What Did They Do? Deriving High-Level Edit Histories in Wikis

Peter Kin-Fong Fong
Department of Computer and Information Science
Faculty of Science and Technology
University of Macau
Macau S.A.R., China
Tel: +853-8397 4296
ma86515@umac.mo

Robert P. Biuk-Aghai
Department of Computer and Information Science
Faculty of Science and Technology
University of Macau
Macau S.A.R., China
Tel: +853-8397 4375
robertb@umac.mo

ABSTRACT

Wikis have become a popular online collaboration platform. Their open nature can, and indeed does, lead to a large number of editors of their articles, who create a large number of revisions. These editors make various types of edits on an article, from minor ones such as spelling correction and text formatting, to major revisions such as new content introduction, whole article re-structuring, etc. Given the enormous number of revisions, it is difficult to identify the type of contributions made in these revisions through human observation alone. Moreover, different types of edits imply different edit significance. A revision that introduces new content is arguably more significant than a revision making a few spelling corrections. By taking edit types into account, better measurements of edit significance can be produced. This paper proposes a method for categorizing and presenting edits in an intuitive way and with a flexible measure of significance of each individual editor's contributions.

Categories and Subject Descriptors

H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces – *collaborative computing*. I.7.1 [**Document and Text Processing**]: Document and Text Editing – *Version control.*

General Terms

Algorithms, Measurement, Design, Experimentation.

Keywords

Wiki, revision history, text differencing, edit categorization, edit significance.

1. INTRODUCTION

Personal computers have long become the main writing tool for many people, and collaborative writing has similarly moved to computer-based applications. Web-based collaborative writing platforms, such as wikis, allow more speedy and convenient collaborative writing than before.

Wikis support easy creation and editing of interlinked pages by using a simplified markup language, and editing directly within a web browser [10]. They are designed in this way to encourage broad participation in content creation. People use wikis to facilitate interaction and collaboration, including on community websites, corporate intranets, and knowledge management systems. Wikipedia is a well-known example of a wiki-based website, which uses a wiki system to develop an online encyclopedia that is written and edited in its entirety by a community of readers, or rather reader-editors.

A wiki software usually saves all previous versions, also called *revisions*, of any single page. Typically a "history" view is provided to present all previous versions to the user, enabling all participating writers to track the edit process and progress of an article. The history view normally offers a "diff" function which displays the difference (addition, deletion and modification) between any two distinct versions of an article, so editors do not need to manually compare the text by themselves [5]. In addition, summaries of each edit may also be provided, including the size difference in characters from the previous version, a short line of summary text provided by the editor, etc. These summaries can help other editors to understand what has changed in the new version without having to look into the text. However, the difference counted as a number of characters does not expose the nature of the edit, and a summary text is not always provided. Even where a summary text is provided it may be unable to comprehensively reflect the nature of the edit.

There are scenarios when we wish to know which kind and how significant a change has happened in an edited text without having to manually examine each version of a document. For example, in a large wiki with many active editors, the number of documents and their respective versions can be enormous, making it difficult, if not impossible, to identify the role of each writer (e.g. who is content contributor, who is editor, who formats content, who is proofreader etc.) through unaided human effort. A computer aid to derive an *intuitive editing history*, reflecting high-level changes between versions in a way close to human perception, would allow users and other applications to know what kind of changes actually happened in each edit, rather than just a "differences of words" which wiki systems offer nowadays.

WikiSym '10, July 7-9, 2010, Gdańsk, Poland
Copyright © 2010 ACM 978-1-4503-0056-8/10/07... $10.00

An edit history analyzer can help us to solve the above problem. Such an analyzer takes two versions of a document as input, processes and analyzes the relation between these versions, and then outputs a list of edits that have been made, identifying their type and significance. From that detailed edit information one can further derive more summarized information, including a classification of authors by their usual type of edit, automatically generate an edit summary, measure an edit's significance, and many others. This paper proposes such an edit history analyzer.

Significance is a subjective measure of how important a given editor's contribution is. Significance depends on the *type* of a given edit, as well as its *volume*. Edit types include actions such as adding text, inserting references, formatting text, correcting spelling, etc. Adding new text to an article is usually regarded as the most significant type of edit, whereas fixing spelling and punctuation errors could be considered one of the least significant ones. The exact ranking and relative value of these types is dependent on the value system of a given community of wiki editors and should be determined through group consensus. Given a new revision of a text, the individual edits within it can be identified and categorized over the set of defined edit types. The volume of text involved in these types of edits (such as adding new text), or the number of occurrences of a particular type of edit (such as inserting an internal link) determines the magnitude of edit significance. With a system of edit significance in place, it can be calculated to determine who the most significant authors of an article are. Thus, similar to how author names appear in their order of significance of contribution on a traditional publication such as this paper, a wiki article too could show an ordered list of its authors based on calculated significance taken over all of its revisions.

Intuitiveness means that a software's interpretation of a fact, such as what types of edits someone has performed, is in agreement with a majority of users.

The remainder of this paper is structured as follows. After reviewing prior work on differencing, categorizing versioned text, and analyzing editing histories in wikis in Section 2, we propose our method of high-level edit history analysis in Section 3. In Section 4 we present a prototype implementation of our method, and show a preliminary evaluation of our work, using real world examples, in Section 5. Finally we discuss some potential applications of our method in Section 6.

2. RELATED WORK

Several researches have focused on analyzing and visualizing edit histories of articles in wiki systems. Edit histories have been visualized as a history flow diagram (Viégas et al. [18]), and as a tree of versions (Sabel [13], Ekstrand and Riedl [6]). Those visualizations are generally based on simple equality comparisons, or similarities between versions based on sentence or word differences. None of them classifies edits into different categories.

Author contribution measure is another area that frequently digs into the version history of articles. Existing methods depend on word-based text differences (Adler et al. [2], Kittur et al. [9]) with no distinction of types of content (body text, reference, image, markup etc.). To the best of our knowledge, we are not aware of any prior work that attempts to categorize edits in a wiki by an algorithm as we propose, besides labeling reverted versions.

In the following paragraphs, we review some text differencing algorithms used by other wiki researchers, as well as some research related to categorizing edits.

2.1 Differencing Algorithms

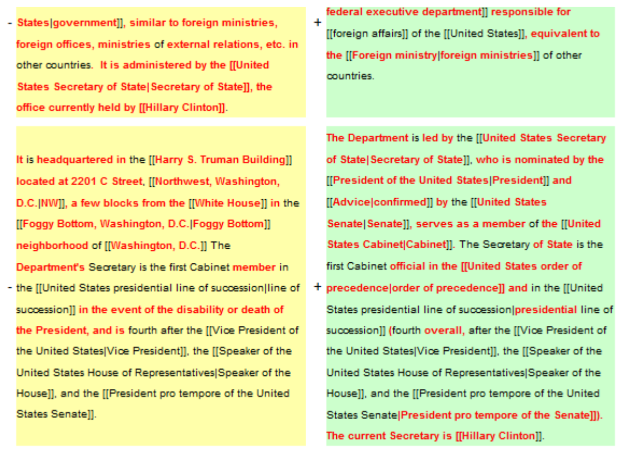
Longest common subsequence (LCS) based text differencing methods, represented by the Unix *diff* utility [8], produce a difference statement in terms of insertion, deletion and replacements relative to an old version of a text. MediaWiki, the software which Wikipedia runs on, employs this kind of method to display differences of wikitext. Some methods are based on it to calculate edit distances as well, e.g. [9] and [13]. However, the kind of difference statement produced is not very satisfactory, as it does not recognize text movements, i.e. where a piece of text has moved up or down relative to its previous location. In this case the output of these methods is a deletion of the text in the old version and the addition of the same piece of text in the new version. Text movement is a possible edit action in many situations, for example when re-structuring an article. Therefore, a difference engine that marks the differences in terms of insertion, deletion, block moves, and replacements would be preferable.

Tichy [17] has proposed a different approach for text differencing. His algorithm generates an edit script in terms of copying blocks of characters from the old version, and then adding missing characters to construct the new version. While the algorithm originally aims to produce a minimal edit script, it can be modified to produce a difference statement in terms of insertion, deletion, replacement, and block match. Especially block matches can be out of order, which makes the algorithm more preferable than the LCS-based method when content movements are of interest to the user. This algorithm was adopted and refined by Adler and de Alfaro [1] for author trust calculation in Wikipedia, implemented in their WikiTrust system, and later used in author contribution measures [2].

In terms of granularity, differencing on word level alone cannot generate a good difference statement for human understanding, as Neuwirth et al. [13] suggest. Their proposed algorithm is based on Myers' basic diff algorithm [12]. It differences two versions of a document with hierarchical decomposition strategy that exploits the grain size (paragraph, sentence, phrase, word, character). Their method differences versions of a document of coarse grain size (e.g. paragraph) first, and if the compared strings have little in common it is reported as "whole string has changed". On the contrary, if the compared string has many commonalities, finer grain size differencing is conducted. This process repeats until differencing is performed on the finest level. Since it only shows fine level differences when the edit distances are small, and concludes that two blocks are entirely different when the edit distances are big, fragile difference statements are avoided.

MediaWiki currently performs two levels of text differencing, paragraph level and word level, but word level differencing is presented even if a paragraph is changed heavily. This approach sometimes generates hard-to-read difference statements. Figure 1 shows an example illustrating this problem, extracted from an article in the English Wikipedia[1].

[1] `http://en.wikipedia.org/w/index.php?title=United_States_Department_of_State&diff=prev&oldid=321890088`



-	States|government]], similar to foreign ministries, foreign offices, ministries of external relations, etc. in other countries. It is administered by the [[United States Secretary of State|Secretary of State]], the office currently held by [[Hillary Clinton]].	+	federal executive department]] responsible for [[foreign affairs]] of the [[United States]], equivalent to the [[Foreign ministry|foreign ministries]] of other countries.
-	It is headquartered in the [[Harry S. Truman Building]] located at 2201 C Street, [[Northwest, Washington, D.C.|NW]], a few blocks from the [[White House]] in the [[Foggy Bottom, Washington, D.C.|Foggy Bottom]] neighborhood of [[Washington, D.C.]] The Department's Secretary is the first Cabinet member in the [[United States presidential line of succession|line of succession]] in the event of the disability or death of the President, and is fourth after the [[Vice President of the United States|Vice President]], the [[Speaker of the United States House of Representatives|Speaker of the House]], and the [[President pro tempore of the United States Senate]].	+	The Department is led by the [[United States Secretary of State|Secretary of State]], who is nominated by the [[President of the United States|President]] and [[Advice|confirmed]] by the [[United States Senate|Senate]], serves as a member of the [[United States Cabinet|Cabinet]]. The Secretary of State is the first Cabinet official in the [[United States order of precedence|order of precedence]] and in the [[United States presidential line of succession|presidential line of succession]] (fourth overall, after the [[Vice President of the United States|Vice President]], the [[Speaker of the United States House of Representatives|Speaker of the House]], and the [[President pro tempore of the United States Senate|President pro tempore of the Senate]]). The current Secretary is [[Hillary Clinton]].

Figure 1. A hard-to-read difference statement in English Wikipedia

2.2 Edit Categorization

A wiki teaching environment by de Pedro [4] categorizes edits for student evaluation purpose. The system requires its student editors to categorize their own edit into one or more categories, for example "markup improvement" or "new information". If the system were able to suggest edit categories for students, as we are proposing here, time used to categorize edits could be reduced.

A study of the evolution of a concept in the Wikipedia article "Web 2.0" conducted by Gorgeon and Swanson [7] classifies edits into several categories. Some of these categories are already in common use among Wikipedia editors, such as "vandalism" and "spam". Others are defined by the authors for their study purpose, for example "unchallenged" and "challenged" edits. They examine each of the 3,665 edits individually to classify them, a task that is described in the paper as "simple but tedious". This would be another suitable candidate for automated classification by software. If the task could be automated, or at least semi-automated, articles and versions could be examined on a larger scale, and a bigger picture of article evolution in general could be drawn.

3. EDIT HISTORY ANALYZER

Our proposed edit history analyzer takes two versions of a document as input, analyses the differences, and outputs a list of summaries of changes. The analyzer is divided into four parts, working step-by-step. Following the order of steps, they are *lexical analyzer*, *text difference engine*, *action categorizer*, and *history summarizer*. Figure 2 depicts their relationship.

The lexical analyzer breaks the raw text into tokens and sentences so that the text can be analyzed easier in the following steps. The text difference engine compares two versions of the text, and produces a list of edits in terms of basic edit actions, such as insertion, deletion, movement and replacement. The action categorizer takes the basic edit actions and classifies them into different categories, from minor ones like spelling correction, sentence re-arrangement, inter-language links, internal link insertion, to major revisions like new content addition, whole article re-structuring, and many others. The history summarizer collects all action meta-information recognized in the previous step, and makes summarized statements about the revision, such

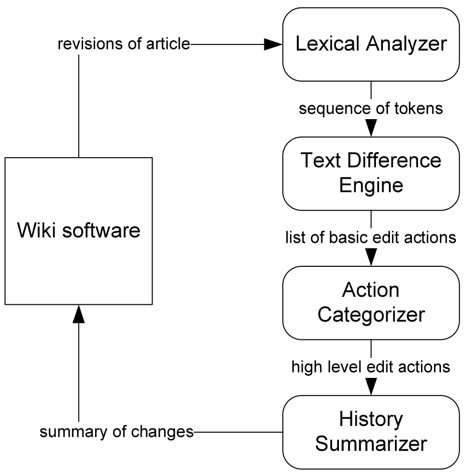


Figure 2. General structure of proposed edit history analyzer

as the proportion of newly added content, spelling corrections, formatting, etc.

The following sub-sections describe each step of the proposed method in more detail.

3.1 Lexical Analyzer

When the system receives a new version of an article, it first passes the text into a lexical analyzer. In our system, the lexical analyzer is used to break an article's raw character stream into tokens of words, punctuation and markup symbols, and then break the token stream into sentences.

3.1.1 Tokenizer

The tokenizer transforms a sequence of characters into a sequence of tokens by applying certain grammar rules. It is used in our system to make sure markup symbols are not broken apart, and to identify those markups. Each wiki software usually has its own specific set of system-specific markups. For example, MediaWiki and Twiki use a different set of markup symbols. In the tokenizing process we translate the markup symbols into tokens that reflect the semantics of the markup, thus platform-neutral differencing and categorizing is possible. Since different markup languages have different language specifications, a different set of grammar rules is needed for the tokenizer. Table 1 shows some examples of wiki markup used by MediaWiki.

To illustrate the usage of the tokenizer, here is an example wikitext in MediaWiki markup format. The original source text is:

```
'''Paris''' ({{pron-en|'parɪs}} in [[English
language|English]]) is the [[Capital
(political)|capital]] and [[primate city]] of
[[France]].
```

After lexical analysis, it is broken into a string of tokens. Symbol characters that belong to wiki markup are grouped together as a single token.

```
''' Paris ''' ( {{ pron-en | ' parɪs }} in
[[ English language | English ]] ) is the
[[ Capital ( political ) | capital ]] and
[[ primate city ]] of [[ France ]] .
```

Table 1. Some basic wiki markup in MediaWiki

Markup	Meaning	Occurrence
`[URI Text]`	External link	Anywhere
`[[Text]]`	Internal link	Anywhere
`[[File:name.jpg]]`	Image	Anywhere
`{{Text}}`	Template	Anywhere
`'''Text'''`	Bold	Anywhere
`''Text''`	Italic	Anywhere
`<ref>Text</ref>`	Reference	Anywhere
`* Text`	Unordered list item	Beginning of line
`# Text`	Ordered list item	Beginning of line
`----`	Horizontal Line	Beginning of line
`== Text ==`	Title (2nd level)	Beginning of line
`=== Text ===`	Title (3rd level)	Beginning of line

A type is given to every token during the lexical analysis process. The first token of the above example is of type "bold-open", followed by a "word" token, "bold-close", "punctuation", "template-opening", "template-name", and so on. Type of token is used in the following steps to split token sequences into sentences, and categorize edit actions.

3.1.2 Sentence Splitter

After tokenizing the source wikitext, the token sequence is passed to a sentence splitter, which splits the sequence into a number of sentences.

Methods have been proposed on this topic by researchers in the natural language processing area, e.g. SATZ, which uses neural network based methods [14]. We are currently using a naïve sentence splitting method, which delimits the sequence when it encounters sentence-ending punctuations, such as a period (full stop), exclamation mark or question mark. A few exception rules have also been implemented, e.g. "repetition of single character and dot" pattern is recognized as an acronym. While this naïve method is not perfect, it works in most test cases in Simple English Wikipedia. A more sophisticated method may be employed in the future.

Besides basic plaintext sentence delimiting, we should also consider a few additional situations in the context of wikitext:

- A list item markup symbol should be treated as a sentence delimiter.
- A heading should be treated as a sentence by itself. The same applies to category and inter-language links.
- In-line references should be extracted from the content text and treated as a separate sentence. This is because references are presented separately in the article.

3.2 Text Difference Engine

The next step of analysis is to calculate the difference to the previous revision, in order to find out what has changed from the previous version to the current version.

As discussed in Section 2.1, single level text differencing is not capable of generating a good difference statement for human understanding, thus two levels of differencing are performed. The two levels we use are sentence level and token level. Unlike MediaWiki, we use sentence level instead of paragraph level as the larger unit, because it is the basic linguistic unit in human language. Tokens are used instead of just words, because we want the system to be aware of markup changes.

First, sentence level matching is performed, looking for exact matches between whole sentences. Then for all sentences not matching exactly, token level differencing is performed. Finally, sentence matching rates between old version sentences and new version sentences are calculated based on token level difference results for approximate sentence matching.

3.2.1 Basic Edit Actions

Given a document D with versions $V_1, V_2, \ldots, V_n$, each of them contains a sequence of tokens $V_i = [t_1, t_2, \ldots, t_l]$. In the following paragraphs, we denote

- $V_i[x]$: x^{th} token in the token sequence of V_i
- $V_i[x\ldots y]$: subsequence of tokens of V_i, from x^{th} token to y^{th} token ($1 \le x \le y \le l$). Note that $V_i[x\ldots x] = V_i[x]$. We define $V_i[x\ldots y] =$ [] (empty sequence) when $x > y$.
- l_i: number of tokens in V_i.
- a^b: concatenation operation, which creates a new token sequence from all tokens in first operand a, followed by all tokens in second operand b.

A *basic operation op* transforms a sequence of tokens from one state to another state, and includes *insertion*, *deletion*, *movement* and *replacement*. A *revision* R_i is a sequence of basic operations $op_{i,j}$ that are applied to V_i to construct V_{i+1}. The following defines each of the four basic operations:

- *Insertion* is an operation that introduces a sequence of n new tokens $T_{\text{new}} = [t'_1, t'_2, \ldots, t'_n]$ after a certain position p.

  $$\text{Insert}(T_{new}, p)\text{: } V_i \rightarrow V_i\,[1\ldots p] \wedge T_{new} \wedge V_i\,[p{+}1\ldots l_i],\quad (0 \le p \le l_i)$$

- *Deletion* is an operation that removes a series of k tokens from position p.

  $$\text{Delete}(k, p)\text{: } V_i \rightarrow V_i\,[1\ldots p{-}1] \wedge V_i\,[p{+}k\ldots l_i],\quad (1 \le p \le l_i\,, p < p{+}k \le l_i{+}1)$$

- Movement is an operation that moves a series of k tokens from position p to position q.

  $$\text{Move}(k, p, q)\text{: } \begin{cases} V_i \rightarrow V_i\,[1\ldots q{-}1] \wedge V_i\,[p\ldots p{+}k{-}1] \wedge V_i\,[q\ldots p{-}1] \wedge V_i\,[p{+}k\ldots l_i], \\ \quad \text{when } 1 \le q < p < p{+}k \le l_i + 1\text{ ;} \\ V_i \rightarrow V_i\,[1\ldots p{-}1] \wedge V_i\,[p{+}k\ldots q] \wedge V_i\,[p\ldots p{+}k{-}1] \wedge V_i\,[q{+}1\ldots l_i], \\ \quad \text{when } 1 \le p < p{+}k \le q \le l_i + 1 \end{cases}$$

- Replacement is defined as a combination of insertion and deletion operation, in the way that the insertion operation of n new tokens immediately follows the deletion operation of k tokens at position p.

  $$\text{Replace}(k, t_{\text{new}}, p)\text{: Delete}(k, p)\text{; Insert}(T_{\text{new}}, p),\quad \text{where } T_{new} = [t'_1, t'_2, \ldots, t'_n]$$

3.2.2 Basic Differencing Algorithm

The basic token differencing algorithm is based on the text matching algorithm used by WikiTrust [1], which is in turn a variation of a greedy matching algorithm ([3], [17]). The basic idea of the differencing method is to match the old version's chunks of sentences or tokens in the new version, and to label unmatched old version chunks as deleted and unmatched new version chunks as added.

The first step of differencing is matching. First, the chunks (sentences or tokens) in the new version are indexed and placed into a hash table. Then the old version is scanned from beginning to end. For each chunk (containing one or more sentences or tokens), find occurrences in the new version. If such a chunk is found in the new version, advance the cursor position in the old version and new version simultaneously, and match the chunks until no more matches can be found. Matched positions are stored into a maximum heap, ordered by length of the match.

After the entire old version has been scanned through, matches are removed one by one from the heap and recorded. Using a maximum heap can guarantee that the longest possible matches are found first, while short matches contained within can be ignored.

After matching is finished, unmatched portions can be processed. The process of handling unmatched sentences is different from that of unmatched tokens. Unmatched sentences are concatenated and passed to the same matching algorithm, to conduct token level differencing. Unmatched token handling is discussed in the next section.

3.2.3 Token Differencing

Unmatched tokens appearing only in the old version can be marked as deleted, and unmatched tokens appearing only in the new version can be marked as inserted. Replacement can be marked as deletion and insertion at the same position. This can be discovered by checking previous matched chunk and next matched chunk of each deletion and insertion, to see if they are the same matched chunk or not.

3.2.4 Sentence Differencing

For each sentence in the old version, a matching rate is computed against every sentence in the new version. If we denote:

- The number of tokens in the i^{th} sentence of old version as lo_i,
- The number of tokens in the j^{th} sentence of new version as ln_j,
- The number of common tokens between the above two sentences as $lc_{i,j}$,
- The matching rate between the above two sentences as $m_{i,j}$

Then the matching rate can be calculated by the formula below

$$m_{i,j} = 2 \times lc_{i,j} / (lo_i + ln_j)$$

Since the number of common tokens will never exceed the two numbers of tokens in both sentences, the upper bound of matching rate is 100%, which happens when two sentences are identical. The lower bound is 0%, which happens when two sentences have no common tokens. If the matching rate is greater than a certain threshold, the two sentences are considered approximately matched, which means the sentence in the new version is based on the sentence in the old version. We currently have set the sentence matching threshold to 40%. This value was determined empirically through several experiments on Simple English Wikipedia data, and can be changed as needed.

If the text revision contains sentence merging (combining two or more sentences into one new sentence) or splitting (separating one sentence into two or more sentences), only calculating a matching rate against a single sentence may not reflect the edit situation very well. However, this occurs quite frequently in edited text. For example, the old version may contain these two consecutive sentences: "Apple is a fruit. It usually has red skin." It could be merged in the new version like: "Apple is a fruit that usually has red skin." In order to detect these cases, we attempt to combine consecutive sentences in the old version, and see if there is any improvement in the matching rate:

$$\frac{2 \times (lc_{i,j} + lc_{i+1,j})}{lo_i + lo_{i+1} + ln_j} > m_{i,j}$$

If the matching rate increases, the two sentences in the old version are marked as merged into one sentence in the new version. In the case of the above example we would find that neither of the two sentences in the old version taken individually match the merged sentence in the new version very well (53% and 50%, respectively), but when merging these two sentences from the old version the matching rate increases significantly (to 76% in this example). The merging process can be repeated, trying to merge more sentences and to maximize the matching rate. The same method can be applied to sentences in the new version, the result of which reflects sentence split.

A sentence with its maximum matching rate to other sentences lower than the defined threshold is considered as completely removed (if it is in the old version) or completely new (if it is in the new version). This decision is based on the finding in [13]: reporting them as completely new or removed is more close to human understanding.

3.2.5 Movement Detection

Conceptually, a matching chunk (of either sentences or tokens) whose relative order in the sequence of chunks has changed is considered a case of movement. We can check if there are movements of chunks by labeling each match with its ordinal position in the old version, then check if they are sorted or not in the new version. If it is not, label the match with the greatest position offset as moved, and remove it from the match set. Repeat the above process until no more inversion is found. All movements should be labeled at that time.

In practice, we find that matches shorter or equal to three tokens are usually common word phrases. For instance in an article on "green tea" it is common for many occurrences of this phrase to appear. Displaced short matches are unlikely to be real moves but instead are more likely cases of deletion and insertion of the same word phrase that happen in different locations. Thus we only consider a displaced match as a move if it is longer than three tokens.

3.3 Action Categorizer

With differences of revisions on hand, we can do further analysis on those basic edit actions and try to discover and categorize higher level edit actions. A rule-based categorizer can be used to extract various such edit actions. Rules can be customized to better capture these high-level edit actions. Additional comparison or computation can be done following the rule-based categorizer to extract the detailed nature of the edit action. For example, we can calculate the character level edit distance of original and current text when a short phrase replacement is encountered (for instance when "haelth" is replaced with “health”), to find out if it is a spelling correction or a term replacement.

3.3.1 Rule-based Categorization of Edit Action

With a list of basic edit actions, we can further categorize them based on the type of action, type or content of tokens in the edit, or a combination of them.

In the following paragraphs, we denote

- a: Basic edit action (Insert, Delete, Replace, Move)
- t: Token (Belongs to certain type such as markup tags, words, link content, etc.)
- string(t): The string representation of t
- type(t): The token type of t
- distance(s_1, s_2): Character edit distance between s_1 and s_2
- $T_{\text{old}} = V_{old}[p\ldots p+k\text{-}1]$: a shorthand notation that denotes either deleted token sequence in Delete(k, p), or replaced token sequence in Replace(k, t_{new}, p)

With the symbols above, we can define rules about edit actions. Edits that match the rule are classified into the category the rule represents.

Consider this simple example. The wiki markup contains the opening tag of a reference “`<ref>`”. If we encounter this tag in a chunk of newly inserted text we know that an instance of a reference was added. This edit, which on the level of basic edit actions is simply an Insert, can further be categorized as being of type Reference by following rule:

$$a = \text{Insert}(T_{\text{new}}, p) \wedge \text{type}(T_{\text{new}}[1]) \in (\text{ref-open})$$

(which says the action is an Insert at position p and the first inserted token is an opening tag of a reference).

The rule can examine both type and content of the token, and check every token in the edit. For example, an inter-language link addition can be defined as follows

$$a = \text{Insert}(T_{\text{new}}, p) \wedge \exists\, t_i \in T_{\text{new}}, (\text{type}(t_i) \in (\text{intlink-prefix}) \wedge \text{string}(t_i) \in (\text{InternationalPrefix}))$$

$$\text{where InternationalPrefix} = \{\text{ar, de, en, } \ldots\}$$

This rule checks each token in the inserted token sequence, checks if an internal link prefix token exists, and if it contains a language code that exists in the language list.

A more complicated rule may depend on multiple basic edit actions. Here is a (simplified) definition of *wikify*, a high-level edit action which formats an article by surrounding existing text with wiki markup.

$$\begin{aligned} a_1 = \text{Insert}([t_{\text{n1}}], p_1) \wedge a_2 = \text{Insert}([t_{\text{n2}}], p_2) \wedge p_1 < p_2 \wedge (\\ (\text{type}(t_{\text{n1}}) \in (\text{bold-open}) \wedge \text{type}(t_{\text{n2}}) \in (\text{bold-close}))\\ \vee (\text{type}(t_{\text{n1}}) \in (\text{italic- open}) \wedge \text{type}(t_{\text{n2}}) \in (\text{italic- close}))\\ \vee (\text{type}(t_{\text{n1}}) \in (\text{intlink- open}) \wedge \text{type}(t_{\text{n2}}) \in (\text{intlink- close}))\\ \vee (\text{type}(t_{\text{n1}}) \in (\text{extlink- open}) \wedge \text{type}(t_{\text{n2}}) \in (\text{extlink- close}))\end{aligned}$$

This rule recognizes a pair of markup tags inserted in the existing text, and checks if the pair is of the same markup type.

3.3.2 Additional Comparisons

After an edit action is categorized, we can apply additional comparisons to find out more detail about the edit. For example, a replacement of pure word tokens (i.e. no markup tags) can be further examined by calculating the character edit distance between them.

Table 2. Some edit actions and edit categories

Edit Action	Content Category
(Basic)	Editorial Template
Insertion	Info Template
Deletion	Table
Replacement	Wiki Markup
Movement	Reference
(High Level)	Image
Spelling correction	Inter-language Link
Remove ambiguity	Category
Layout change	Body Text
…	…

$$\begin{aligned} &a = \text{Replace}(k, T_{\text{new}}, p)\\ &\wedge \forall\, t_i \in T_{\text{old}}, \text{type}(t_i) \in (\text{word})\\ &\wedge \forall\, t_i \in T_{\text{new}}, \text{type}(t_i) \in (\text{word})\\ &\wedge \text{distance}(\text{string}(T_{\text{old}}), \text{string}(T_{\text{new}})) < 3\end{aligned}$$

The character edit distance used is the Levenshtein distance [11], which is the minimum number of character edits required to transform one word into another. If this distance is smaller than a certain threshold, for example 3 as in the above example, then the edit can be categorized as a spelling correction.

3.4 History Summarizer

After all the individual edits have been categorized, we can summarize the list of edits. This step includes grouping categories of edits into more abstract groups, such as copy edit, rule enforcement, content addition, etc., and then counting the number of changes and their proportion in the edit for each group. Other statistics that are useful for the user or other applications can also be calculated in this step.

In particular, we are interested in the calculation of edit significance. A calculation method of edit significance can be based on a weighted sum formula. In the previous step, some edit actions were classified into categories of high-level edit actions, whereas others do not belong to any such category. Table 2 shows some examples of edit actions and content categories.

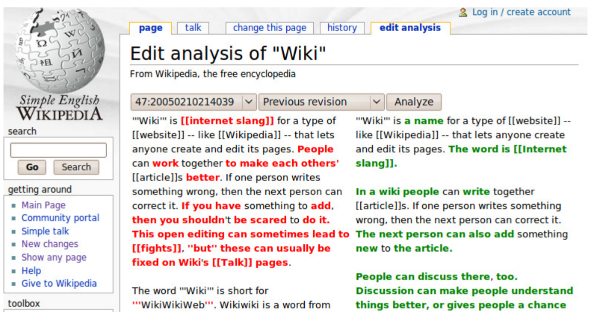
Every edit performed is categorized at the highest possible level and only appears once in the list of edits. That is, if an edit action is classified as a certain basic edit action and later as a high-level edit action, it is only listed as the high-level edit action and not as the basic edit action. For instance, an edit may be categorized as the Replacement basic edit action, whereas this same edit is later recognized to be a Spelling Correction. In this case it is only listed as the latter edit action and not the former. This is in order to avoid repeated occurrence of the same edit.

Once all edits have been categorized to the highest possible level, edit significance can be calculated, according to following weighted sum formula:

$$s = s_{high} + s_{basic}$$

$$s_{high} = \sum_{x=1}^{m} \sum_{i=1}^{n} w_{x,i} c_{x,i}$$

$$s_{basic} = \sum_{i=1}^{n} w_{ins,i} c_{ins,i} + w_{del,i} c_{del,i} + w_{repl,i} c_{repl,i} + w_{mov,i} c_{mov,i}$$



Edit analysis of "Wiki"

From Wikipedia, the free encyclopedia

'''Wiki''' is **[[internet slang]]** for a type of [[website]] -- like [[Wikipedia]] -- that lets anyone create and edit its pages. **People** can **work** together **to make each others'** [[article]]s **better**. If one person writes something wrong, then the next person can correct it. **If you have** something to **add, then you shouldn**'t **be scared** to **do it. This open editing can sometimes lead to [[fights]]**, **''but'' these can usually be fixed on Wiki's [[Talk]] pages**.

The word '''Wiki''' is short for '''WikiWikiWeb'''. Wikiwiki is a word from

'''Wiki''' is **a name** for a type of [[website]] -- like [[Wikipedia]] -- that lets anyone create and edit its pages. **The word is [[Internet slang]].**

In a wiki people can **write** together [[article]]s. If one person writes something wrong, then the next person can correct it. **The next person can also add** something **new** to **the article.**

People can discuss there, too. Discussion can make people understand things better, or gives people a chance

Figure 3. Edit analysis tab extension in MediaWiki

where

- $w_{x,i}$ – weight of content category i on edit action x. ($w_{x,i} \geq 0$)
- $c_{x,i}$ – edited content scale of content category i on edit action x. ($c_{x,i} \geq 0$)

In certain categories, such as body text, the length of the edited text is relevant to the edit significance, whereas in other categories, such as reference, the length is irrelevant but the number of occurrences is. Therefore, $c_{x,i}$ can either represent length or number of occurrences, depending on the category. Since both $w_{x,i}$ and $c_{x,i}$ are unbounded, the significance value has the range $[0, \infty)$.

Weights need to be subjectively assigned according to perceived relative significance of different edit actions and different content categories. We expect that a wiki user community, including its active contributors, would collectively determine these weights to match the group's perception of value. Thus unless standard weights are used, a significance measure is not portable across different communities.

4. PROTOTYPE IMPLEMENTATION

We have implemented a prototype of our analyzer in the Java programming language. It takes two pieces of text as input and produces a list of categorized edits as output. The analyzer is designed to work on MediaWiki texts. Currently our prototype can process the first three steps of our analyzer, while the history summarizer is still under development.

In addition, we have developed a MediaWiki extension in PHP. This extension extracts two versions of an article from the wiki database, calls the above Java program to analyze the revision process, and formats the output to show the differences between versions, categorized by the analyzer. An "edit analysis" tab is added on top of every article in the main namespace (i.e. excluding talk and other pages), which links to the analyzer when clicked. Figure 3 shows an example of the edit analysis page.

To demonstrate our prototype analyzer and MediaWiki extension, we installed them on a MediaWiki server using a database dump of the Simple English Wikipedia[2]. For experimenting with our analyzer on the server, we chose several articles and performed an edit analysis on them. Following is the demonstration of our edit analyzer.

4.1 Differencing

We compare our difference engine output with the one used in MediaWiki. As discussed in Section 2.1, the longest common subsequence (LCS) based algorithms cannot recognize content movements. We also have shown that paragraph-word level differencing used by MediaWiki may produce hard-to-read difference statement. By using a greedy block matching algorithm, and sentence-token level differencing, we expect our algorithm to produce better difference statements in terms of intuitiveness, i.e. in close agreement with what a human evaluator would determine.

Testing both algorithms through several example articles shows that our algorithm meets our expectations. To illustrate this, we tested the differencing on a pair of consecutive versions of the article "Film noir"[3]. MediaWiki shows that there are two paragraphs changed and one new paragraph added (Figure 4), while our algorithm shows there is a sentence moved, a new sentence added, a sentence changed, and a sentence expanded to two sentences in the new version (Figure 5), which appears to reflect the actual edit situation more closely. In particular, because of MediaWiki's paragraph-level differencing, two sentences in the first paragraph are incorrectly marked as deleted, then added into the same relative position. Sentence level differencing avoids this situation, ignoring the boundary between paragraphs.

4.2 Edit Action Categorization

We have defined several categorization rules in our prototype action categorizer. The rules are some frequent edit actions in Wikipedia, such as wikify, inter-language link addition, category modification, reference addition, content modification, spelling correction, etc. Our tests applied the categorizer to several articles in the Simple English Wikipedia. Results show that all defined edit actions can be correctly categorized. Take two versions of the article "Tropical Storm Barry (2007)"[4] as an example (Figure 6). Categorized edits showed that two pieces of text were wikified, one word with its markup removed, two numbers were substituted with the spelled-out variant, a piece of text was added and a piece of text was removed. The derived edit list appears as follows:

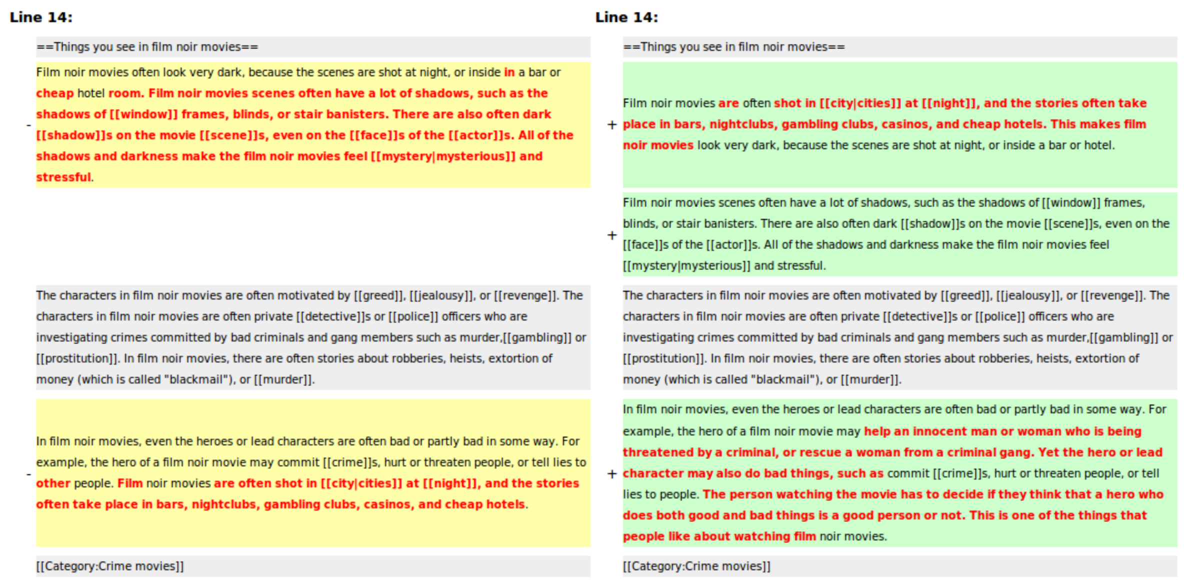
Wikify Ins([`[[miles per hour|`], 206); Ins([`]]`], 212);
Wikify Ins([`[[wiktionary: wildfire|`], 425); Ins([`]]`], 430);
Dewikify Del([`[[`], 352); Del([`]]`], 354);
ContentSubstitution Repl([`3 `], 291, [`three `], 298);
ContentSubstitution Repl([`2 `], 378, [`two `], 383);
ContentRemoval Del([`quickly forming `], 125);
ContentAddition Ins([`that formed quickly`], 129);

Note the first two and the last edit which each involved two spatially separated chunks, the first containing a starting tag and the second the matching closing tag. This text matching and categorization goes far beyond the ability of MediaWiki and other text differencing applications which do not consider the type of the text analyzed and thus only look at words in contiguous

[2] 31st December 2009 database dump, with 57,448 articles.

[3] http://simple.wikipedia.org/wiki/index.php?title=Film+noir&diff=296841&oldid=prev

[4] http://simple.wikipedia.org/w/index.php?title=Tropical_Storm_Barry_%282007%29&diff=next&oldid=1341489


Line 14:

==Things you see in film noir movies==

- Film noir movies often look very dark, because the scenes are shot at night, or inside in a bar or cheap hotel room. Film noir movies scenes often have a lot of shadows, such as the shadows of [[window]] frames, blinds, or stair banisters. There are also often dark [[shadow]]s on the movie [[scene]]s, even on the [[face]]s of the [[actor]]s. All of the shadows and darkness make the film noir movies feel [[mystery|mysterious]] and stressful.

The characters in film noir movies are often motivated by [[greed]], [[jealousy]], or [[revenge]]. The characters in film noir movies are often private [[detective]]s or [[police]] officers who are investigating crimes committed by bad criminals and gang members such as murder,[[gambling]] or [[prostitution]]. In film noir movies, there are often stories about robberies, heists, extortion of money (which is called "blackmail"), or [[murder]].

- In film noir movies, even the heroes or lead characters are often bad or partly bad in some way. For example, the hero of a film noir movie may commit [[crime]]s, hurt or threaten people, or tell lies to other people. Film noir movies are often shot in [[city|cities]] at [[night]], and the stories often take place in bars, nightclubs, gambling clubs, casinos, and cheap hotels.

[[Category:Crime movies]]

Line 14:

==Things you see in film noir movies==

+ Film noir movies are often shot in [[city|cities]] at [[night]], and the stories often take place in bars, nightclubs, gambling clubs, casinos, and cheap hotels. This makes film noir movies look very dark, because the scenes are shot at night, or inside a bar or hotel.

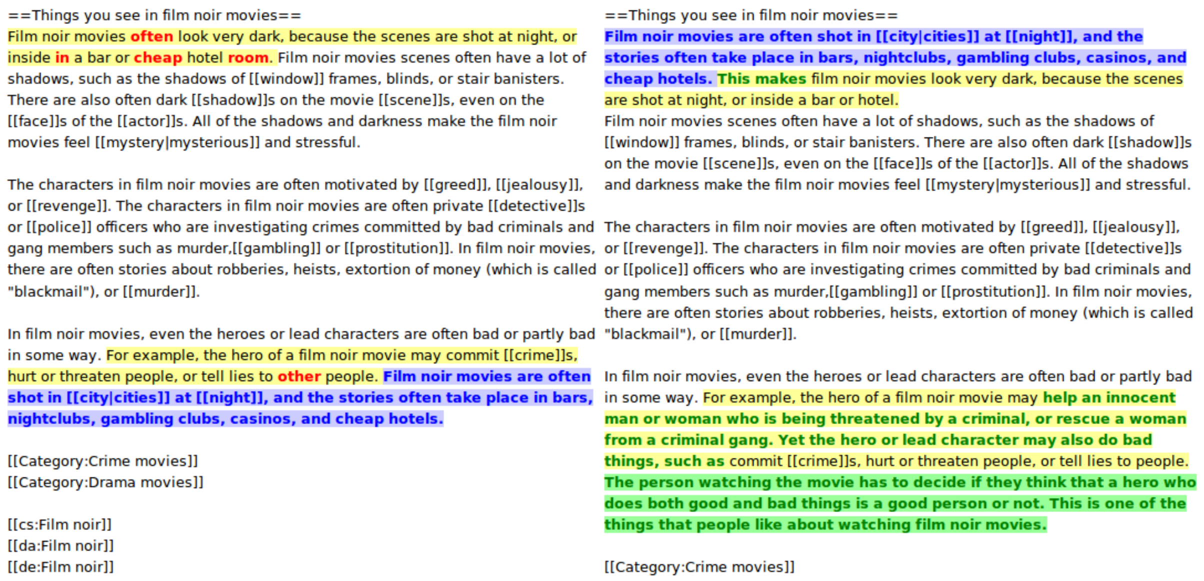
+ Film noir movies scenes often have a lot of shadows, such as the shadows of [[window]] frames, blinds, or stair banisters. There are also often dark [[shadow]]s on the movie [[scene]]s, even on the [[face]]s of the [[actor]]s. All of the shadows and darkness make the film noir movies feel [[mystery|mysterious]] and stressful.

The characters in film noir movies are often motivated by [[greed]], [[jealousy]], or [[revenge]]. The characters in film noir movies are often private [[detective]]s or [[police]] officers who are investigating crimes committed by bad criminals and gang members such as murder,[[gambling]] or [[prostitution]]. In film noir movies, there are often stories about robberies, heists, extortion of money (which is called "blackmail"), or [[murder]].

+ In film noir movies, even the heroes or lead characters are often bad or partly bad in some way. For example, the hero of a film noir movie may help an innocent man or woman who is being threatened by a criminal, or rescue a woman from a criminal gang. Yet the hero or lead character may also do bad things, such as commit [[crime]]s, hurt or threaten people, or tell lies to people. The person watching the movie has to decide if they think that a hero who does both good and bad things is a good person or not. This is one of the things that people like about watching film noir movies.

[[Category:Crime movies]]

Figure 4. Difference page of article "Film noir" produced by MediaWiki analyzer (background colours: yellow/green – old/new version paragraph changed, gray – paragraph unchanged; text colour: red – word changed (deleted, inserted))


==Things you see in film noir movies==
Film noir movies often look very dark, because the scenes are shot at night, or inside in a bar or cheap hotel room. Film noir movies scenes often have a lot of shadows, such as the shadows of [[window]] frames, blinds, or stair banisters. There are also often dark [[shadow]]s on the movie [[scene]]s, even on the [[face]]s of the [[actor]]s. All of the shadows and darkness make the film noir movies feel [[mystery|mysterious]] and stressful.

The characters in film noir movies are often motivated by [[greed]], [[jealousy]], or [[revenge]]. The characters in film noir movies are often private [[detective]]s or [[police]] officers who are investigating crimes committed by bad criminals and gang members such as murder,[[gambling]] or [[prostitution]]. In film noir movies, there are often stories about robberies, heists, extortion of money (which is called "blackmail"), or [[murder]].

In film noir movies, even the heroes or lead characters are often bad or partly bad in some way. For example, the hero of a film noir movie may commit [[crime]]s, hurt or threaten people, or tell lies to other people. Film noir movies are often shot in [[city|cities]] at [[night]], and the stories often take place in bars, nightclubs, gambling clubs, casinos, and cheap hotels.

[[Category:Crime movies]]
[[Category:Drama movies]]

[[cs:Film noir]]
[[da:Film noir]]
[[de:Film noir]]

==Things you see in film noir movies==
Film noir movies are often shot in [[city|cities]] at [[night]], and the stories often take place in bars, nightclubs, gambling clubs, casinos, and cheap hotels. This makes film noir movies look very dark, because the scenes are shot at night, or inside a bar or hotel.
Film noir movies scenes often have a lot of shadows, such as the shadows of [[window]] frames, blinds, or stair banisters. There are also often dark [[shadow]]s on the movie [[scene]]s, even on the [[face]]s of the [[actor]]s. All of the shadows and darkness make the film noir movies feel [[mystery|mysterious]] and stressful.

The characters in film noir movies are often motivated by [[greed]], [[jealousy]], or [[revenge]]. The characters in film noir movies are often private [[detective]]s or [[police]] officers who are investigating crimes committed by bad criminals and gang members such as murder,[[gambling]] or [[prostitution]]. In film noir movies, there are often stories about robberies, heists, extortion of money (which is called "blackmail"), or [[murder]].

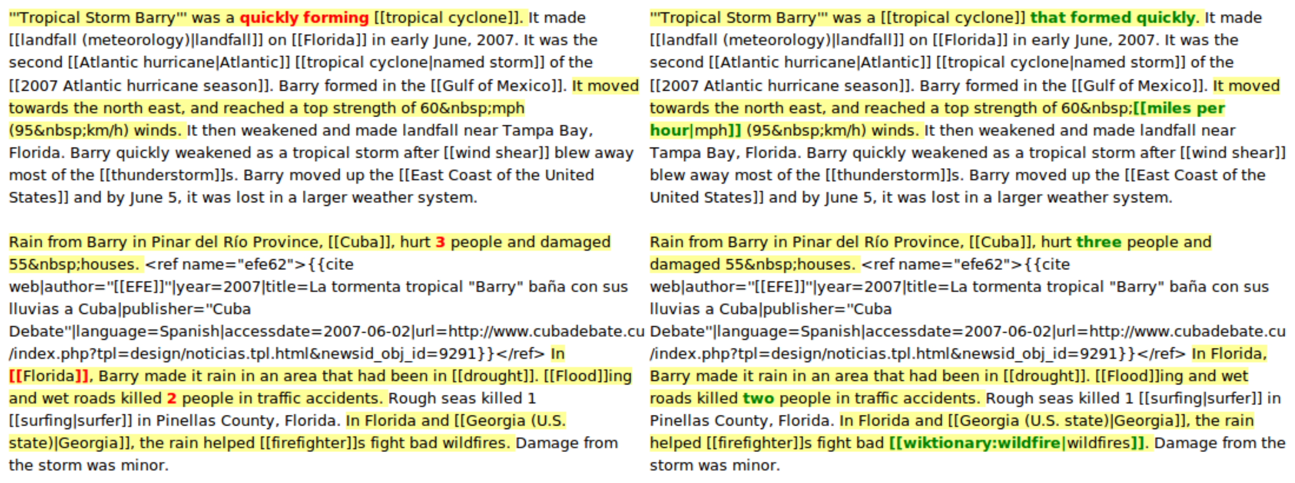
In film noir movies, even the heroes or lead characters are often bad or partly bad in some way. For example, the hero of a film noir movie may help an innocent man or woman who is being threatened by a criminal, or rescue a woman from a criminal gang. Yet the hero or lead character may also do bad things, such as commit [[crime]]s, hurt or threaten people, or tell lies to people. The person watching the movie has to decide if they think that a hero who does both good and bad things is a good person or not. This is one of the things that people like about watching film noir movies.

[[Category:Crime movies]]

Figure 5. Difference page of article "Film noir" produced by our edit history analyzer (background colours: yellow – sentence changed, blue – sentence moved, green – sentence added; text colours: red – word deleted; blue – word moved, green – word inserted)

chunks, not as separated pairs such as matching opening/closing tags. Again, the result obtained closely matches a human evaluation of the revisions performed. Defining a comprehensive set of categorizing rules is still a work in progress. We will perform more testing on various articles once this definition is finished.

Whereas we have not yet performed any formal performance evaluation, running the text differencing and edit categorization for pairs of revisions from the Simple English Wikipedia on a standard desktop PC results in acceptable performance in the range of 0.1 to 2 seconds, depending on the size of the revisions and the amount of changes. For a medium-sized production server it would be desirable to hook the edit categorization to the page save event, thus creating (and saving in the database) edit lists on an ongoing basis. Moreover, it may also be desirable for this to run on a separate analysis server machine rather than the main wiki server machine to prevent an adverse performance impact on wiki users.



'''Tropical Storm Barry''' was a **quickly forming** [[tropical cyclone]]. It made [[landfall (meteorology)|landfall]] on [[Florida]] in early June, 2007. It was the second [[Atlantic hurricane|Atlantic]] [[tropical cyclone|named storm]] of the [[2007 Atlantic hurricane season]]. Barry formed in the [[Gulf of Mexico]]. It moved towards the north east, and reached a top strength of 60 mph (95 km/h) winds. It then weakened and made landfall near Tampa Bay, Florida. Barry quickly weakened as a tropical storm after [[wind shear]] blew away most of the [[thunderstorm]]s. Barry moved up the [[East Coast of the United States]] and by June 5, it was lost in a larger weather system.

Rain from Barry in Pinar del Río Province, [[Cuba]], hurt **3** people and damaged 55 houses. <ref name="efe62">{{cite web|author="[[EFE]]"|year=2007|title=La tormenta tropical "Barry" baña con sus lluvias a Cuba|publisher="Cuba Debate"|language=Spanish|accessdate=2007-06-02|url=http://www.cubadebate.cu/index.php?tpl=design/noticias.tpl.html&newsid_obj_id=9291}}</ref> In **[[Florida]],** Barry made it rain in an area that had been in [[drought]]. [[Flood]]ing and wet roads killed **2** people in traffic accidents. Rough seas killed 1 [[surfing|surfer]] in Pinellas County, Florida. In Florida and [[Georgia (U.S. state)|Georgia]], the rain helped [[firefighter]]s fight bad wildfires. Damage from the storm was minor.

'''Tropical Storm Barry''' was a [[tropical cyclone]] **that formed quickly.** It made [[landfall (meteorology)|landfall]] on [[Florida]] in early June, 2007. It was the second [[Atlantic hurricane|Atlantic]] [[tropical cyclone|named storm]] of the [[2007 Atlantic hurricane season]]. Barry formed in the [[Gulf of Mexico]]. It moved towards the north east, and reached a top strength of 60 **[[miles per hour|mph]]** (95 km/h) winds. It then weakened and made landfall near Tampa Bay, Florida. Barry quickly weakened as a tropical storm after [[wind shear]] blew away most of the [[thunderstorm]]s. Barry moved up the [[East Coast of the United States]] and by June 5, it was lost in a larger weather system.

Rain from Barry in Pinar del Río Province, [[Cuba]], hurt **three** people and damaged 55 houses. <ref name="efe62">{{cite web|author="[[EFE]]"|year=2007|title=La tormenta tropical "Barry" baña con sus lluvias a Cuba|publisher="Cuba Debate"|language=Spanish|accessdate=2007-06-02|url=http://www.cubadebate.cu/index.php?tpl=design/noticias.tpl.html&newsid_obj_id=9291}}</ref> In Florida, Barry made it rain in an area that had been in [[drought]]. [[Flood]]ing and wet roads killed **two** people in traffic accidents. Rough seas killed 1 [[surfing|surfer]] in Pinellas County, Florida. In Florida and [[Georgia (U.S. state)|Georgia]], the rain helped [[firefighter]]s fight bad **[[wiktionary:wildfire|wildfires]].** Damage from the storm was minor.

Figure 6. Difference page of article "Tropical Storm Barry (2007)" produced by our edit history analyzer

5. PRELIMINARY EVALUATION

To evaluate the effectiveness of our prototype system we conducted a small scale evaluation. Ten human evaluators were given a pair of consecutive revisions for each of two Simple English Wikipedia articles, and were asked to manually compare them and point out the changes between them. After the completion of this task they were presented with the edit list corresponding to these pairs of revisions as produced by our program and asked to indicate whether they agreed or disagreed with our program's interpretation of what had changed in the articles. If they disagreed they could state a reason why they thought they disagreed.

Ten articles were selected from the same Simple English Wikipedia database mentioned in the previous section. The articles were selected based on the length of their latest revision, with lower and upper thresholds of 2000 and 41000 characters, respectively, and about equal intervals on a quadratic distribution within this range. For each chosen article, we selected the most recent pair of consecutive revisions for which our program output the longest edit list below an upper limit of 20 edits (to make the human review task reasonably short). The 10 chosen revisions contained 2 to 18 edits.

We invited 10 student volunteers to participate in the evaluation. They were about equally distributed in terms of gender (6 male, 4 female), technical background (4 from computing science majors, 6 from business or humanities majors), and education level (5 undergraduates, 5 postgraduates). Each of them was given two articles to review, a shorter and a longer one. As we had selected 10 articles, each one was evaluated twice by two distinct evaluators. Printouts of both old and new revisions were presented to them (end-user view, not source wiki text). Identical paragraphs in old and new revision had been removed from the article before it was presented, to reduce evaluators' workload. We then asked them to mark and categorize changes between the two revisions manually. After they finished marking, the edit list generated by our prototype was presented to the evaluators. For each item in the list, we asked them if they agreed with the machine categorization or not, and to state the reason of disagreement if any. The whole process took about 30 minutes for each evaluator.

The result of the evaluation shows that our prototype system can produce edit histories that are largely in agreement with human interpretation of changes. 11 out of 20 evaluations agreed 100% to our edit list, and the remaining 9 evaluations ranged from 33.3% to 88.9% agreement. Overall the average agreement rate from all 2 x 10 evaluations was 84.1%. No significant differences were found between the different evaluator groups (male/female, technical/non-technical, undergraduate/postgraduate). The low agreements (33.3% in the case of the article "Nuclear physics") were due to an expectation on the part of the evaluators regarding how a change should be interpreted, specifically expecting our program to be more or less as "clever" as themselves. To illustrate this: in the old revision of this article the sentence fragment "electrons quickly go around the nucleus" became "electrons move around the nucleus very quickly" in the new revision (changed parts underlined). One evaluator expected that replacing the word "go" with "move" and moving the word "quickly" to the end should be considered as a single edit, as these two changes taken together preserve the sentence's meaning. The insertion of the word "very", on the other hand, could be considered a separate edit. However, as our program tries to identify the longest sequence of words in a single edit, and has no understanding of the grammar of the underlying language (here English), it considers "quickly go" as one sequence in the old revision, and "very quickly" as one sequence in the new revision. This particular example led us to add another rule to our differencing algorithm: if the move of a sequence of tokens takes place within a sentence, it will be considered a move regardless of the length of the token sequence (unlike moves crossing sentence boundaries for which a threshold applies, in our case a minimum length of four tokens).

Through this evaluation we collected valuable feedback that we are using to make further adjustments to our prototype. The results, however, confirm to us that our edit lists are on the whole close to how human evaluators interpret changes in the text.

6. CONCLUSIONS AND FUTURE WORK

Research in text differencing algorithms goes back many years. However, when edits are non-trivial, text difference statements produced by these algorithms for a pair of texts can indicate larger and more complicated changes than a human evaluation of those texts would produce. In this paper we have proposed a new text differencing and edit categorization method. By adding a tokenizing step and some tweaks to existing differencing algorithms, we can produce a difference statement closer to human evaluation. This difference statement can be used to classify edit actions into categories, and generate summary statements about the edit. Moreover, our algorithm and method are largely language-independent, and are applicable to any alphabet-based language that uses common sentence-ending tokens (full stop, exclamation mark, question mark etc.) and whitespace to separate words. This mainly excludes East Asian languages (Chinese, Japanese, Korean) and possibly some others.

Our preliminary evaluation on articles from the Simple English Wikipedia shows that our method can produce a better difference statement compared with the MediaWiki differencing engine, and correctly classify each edit into its appropriate category. We see the potential for our method to be applied to a broad range of problems, including automatic summarization, edit classification, edit significance calculation, author contribution calculation, and author interest classification. Using our edit history analyzer to scan through the entire Wikipedia database, interesting observations of user edit patterns could be obtained.

At the time of writing, our history summarizer and edit significance calculation are not fully yet implemented. Upon full implementation we are planning to use categorized edit actions as the basis to refine the edit significance calculation. We will seek feedback from the wiki community to determine suitable weights to be used in our proposed formula, and then perform a comprehensive evaluation on the articles of English Wikipedia.

Our edit significance calculation model could also be applied to our previous work on co-authorship degree calculation [16]. That work only made a very simple determination of edit significance. Using our new edit significance calculation model will allow a more accurate result to be obtained.

Finally, we plan to release our program code as open source in the near future.

7. REFERENCES

[1] Adler, B. T. and de Alfaro, L. 2007. A content-driven reputation system for the Wikipedia. In *Proceedings of the 16th international Conference on World Wide Web* (Banff, Alberta, Canada, May 08 - 12, 2007). ACM, 261-270.

[2] Adler, B.T., de Alfaro, L., Pye, I., and Raman, V. 2008. Measuring Author Contributions to the Wikipedia. In *Proceedings of the 2008 international Symposium on Wikis* (Porto, Portugal, 2008). WikiSym '08. ACM.

[3] Burns, R. and Long, D. 1997. A linear time, constant space differencing algorithm. In *Proceedings of the Performance, Computing, and Communication Conference* (Phoenix, Arizona, USA, Feb. 5-7, 1997). IEEE, 429–436.

[4] de Pedro Puente, X. 2007. New method using Wikis and forums to evaluate individual contributions in cooperative work while promoting experiential learning: results from preliminary experience. In *Proceedings of the 2007 international Symposium on Wikis* (Montreal, Quebec, Canada, October 21 - 25, 2007). ACM, 87-92.

[5] Ebersbach, A. 2008. *Wiki: Web Collaboration*. Springer, 2nd edition.

[6] Ekstrand, M. D. and Riedl, J. T. 2009. rv you're dumb: identifying discarded work in Wiki article history. In *Proceedings of the 5th international Symposium on Wikis and Open Collaboration* (Orlando, Florida, October 25 - 27, 2009). WikiSym '09. ACM, 1-10.

[7] Gorgeon, A. and Swanson, E. B. 2009. Organizing the vision for web 2.0: a study of the evolution of the concept in Wikipedia. In *Proceedings of the 5th international Symposium on Wikis and Open Collaboration* (Orlando, Florida, October 25 - 27, 2009). WikiSym '09. ACM, 1-4.

[8] Hunt, J. W. and McIlroy, M. D. 1976. An Algorithm for Differential File Comparison. *Computing Science Technical Report, Bell Laboratories* 41.

[9] Kittur, A., Chi, E., Pendleton, B. A., Suh, B. and Mytkowicz, T. 2007. Power of the few vs. wisdom of the crowd: Wikipedia and the rise of the bourgeoisie. *Alt.CHI*, 2007, San Jose, CA.

[10] Leuf, B. and Cunningham, W. 2001. *The Wiki Way: Collaboration and Sharing on the Internet*. Addison-Wesley Professional.

[11] Levenshtein, V. I. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady* 10, 707–710.

[12] Myers, E. 1986. An O(ND) Difference Algorithm and Its Variations. *Algorithmica*, 1(2): 251–266.

[13] Neuwirth, C. M., Chandhok, R., Kaufer, D. S., Erion, P., Morris, J., and Miller, D. 1992. Flexible Diff-ing in a collaborative writing system. In *Proceedings of the 1992 ACM Conference on Computer-Supported Cooperative Work* (Toronto, Ontario, Canada, November 01 - 04, 1992). CSCW '92. ACM, 147-154.

[14] Palmer, D. D. and Hearst, M. A. 1997. Adaptive multilingual sentence boundary disambiguation. *Computational Linguistics*.

[15] Sabel, M. 2007. Structuring wiki revision history. In *Proceedings of the 2007 international Symposium on Wikis* (Montreal, Quebec, Canada, October 21 - 25, 2007). WikiSym '07. ACM, New York, NY, 125-130.

[16] Tang, L. V.-S., Biuk-Aghai, R. P., and Fong, S. 2008. A Method for Measuring Co-authorship Relationships in MediaWiki. In *Proceedings of the 2008 international Symposium on Wikis* (Porto, Portugal, 2008). ACM.

[17] Tichy, W. F. 1984. The string-to-string correction problem with block moves. *ACM Trans. Comput. Syst.* 2, 4 (Nov. 1984), 309-321.

[18] Viégas, F. B., Wattenberg, M., and Dave, K. 2004. Studying cooperation and conflict between authors with history flow visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vienna, Austria, April 24 - 29, 2004). CHI '04. ACM, 575-582.