Mutual Evaluation of Editors and Texts for Assessing Quality of Wikipedia Articles

Yu Suzuki
Information Technology Center, Nagoya University
Furo, Chikusa, Nagoya
Aichi 4648601, Japan
suzuki@db.itc.nagoya-u.ac.jp

Masatoshi Yoshikawa
Graduate School of Informatics, Kyoto University
Yoshida Honmachi, Sakyo, Kyoto
Kyoto 6068501, Japan
yoshikawa@i.kyoto-u.ac.jp

ABSTRACT

In this paper, we propose a method to identify good quality Wikipedia articles by mutually evaluating editors and texts. A major approach for assessing article quality is a text survival ratio based approach. In this approach, when a text survives beyond multiple edits, the text is assessed as good quality. This approach assumes that poor quality texts are deleted by editors with high possibility. However, many vandals delete good quality texts frequently, then the survival ratios of good quality texts are improperly decreased by vandals. As a result, many good quality texts are unfairly assessed as poor quality. In our method, we consider editor quality for calculating text quality, and decrease the impacts on text qualities by the vandals who has low quality. Using this improvement, the accuracy of the text quality should be improved. However, an inherent problem of this idea is that the editor qualities are calculated by the text qualities. To solve this problem, we mutually calculate the editor and text qualities until they converge. We did our experimental evaluation, and we confirmed that the proposed method could accurately assess the text qualities.

Categories and Subject Descriptors

H.1.2 [**Models and Principles**]: User/Machine Systems

Keywords

Wikipedia, Quality, Peer Review, Edit History, Link Analysis

1. INTRODUCTION

Wikipedia[1] is one of the most successful and well-known User Generated Content (UGC) websites. Any user can edit any article, Wikipedia has more and fresher information than existing paper-based encyclopedias. Many experts submit texts to Wikipedia, and the texts should be informative for readers. However, these texts are not reviewed by experts, then the number of poor quality texts are also dramatically increase. On the other hand, many readers cannot easily identify texts which are good quality or not, because not all readers are experts. Therefore, there is a need for automatically identifying which texts in Wikipedia are good quality or not.

In this paper, we propose a method for identifying good quality texts from edit history. The definition of quality has many aspects such as credibility, expertise, and correctness, then measuring excellence is difficult task. In this paper, we use survival ratio based approach for calculating text qualities, which is one of the major approaches for measuring text qualities [1]. We measure the number of times which editors decide the text should remain, which is a key idea of survival based approach. When many readers feel excellent for a text, the quality of this text is good, but when many readers feel that a text should remove, the quality of this text is poor. Adler et al. [2] investigated that 79% of poor quality texts are short-lived. From this result, if editors find poor quality texts, many editors remove them. This means that if a text survives beyond multiple edits by the other editors, the text should be good quality. Therefore, using the *survival ratio* of texts, the system calculates the text quality.

Example 1: Let us consider a motivating example. One editor e_a writes a text $p(e_a)$. Then, another editor e_b edits another text $p(e_b)$, but keeps $p(e_a)$ intact. In this case, we assume s/he judged $p(e_a)$ to be good quality because e_b remains $p(e_a)$ as it is. Next, another editor e_c deletes $p(e_a)$. We assume that e_c judged $p(e_a)$ to be poor quality, hence s/he deleted the text. As a result, $p(e_a)$ is confirmed by e_b, but not confirmed by e_c. If e_c had not delete $p(e_a)$, the quality in this case would have been higher than that in the former case, because the paragraph $p(e_a)$ is trusted by one editor in the former case whereas it was trusted by two editors in the latter case. In this case, the *survival ratio* of $p(e_a)$ is 1 when e_b edits, and 0 when e_c edits. Therefore, the overall survival ratio of $p(e_a)$ is 0.5.

In this method, they assume that the quality of article becomes good according to the number of edits, because they assume all editors delete only poor quality texts. However, this assumption is not always true because of edits by vandals. Vandals delete not only poor quality texts but also good quality texts. If vandals delete a text, the survival ratio of the text is overly decreased. In Example 1, if e_c was a vandal, survival ratio of e_a's text should not be decreased.

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

To solve this problem, we need to detect which editors are vandals and which are not, and re-adjust survival ratios of texts in accordance with the editor qualities.

In this paper, we propose a method for mutually calculating text qualities using both survival ratios of texts and editor qualities. We assume that vandals rarely submit good quality texts, whereas good quality editors frequently submit good quality texts. We define an editor quality as the average text qualities written by the editor. However, text qualities are calculated on the basis of editor qualities. In short, one quality is calculated by another quality. Therefore, it is hard to calculate the text qualities using editor qualities. To solve this problem, we first set editor qualities as constant values and calculate text qualities. Next, we calculate the editor qualities by using text qualities. Again, we calculate the text qualities using the editor qualities. In this way, we mutually calculate editor and text qualities. Using this method, we can calculate a text quality that takes into consideration that of its editor qualities.

In our method, we use both edit history analysis techniques, such as Adler et al. [1–3], Hu et al. [4], and Wilkinson et al. [5], and link analysis techniques, such as HITS [6], PageRank [7], and SALSA [8]. In our system, we can draw a bipartite graph using sets of editors and texts as nodes. Therefore, using link analysis techniques, we can calculate editor and text importance [9].

We implemented our proposed system and baseline system as a Web application, and evaluated the accuracy of text qualities. We implement the method which use only text survival ratio to the baseline system. We evaluated our proposed system using the Japanese version of Wikipedia's edit history data. From the results, we found that when we pick up 100 articles using baseline method and our proposed system, our proposed system can identify 78 good quality articles while the baseline system can identify only 61 good quality articles.

The rest of this paper is organized as follows. First, in section 2, we summarize related works about measuring quality of Wikipedia, which use explicit and implicit features. In section 3, we describe how to measure the text, editors, and version qualities. In section 4, we discuss the evaluation results. Finally, in section 5, we close with conclusions and future work.

2. RELATED WORK

There has been much research in calculating quality degrees of products, people, and objects using reputation based method [10,11]. A key concept for evaluating Wikipedia articles is *the peer review process*. Wikipedia is not thought to have a peer review system because most texts are instantly made and saved, though no one reviews these texts. However, Stivila et al. [10] mentioned that the open edit system is a kind of peer review system where editors of the system vote on implicit features of the texts.

In these investigations, many features are extracted from Wikipedia data in many studies, and they can be divided into two types: explicit and implicit features. Explicit features are the user's decision which are directly input to the system by users, and implicit features are the user's decision which the system presumes from their behaviors. In this section, we describe the studies that have used explicit and implicit features and also describe why we choose to use implicit features.

2.1 Explicit Features

Explicit features are commonly used to evaluate quality of information, products, and objects. For example, many online shopping sites like Amazon.com[2] have voting systems for users to evaluate products. When users want to evaluate how satisfied or not they are with a product they have bought, they give the product 1-5 stars and submit review texts. Then, the system presents the average number of stars along with the reviews. If the other users want to know the quality or the satisfaction of the products, they refer to the number of stars and reviews, and decide whether to buy it or not. This system has been implemented as a part of many online Web services, such as YouTube[3] and Google+[4], because it is easy to implement and the process of calculating the number of stars is easy and clear.

Kramer [12] implemented the voting system on MediaWiki[5] for educational use, and also implemented at the English version of Wikipedia as Article Feedback Tool [6]. Using these systems, users easily understand which are good quality articles by referencing these votes. However, one problem with this system is that not every user always appropriately evaluates or reviews. In fact, according to statistics about YouTube, almost all users who vote give the highest score to almost all videos they votes on [13]. Moreover, the survey of the Article Feedback Tool by Wikipedia[7] shows that 90.9% rates are the highest score. From this statistics, we find that users rate only good targets, but they do not rate poor targets.

The advantage of this system is that users can directly evaluate quality of targets. However, the disadvantage is that only a small number of users input negative ratings. Therefore, if there is a target with a small number of voting, we cannot identify the target as either the poor quality target or the non-reviewed target. One reason of this problem is a lack of negative ratings, which is hard to recover by analysis of ratings. Therefore, we do not use explicit features.

2.2 Implicit Features

Implicit features are the user's decisions which the system presumes from their behaviors. When the system uses these features, users do not need to input the evaluation of items. Our proposed method uses this method. However, how can users' evaluations be presumed from their behavior?

Lifecycles of texts are used for calculating qualities of texts or articles. Wöhner et al. [14] calculate Wikipedia article qualities using the lifecycle of texts. In this method, they discovered that the quality and lifecycle of a text have a relationship. Halfaker et al. [17] presume implicit features from contribution degrees for editors. In this method, they proposed six heuristic assumptions about why editors contribute to Wikipedia. These ideas are appropriate when the articles are frequently edited. However, the frequency of edits are different, then the lifecycle of a text is different for every article. In addition, when edit warring occur, this

[2]http://www.amazon.com/
[3]http://www.youtube.com/
[4]https://plus.google.com/
[5]http://www.mediawiki.org/
[6]http://en.wikipedia.org/wiki/Wikipedia:Article\%20Feedback\%20Tool
[7]http://www.mediawiki.org/wiki/Article_feedback/Survey

method cannot calculate appropriate qualities. Our method can calculate appropriate qualities if articles are not only infrequently edited but also suffering an edit war because we consider editor qualities.

Adler et al. [1–3], Hu et al. [4], and Wilkinson et al. [5] proposed a method for calculating quality values from edit histories. This method is based on survival ratios of texts. However, they did not consider editor qualities. Thus, if vandals delete articles frequently, the qualities of deleted texts decrease, and so the system cannot calculate appropriate text qualities. In addition, this method cannot calculate the quality values for new, up-to-date texts. In our method, we use editor quality values as well as text qualities. Therefore, we can calculate qualities for new texts by using the editor qualities.

Adler et al. described that reputation systems can be classified into two categories: *chronological*, where a reputation is computed from the chronological sequence, and *fixpoint*, where a reputation is computed via a fixpoint calculation. Their algorithms are classified into *chronological* algorithms, because they aim to develop realtime quality calculation method. However, our proposed algorithm is classified into *fixpoint* algorithm, because the aim of our proposed method is to calculate accurate text qualities, even if the method is computationally heavyweight. However, this computational cost is not an important problem, because our method can be easily implemented to distributed computing frameworks such as Hadoop[8].

3. PROPOSED METHOD

Our goal is to assess qualities of articles by mutually evaluating text and editor qualities. First, in section 3.1, we describe the key idea. Next, in section 3.2, we define several notations that are used throughout this paper. Then, in section 3.3, we describe how to calculate quality values. In section 3.4, we describe why our proposed system is resistant to edit war.

3.1 Key Idea

The goal of this study is to calculate article qualities using survival ratio based approach. However, using this approach, good quality articles attacked by vandals are identified as poor quality, because when the vandals edit articles, they may delete good quality texts and add poor quality texts. Moreover, if the vandals try to reduce qualities of the other editors, the vandals delete texts written by those editors, because the survival ratios of the texts can easily be reduced by the vandals. In short, the disadvantage of this method is its weakness to vandalism. To solve this problem, we must detect which editors are vandals or which are not.

Our key idea is that we adjust the text qualities by using the editor qualities. We believe that survival ratio of text is an important factor, but a quality of editor who deletes a text is also an important factor for calculating text quality. We assume that vandals rarely write good quality texts, so their text qualities should be low. Therefore, if an editor deletes a text and has a low quality value, we adjust the decreased survival ratio of this text so that it increases, because this deletion should be considered inappropriate. By using our proposed method, the accuracy of text qualities should improve.

[8]`http://hadoop.apache.org`

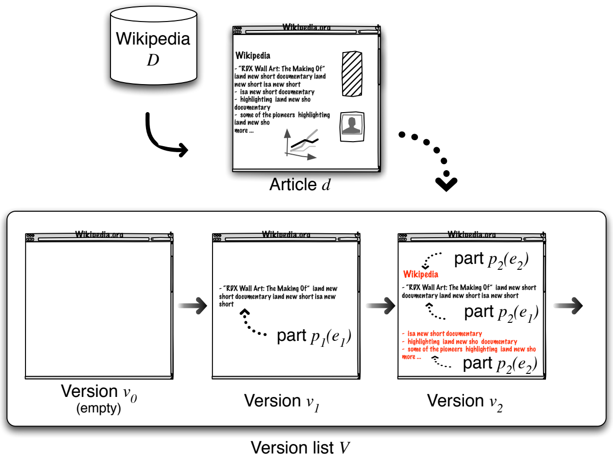


Figure 1: Notations of this paper.

3.2 Modeling

In this section, we define notations that are used throughout this paper as shown in Figure 1. On Wikipedia, every article has a version list $V = \{v_i | i = 0, 1, \cdots, N\}$ where i is the version number, and v_N is the latest version. We denote that if $i = 0$, v_0 is a version with empty contents and no editor. When editor e in all Wikipedia editors E creates a new article, the system automatically makes two versions, v_0 and v_1, and then the system stores the text of editor e in v_1 which consist of one text $p(e)$. We identify editors using editor names or IP address for anonymous editor. Then, we define version $v_i = \{p(e)|e \in E\}$ as a set of complete texts that is stored at i-th edit and that consists of a text by $1, 2, \cdots, i$-th editors. $p(e)$ is a text added by editor e. If e deletes all texts from i-th version, v_i is an empty set.

Editor e creates a set of texts $P(e) = \{p(e)\}$ where $p(e)$ is a text created by e for all articles in Wikipedia. When editors edit one article by same user more than twice consecutively, the system keeps the last version and deletes the other versions created by the user. That is, the editor of a version and that of next version are always different.

The aim of our proposed method is calculating text quality $\tau(d, e)$ of text $p(e)$. To accomplish our mission, we should calculate converged text quality $\tau_K(d, e)$ on article d by editor e, and converted editor quality $u'_K(e)$ of editor e. $\tau_0(d, e)$ is an initial text quality, and $u'_0(e)$ is an initial editor quality. K is a number of process of $\tau_k(d, e)$ and $u'_k(e)$ converges. In step 6. at section 3.3, we mutually calculating k-th text quality $\tau_k(d, e)$ and k-th editor quality $u'_k(e)$ until converge. $\tau_K(d, e)$ and $u'_K(e)$ is the converged text and editor quality, respectively.

3.3 Calculation Method of Text and Editor Quality

Figure 2 shows the overview of calculating text qualities. Our proposed system consists of the following seven steps.

1. Extract articles from Wikipedia edit history, and identify texts and their editors from edit history. (Section 3.3.1)

2. Calculate initial text quality using survival ratios of texts. (Section 3.3.2)

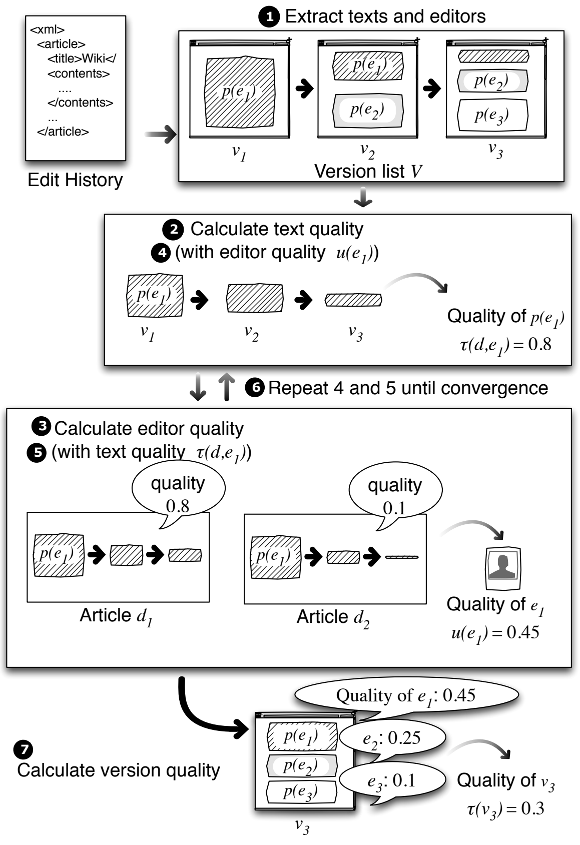


Figure 2: Overview of our proposed method.

3. Calculate initial editor qualities using text qualities. (Section 3.3.3)

4. Calculate adjusted text qualities using both editor and text qualities. (Section 3.3.4)

5. Calculate editor qualities using text qualities. (Section 3.3.5)

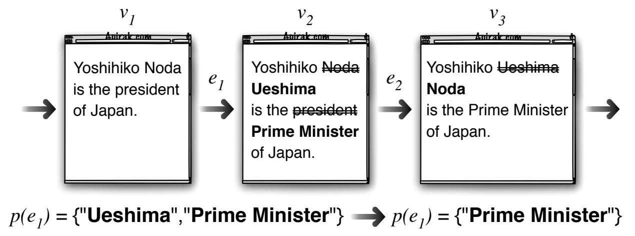
6. Repeat processes 4. and 5. until the text quality converges.

7. Calculate version qualities using converged version qualities. (Section 3.3.6)

3.3.1 *Extract Texts and Editors from Edit History*

First, we extract all articles from the Wikipedia edit history, and identify which editor edited which texts. Edit history stores the extract title, editor's name, and a snapshot of the article for every version. We extract these data, and store them in a database system. At this time, we identify the editors of the texts using diffs. The texts that editors have added are the texts that differ between the current and previous versions. When a text is not in the previous version but is in the current version, the text must have been written by the editor of the current version. Using this policy, we identify the editor of every text.

Identifying the editors of the texts is important and difficult issue. In our system, we use a letter based method which are similar to the difference page by MediaWiki. Fong et al. proposed an intuitive method [18], but this method



v_1 v_2 v_3
Yoshihiko Noda is the president of Japan.
e_1
Yoshihiko ~~Noda~~ **Ueshima** is the ~~president~~ **Prime Minister** of Japan.
e_2
Yoshihiko ~~Ueshima~~ **Noda** is the Prime Minister of Japan.
$p(e_1)$ = {"**Ueshima**","**Prime Minister**"}
$p(e_1)$ = {"**Prime Minister**"}

Figure 3: Example of edit history (Example 2.)

cannot be easily adopted to our system. Because in East Asian languages (Chinese, Japanese, Korean), each term is not separated by space character.

When we extract versions, we should consider reverted versions, which are the versions that have been changed back into their previous versions. In this case, when we simply use this policy, we identify the editor of the current (reverted) version who wrote the text that differs between the current and previous versions. However, if this reversion is during an edit war, the survival ratio of the text decreases, leading the text qualities to decrease. To solve this problem, we identify the editors of a reverted version to be the editors of the original version, and the editors who revert the articles back to their previous versions are treated as neither adding nor deleting anything. Using this policy towards reversions, the text qualities are not affected by vandalism or inappropriate edit warring. In section 3.4, we discuss why we use this method.

3.3.2 *Initial Text Quality*

Next, we calculate initial texts' quality $\tau_0(d,e)$ in article d and editor e. We defined the text quality as a survival ratio beyond multiple edits, which is similar to Adler's definition of quality. When editors edit articles, they evaluate and remove any inappropriate or poor quality texts. Therefore, if texts survive beyond multiple edits, they should be considered good quality. Using this policy, we calculate text qualities.

One important policy is that editors do not evaluate themselves. When editors add texts, the texts are not evaluated at that time. We use this policy of self evaluation, because, if we permit editors to evaluate themselves, vandals can easily increase survival ratios of texts they edit, which easily increases qualities of the vandals.

Example 2: We explain this policy using a specific example. Figure 3 shows the example of an edit history from 1st version to 3rd version. Using this example, we explain how to calculate text quality $p(e_1)$ that are added by editor e_1 in version v_1. First, we identify the texts that are added in version v_1. In this example, the editor e adds the texts "Ueshima" and "Prime Minister" as $p(e_1)$ to version v_2. The number of letters of $p(e_1)$ is 21 including spaces. Next, we focus on the survival ratio of $p(e_1)$. The editor e_2 of v_3 deletes the text "Ueshima" but keeps the text "Prime Minister". In other words, e_2 only accepts 14 letters. Therefore, the number of letters of $p(e_1)$ is 14. From these edits, editor e_1 obtains the quality $\tau_0(d,e_1) = \log_2 7 + \log_2 14 - \log_2 7 \simeq 3.7$ from this article.

When we calculate text quality, we use the number of letters in a log scale instead of the raw number of letters,

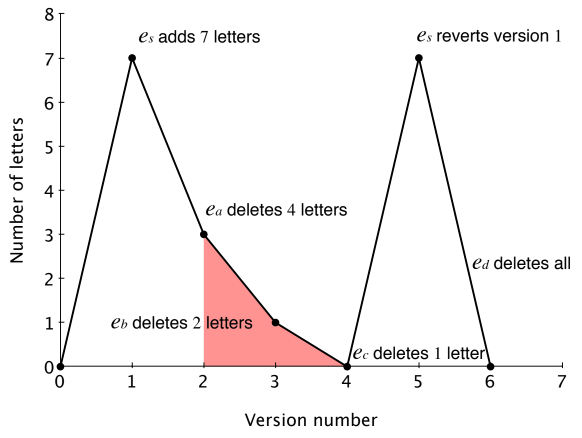


Figure 4: Example of edit history (Example 3.)

because we face a problem when the editor adds long texts. If an editor adds 10,000 letters to the article, and the texts survive only one edit, this text quality value is 10,000, which is the same quality as a 100-letter texts that survives beyond 100 edits. We think that the latter text is higher quality than the former text. Therefore, we count the number of letters using a log scale.

Here, we define the text quality $\tau_0(d,e)$ in article d by editor e as follows:

$$\tau_0(d,e) = \sum_{p(e)\in\bar{P}} \log_2(|p(e)|+1) \tag{1}$$

where $\bar{P}$ is a set of texts which is not on the version edited by e, and $|p(e)|$ is the number of letters in $p(e)$. This equation means the summation of the number of letters on texts that are written by e. We remove the number of letters on the version edited by e himself/herself because of the policy of non-self-evaluation.

We give an example below to clarify how to calculate text quality.

Example 3: We explain more complex example than example 2. Figure 4 shows an example of edit history for editor e_s. First, editor e_s writes 7 letters. Then, e_a deletes 4 letters and leaves 3 letters. At this time, e_a confirms that the remaining 3 letters are high quality, so e_s obtains the quality $\log_2(3+1)=2$. We should note that e_s does not obtain the quality $\log_2(7+1)$ from the first 7 letters, because this text is not confirmed by the other editors. Next, e_b leaves 1 letters, so e_s obtains $\log_2(1+1)=1$ as a quality from e_b. e_c deletes all texts by e_s, so e_s obtains no quality from e_c. e_s reverts the article from version v_4 to v_1 as version v_5. As we already mentioned in section 3.3.1, if e_s reverts the article, the reverted text belongs to e_s. In other words, this case is the same as that in which e_s writes 7 letters. However, as we mentioned above, e_s does not evaluate himself/herself. Therefore, e_s does not obtain the quality $\log_2(7+1)=3$. Finally, e_d deletes all texts by e_s, so e_s does not obtain a quality. In short, e_s leaves 7, 3, 1, 7 letters each to the article, so e_s obtains $log_2 8+log_2 4+log_2 2+log_2 8=9$. However, first and last 7 letters are not confirmed to be correct by the other editors, so we should reduce $log_2 8+log_2 8$ from e_s's quality. As a result, e_s obtains 3 as the quality from this edit history. If e_d not e_s reverts texts of e_s at version v_5, e_d can confirm e_s's texts at version v_5. In this case, e_s obtains 6 as the quality.

In our method, we measure the survival ratio using number of edits, we do not use length of survival time. This is because, Adler et al. [1] discovered by experiments that using number of edits makes better accuracy than using length of survival time.

3.3.3 Initial Editor Quality

We calculate editor quality using the text quality calculated at section 3.3.2. We define the initial editor quality $u_0(e)$ of editor e as follows:

$$u_0(e) = \frac{\sum_{d\in D(e)} \tau_0(d,e)}{|D(e)|} \tag{2}$$

where $D(e)$ is a set of Wikipedia articles that e edits, and $|D(e)|$ is the number of articles in $D(e)$. If we calculate $u_0(e)$, we remove texts of articles that are created for specific purposes, such as notes, rules of Wikipedia, editors' private articles, and so on. This is because editors mainly write these texts to express their opinions and do not always delete them. Therefore, the qualities of these texts tend to be higher than those of general articles.

We normalize $u_0(e)$ to range between 0 and 1 as follows:

$$u'_0(e) = \frac{u_0(e) - \min_{e'\in E} u_0(e')}{\max_{e'\in E} u_0(e') - \min_{e'\in E} u_0(e')} \tag{3}$$

3.3.4 Text Quality

Next, we calculate the text quality using editor quality. This phase is derived from initial text quality calculation method written in section 3.3.2. In this phase, we integrate the survival ratio of texts and those of the editors who delete them using weighted summation, whereas the initial quality calculation method only use the survival ratio of texts.

We calculate the text quality $\tau_k(d,e)$ as follows:

$$\begin{aligned}\tau_k(d,e) =& \alpha\cdot\tau_0(d,e) \\ &+ (1-\alpha)\cdot\sum_{e'\in E'} |\delta(e')|\cdot(1-u'_{k-1}(e'))\}\end{aligned} \tag{4}$$

where E' is a set of editors who delete $p(e)$, $\delta(e')$ is the letters in $p(e)$ deleted by e', $|\delta(e')|$ is the number of letters in $\delta(e')$, and α $(0\le\alpha\le 1)$ is the parameter to control the effect of editor quality. The first part of expression means the initial text quality which is the same as section 3.3.2, and the second part means the number of deleted letters with qualities of editors who delete them. If an editor who has a poor quality deletes a text $p(e)$, then the value of the second part is low, then the value of $\tau_k(d,e)$ is almost the same as $\tau_0(d,e)$. Therefore, if editor quality is low, the editor quality does not affect the text quality. In this case, if the editor who deletes the text has a good quality, the second expression has a high value. Thus, the value of $\tau_k(d,e)$ decreases more than $\tau_{k-1}(d,e)$.

Example 4: We explain how to adjust text quality using editor quality. Figure 5 shows the example of an edit history that is a part of the edit history in Figure 4. In this example, e_s adds 7 letters at version 1, e_a deletes 4 letters at version 2, and e_b deletes 2 letters at version 3.

We adjust the number of letters in version 2. In this version, e_a deletes 4 letters, and α is set to 0.5, so we assume

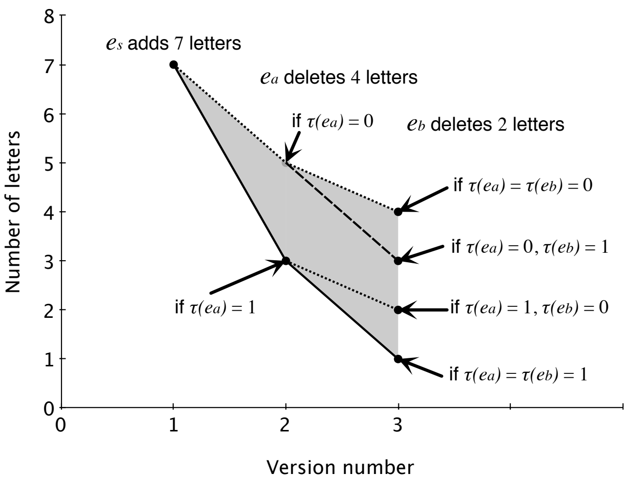
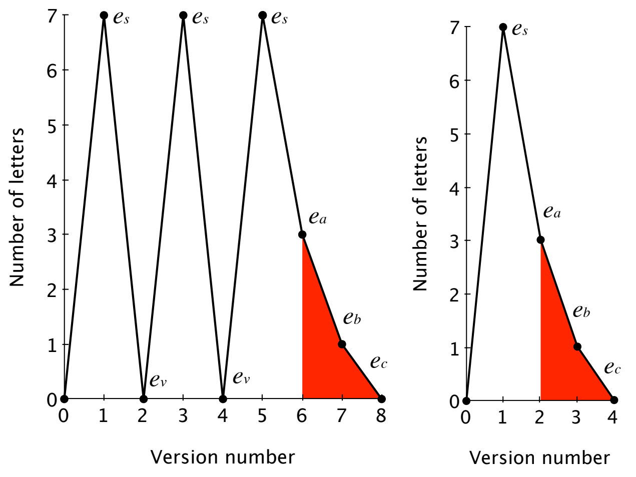


Figure 5: Example of edit history (Example 4.)



(a) Edit warring (b) Without edit warring

Figure 6: Example of edit history (Example 5.)

e_a deletes $2(= 4 \times 0.5)$ letters. We decide whether the other 2 letters should be deleted or not using the quality of e_a. If $\tau_{k-1}(d, e_a) = 0$, e_a is a poor quality editor, so it is possible that e_a deletes appropriate text. In this case, we assume 2 letters are left, not deleted by e_a. On the other hand, if $\tau_{k-1}(d, e_a) = 1$, e_a is a good quality editor, e_a should delete inappropriate text. Then, we assume 4 letters are deleted by e_a.

Next, we adjust the number of letters in version 3. In this version, e_b deletes 2 letters. Then, we assume e_b deletes 1 $(= 2 \times 0.5)$ letter. We decide whether another 1 letter should be deleted or not in the same way. If $\tau_{k-1}(d, e_b) = 0$, we assume 1 letter is left, but if $\tau_{k-1}(d, e_b) = 1$, 1 letter is deleted. As a result, if both $\tau_{k-1}(d, e_a)$ and $\tau_{k-1}(d, e_b)$ are 0 and if both e_a and e_b are low quality, the number of letters by e_s is 7 in version 1, 5 $(= 7 - 2)$ in version 2, and 4 $(= 7 - 2 - 1)$ in version 3.

3.3.5 Editor Quality using adjusted Text Quality

Using adjusted text quality $\tau_k(d, e)$, we define the editor qualities $u_k(e)$ of e as follows:

$$u_k(e) = \frac{\sum_{d \in D(e)} \tau_k(d, e)}{|D(e)|} \tag{5}$$

This equation is almost the same as the equation (2) described in section 3.3.3.

We normalize $u_k(e)$ to range between 0 and 1 as follows:

$$u'_k(e) = \frac{u_k(e) - \min_{e' \in E} u_k(e')}{\max_{e' \in E} u_k(e') - \min_{e' \in E} u_k(e')} \tag{6}$$

We repeat the processes in sections 3.3.4 and 3.3.5 until the values of $\tau_k(d, e)$ and $u_k(e)$ converge.

Here we discuss the possibility of convergence of $\tau_k(d, e)$ using intuitive example. Figure 5 shows the number of letters by e_s, and $\tau_k(d, e_s)$ is defined by sum of number of letters with log scale. We do not consider the number of letters of the first version. The number of letters of the second value is between 3 and 5 when α is set to 0.5, which is a finite number, if editor e_a is a high quality editor or not. In practice, this value is between 0 and 7 if α is set to any number. In the same way, the number of letters of the third version is between 0 and 7, which is at most 7. Moreover, the value of $\tau_k(d, e)$ is always in the grey area of figure 5 if editor quality is any value. Therefore, in this case, $\tau_k(d, e)$ is always converge because this value is at most the number of letters times the number of the remained version.

In our experiment 1. at section 4.2, we confirm that text and editor qualities converge.

3.3.6 Qualities of Versions

Using the converged value of $u'_K(e)$, we define the version quality $T(v_i)$ of version v_i as follows:

$$T(v_i) = \frac{\sum_{e \in P(e)} u'_K(e) \cdot |p(e)|}{|v_i|} \tag{7}$$

where $|v_i|$ is the number of letters in v_i, $|p(e)|$ is the number of letters in $p(e)$, and $u'_K(e)$ is the editor quality of e. This function means that the version quality is the weighted averaging value of text quality, and the weight is the number of letters in the text.

3.4 Edit War

Our proposed method is resistant to vandalism. To confirm that the qualities of editors are not increased or decreased by vandalism, we discuss an example.

Example 5: We consider two cases: edit history (a) with edit warring and (b) without edit warring. Figure 6 shows the example of an edit history. In case (a), e_s writes 7 letters, and then a vandal e_v deletes all text of e_s. e_s reverts to version 1, but e_v deletes all text of e_s again. e_s reverts to version 1 again. Then, e_a deletes 4 letters, e_b deletes 2 letters, and e_c deletes 1 letter. Case (b) is the same edit history without the vandal e_v. First, e_s writes 7 letters, then e_a, e_b, and e_c delete 4, 2, and 1 letters, respectively.

From these cases, we calculate the text quality by e_s using equation (1) described in section 3.3.2. In case (a), e_s leaves 7, 0, 7, 0, 7, 3, 1, and 0 letters at from version 1 to 8, but version 1, 3, and 5 are not verified by the other editors.

Then, text quality by e_s is $\log_2(3+1)+\log_2(1+1)+\log_2(0+1) = 3$. In case (b), e_s leaves 7, 3, 1, and 0 letters at from version 1 to 4, but version 1 is not verified by the other editors. Then, text quality by e_s is $\log_2(3+1)+\log_2(1+1)+\log_2(0+1) = 3$, which is the same value as case (a). In short, vandals do not affect the qualities.

In general, the behavior of vandals, such as inappropriately adding and deleting good quality texts, is not permitted by the other editors, so many non-vandal editors try to counter the behavior of vandals. In our proposed system, if the behavior of an editor is permitted by the other editors, the quality of the editor increase. As a result, vandals do not affect the qualities of the other editors.

4. EXPERIMENTS

To determine the accuracy of the article quality calculated by our proposed system, we did two experimental evaluations. In these evaluations, we tried to confirm that when we use the editor quality to calculate the text quality, the accuracy of the text quality should improve. However, we cannot identify which text is good quality or not, because the unit of text is too small. Therefore, we evaluate article quality, which is a latest version quality calculated at section 3.3.6.

We compared three systems: (*baseline*) our proposed system without using editor quality, (*once*) our proposed system using editor qualities at once, and (*proposed*) our proposed system using both converged editor and text qualities. When we calculate version quality $T(v_i)$, in *baseline*, we use steps 1, 2, and 7 from section 3.3, then we set all editors' quality $u'_K(e)$ to 1 as a constant value because we do not execute step 3, so editor qualities are not defined. We set *baseline* as almost equal to Adler et al.'s system [1] to compare our work with related work. In *once*, we use all steps except step 6. In *proposed*, we use all seven steps. We created three article lists, which were ordered by the newest versions of the qualities of these three systems. These lists are query-independent order.

In experiment 1, we set the answer set of "featured" and "good" articles as a correct answer set. Featured and good articles are selected by the votes of Wikipedia users. These articles are evaluated by "Featured article criteria"[9] and peer reviewed by many active users. If vandals nominate poor quality articles for featured or good articles, the nomination is rejected by administrators. Therefore, we believe that these articles are good quality, so we could use featured and good articles as good quality articles for the test set.

In this experiment, we compared three systems using relative precision ratios. In this evaluation method, we compare the answer set with the list of articles in ascending order of their qualities. If articles in the answer set are ranked higher, we will be able to confirm that the system calculates accurate qualities. The key in this evaluation is the appropriateness of answer sets. In current information system retrieval evaluation, observers create answer sets by judging relevance of articles. However, judging the quality of articles is difficult, so we cannot confirm the appropriateness of quality judgments of articles. Therefore, we put only featured and good articles in the answer set.

Experiment 1 was objective because general Wikipedia editors created the answer set. However, this experiment was insufficient, because many good quality articles were not in the answer set because of the strict criteria articles must meet to be deemed featured or good. Therefore, we did one more manual evaluation: experiment 2 confirmed that our proposed method did not judge poor quality articles to be of good quality.

In both experiment 1 and 2, we set $\alpha = 0.8$ as a parameter of equation (4). Before these experiments, we set α from 0.1 to 1 in 0.1 increments and calculate averaging precision ratio as preliminary experiment. In this result, when we set 0.2, we got the highest averaging precision ratio of our proposed system.

4.1 Data Sets

In these experiments, we used the Japanese version of the Wikipedia edit history dumped on November 2, 2010, which can be downloaded at the Wikipedia dump download site[10]. These data include $1,889,129$ articles and $24,054,128$ versions. The number of editors is $2,178,003$ including not registered editors who are identified by IP addresses, and bots which are listed[11]. This edit history is compressed by bzip2 and is about 20.1GB. From these articles, we removed the articles that does not contain link to Wikipedia articles. We also removed the articles for specific purposes, such as redirect pages, notes and rules of Wikipedia. We referred to Wikipedia statistics[12] to decide this definition. As a result, the number of articles is $713,002$.

4.2 Experiment 1: Evaluation using Featured and Good Article

In this experiment, we did the evaluation using a relative precision ratio per each recall level. Relative precision ratio P is P_t, a number of correct articles selected by the target system, divided by P_b, the number of correct articles selected by the baseline system. Relative precision ratio P is defined as follows:

$$P = \frac{P_t}{P_b} \tag{8}$$

When P is larger than 1, the target system is more accurate than the baseline system. We set the baseline system as *baseline* and the target system as *once* and *proposed*. Therefore, when we draw a relative recall-precision graph, we first draw a general interpolated 11-pt recall precision graph [19]. Then, we calculate the relative precision ratio P for each recall level. Finally, we draw a relative precision ratio for each recall level.

As we already mentioned in section 4, we set the answer set of articles as featured and good articles from Japanese Wikipedia. Since there were 87 featured articles and 499 good articles, we selected 586 articles for the answer set. Next, we calculated articles' qualities using our proposed method and the baseline method and listed articles in descending order of quality values. Finally, we calculated the relative precision ratio for each recall level using the article list and the answer set.

[9] `http://en.wikipedia.org/wiki/Wikipedia:Featured_article_criteria`

[10] `http://dumps.wikimedia.org/jawiki/20101102/`

[11] `http://ja.wikipedia.org/wiki/WP:BOTST`

[12] Wikipedia: What is an article?: `http://en.wikipedia.org/wiki/Wikipedia:What_is_an_article`

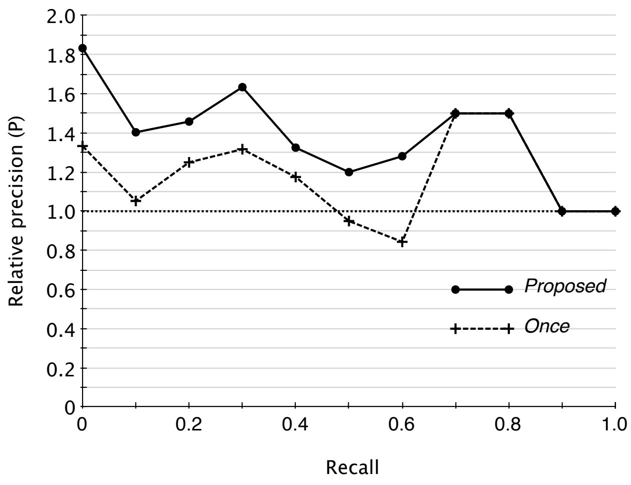


Figure 7: 11-pt relative recall-precision graph (Experiment 1.)

Figure 7 shows a relative precision ratio per each recall level of *once* and *proposed* in comparison with *baseline*. From this graph, we discovered that *proposed*, our proposed method calculates article qualities more accurately than *baseline*, the baseline method. We also confirmed that when we use editor qualities for multiple times until convergence, the relative precision ratio increases.

In this experiment, we also confirmed that both text and editor qualities were converged when we calculate qualities 18 times. We did not observe diverged and oscillate values. However, if we use the other language version of Wikipedia as a dataset, these qualities may diverge or oscillate. This is because, when linked graph of editors and texts are separated to multiple graphs, the values may not converge. In our experiment, we do not face this problem. However, if we face this problem, we should develop a method to integrate multiple graphs into single graph.

In the details of experimental results, we found that our proposed method is effective if vandals attack the articles and cause an edit war, which involves many inappropriate additions and deletions. In the results of *proposed*, there are 36% of articles attacked by vandals in the top 100 positions, whereas there are 0% of articles in *baseline*, and 28% of articles in *once*. When we count articles attacked by vandals, we use list of "Wikipedia: most vandalized pages"[13]. When edit wars happen, vandals delete texts even if they are good quality. Generally, vandals do not indiscriminately delete texts; they delete the texts of specific editors whose opinions they oppose. The articles about religion and politics especially face this kind of edit warring. As a result, vandals decrease qualities of texts by good quality editors. Using our proposed method, the quality values of texts by good quality editors increase, and the quality values of versions that face vandalism increase. On the other hand, the qualities of versions that do not face vandalism neither increase nor decrease. This is not a problem for our proposed system, because when the articles do not face vandalism, the system can calculate appropriate qualities. This means that our proposed method is effective for the articles that face vandalism.

At recall level from 0.5 to 0.6, relative precision ratio of *once* is lower than 1, which means that accuracy of *once* is lower than *baseline*. This is because of the editors who edit a small number of articles. Even editors who have high qualities do not always submit good quality texts. Therefore, if there is a good quality text that has survived beyond multiple edits, but the editor's quality is low, the text is considered low quality by *once*. However, generally the editor's low quality is caused by vandalism. Therefore, when we calculate editor and text qualities, we can recover this problem, so relative precision ratio of *once* is improved and is higher than 1.

At a recall level from 0.7 to 0.8, relative precision ratios of *once* and *proposed* are almost the same, because *baseline* cannot find good quality articles at this level. *baseline* can find good quality articles when the articles have edit histories long enough for the text qualities to be calculated. On the other hand, both *once* and *proposed* can find good quality articles with short edit histories, because these systems calculate qualities of articles using qualities of editors. Editors generally edit multiple articles, so if a good quality article has a short edit history and if editors obtain high qualities from the other articles, *baseline* calculates a low quality for the article whereas both *once* and *proposed* calculate a high quality.

In these experimental results, we also found that the average precision ratio of *proposed* is low, about 0.035. This means that when we select the articles with the top 100 qualities, we found that only about 3.5 articles are featured or good, which is an extremely small number when compared with a general information retrieval method. However, in our results, we think that our proposed system outputs appropriate qualities, because many good quality articles are not selected as featured and good articles. Therefore, we did experiment 2. In this experiment, we manually decided whether articles were high quality or not. Moreover, we also discuss why the average precision ratio of experiment 1 is low.

4.3 Experiment 2: Manual Evaluation

In this section, to confirm that our proposed method did not judge poor quality articles to be of good quality, we measure the accuracy of article qualities using the answer set that was manually selected. In this experiment, we created three article lists, which were ordered by the qualities of *baseline*, *once*, and *proposed*, and pick up the top 100 articles from each list. Then, the three observers, graduate school students who do not know about these systems, manually decided whether the articles were good quality or not by hand. The observers evaluated 193 articles with no duplicates. Then, when the three observers decided an article as good quality article, we decided as good quality article. When evaluate articles, the system shuffled these three lists, so the observer did not know which lists the observer evaluates. We decided that an article was good quality only if all its texts were high quality. If it had at least one poor quality text, the observers decided the article was poor quality. The observers also decide very short articles as poor quality.

Figure 8 shows the precision ratio at each rank. From this figure, we found that 78 articles from *proposed*, 69 articles

[13]Wikipedia: most vandalized pages (in Japanese) : `http://ja.wikipedia.org/wiki/Wikipedia:%E8%8D%92%E3%82%89%E3%81%95%E3%82%8C%E3%82%84%E3%81%99%E3%81%84%E3%83%9A%E3%83%BC%E3%82%B8%0D`

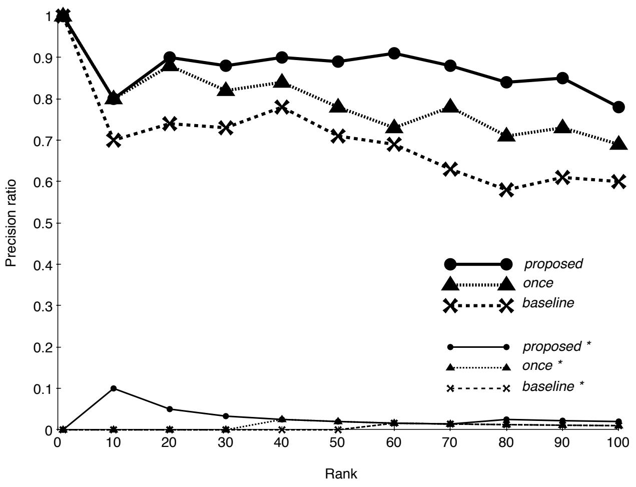


Figure 8: Precision ratio of each rank. *proposed, once, baseline* **are the results using manual evaluation, and** *proposed *, once *, baseline ** **are the results using featured and good articles.**

from *once* and 61 articles from *baseline* were good quality articles. We determined that our proposed method could calculate more appropriate article qualities than the baseline method.

We should discuss the correctness of experiment 1 using this result. In Figure 8, we also show the precision ratios of the three methods when we use only featured and good articles as good quality. From this result, we found only two articles from *proposed* and one article from *once* and *baseline* as good quality from top 100 articles. These articles are also decided as good quality articles by manual evaluation. From this result, the number of good quality articles at experiment 2 is 40 times larger than them at experiment 1. Then, when we manually evaluate all articles, the precision ratio of experiment 1 should increase.

The articles consisting of lists or data (such as the articles listing all a TV program's episode list, F1 driver list, and famous people list born in 1900) are ranked higher than the other articles. This is because, these articles are very long, and many editors may not read articles from top to bottom, then many texts tend to survive beyond long edits. However, we believe that there are higher quality articles on Wikipedia. Therefore, we should categorize articles into several types, and develop an appropriate calculation method for qualities for typical types of articles.

5. CONCLUSION

Wikipedia is one of the most popular and highest quality encyclopedias created by many editors. The information on Wikipedia keeps expanding, but its quality is not proportional to its quantity. In this paper, we have presented a method to identify good quality articles by mutually evaluating qualities of editors and texts.

In this paper, we introduced a combination of a survival ratio method and a link analysis method. There are many vandals in Wikipedia, and many vandals attack Wikipedia by deleting good quality texts. In our proposed method, editor qualities affect text qualities instead of using only text survival ratio. Therefore, when the vandals delete good quality texts, they do not affect the survival ratio of the texts, because the editor qualities of the vandals are low value. As a result, the text qualities which are attacked by vandals do not decrease. Using this method, we can calculate accurate text qualities using editor qualities.

Our proposed approach's strongest point is the resistance to vandalism. In experiment 1, 36% of all good quality articles attacked by vandals are identified as good quality articles using our proposed method, but the baseline system identify all good quality articles as poor quality. Moreover, using manual evaluation in experiment 2, our proposed system could find 78 good quality articles, whereas naive survival ratio based system could find 61 good quality articles. From these results, we confirmed that our proposed system could calculate accurate quality using editor qualities.

Quality of information is becoming increasingly important in information retrieval research field. An information retrieval system retrieves the documents that are relevant to the user's query, but the system is not concerned about whether the documents are good quality or not. However, if the retrieved documents are poor quality, they should not be retrieved even if they are relevant. Therefore, as Toms et al. [20] already mentioned, when an information retrieval systems and a document quality measurement system are integrated, we will develop an information retrieval system more accurate than current information retrieval systems.

Finally, we describe five open problems:

Vagueness of quality: In this paper, we calculated editor qualities in sections 3.3.3 and 3.3.5. However, this editor quality is not always distinct because the frequency of editing is different for each editor. We suppose that if an editor rarely edits Wikipedia articles, the editor may just happen to obtain a high quality. However, the vagueness of the editor quality should be high, because there is less evidence to calculate the editor quality for this editor. Therefore, we should develop a method to calculate both editor quality and the vagueness degree of editor quality.

Use of natural language processing techniques: In our proposed method, we do not analyze linguistic structures; we only count the number of letters in texts. A strong point of our proposed system is that it can adapt to different language versions of Wikipedia articles. However, a weak point is that it cannot use important features that come from linguistic features. In our experiment, we found that good quality articles are always written in formal language. Moreover, Sabel et al. [21] said that text analysis is useful for calculating qualities. For example, if an editor changes "A is a B." to "A is not a B." the number of letters changes by only three, but the meanings of these sentences are completely different. Therefore, we should use natural language analysis techniques for calculating survival ratio of texts.

User interface and visualization: We developed a Web-based user interface. In this user interface, all users use the same Web pages as a result. However, we believe that undemanding and demanding users will want to browse different Web pages [22]. For example, if a text were mostly low quality, the system would determine it to be low quality for demanding users but high quality for undemanding users. Holloway et al. [23] and Otjacques et al. [24] have already discussed the user interface of Wikipedia before. Therefore, we should develop a user interface that is useful for every user.

Scalability: We implemented our system on a single PC with four CPUs and a 24 GB memory. As a result, this system took about 10 days to analyze all articles on the Japanese version of Wikipedia. Therefore, to analyze the English version of Wikipedia, we will need more than 100 days because the edit history of English Wikipedia is 10 times larger than that of Japanese Wikipedia. However, the algorithm of our proposed method is similar to existing HITS and PageRank algorithms. These algorithms can easily be adopted to map/reduce frameworks. Therefore, if we use multiple cluster PCs and implement our methods using these frameworks, we will reduce calculation time of article analysis. We will therefore work on developing systems that are more scalable.

6. ACKNOWLEDGMENTS

This work was partially supported by Japan Society for the Promotion of Science, Grants-in-Aid for Scientific Research (20300036, 23700113).

7. REFERENCES

[1] B. T. Adler, K. Chatterjee, L. de Alfaro, M. Faella, I. Pye, and V. Raman. Assigning Trust to Wikipedia Content. In *Proceedings of the International Symposium on Wikis (WikiSym '08)*. ACM, 2008.

[2] B.T. Adler and L. de Alfaro. A content-driven reputation system for the Wikipedia. In *Proceedings of the 16th international conference on World Wide Web (WWW '07)*, pages 261–270, 2007.

[3] B.T. Adler, K. Chatterjee, L. de Alfaro, M. Faella, I. Pye, and V. Raman. Measuting Author Contributions to the Wikipedia. In *Proceedings of the 2008 International Symposium on Wikis (WikiSym '08)*, 2008.

[4] M. Hu, E. Lim, A. Sun, H. W. Lauw, and B. Vuong. Measuring Article Quality in Wikipedia: Models and Evaluation. In *Proceedings of ACM International Conference on Information and Knowledge Management (CIKM 2007)*, pages 243–252, 2007.

[5] D. M. Wilkinson and B. A. Huberman. Cooperation and quality in wikipedia. In *Proceedings of the 2007 international symposium on Wikis (WikiSym '07)*, pages 157–164. ACM, 2007.

[6] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46:604–632, 1999.

[7] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the International Conference on World Wide Web (WWW'97)*, 1997.

[8] R. Lempel and S. Moran. SALSA: the stochastic approach for link-structure analysis. *ACM Trans. Inf. Syst.*, 19(2):131–160, April 2001.

[9] Monika Henzinger. Link analysis in web information retrieval. *IEEE DATA ENGINEERING BULLETIN*, 23:3–8, 2000.

[10] B. Stvilia, M. Twidale, L. Smith, and L. Gasser. Information quality work organization in wikipedia. *J. Am. Soc. Inf. Sci. Technol.*, 59(6):983–1001, 2008.

[11] B. Stvilia, L. Gasser, M. B. Twidale, and L. C. Smith. A Framework for Information Quality Assessment. *Journal of the American Society for Information Science and Technology*, 58(12):1720–1733, 2007.

[12] M. Kramer, A. Gregorowicz, and B. Iyer. Wiki trust metrics based on phrasal analysis. In *Proceedings of International Symposium on Wikis (WikiSym '08)*, 2008.

[13] M. G. Siegler. Youtube comes to a 5-star realization: Its ratings are useless: http://www.techcrunch.com/2009/09/22/youtube-comes-to-a-5-star-realization-its-ratings-are-useless/, September 2009.

[14] T. Wöhner and R. Peters. Assessing the quality of Wikipedia articles with lifecycle based metrics. In *Proceedings of the International Symposium on Wikis and Open Collaboration (WikiSym '09)*, 2009.

[15] Reid Priedhorsky, Jilin Chen, Shyong (Tony) K. Lam, Katherine Panciera, Loren Terveen, and John Riedl. Creating, destroying, and restoring value in wikipedia. In *Proceedings of the 2007 international ACM conference on Supporting group work*, GROUP '07, pages 259–268, New York, NY, USA, 2007. ACM.

[16] Katherine Panciera, Aaron Halfaker, and Loren Terveen. Wikipedians are born, not made: a study of power editors on wikipedia. In *Proceedings of the ACM 2009 international conference on Supporting group work*, GROUP '09, pages 51–60, New York, NY, USA, 2009. ACM.

[17] A. Halfaker, A. Kittur, R. Kraut, and J. Riedl. A Jury of Your peers: Quality, Experience and Ownership in Wikipedia. In *Proceedings of the International Symposium on Wikis and Open Collaboration (WikiSym '09)*, pages 1–10, 2009.

[18] Peter Kin-Fong Fong and Robert P. Biuk-Aghai. What did they do? deriving high-level edit histories in wikis. In *Proceedings of the 6th International Symposium on Wikis and Open Collaboration*, WikiSym '10, pages 2:1–2:10, New York, NY, USA, 2010. ACM.

[19] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval: the concepts and technology behind search*. Addison-Wesley, 2011.

[20] E. G. Toms, T. Mackenzie, C. Jordan, and S. Hall. wikiSearch: enabling interactivity in search. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2009)*, page 843, 2009.

[21] M. Sabel. Structuring Wiki revision history. In *Proceedings of the 2007 international symposium on Wikis (WikiSym '07)*, pages 125–130. ACM, 2007.

[22] Marti A. Hearst. *Search User Interfaces*. Cambridge University Press, 2009.

[23] T. Holloway, M. Bozicevic, and K. Börner. Analyzing and visualizing the semantic coverage of Wikipedia and its authors. *Complexity*, 12(3):30–40, 2007.

[24] B. Otjacques, M. Cornil, and F. Feltz. Visualizing Cooperative Activities with Ellimaps: The Case of Wikipedia. In Y. Luo, editor, *Cooperative Design, Visualization, and Engineering (CDVE '09)*, volume 5738 of *Lecture Notes in Computer Science*, pages 44–51. Springer, 2009.