Enhancing an Incremental Clustering Algorithm for Web Page Collections

Gavin Shaw & Yue Xu
Faculty of Science & Technology, Queensland University of Technology, Brisbane, Australia
g4.shaw@student.qut.edu.au, yue.xu@qut.edu.au

Abstract

With the size and state of the Internet today, a good quality approach to organizing this mass of information is of great importance. Clustering web pages into groups of similar documents is one approach, but relies heavily on good feature extraction and document representation as well as a good clustering approach and algorithm. Due to the changing nature of the Internet, resulting in a dynamic dataset, an incremental approach is preferred. In this work we propose an enhanced incremental clustering approach to develop a better clustering algorithm that can help to better organize the information available on the Internet in an incremental fashion. Experiments show that the enhanced algorithm outperforms the original histogram based algorithm by up to 7.5%.

1. Introduction

The growing size of the Web means that more and more information is available to people. However the downside to such a large collection of documents is that it becomes difficult to find documents relevant to the user’s needs. Clustering web page collections can make it easier to find relevant documents as clustering brings similar documents together and can make finding information easier and quicker. Traditionally, datasets have been static (they do not change) so clustering algorithms were developed take advantage of this. These algorithms are known as static algorithms and they cluster the dataset once. Should the dataset change (new documents added, documents deleted or modified) then it was necessary to perform a complete reclustering.

Although incremental algorithms are the best method for clustering dynamic data, they suffer from problems. Two of these problems, are the effectiveness of the overall algorithm/approach (especially over time as the algorithm runs the collection through multiple iterations) and the insertion order of new documents into the existing collection.

Effectiveness determines how correct/accurate the results will be. For incremental algorithms this is important as it affects the results over time. Because an incremental algorithm will be executed many times an incremental algorithm needs not just a high initial effectiveness, but one that can be maintained throughout each iteration, keeping the results accurate and meaningful.

The second problem is the insertion order issue. To some extent, incremental algorithms are affected by the order that new documents arrive in to be added to the clustered results. Ideally, incremental algorithms should give the same results for a dataset/collection regardless of the order that documents arrive in (order-independent). The goal is to reduce the effect that the insertion order problem has on the results or remove it.

The aim of this work is to improve the histogram based incremental clustering approach [3] to reduce the impact these issues have when clustering web page collections. This will result in a better algorithm.

The paper is organized as follows. Section 2 discusses related work. The proposed enhanced algorithm is presented in Section 3. Experiments and results are presented in Section 4. Lastly, Section 5 concludes the paper.

2. Related Work

Traditionally, clustering took a dataset, processed it and produced a result set. If the dataset was changed, the entire dataset had to be reclustered from scratch. This reclustering could be costly in terms of processing time. Incremental clustering is one technique developed to avoid this. It is based on the idea that it is possible to process documents one at a time and assign them to a cluster, without significantly affecting the state of the existing clusters. In short, this means documents that are already clustered do not have to be reclustered when a new document is added to the dataset. This is the primary difference between traditional clustering and incremental clustering, the ability to handle new data as it is added to the dataset [3].
Today very few databases are static and most databases are large in size. Going through the process of feature extraction etc for each document and then reapplying the clustering algorithm across the entire set can be prohibitively expensive, whereas incremental clustering algorithms may only be needed on a small proportion of the entire dataset size. In the worst case an incremental algorithm will degenerate into a reclustering with the same cost as a non-incremental algorithm [1,2].

It has been claimed that when it comes to web page clustering there are four main issues to be looked at. They become especially important for an incremental clustering algorithm. The four issues are efficiency, effectiveness, incrementality and noise-tolerance [5].

The second issue is effectiveness. This is basically the ability for a clustering algorithm to group similar web pages together, while also ensuring that dissimilar web pages are kept separate and therefore are not placed together in the same cluster. It is important that an incremental algorithm is just as effective and accurate as a non-incremental algorithm. If it is not, then there will be no point in using it. It may take less time, but the results are of a poorer quality. Because an incremental algorithm will also be used many times, as the web page collection changes, the algorithm must be able to maintain a comparable level to non-incremental techniques or the quality of the results will decrease over time.

The final issue is noise-tolerance. Web pages are perhaps the only form of data that contain huge amounts of noise or unimportant contents such as banners, advertisements etc. All of these often have very little to do with the actual content of the page. Noise-tolerance is the ability for an incremental algorithm to deal with web pages, which are not similar to any other page already in the clustered set. To maintain the quality of the clusters, the algorithm should not have to group this noise page in an existing cluster (as this would decrease the quality). Instead it should be able to place this page in a new cluster that is then added to the existing cluster structure as appropriate [1,6]. This then preserves the quality of the existing clusters.

Our work mainly focuses on the second and fourth issues; effectiveness & noise-tolerance, to produce a better incremental clustering algorithm.

3. Enhanced Incremental Algorithm

Here we present an overview of the proposed enhanced similarity histogram clustering algorithm. The original algorithm (known as SHC) was first presented in [3]. Full details of the base version are omitted here due to space, but can be found in [3,4].

3.1 Enhanced Similarity Histogram Clustering Using Intra Centroid Vector Similarity (ESHSC-IntraCVS)

This enhanced version of the similarity histogram clustering algorithm takes into account the similarity that a document has with the centroid vector of a cluster when it comes to determining which cluster is the best for the document to be placed in. The centroid vector is a vector space model (VSM) representation of the centre or average of the cluster and the documents it contains. This enhanced algorithm works on the idea that the cluster that contains a large number of similar documents to the current document being clustered, will have a centroid vector that has a high similarity to the current document. Therefore the cluster whose centroid vector is the most similar to the document’s vector representation is the one that most likely contains the greatest number of documents that are the most similar to the current document. Adding the new document to this cluster (when possible) will probably give the greatest benefit to that cluster and the entire dataset.

The biggest change in how this algorithm works is that instead of placing the document in the cluster which would receive the best histogram ratio change (like the original algorithm), this enhanced version instead adds it where possible to the cluster that has the most similar centroid vector to the document. The idea with this approach is that the cluster with the highest similarity to the document will have the greatest number of similar documents in it and would be the best cluster to place the document in. By using the similarity to the cluster centroid vector the cohesiveness of the clusters can be maintained and any tendency for a cluster to spread out over vector space can be limited. This should give rise to more tightly packed cluster which are more distinct from each other, with minimal or ideally no overlap between them.

For this enhanced algorithm, the centroid vector is calculated by determining the average weight of every term that is in at least one document vector present for that particular cluster. Mathematically, the centroid vector (CV) is defined as the following:

\[
CV = \frac{1}{|S|} \sum_{\mathbf{d} \in S} \mathbf{d}
\]  
(Eq. 1)

where \( S \) is the set of documents, \( \mathbf{d} \) is the term vector representation for document \( d \) in the cluster.
Thus the algorithm for the implementation of this enhanced algorithm is as follows:

```
1: L ← Empty List [Cluster List]
2: for each document D do
3:    for each cluster C in L do
4:       HR_{old} = HR_C
5:       Simulate adding D to C
6:       HR_{new} = HR_C
7:       if (HR_{new} ≥ HR_{old}) OR ((HR_{new} > HR_{min})
         AND (HR_{old} − HR_{new} < ε)) then
8:          Sim_{DC} = Cosine similarity between D
            and CV of C (without updating
            to include D)
9:          store details in List (P)
10: end if
11: end for
12: order List P in decreasing Sim_{DC}
13: take the first entry in List P and determine C
14: Add D to C
   (which was the first entry in List P)
15: if D was not added to any cluster then
16:    Create a new cluster C
17: ADD D to C
18: ADD C to L
19: end if
20: end for
```

Figure 1. Algorithm for ESHC-IntraCVS.

In this implementation, the algorithm looks at each cluster and simulates the addition of the document to it, thus determining both the new and old ratios (lines 4 – 6). The cluster is then assessed to see if it is allowed to potentially take the document by checking that the new histogram ratio is okay (line 7). If the new ratio is okay, then the similarity between the document (D) and the cluster (C) is calculated (line 8). All of this information is then stored in a list (line 9). After going through all the clusters that currently exist, the algorithm then orders the list (called P) in order of decreasing similarity to the cluster centroid vector (Sim_{DC}) (line 12). This ensures that the cluster that the document (D) is most similar to, will be the cluster that receives the new document. The document is then added to the first cluster in the candidate list (P) (line 14). If there were no clusters suitable to take the new document, then a new cluster is generated and the document is added to it (lines 15 – 19).

This enhanced version algorithm can also implement the same document reassignment strategy that the original SHC algorithm uses [3,4], except that it also uses the similarity between a document and a cluster’s centroid vector to determine which is the best cluster to move the document to.

Thus the algorithm to implement the reassignment strategy is as follows:

```
1: after new document has been clustered
2: for each cluster in the Cluster List
3:    determine the documents that are candidates for reassignment
4:    for each candidate
5:       for each cluster in the Cluster List, except for the current cluster
6:          HR_{old} = HR_C
7:          Simulate adding D to C
8:          HR_{new} = HR_C
9:          HR_{change} = HR_{new} − HR_{old}
10: if (HR_{change} > 0) then
11:     Sim_{DC} = Cosine similarity between D and CV of C (without updating
         to include D)
12: store details in List (P)
13: end if
14: end for
15: order List P in decreasing Sim_{DC}
16: if (List P has >= 1 entries) then
17:    reassign candidate document to the first cluster in List P
18: end if
19: end for
20: end for
```

Figure 2. Document reassignment algorithm for ESHC-IntraCVS.

4. Experimental Results

In this section we present experimental results of our proposed enhanced incremental clustering algorithm and compare it against the original approach.

4.1 Dataset & Setup

For this experiment a single dataset was used as the primary source of input to the algorithms. This dataset was the UW-CAN dataset that was used by the original authors of the similarity histogram algorithm. The dataset is available freely on the Internet and is located at http://pami.uwaterloo.ca/~hammouda/webdata/. This dataset contains 314 web pages that have been taken from the University of Waterloo and various Canadian web sites. The pages are pre-classified into 10 (black-bear-attack, campus-network, Canada-transportation-roads, career-services, co-op, health-services, river-
fishing, river-rafting, snowboarding-skiing & winter-
canada) different categories/classes. We use this
existing classification as our baseline on how the
dataset should be clustered. The following measures:
Precision, Recall, F-Measure & Weighted F-Measure,
were used during the experiment to evaluate both the
original approach and our proposed enhanced
algorithm to compare their performance.

4.2 Results

We undertook eight test cases for both algorithms,
using different vector space weighting schemes, initial
dataset status and document reassignment support. The
eight test cases were:

<table>
<thead>
<tr>
<th>Case</th>
<th>Feature Weighting</th>
<th>Initial Dataset</th>
<th>Document Reassignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TF</td>
<td>Empty</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>TF</td>
<td>Empty</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>TF</td>
<td>150 pre-classified</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>TF</td>
<td>150 pre-classified</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>TF-IDF</td>
<td>Empty</td>
<td>No</td>
</tr>
<tr>
<td>6</td>
<td>TF-IDF</td>
<td>Empty</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>TF-IDF</td>
<td>150 pre-classified</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>TF-IDF</td>
<td>150 pre-classified</td>
<td>Yes</td>
</tr>
</tbody>
</table>

The following tables summarize the results of our
experiments.

<table>
<thead>
<tr>
<th>Case</th>
<th>Feature Weighting</th>
<th>Initial Dataset</th>
<th>Document Reassignment</th>
<th>Ave.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TF</td>
<td>Empty</td>
<td>No</td>
<td>0.624</td>
</tr>
<tr>
<td>2</td>
<td>TF</td>
<td>Empty</td>
<td>Yes</td>
<td>0.633</td>
</tr>
<tr>
<td>3</td>
<td>TF</td>
<td>150 pre-classified</td>
<td>No</td>
<td>0.851</td>
</tr>
<tr>
<td>4</td>
<td>TF</td>
<td>150 pre-classified</td>
<td>Yes</td>
<td>0.813</td>
</tr>
<tr>
<td>5</td>
<td>TF-IDF</td>
<td>Empty</td>
<td>No</td>
<td>0.675</td>
</tr>
<tr>
<td>6</td>
<td>TF-IDF</td>
<td>Empty</td>
<td>Yes</td>
<td>0.691</td>
</tr>
<tr>
<td>7</td>
<td>TF-IDF</td>
<td>150 pre-classified</td>
<td>No</td>
<td>0.916</td>
</tr>
<tr>
<td>8</td>
<td>TF-IDF</td>
<td>150 pre-classified</td>
<td>Yes</td>
<td>0.888</td>
</tr>
</tbody>
</table>

The following tables summarize the results of our
experiments.

<table>
<thead>
<tr>
<th>Case</th>
<th>Feature Weighting</th>
<th>Initial Dataset</th>
<th>Document Reassignment</th>
<th>Ave.</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>TF</td>
<td>Empty</td>
<td>No</td>
<td>0.485</td>
</tr>
<tr>
<td>6</td>
<td>TF</td>
<td>Empty</td>
<td>Yes</td>
<td>0.518</td>
</tr>
<tr>
<td>7</td>
<td>TF</td>
<td>150 pre-classified</td>
<td>No</td>
<td>0.757</td>
</tr>
<tr>
<td>8</td>
<td>TF</td>
<td>150 pre-classified</td>
<td>Yes</td>
<td>0.757</td>
</tr>
</tbody>
</table>

The ESHC-IntraCVS approach we have proposed
here consistently outperforms the original SHC
approach in all of our experiments. As shown in Tables
2 & 3 the ESHC-IntraCVS algorithm gives an
improvement from 0.026 (Case 7) up to 0.075 (Case 4)
in terms of the F-Measure score. Our results also show
that having a document reassignment also improves the
effectiveness of the clustering algorithm. Finally,
having part of the dataset already clustered by domain
experts (Cases 3, 4, 7 & 8) and then applying an
incremental algorithm to the remaining web pages
gives a much better result. With this pre-classifying our
proposed ESHC-IntraCVS algorithm manages to
achieve a result of 0.916 in one of the test cases.

5. Conclusion

In this paper we proposed an enhancement to the
incremental similarity histogram clustering algorithm
by adding in the use of a centroid vector for each
cluster. This approach uses the same pair-wise
document similarity representation and distribution
approach for each cluster that the original algorithm
does, but also uses additional information about the
cluster to determine the best cluster to place the new
document in. For the ESHC-IntraCVS algorithm the
cosine similarity between the document and the
cluster’s centroid is used (intra-similarity). The
experiments show that the ESHC-IntraCVS
implementation has an improved performance over the
original SHC algorithm. One drawback of the proposed
approach is that it is not as efficient as the original
algorithm due to the use of the centroid vector
requiring extra computation.

Future work includes improving the efficiency of
our proposed algorithm, along with testing its
application to larger datasets.

6. References

Information Processing. ACM Transactions on Information
for Dynamic Document Databases. In Applied Computing
Symposium.
Document Clustering Using Cluster Similarity Histograms. In
Int. Conference.
Clustering Algorithm for Web Page Collections, Honours
dissertation. Queensland University of Technology.
Web Page Classification, Masters manuscript, full version.
The Chinese University of Hong Kong.
Clustering for Web Page Classification. In Info. Society in the
21st century: emerging technologies and new challenges