INLS 490-154: Introduction to Information Retrieval System Design and Implementation. Fall 2008.

13. Information Organization

Chirag Shah*
School of Information & Library Science (SILS)
UNC Chapel Hill NC 27514
chirag@unc.edu

1 Introduction

This time we will focus on organizing the collected or retrieved information. While information organization is a context-dependent concept, classification or clustering is often used while organizing a bunch of documents. Such an organization can help a user to get a better idea about the inherent structure of the information as well as various relationships that may exist among different pieces of the information. For instance, Clusty[1] is a search engine that clusters the retrieved documents before presenting the results to the user. This allows the user to see how the results are distributed in various categories, and also refine his search to a particular subdomain or concept.

2 Term-clouds

Before we look at clustering, let us see a simple way of presenting an organized structure to the user to give him an idea of the underlying information. To be specific, we will produce a term-cloud structure, inspired by tag-clouds in many of the Web 2.0 application. This structure can be produced in many ways. To keep things simple, we will simply take the frequencies of the terms present in a collection and use that information to create a term-cloud.

An algorithm, along with the ways to execute those steps, to produce this cloud is given below.

[1] http://www.clusty.com

1. Get the vocabulary from the index. With Lemur, we can use 'dumpindex' application to get this vocabulary:
   `dumpindex <index_name> v`

2. Extract the terms and their frequencies. Since the output of the above 'dumpindex' command produces a list where each line is
   `<term_name> <term_freq_in_collection> <number_of_documents>`,
   we can pipe that output to `awk` as the following.
   `dumpindex <index_name> v | awk '{print $1,$2;}'`
   This will print one term per line along with its frequency, which you can redirect to a file for storage and further processing. Make sure to delete the first line from the above output as it will contain the frequency count for all the terms.

3. Use relative frequencies to size the terms for the display. The details of this process is beyond the scope of this document, but you can download a script to execute this step from the class website.

The output of the above algorithm will be a term-cloud that can be displayed to a user to give him a good idea of what terms/concepts are predominant in the given collection. Similar process can be done on the collection of retrieved documents. Furthermore, the displayed terms/concepts can be hyperlinked, so that when the user clicks on it, the system takes that term as a query and returns the results matching that term.

3 Clustering

Clustering refers to partitioning of the given data set into subsets (clusters). Typically, cluster labels, and sometimes even the number of clusters to create, are not known. We will look at three of the most popular way of clustering a collection of documents.

3.1 k-means or centroid-based clustering

This is a clustering method that takes n objects and assigns them in k partitions ($k < n$). This is done by starting with an arbitrary number of clusters and continuing to reassign the objects to different clusters so that the intra-cluster variance is minimized. This intra-cluster variance is computed as

$$V = \sum_{i=1}^{k} \sum_{x_j \in P_i} (x_j - \mu_u)^2 \tag{1}$$

Here, k is the number of clusters that we want, and μ_i is the centroid (mean point) of all the points x_j that belong to the partition P_i. The most famous way of implementing k-means is using Lloyd's algorithm, which is outlined below.

1. Start by selecting k points as the initial centroids.

2. Assign all the n points ($n > k$) to the nearest centroid.

3. Recalculate the centroid of each cluster.

4. If the newly computed centroids are the same, or no points switched clusters, or we reached the maximum iterations allowed, stop. Otherwise, repeat the above two steps.

3.2 Bisecting k-means or top-down clustering

While k-means is probably one of the most popular clustering method, several other techniques exist that can be broadly classified into top-down approach, and bottom-up approach. The top-down approach starts with a large cluster and keep splitting it until a stopping criteria. The bottom-up approach is exactly opposite, in which it starts with the smallest clusters and keeps combining them to form larger clusters.

Following is an algorithm for bisecting k-means algorithm, which is a top-down approach to clustering.

1. Pick a cluster to split (the largest).

2. Find two sub-clusters using the basic k-means clustering. (bisecting stage)

3. Repeat step-2 for $bkIters$ times and take the split that produces the clustering with the highest overall similarity.

4. Repeat the above steps until the desired number of clusters achieved.

3.3 Agglomerative or bottom-up clustering

Now, let us look at a bottom-up approach. Agglomerative clustering is by definition bottom-up. It starts by having one element per cluster and continues agglomerating them to form larger clusters.

An algorithm to show how this approach works is given below.

1. Begin with each element as a cluster.

2. Merge "similar" elements in one cluster.

3. Keep merging elements/cluster to create larger clusters until we achieve a certain number of clusters, or we have gone through a certain number of iterations, or the process converges (no more meaningful merging).

4 Comparison of clustering methods

- Bisecting k-means tends to produce relatively uniform sized clusters.
- Agglomerative clustering may be fast, but may produce bad clusters.
- k-means is better suited for document clustering.
- Bisecting k-means is shown to be better than k-means and as good as or better than the hierarchical approaches.

5 Clustering with Lemur

Lemur comes with a good support for performing a variety of clustering tasks. It has an application called 'Cluster', which can be used for online clustering, which means it can assign an item as it sees it without looking at the entire collection. This is done by matching a new item with the existing clusters and seeing if it could fit in one of them based on their similarities. If it does not fit in an existing cluster, a new cluster is created for that item. This is a very efficient method, but it may result in poor clustering.

Lemur also comes with functions and classes that let one do other forms of clustering. To help one get started, Lemur has an example application called 'OfflineCluster'. Sample parameter files for both of these applications are shown below.

Code 1: Parameter file for 'OfflineCluster'

```
<parameters>
    <index>myindex</index>
    <clusterType>centroid</clusterType>
    <simType>COS</simType>
    <numParts>10</numParts>
    <maxIters>10</maxIters>
</parameters>
```

Code 2: Parameter file for 'Cluster'

```
<parameters>
    <index>myindex</index>
    <clusterType>agglomerative</clusterType>
    <simType>COS</simType>
    <threshold>0.25</threshold>
</parameters>
```

6 Summary

- Clustering helps in organizing information based on content similarity.
- We looked at k-means, bisecting k-means, and agglomerative clustering.
- Lemur has 'Cluster' application for online clustering.
- Lemur has 'OfflineCluster' example application, which can be easily modified to achieve desired clustering.