Improving revision graph extraction in Wikipedia based on supergram decomposition

呉 建民† 岩井原瑞穂‡
†早稲田大学大学院情報生产システム研究科
〒808-0135 福岡県北九州市若松区ひびきの 2 - 7
E-mail: †jianmin.wu@moegi.waseda.jp, ‡iwaihara@waseda.jp

Abstract As one of the popular social media that many people turn to in recent years, collaborative encyclopedia Wikipedia provides information in a more “Neutral Point of View” way than others. Towards this core principle, plenty of efforts have been put into collaborative contribution and editing. The trajectories of how such collaboration appears by revisions are valuable for group dynamics and social media research, which suggest that we should extract the underlying derivation relationships among revisions from chronologically-sorted revision history in a precise way. With this paper, we propose a revision graph extraction method based on supergram decomposition in the document collection of near-duplicates. We show that this method can effectively perform the task than existing methods.

Keyword Social Media, Web Mining, Wikipedia

1. Introduction

In recent years, social media becomes more and more attractive to many people since it involves means of interactions among people in which they create, share, exchange and comment contents among themselves in virtual communities and networks [2]. As a collaborative project, online encyclopedia Wikipedia receives contribution from all over the world [5] and its content is well accepted by those who want reliable social news and knowledge.

Guiding by the fundamental principle of “Neutral Point of View”, Wikipedia articles need plenty of extra editorial efforts other than simply content expanding and fact updating. Users can choose to edit on an existing revision and override the current one or revert to a previous revision. However, there is no explicit mechanism in Wikipedia to trace such relationship among revisions, while the trajectories how such collaboration appears in Wikipedia articles in terms of revisions are valuable for group dynamics and social media research [3]. Also, research exploiting revision history for term weighting requires clean history without astray, which can be accomplished by such trajectories.

Wikipedia now keeps all the versions’ contents for each article and make the edit history publicly available. Other useful information, such as timestamps, contributors, and edit comments is also recorded. Figure 1.1 shows a snapshot of typical Wikipedia edit history. Most existing research modeling Wikipedia’s revision history choose trees or graphs to represent the relationship, but few of them concern about the accuracy of their models.
convert to \( r \). More specifically:

a) Adding takes more efforts than deleting.

b) Long edits take more efforts than short edits.

c) Multiple short edits take more effort than single long
short edits.

To find candidates that meet the above requirements is
different from nearest neighbor search (NNS) in text
mining. The conventional NNS deals with text corpus that
is generally heterogeneous, while in our research the text
content is mutually highly similar in the revision
collection. Common text clustering methods like kNN and
There is another issue we should notice. The overview of
Wikipedia mining [4] shows that the text amount of diff
between two adjacent revisions is not proportional to the
length of the article, that is, users would not contribute
more text because of a longer article. With the relatively
stable edit contribution amount, the longer an article
grows, the less difference can be told by Jaccard distance,
which suggests that we need absolute measure.

In this paper, we first introduce existing work related to
our research. In Section 3 we explain our motivation and
basic process of supergram decomposition. We extend the
model in Section 4 by exploiting dependencies among
revisions and narrowing down comparison scope for
 scalability. Section 5 evaluates the result generated by our
method and compare with other representative methods.
Finally we conclude our paper by summarizing findings
and discussing several key issues.

2. Related Work

Basically, a revision history modeling method should
include three components: text differencing method,
similarity measurement and comparison strategy. Most
existing work focused on the first component. P. Fong et al
[3] proposed a detail text differencing algorithm that finds
all the different parts, including the case of phrase
movement and sentence re-writing, between two given
revisions based on hierarchial decomposition and the
longest common string method, which is however way
too computationally expensive in terms of large scale
revision comparison.

In an investigation on structure and dynamics of
Wikipedia’s breaking news collaborations [3], Keegan et
al. construct article trajectories of editor interactions as
they coauthor an article. Examining a subset of this corpus,
their analysis demonstrates that articles about current
events exhibit structures and dynamics distinct from those
observed among articles about non-breaking events.
However, the similarity metric adopted in this research is
over-simplified and the correctness of the trajectories they
build is not assured.

Cao et al. [2] 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
revisions and then largest common subsequences are used
for more precise comparison, which is closer to string
matching problem.

Wu et al. [4] proposed a revision graph extraction
method for Wikipedia articles based on n-gram cover. This
research uses 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’. But this method still suffers from
error rate due to the plain model of n-gram diff score.

3. Supergram Decomposition

As the further research of [4], we carefully consider the
model of n-gram cover. The n-gram frequency comparison
method in n-gram cover model is from the shingling
method, which has been a conventional method in nearest
neighbor searching [9][10]. In n-gram cover, only the
different text among revisions has been noted and
measured. Diff caused by edit behaviors will be detected
as changes in k-gram frequency distribution. Although the
positional information among tokens can be reserved
partially by longer shingle (bigger n), the integrity of
different edits cannot be recovered. On the other hand, it
takes too much time to achieve integrity by longest
common subsequence based diff algorithm.

We find that there are some token sequences that keep
as a unit throughout the whole revision collection. For a
small revision collection of several revisions, such token
sequences is little but with long length. As the size of the
revision collection grows larger, long token sequences are
split into shorter fragile due to modifications. Formally,
we define such units as:

**DEFINITION 3.1. Supergram**

A supergram \( s=t_1t_2...t_n \) in a revision collection \( R \) is an
n-gram (\( n\geq2 \)) such that \( s \) occurs in at least one revision in
\( R \), and if a token \( t_i \) (\( 1\leq i \leq n \)) occurs in a revision \( R' \), then
\( t_{i+1} \) always occurs just after \( t_i \) in \( R' \).

Basically, for the revision collection \( R \) of an article, we
extract the supergrams by path contraction on word
transition graph, and utilize supergram diff to compare
revisions. More concretely, our method consists of the following steps:

1. **Pre-processing.** After text-cleansing and URL replacement, split all revisions into bigrams and construct a global inverted index $I$ of bigrams on revisions.

2. **Word transition graph construction.** By scanning each revision, construct a word transition graph $G$ for the revision collection. Compact $G$ into a weighted multigraph $G'$ by path contraction, extract the edges’ weights in $G'$ to construct the supergram list $S$.

3. **Supergram decomposition.** Decompose each revision based on $S$, then construct an inverted index of $S$ on revisions. Regarding all terms appearing in $S$, construct an inverted index of terms on $S$.

### 3.1 Pre-processing

We first split the original revision text into a unigram token sequence. The text content in the original revision files contains plenty of *Wiki Markups*, which give specific semantic tags on plain text. While splitting the text, such markups are extracted by regular expression and will be reserved as single tokens in the following steps. The second task is replacing the URLs appearing in the text. No matter how many terms a URL involves, it has no more contribution to add a new URL than to add a single word. We replace each URL with a 16-byte string generated by MD5 for consistency.

### 3.2 Word transition graph construction

Given an article $R$ with versions $r_1, r_2, ..., r_n$, each of them contains a sequence of tokens $D_i = \{t_1, t_2, ..., t_l\}$. In the following paragraphs, we denote

- $v_i$: vertex $i$ labeled with $t_i$;
- $e_{ij}(v_i, v_j)$: edge $x$ from $v_i$ to $v_j$, labeled with the collection frequency of bigram $t_it_j$;
- $out(v_i)$: set of all edges from $v_i$;
- $in(v_i)$: set of all edges to $v_i$;
- $src(e)$: source vertex of edge $e$;
- $tar(e)$: target vertex of edge $e$.

**DEFINITION 3.2 Word transition graph**

Given a document $r$ on vocabulary $D$, a **word transition graph** $G=(V, E)$ is a directed weighted graph such that each vertex $v_i \in V$ denotes a term $t_i \in D$. For two terms $t$ and $t_j \in D$, a weighted directed edge $e(v_i, v_j) \in E$ exists between their corresponding vertices $v_i$ and $v_j$ if and only if the bigram $t_it_j$ has a frequency $f(t_it_j) > 0$ in $R$, and $f(t_it_j)$ is assigned as the edge weight.

The word transition graph is allowed to contain cycles since the multiple appearance of frequent terms causes path that starts and ends at the same vertex but otherwise has no repeated vertices or edges. On the other hand, there exist chain-like subgraphs at which only one path exists, which correspond to **Definition 3.1.** Here we define such structure formally:

**DEFINITION 3.3 Chain**

A **chain** $C=(V', E')$ is a subgraph of $G$ containing only one connected component, with the property that every vertex $v' \in V'$, except the two ends, has only one incoming edge and one outgoing edge, i.e. $|out(v'_i)| = |in(v'_i)| = 1$, and $v'_i$ is called a **chain vertex**. The starting vertex of $C$, namely the **source**, is defined as the vertex $v_s$ such that $|out(v_s)| = 1$ and $|in(v_s)| \neq 1$. The **sink** is defined similarly.

**Path contraction**

By path contraction, each edge $e'(v_i, v_j)$ should satisfy both:

- **Correctness.**
  - For any bigram $titj$ in revision collection $R$, its frequency $f(titj)$ is equal to the supergram frequency $f(sk)$ in $R$, where $sk$ is the supergram that contains $titj$.

- **Compactness.**
  - If the source of $e'$ has no in-degree ($|in(src(e'))| = 0$), the target of $e'$ should have more than 1 out-degree ($|out(tar(e'))| > 1$). Otherwise the total degree of source and target should be more than 3.

Regarding such requirements, we describe the algorithm as follows:

![Figure 3.1 Process of supergram decomposition.](image-url)
Algorithm for path contraction:
Input: G = (V, E)
For each vertex v_i ∈ V. If |out(v_i)| > 0,
    for each v_j ∈ out(v_i),
        If v_j is a chain vertex with an outgoing edge e(v_j, v_k), create a new edge e′(v_i, v_k) and label it with the concatenation of the label of e(v_j, v_k) and v_k’s corresponding term t_k, and delete v_j from G.

Notice that each revision can be treated as a token sequence starting from the same source “$” and sinking with the same terminator “^”, there is no need to consider the cases of |out(v_i)| = 0, or |in(v_i)| = 0.

After path contraction, the original word transition graph is contracted to a multigraph such that each vertex v_i’s corresponding term t_i has a frequency f(t_i) > 0 in at least one revision, and each edge e(t_i, t_j) representing a supergram s in either ways:

a) If the edge label is a concatenation of (freq|terms), s is a new concatenation of t_i + terms + t_j.

b) If the edge label is an integer, s = t_i, t_j  

3.3 Supergram decomposition
Decompose each revision based on supergram set S, construct an inverted index of S on revisions. Regarding all terms appearing in S, construct an inverted index of terms on S. Then all revisions’ supergram frequency distribution can be compared based on S.

4. Comparison
Recall the observation of supergram we mentioned before: a narrower scope will produce longer supergrams. This is because the number of edits is proportional to the scope length and fewer edit mean smaller chances, and supergrams tend to be undivided. Longer supergram is preferable in supergram decomposition because it reserve more integrity and reduce the total number of supergrams. Another strong reason for scoping is scalability. In Wikipedia, articles pose various numbers of revisions from tens to tens of thousands. Without any heuristics, it takes O(N^2) time to perform full pairwise comparison for an article posing N revisions, which is an unbearable expense of computing resource especially for those popular articles with thousands of revisions. Regarding these issues, we extend the global decomposition by introducing a sliding comparison scope and finish the whole comparison on revision collection. The comparison stage consists of 4 parts:

1. Comparison scope determination
For each revision r_i, calculate r_i’s comparison scope C_i based on r_i’s timestamps and global bigram inverted index I.

2. Sliding decomposition
Construct a word transition graph G_i of all the revisions within C_i and perform path contraction for G_i to get G_i’. Decomposed r_i’ and all revisions in C_i into supergram frequency distribution based on the supergram set S extracted from G_i’.

3. Supergram diff score computing.
Compare r_i’’s supergram frequency distribution with all revisions’ in C_i and compute their supergram diff scores.

4. Candidate selection.
Pick up the revisions with lowest k supergram diff score as the candidates for parents.
Before we limit the comparison scope to a fixed number of previous revisions, we consider the frequent edit behavior within a certain period of time as another important factor according to the timestamps in the edit history’s meta information. Intense editing activity could be caused by edit wars, increasing popularity of the article, or immediate updates after related events happen, and the total number of edits in a week could easily exceed any preset number. Figure 4.1 shows the edit count of Wikipedia article "Barack Obama" during 2008, the year of the U.S. presidential election.

But before we limit the comparison scope to a fixed number of previous revisions, we consider the frequent edit behavior within a certain period of time as another important factor according to the timestamps in the edit history’s meta information. Intense editing activity could be caused by edit wars, increasing popularity of the article, or immediate updates after related events happen, and the total number of edits in a week could easily exceed any preset number. Figure 4.1 shows the edit count of Wikipedia article "Barack Obama" during 2008, the year of the U.S. presidential election, and significant peak can be found in November, when the election was held. Thus, all the previous revisions within certain time span should be examined, regarding the fact that contributors’ attention can last for a period of time.

Fixed scope would not be able to capture the whole process of the intense edit activity, while fixed time span can cover only little revisions. Considering such trade-off, we employ maximum comparison scope to denote the biggest number of previous revisions to be compared, which is defined as below.

**DEFINITION 4.1 Maximum comparison scope:**

Given a revision history \( H = \{(r_1, t_1), (r_2, t_2), \ldots, (r_m, t_m)\} \), where \((r_i, t_i)\) denote a revision \( r_i \), with its timestamp \( t_i \), the maximum comparison scope \( C \) for revision \( r_k \) is determined by either:

a) If \( t_k - t_{k-p} > T \), \( C = S_i \) or

b) \( \exists p > 0 \) such that \( t_k - t_{k-p} \leq T \) and \( t_k - t_{k-p+1} > T \), \( C = p \)

where \( S_i \) denotes the least scope to ensure enough comparison for unpopular documents, \( T \) denotes the least time span for intense edits.

Notice that there could be a series of consecutive edits by the same contributor, we take the latest revision only and omit the others, since we focus on the collaborative authoring and editing process rather than individual perspective.

Another issue we should notice is the phenomenon of remote copy, which is the behavior that copying a piece of text from an ancient revision such that there is no appearance of such text within the scope of Maximum comparison scope. Simply expanding the scope to that ancient revision includes unnecessary revisions and lowers the efficiency. We choose to include this kind of ancient revision as individual revision alone. Formally, an ancient revision is identified as follows:

A revision \( r_j \) is a potential remote ancestor of \( r_i \) if and only if there is a bigram \( b_k \) that appears in \( r_j \) but not in revisions between \( r_j \) and \( r_i \).

**4.2 Supergram diff score computing**

For pairwise revision comparison, we first create the supergram diff for two revisions, and then calculate the supergram diff score to measure their difference.

**DEFINITION 4.2. Supergram diff**

Given a supergram set \( S \), we denote the supergram frequency distribution of revision \( r_a \) as \( f(s, r_a) \) (\( s \in S \)). For two revisions \( r_a \) and \( r_b \), the supergram diff \( SD \) is the set of supergrams with a non-zero residual frequency between \( r_a \) and \( r_b \):

\[
SD(r_a, r_b) = \{ s \in S | |f(s, r_a) - f(s, r_b)| > 0 \}
\]  

(4.1)

**DEFINITION 4.3. Supergram diff score**

\[
diffScore(r_a, r_b) = w_1 \cdot \sum_{s \in SD_{add}} |f(s, r_a) - f(s, r_b)| \cdot |s| + w_2 \cdot \sum_{s \in SD_{del}} |f(s', r_a) - f(s', r_b)| \cdot \log |s|
\]

(4.2)

where \( SD_{add} \) is the set of all supergrams such that \( f(s, r_a) - f(s, r_b) > 0 \), and \( SD_{del} \) is defined similarly, \( w_1 \) is the weight for discrimination between adding and deleting operations. As heuristics, the logarithms are to the base of 10, since the deleting operations is less effort-taking job.

**4.3 Candidate selection**

It is possible that with the lowest supergram diff score there come multiple revisions with the same score. For example, if a revision \( r \) just revert to one previous revision \( r' \), the child of \( r \) has at least 2 revisions that have the same supergram diff score. To avoid ambiguity, we define that the parent revision should be the one with the latest timestamp in such cases.

---

5. Experimental evaluation

We conduct experimental evaluation on our method against several existing revision relationship modeling methods on a collection of Wikipedia articles.

<table>
<thead>
<tr>
<th>Article #</th>
<th>Article Title</th>
<th># of Branches</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Racism</td>
<td>23</td>
</tr>
<tr>
<td>2</td>
<td>2006 Israel–Gaza conflict</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>PhpBB</td>
<td>37</td>
</tr>
<tr>
<td>4</td>
<td>Edith Wharton</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>Federal republic</td>
<td>33</td>
</tr>
<tr>
<td>6</td>
<td>Sarkar Raj</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>Grade inflation</td>
<td>24</td>
</tr>
<tr>
<td>8</td>
<td>Natal chart</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>Muhammad Naguib</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>Clarinet Concerto</td>
<td>12</td>
</tr>
</tbody>
</table>

Table 5.1 Ground Truth Statistics

To evaluate the performance of the proposed method, we conduct a series of accuracy evaluation with 3 representative methods: sentence-level Jaccard distance, keyword clustering, n-gram cover. For each method, we compare its result revision graph with manually constructed graphs on the existing ground truth [4]. As shown in Table 5.1, the data set contains 10 Wikipedia articles totaling 2000 revisions. The branch counts indicate the total revert events and diversity, potentially suggesting the degree of popularity and controversy. All the revisions have been pre-processed according to Section 3.1 so that all methods start with the same token sequence.

The parent accuracy is evaluated by the percentage of the revisions that has the correct parent.

\[
C(G_1, G_2) = \frac{2 |\mathsf{overlap}(G_1, G_2)|}{|\mathsf{parent}(G_1)|^2} \tag{5.1}
\]

where \( G_1^+, G_2^+ \) are the transitive closures of \( G_1, G_2 \). \(|D|^2/2\) is half the number of all the node pairs. By formula (5.1) we focus on how far (in terms of number of total descendant revisions) an error can reach, so errors that happen in the early stage or those that involve more succeeding revisions have greater loss in accuracy.

6. Conclusion

In this paper, we proposed supergram technique for accurate reconstruction of Wikipedia revision history. Supergrams are extracted from a word transition graph by path contraction. Our proposed method outperforms previous approach by the n-gram cover.

References

