Wikipedia revision graph extraction based on n-gram cover

呉　建民† 　岩井原瑞穂‡

†早稲田大学大学院情報生産システム研究科

〒808-0135 福岡県北九州市若松区ひびきの 1-15

E-mail: †jianmin.wu@moegi.waseda.jp, ‡iwaihara@waseda.jp

Abstract During the past decade, mass collaboration systems have emerged and thrived on the World-Wide Web, with numerous user contents generated. As one of such systems, Wikipedia allows users to add and edit articles in this encyclopedic knowledge base and piles of revisions have been contributed. Wikipedia maintains a linear record of edit history with timestamp for each article, which includes precious information on how each article has evolved. However, meaningful revision evolution features like branching and revert are implicit and needed to be reconstructed. Also, existence of merges from multiple ancestors indicates that the edit history shall be modeled as a directed acyclic graph. To address the issue, we propose a revision graph extraction method based on n-gram distribution covering that effectively find branching and revert. We evaluate the accuracy of our method by comparing with manually constructed revision graphs.

Keyword Wikipedia Revision graph, Mass collaboration

1. Introduction

During the past decade, online mass collaboration systems have emerged and thrived on the World-Wide Web. This form of collective action involves large numbers of contributor and numerous user contents are generated. Contributors are often organized under certain projects but works independently. Typical examples include Wikipedia, Linux, Yahoo! Answers, Mechanical Turk-based systems [1].

Wikipedia, one of the most successful systems, is a free, multilingual Internet encyclopedia written collaboratively by volunteers around the world [2]. Users can edit almost every article based on his knowledge, which makes Wikipedia continually changing and evolving. For each article, Wikipedia simply keeps both the current version and all the past versions and this edit history is publicly available. Other useful information including timestamps, contributor, and edit comments are also recorded.

While most contributors edit an article based on the latest revision, they sometimes might have different point of views and argue against others' work, which causes forks in the evolution process. If we review all revisions of an article, we could find many meaningful things from it. However, unlike what is very common in software development, Wikipedia does not maintain an explicit revision control system that manages the detailed change of revisions and revisions are organized by date and time in descending order. In this paper, we intend to extract articles' evolution process called revision graph from the linear edit history.

As shown in Figure 1.1, a revision graph is a DAG (directed acyclic graph) in which each node represents one revision with directed edges indicating their reference relationship. Regarding the fact that users edit articles based on the current revision and sometimes past revisions, each node should have at least one reference source and together they form such hierarchical structure. We need to identify the source of each revision in given collection of text with high similarity. Currently few efforts appeared on this issue.

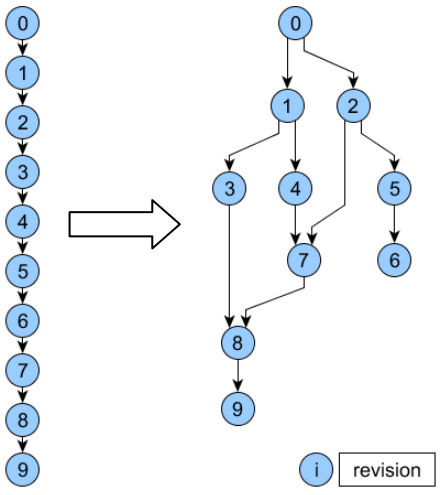

Figure 1.1 Example of revision graph

To address the challenges mentioned above, in this paper, we propose an approach on revision graph extraction based on n-gram distribution coverage. More specifically:

- We use n-gram distribution to denote revisions of the given articles with timestamps and find how a revision's n-gram distribution can be formed by specific previous revisions'.
- We utilize smallest n-gram diff score to decrease the complexity in the n-gram coverage process.

This paper is organized as follows. In Section 2 we introduce the background of our research. Section 3 describes basic definitions regarding our problem, and explains algorithms used to extract a revision graph. In Section 4 we show experimental evaluation of our method and the results. Finally, concluding remarks and future works are discussed in Section 5.

2. Background

2.1. Mass collaboration system and Wikipedia

A mass collaboration system enlists a crowd of users to collaborate to build a long-lasting artifact that is beneficial to the whole community [1]. Different kind of mass collaboration systems diverse in many ways such as the way they recruit and retain users, the way to combine user contributions to solve the target problem. But they are all lack of a framework to trace and evaluate contributors' work systematically.

Wikipedia, launched in January 2001, poses more than 20 million articles by now [2]. Wikipedia employs an open, collaborative editing model where contributors can edit articles using wiki markup to format the text and add other elements like images and tables, and Wikipedia would keep articles with their modifications as edit history with additional information stored, such as timestamps, edit comments and contributors' information [3]. Articles can be improved immediately.

The success of Wikipedia also leads to the popular of wiki. Many websites provide a wiki engine for users to contribute their content via a web browser to form a Wikipedia-like knowledge base, which suggests that our research can be applied in more cases.

2.2. Related work

Detecting highly similar documents in a large collection is a common topic in computer-assisted plagiarism detection [7]. Many documents that are published on the Internet are copies or plagiarisms of other documents. Since a plagiarism may not be identical to the original document, using conventional search techniques it can be difficult to distinguish plagiarized documents from those that are simply on the same topic. Existing techniques that can be used to address these problems include fingerprinting, a technique developed specially for detecting duplicates [4]. But all these works are based on a large collection of not similar documents, unlike Wikipedia's edit history.

Cao et al. [5] proposed a version tree reconstruction method for Wikipedia articles based on keyword clustering. This method uses tf-idf (term frequency and inverted document frequency) score to cluster similar revision and largest common subsequence for more precise comparison, which is closer to string matching problem. However, the proposed method fails to keep consistency as the revision number grows. Another main drawback is lack of detecting the trace of merge, as each revision in their tree has only one parent, for the consideration that merging is not frequently occurring in practice.

3. Revision graph extraction method

3.1. Revision graph extraction

In most cases, contributors edit a Wikipedia article based on the current revision. If everyone follows this practice, the edit history should appear as a linear sequence of revisions. However, branches occur when contributors have different opinions on editing. For example, a contributor might restore the article to the previous revision if he thinks the content added in the current revision is inappropriate. Considering the nature of editing behavior, we define the reference relationship as a relationship of which a pair of revision that the latter revision, the child revision, is edited based on the former one, the parent revision, in reality. After all the reference relationships are found, a revision graph that represents the revision evolution process can be extracted.

While considering the revision graph extraction in Wikipedia, these assumptions about edition behaviors should be held:

1. A contributor edits an article based on the existing revisions.
2. A revision can be totally identical as one of its previous revisions.

The main task of our method for revision graph extraction is to identify the reference sources of each

revision. An intuitive solution is to check each previous revision for verbatim text overlaps and evaluate their edit distance in order to find the most similar one, if we treat the past of the current revision as a reference collection. This can be utilized by many approximate string matching methods. However, this solution suffers from expensive computational complexity of *O(mn)*, where m and n are the total numbers of the characters in the comparing revisions, not to mention that we have to perform this pairwise comparison $N(N\text{-}1)/2$ times for N revisions. Another shortcoming has been found in a series of cases that it could lead to a wrong result when identifying relationships among a small set of revisions by edit distance.

In our method, we adopt a hybrid model of bag-of-word analysis and n-gram model for revision comparison. Revisions can be treated as a collection of words with different occurrence so that we can compare revisions by their word frequency distribution with low computational complexity.

Notice that traditional bag-of-word analysis loses all the sequence order of words, which does harm to reflect the syntax difference between revisions like rephrasing. Here we introduce n-gram model in order to keep the string order partially. Instead of using words directly, we construct the frequency distribution for all the n-grams in a revision, where *n* is fixed during the entire comparison process. The larger *n* we use, the more precise comparison can be achieved by longer word sequences.

We compare revisions by their n-gram distributions. Given the fact the revision documents in the collection of a single article's edit history are highly similar, it is more necessary to distinguish slight differences among them than to evaluate their similarity. We develop n-gram diff to represent the difference between revisions. It is a special n-gram distribution of all the different part, while differentiating the contribution of edit behaviors like add, modify and remove. We also develop a measure to quantify the difference. If the difference between two revisions is small enough, we can find their n-gram distributions overlapped, which is like covering an n-gram distribution by the other.

To find a set of revisions whose n-gram distribution can cover the n-gram distribution of the current revision essentially equals to the set cover problem, which was proved to be NP-hard. In our research, we can avoid the disadvantage based on the previous assumptions of edition behaviors. We treat the main reference source as base revision. So all we need is find the base revision.

3.2. Definitions

Here are the formal definitions that our method is based on.

N-gram

A (word-level) *n-gram* is a contiguous sequence of *n* words from the revision text. The text of a revision is also treated as an entire word sequence, while Wikipedia Markup Language symbols is dismissed. In our method, the "*n*" starts from 2.

N-gram distribution

For a revision R, the *n-gram distribution* of R is the frequency distribution of all n-grams appearing in R. Each entry in the distribution contains an n-gram and its frequency in R.

N-gram diff

Given two n-gram distributions with the same n, ngd_1, ngd_2, the *n-gram diff* of them is a special n-gram distribution consisting of their different parts. More specifically, we check each n-gram in two distributions whether it is unique in only one distribution, or exists in both distributions but with different frequency, and then calculate this difference. In order to indicate the unique n-gram, we use "+/-" before the n-gram respectively. So a typical n-gram diff is shown as follows:

Bigram	Frequency	Description
aaa bbb	12	more in ngd_1
abb baa	-5	less in ngd_1
+aab bba	2	unique in ngd_1
-bbb aaa	6	unique in ngd_2

Table 3.1 N-gram diff example: bigram

N-gram diff score

Let R1 and R2 be revisions. The n-gram diff score of R1 and R2, denoted by $DS_n(R1, R2)$, is

$$DS_n(R1,R2) = \sum_{t\in G} |f(t,R1) - f(t,R2)|,$$

Where $G = \text{ngram}(n,R1) \cup \text{ngram}(n,R2)$, $\text{ngram}(n,R)$ is the set of n-gram of revision R, and $f(t, R)$ is the frequency of n-gram t in R.

We develop this measure to evaluate the degree how two n-gram distributions are different. N-grams from base distribution that appear in the collection are excluded

Unique n-grams

The *unique n-grams* UQ of a revision R is the set of all the n-grams in R that never appear in revisions earlier than R. The unique n-grams cannot be covered by any previous

revision.

N-gram cover

Given a collection D of revisions not later than revision R, an *n-gram cover* of R is the set $S \subseteq D$ of revisions such that:

$$\text{ngram}(n, R) \subseteq UQ \cup \bigcup_{R_i \in S} \text{ngram}(n, R_i)$$

where UQ is the unique n-grams of R.

Revision graph

A *revision graph* is a directed graph that shows the reference relationship among revisions. We denote all the revisions with nodes, which are indexed with revision ids. An edge from revision R_i to revision R_j is constructed when R_i is one of the reference sources, or a parent, of R_j. Here, reference sources must be direct, in the sense that there is no intermediate revision R_j' which was created from R_i and R_j was created from R_j'. As an old revision would never referee to a newer revision, there would not exist a path from descendant node to parent node. So this graph is also a directed acyclic graph.

3.3. Algorithm

Basically, our algorithm performs in two stages, (1) base source detection by minimum n-gram diff score and (2) n-gram cover computing. The value of n is fixed during the whole process, so all the n-gram distribution can be compared by the same scale. Results that are generated with different value of n would be treated separately.

① From the second oldest revision to the latest revision, compare every revision with all of its previous revision and calculate their n-gram distribution diff.

② For all the n-gram diffs of a revision, choose the lowest *k* n-gram diff scores as candidates.

③ From the corresponding revisions of candidates, select the latest revision in time order.

4. Experimental Evaluation

In this section, we show experimental evaluation of our method by DAG reachability validation for a collection of Wikipedia articles' revision graphs.

4.1. Data set

We collect articles from Wikipedia randomly to construct a test data set. Articles are filtered by the following criteria:

- English language
- Having text volume more than one page
- Having 200+ revisions

Article Title	Total # of revisions
Racism	10,896
2006 Israel–Gaza conflict	2,456
PhpBB	1,312
Edith Wharton	1,114
Federal republic	717
Sarkar Raj	592
Grade inflation	456
Natal chart	346
Muhammad Naguib	283
Clarinet Concerto	256

Table 4.1 Test Wikipedia articles

In order to keep our selection non-biased, we use the function of "Random Article" provided by Wikipedia and 10 articles are collected. We retrieve the first 200 revisions of each article from the official export page [4] and save all these 2000 revisions as XML files.

For the purpose of building ground truth, we manually review the reference relationship underlying these revisions based on human experience. After reviewing the collected revisions, 10 revision graphs are extracted for corresponding articles.

Article Title	# of Branches	Avg. run-length
Racism	23	4.26
2006 Israel–Gaza conflict	12	8.00
PhpBB	37	2.67
Edith Wharton	16	6.06
Federal republic	33	2.99
Sarkar Raj	15	6.45
Grade inflation	24	4.08
Natal chart	11	8.70
Muhammad Naguib	8	11.76
Clarinet Concerto	12	8.00

Table 4. 2 Ground Truth Statistics

Table 4.2 shows the total number of branches manually discovered, and average running length of linear paths (i.e. avg. run-length) in the graphs. It can be surmised that a controversial topics can cause more branches, which leads to shorter average run-length.

4.2. Data Processing

4.2.1. Content extraction

First we use the open source XML parser Dom4j [8] to extract the required revisions from the original Wikipedia edit history file. Both revision text contents and edition

information are exported, including timestamps, contributors and comments.

The exported revisions are indexed by descending time order, starting from 0. We also remove punctuations and wiki markup language operators.

4.2.2. n-gram distribution generation

For each cleaned revision, we split the cleaned text of the revision into words. Then we concatenate these unigrams into n-gram as their original order in revision text, while the "n" is on demand.

We count the occurrence for the entire n-grams for each revision R and then construct a hash map and each entry maps an n-gram to its occurrence in R. Maps are built by hash in order to make the time complexity of insert and search in a constant time.

4.2.3. Inverted index construction

We construct another hash map for all the n-grams appear in the edit history. Each n-gram is mapped to a list of revision ids, which indicate that it has appeared in revisions with these ids. We scan each n-gram distribution by time order so that the list of n-gram can have an ascending order.

4.2.4. First appearance index construction

N-grams that first appear in a new revision cannot be covered by any previous revision. While computing a cover for a revision, these n-grams should be excluded. We construct an index for them with inverted index simultaneously.

Revision ID	first appeared 2-grams
23	go to, promotion of, that use
24	For every, denotes the, follows from
25	number of, if and

Table 4.3 Example of First appearance index of Edith Wharton

4.3. Result analysis

We compare the result revision graphs that generated by our method with the manually constructed graphs and evaluate the degree of how they match. Also we introduce the result of keyword clustering method (Cao et al. [5]) as a reference.

We first conduct a direct comparison by checking each revision's reference source from the results and evaluate how much percent of revisions have the right parent in each graph. The higher percentage the result is, the better accuracy can be achieved.

As shown in Figure 4.1, both method have accuracy around 90%, where the n-gram cover method performs slightly better in most of the case.

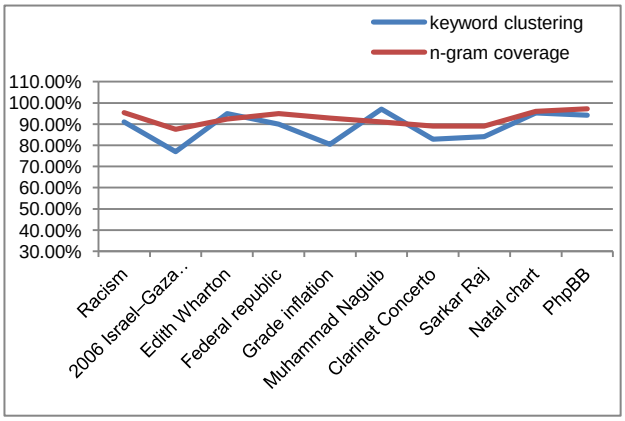

Figure 4.1 Direct comparison of reference source

However, the direct comparison fails to evaluate the errors that happen in the branching nodes. Branching errors that happen in the early stage or involve more revisions deserve more penalties. Other than simply count the number of different reference source, we extend the direct comparison to a complex comparison considering the influence to the descendants.

Notice that a revision graph is a directed acyclic graph (DAG), where a parent node can reach its child nodes and all its descendant nodes. Formally, we define the reachability of a node that node *x* is *reachable* from node *y* if and only if

a. There is an edge from *y* to *x*, or

b. there exists at least one node *p* such that there is an edge from *y* to *p* and *x* is reachable from *p*.

For each node *v*, we check its reachable nodes S(*v*) and S(*v*)'s reachable nodes recursively. The testing graph is represented by an adjacency matrix, where the 1s indicate that node of row index is reachable from node of column index.

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
0	0	0	0	0	0	0	0	0	0
1	**1**	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0
3	1	**1**	0	0	0	0	0	0	0
4	1	1	0	**1**	0	0	0	0	0
5	1	1	0	1	0	0	0	0	0
6	1	0	**1**	0	0	0	0	0	0
7	1	1	0	1	0	**1**	0	0	0
8	1	1	0	1	0	1	0	**1**	0

Table 4. 4 DAG reachability matrix example

By applying this check for every node according to their sequential order, a reachability matrix can be generated. In Table 4.4, we start from the bold 1s to generate the left 1s, where the bold 1s represent direct reference relationship

between nodes. Other than checking nodes pair of bold 1s in the direct comparison, we check each nodes pair of two revision graph and count the number of node pairs with identical reachability. This number would be normalized with the total nodes pairs. Then we can evaluate the accuracy of the result revision graph. In this complex comparison the difference in the higher level of the graph has larger change in result than those in lower level, which reflects the initial purpose that errors that happen in the early stage should be given more penalties.

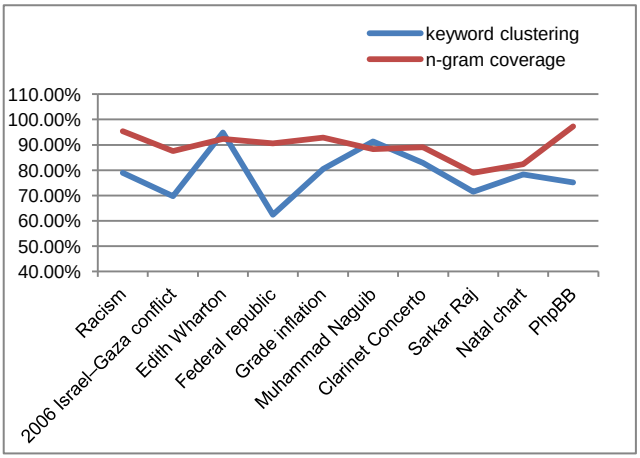

Figure 4.2 Reachability result comparison

As shown in Figure 4.2, when setting the n-gram size *n* to 2, our method can achieve a significant improvement than keyword clustering method in general, especially in the articles about controversial topics.

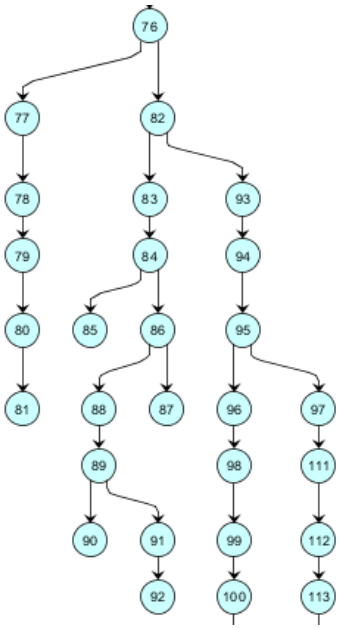

Figure 4.2 Part of revision graph of "PhpBB"

Given the fact that most of the reference sources' n-gram diff score rank within 5 in the candidates, the parameter k in the algorithm is set to 10 in our experiment.

However, our method falls behind in articles that have fewer branches ("Edith Wharton", "Muhammad Naguib"). It is clear that more investigation will be required before a complete understanding of how this phenomenon occurs. In our experience, the adoption of larger *n*, i.e. longer subsequence, does not improve accuracy.

5. Conclusion

In this paper we proposed a revision graph extraction method based on n-gram distribution covering that effectively finds key edit evolution in Wikipedia edit history. We use n-gram distribution to denote revisions of the given articles with timestamps and find how a revision's n-gram distribution can be formed by specific previous revisions'. We utilize smallest n-gram diff score to decrease the complexity in the n-gram coverage process.

For the future work, we would focus on the problem of detecting merge by revising our n-gram distribution diff and n-gram coverage algorithm.

References

[1] A. Doan, R. Ramakrishnan, and A. Y. Halevy. 2011. Crowdsourcing systems on the World-Wide Web. Commun. ACM 54, 4, 86-96, 2011.

[2] Wikipedia: http://en.wikipedia.org/wiki/Wikipedia

[3] Wikipedia Editing : http://en.wikipedia.org/wiki/Wikipedia:How_to_edit_a_page

[4] N. Heintze. Scalable document fingerprinting (extended abstract). In Proc. USENIX Workshop on Electronic Commerce, 1996.

[5] Zhe Cao, Mizuho Iwaihara, Wikipedia version tree reconstruction by clustering revisions through keywords, IEICE Technical Report DE2011-32, 2011

[6] Wikipedia edit history export pages : http://en.wikipedia.org/w/index.php?title=Special:Export&action=submit

[7] Hoad, Timothy; Zobel, Justin, Methods for Identifying Versioned and Plagiarised Documents, Journal of the American Society for Information Science and Technology 54, 2003

[8] Dom4j: http://dom4j.sourceforge.net/