWXML, and outputs MWXML, sans any pages which are in a namespace; pedantically, due to the poor semantics of MWXML, those with colons in the title. This implements the "trim to articles" step of figure 1.

randomsubset Reads and writes MWXML, preserving a random subset of the input pages. In order for this to be O(1) in memory consumption, this does not strictly provide a given proportion of the input; instead, the control is the probability of including a given page in the output. As a result, asking for 50% of the input *may* actually yield anywhere between none and all of the pages: it is just far more likely that the output will be around 50% of the input.[7]

categorise Reads MWXML and categorises the revisions, outputting results to a simple XML format.

cataggr A Perl script which processes the categorisation XML to produce final statistical results and graph data. By this point, the data are small enough that a SAX parser is used to build a custom in-memory document tree, such that manipulation is easier.

The tools are available under the open source MIT license, and can be retrieved from `http://users.ecs.soton.ac.uk/prb/phd/wikipedia/` to recreate the experiment.

6. RESULTS

Because of the known error margin of the approximation of Levenshtein distance, we computed results from both genuine and approximated distances on the 0.01% subset, so as to discover and illustrate the effects of approximation; the computational cost difference between the algorithms was significant: two-and-a-half hours for genuine, eight minutes

[5]Specifically, GNU Zip for intermediate; bzip2, as originally used by Wikipedia, made processing heavily CPU-bound.
[6]`http://xmlsoft.org/`
[7]A better algorithm, which is O(1) with regards to total data size, but O(n) with regards to subset size, is to store a buffer of up to n pages, and probabilistically replace them with different pages as they are encountered. However, even this would be prohibitively memory intensive on statistically significant subset sizes, as each page may have thousands of revisions, each with thousands of bytes of text, all of which must be copied into the buffer.

Edit type	Proportion
Categories	8.71%
Lists	1.97%
Overhead	10.56%

(a) 0.01% subset

Edit type	Proportion
Categories	8.75%
Lists	3.72%
Overhead	12.34%

(b) 0.04% subset

Table 1: Proportions of edits related to index management

for approximated. Results were then generated from the more statistically significant 0.04% subset (27 hours). This latter subset contained some pages on contentious topics, which had seen large numbers of revisions as a result.

6.1 Index management

Table 1 shows the proportions of edits in categories pertaining to index management. "Categories" are changes to the categories in which a page was placed. "Lists" are any change to any 'List of' page; these pages serve as manually-maintained indices to other pages. "Overhead" are changes which fall into either of these categories: because they are not mutually exclusive (lists may be categorised), it is not a sum of the other two values. Because these metrics do not consider the change in 'content' magnitude of a change, they are unaffected by the choice of string distance algorithm.

The ten percent overhead shows a strong case for the need for stronger semantics and querying on Wikipedia; this is one of the key goals, and expected benefits, of the Semantic MediaWiki project. While virtually every 'list of' node could be replaced with a query on appropriate attributes, the gain in category efficiency is harder to measure. Any semantic wiki must still be provided with categorisation metadata such that the type of pages can be used to answer such queries. However, some improvement is to be expected, as there are current Wikipedia categories which could be inferred: either because they are a union of other categories (e.g. 'Free software' and 'Operating systems' cover the existing category 'Free software operating systems') or because they are implied by a more specialised category, and no longer need to be explicitly applied to a page.

The increase in list overhead seen in the larger subset is likely a result of having a more representative proportion of 'List of' pages. Otherwise, the results are largely consistent across sample sizes.

6.2 Link management

Table 2 shows categories related to the management of links. "Links" refers to edits which changed either page-to-page or page-to-URL links. "Links only" refers to such edits *excluding* those edits which also constituted a 'major' content change: they are edits concerned only with links and other structure. "Hyperstructure" is the category of edits which changed any of the navigational capabilities of the wiki: either categories, 'List of' pages, links, or templates. "Content" is simply the category of 'major' edits.

The overestimating effect of the approximate string distance algorithm can be seen as a greater proportion of edits being considered 'major', with a knock-on effect on reducing the ratios of over edits over content edits. However, the results are consistent between the 0.01% subset with the approximated string distance, and the sample set four times the size. As a result, it would appear that the smaller size of the sample set has not introduced significant error in this case, and it is reasonable to assume that a Levenshtein distance comparison of the larger dataset would yield similar results to the 0.01% subset. Therefore, further discussion will focus on the 0.01% subset with Levenshtein distance results.

Category	Registered	Unregistered	Total
List of	1,146	453	1,599
Revert	4,069	679	4,748
Category	6,121	954	7,075
URL link	5,548	2,977	8,525
Indexing	7,174	1,397	8,571
Template	7,992	1,330	9,322
Content	10,275	4,182	14,457
Minor	13,776	9,961	23,737
Link only	20,969	7,877	28,846
Page link	27,205	8,871	36,076
Links	29,671	10,606	40,277
Hyperstructure	38,358	11,701	50,059
Total	57,463	23,733	81,196

Table 3: Categorisation of edits for 0.01% subset, Levenshtein

These figures show the significance of hyperstructure to Wikipedia, to a surprising degree. While we expected that link editing would prove a substantial proportion of edits compared to content, we did not anticipate that *twice as many edits change links alone than those that change content.* Most link changes were page links—those to other pages on the wiki, or metawiki—as opposed to URL links to arbitrary webpages (in some cases, pages on the wiki with special arguments). 36,076 edits modified the former, but only 8,525 the latter.

With such a proportion of editing effort being expended on modifying links on Wikipedia, there is a clear need to improve this process. Introducing richer hypermedia features to wikis, such as generic links, should prove one possible improvement. Generic links are links whose endpoints are defined by matching on criteria of the document content: a basic example being matching on a particular substring. A generic link can specify that a page's title should link to that page, rather than requiring users to manually annotate it: some early wiki systems offered this capability, but only for page titles which were written in the unnatural 'CamelCase' capitalisation. Advanced examples such as local links, present in Microcosm [3, 4], can specify scope limits on the matching. This would help with ambiguous terms on Wikipedia, such as 'Interval', which should be linked to a specific meaning, such as 'Interval (music)'.

6.3 Overall editing distribution

Table 3 shows the categorisation of all edits in the 0.01% dataset, using Levenshtein for string distance, for registered and unregistered users. Note that the edit categories are not mutually exclusive, thus will not sum to the total number of edits by that class of user. "Minor" is the category of edits which did not appear to change anything substantial: either the information extracted from the markup remains the same, and the plaintext very similar; or a registered user annotated the edit as minor. Notably, over 5% of edits are reverts: edits completely rolling back the pre-

Edit type	Proportion
Links	49.60%
Links only	35.53%
Hyperstructure	61.65%
Content	17.81%
Edit type	**Ratio over content**
Links	2.79
Links only	2.00
Hyperstructure	3.46

(a) 0.01% subset, Levenshtein

Edit type	Proportion
Links	49.60%
Links only	23.36%
Hyperstructure	61.65%
Content	35.60%
Edit type	**Ratio over content**
Links	1.39
Links only	0.71
Hyperstructure	1.73

(b) 0.01% subset, Approximated

Edit type	Proportion
Links	49.56%
Links only	25.24%
Hyperstructure	61.90%
Content	35.99%
Edit type	**Ratio over content**
Links	1.38
Links only	0.70
Hyperstructure	1.72

(c) 0.04% subset, Approximated

Table 2: Proportions of edits related to link management

vious edit; this implies that a further 5% of edits are being reverted (presumably as they are deemed unsuitable).[8] A substantial amount of effort is being expended merely keeping Wikipedia 'stationary'.

Figure 2 demonstrates the distribution of users over the total number of edits they have made, in the vein of the Swartz study [10]. There is a sharp falloff of number of users as the number of edits increases (note the logarithmic scale on both axes): by far, most users only ever make very few edits, whether registered or not. Unsurprisingly, registered users tend to make more edits overall, and unregistered users are dominant at the scale of fewer than ten edits.

Figure 3 breaks the low-edit end of this distribution down by basic categories. It is interesting to note that, other than being in close proximity (e.g. "content" and "page link"), the lines do not have any definitive overlaps: the breakdown of edits is consistent regardless of the number of edits the user has made. Users who have made 70 edits have made edits in the same relative proportions (i.e., more "revert" than "list of") as those who have only made five.

Figure 4 shows how the magnitude of edits breaks down by the number of edits of that magnitude, again in the vein of Swartz [10]. Because this is clearly sensitive to the string distancing algorithm, the 0.01% subset was used, with a focus on Levenshtein: the approximate distance for all users is shown as a sparsely dotted line with a consistent overestimate. These results are largely unsurprising: registered users make larger edits, and most edits are small, with the count rapidly falling off as magnitude increases.

6.4 Limitations of detection

There are, unfortunately, several kinds of 'overhead' costs which simply cannot be detected in a computationally feasible manner by this approach. For example, MediaWiki supports a feature called template 'substitution', which actually imports the template, with parameter substitution performed (with some caveats), into the source text of the including node. It is important to note that the relationship between the including and included nodes is lost, and that the benefits of re-use (such as storage efficiency and later corrections) are not available. The information regarding the origin of the text is also lost without manual documentation effort, including any parameters required for the more complicated templates. Because use of this feature is not semantically recorded by MediaWiki, it is largely indistinguishable from the addition of a paragraph of wikitext. As a result, it is not then possible to evaluate the cost of maintaining or documenting these substitutions once the link to the original template has been lost.

It is also not computationally feasible to detect the pattern of a user performing the same fix on multiple pages, which would identify the cost of inadequate, or underused, transclusion. Transclusion is an inclusion-by-reference mechanism, where a selected (fragment of a) document is included 'live' into another, greatly facilitating re-use.

In Wikipedia, it is often desirable to accompany a link to a page with a short summary of that page's topic. In particular, Wikipedia has many cases where articles include a summary of another article, along with a "main article" link. The 'London' page[9], for example, has many sections which consist primarily of summaries of more detailed pages, such as 'Education in London'. However, without some form of transclusion or composition to share text, if the main article's summary changes—possibly because its subject changes—this change must be replicated manually out to any page which also summarises it. A transclusion mechanism would allow a single summary of the subject to be shared by all pages which reference it, including the main article on the subject, if desired.

For example, the 'Education in London' page may begin with a summary of its topic, highlighting the most notable institutions and successful research areas. The article on 'London' may then, within its 'Education' section, transclude this summary from the 'Education in London' page. Should the summary be updated, perhaps because a University gains significant notability in a new research area, this change would be automatically reflected in the 'London' page, as it is using the same text.

While MediaWiki's templates do function as transclusion, they are not employed for this role: common usage and development effort focus on their use as preprocessing macros.

7. CONCLUSIONS

The experiment consisted of the non-exclusive classification of edits made throughout the history of Wikipedia, a large and public wiki system. Classifications included both the areas of "text editing" (assumed to be primarily maintaining the *information content* of Wikipedia: its encyclopædic articles), and "link editing" (maintaining the *navigational structure* of the content). The hypothesis, that link

[8]Actual figures may vary in either direction: this does not detect rollbacks to versions earlier than the immediately preceding version, and 'edit wars' of consecutive rollbacks *will* be entirely included in the first 5%, not belonging in the latter.

[9]`http://en.wikipedia.org/w/index.php?title=London&oldid=155695080`

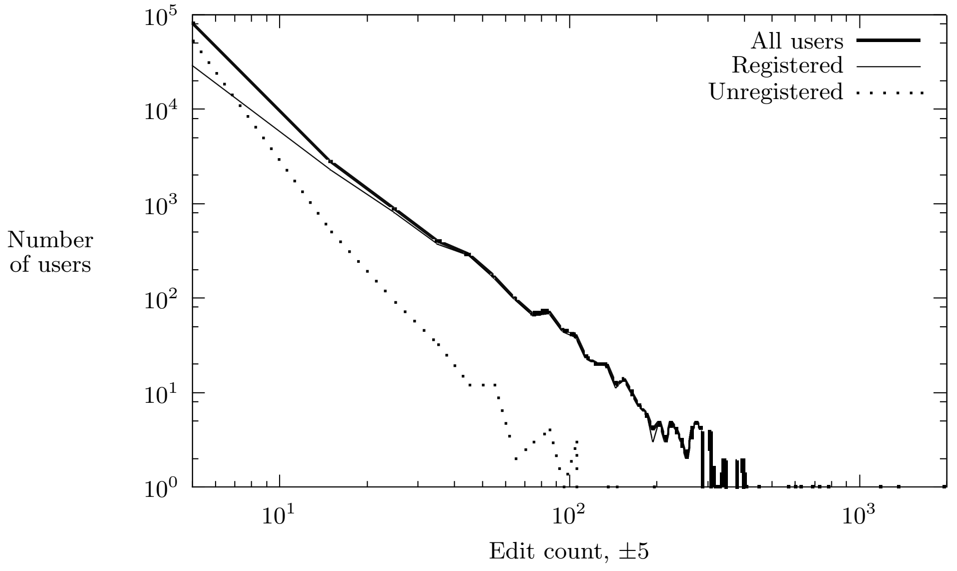

Figure 2: User distribution over total number of edits made; 0.04% subset

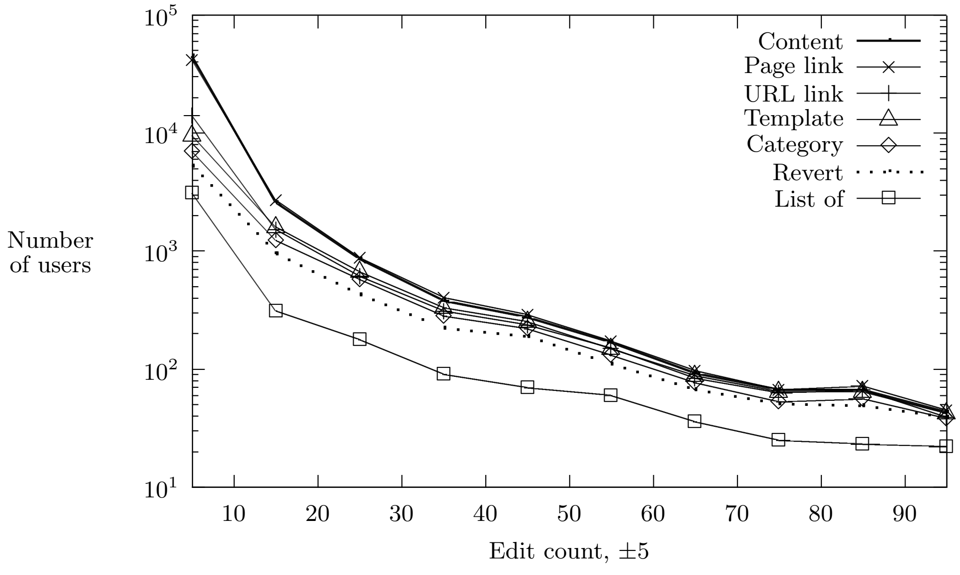

Figure 3: User distribution over total number of edits made, by category; 0.04% subset

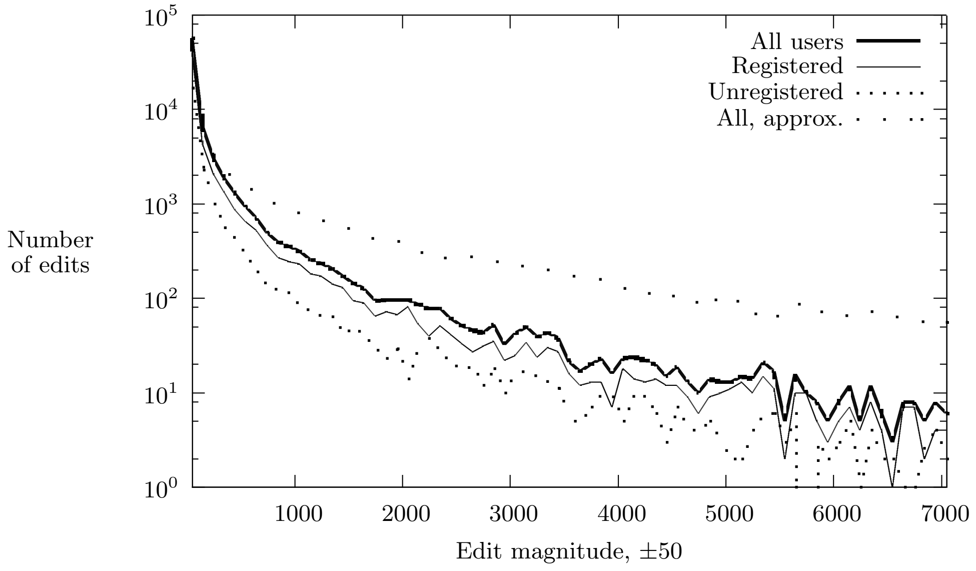

Figure 4: Edit distribution over magnitude of edit; 0.01% subset

editing formed a substantial proportion of total editing effort, which may potentially be automated, was supported by the results. Twice as many edits changed links alone, not affecting the article text. Edits which maintained manual indexes of pages constituted approximately a tenth of total edits.

We are continuing this work with a more detailed, small-scale experiment, to understand better the patterns of real-world wiki editing. It is being treated as a knowledge elicitation task, to gather information on the mental processes behind wiki editing: information on the tasks editors set themselves, and how their actions are used to achieve them. Our long-term goal is to continue this research by means of development and evaluation of a prototype system, informed by these studies, which can be used to test the hypothesis that increased hypermedia features actually result in benefits such as a decrease of editing overhead.

8. REFERENCES

[1] T. Berners-Lee, R. Cailliau, J.-F. Groff, and B. Pollermann. World-Wide Web: The Information Universe. *Electronic Networking: Research, Applications and Policy*, 1(2):74–82, 1992.

[2] V. Bush. As We May Think. *The Atlantic Monthly*, 176:101–108, Jul 1945.

[3] H. Davis, W. Hall, I. Heath, G. Hill, and R. Wilkins. Towards an integrated information environment with open hypermedia systems. In *ECHT '92: Proceedings of the ACM conference on Hypertext*, pages 181–190, New York, NY, USA, 1992. ACM Press.

[4] A. M. Fountain, W. Hall, I. Heath, and H. Davis. MICROCOSM: An open model for hypermedia with dynamic linking. In *European Conference on Hypertext*, pages 298–311, 1990.

[5] F. Halasz and M. Schwartz. The Dexter hypertext reference model. *Communications of the ACM*, 37(2):30–39, 1994.

[6] M. Krötzsch, D. Vrandečić, and M. Völkel. Wikipedia and the semantic web - the missing links. In *Proceedings of the WikiMania2005*, 2005. Online at `http://www.aifb.uni-karlsruhe.de/WBS/mak/pub/wikimania.pdf`.

[7] V. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, Feb 1966.

[8] F. Manola and E. Miller. RDF Primer. Technical report, W3C, Feb 2004.

[9] T. Nelson. *Literary Machines*. Mindful Press, Sausalito, California, 93.1 edition, 1993.

[10] A. Swartz. Who Writes Wikipedia? Online at `http://www.aaronsw.com/weblog/whowriteswikipedia`, Sep 2006.