Clustering Data in Peer-to-Peer Systems

Mei Li
Microsoft Corporation, USA

Wang-Chien Lee
Pennsylvania State University, USA

INTRODUCTION

With the advances in network communication, many large scale network systems have emerged. Peer-to-peer (P2P) systems, where a large number of nodes self-form into a dynamic information sharing system, are good examples. It is extremely valuable for many P2P applications, such as market analysis, scientific exploration, and smart query answering, to discover the knowledge hidden in this distributed data repository. In this chapter, we focus on clustering, one of the most important data mining tasks, in P2P systems. We outline the challenges and review the start-of-the-art in this area.

Clustering is a data mining technique to group a set of data objects into classes of similar data objects. Data objects within the same class are similar to each other, while data objects across classes are considered as dissimilar. Clustering has a wide range of applications, e.g., pattern recognition, spatial data analysis, custom/market analysis, document classification and access pattern discovery in WWW, etc.

Data mining community have been intensively studying clustering techniques for the last decade. As a result, various clustering algorithms have been proposed. Majority of these proposed algorithms is designed for traditional centralized systems where all data to be clustered resides in (or is transferred to) a central site. However, it is not desirable to transfer all the data from widely spread data sources to a centralized server for clustering in P2P systems. This is due to the following three reasons: 1) there is no central control in P2P systems; 2) transferring all data objects to a central site would incur excessive communication overheads, and 3) participants of P2P systems reside in a collaborating yet competing environment, and thus they may like to expose as little information as possible to other peers for various reasons. In addition, these existing algorithms are designed to minimize disk access cost. In P2P system, the communication cost is a dominating factor. Therefore, we need to reexamine the problem of clustering in P2P systems.

A general idea to perform clustering in P2P systems is to first cluster the local data objects at each peer and then combine the local clustering results to form a global clustering result. Based on this general idea, clustering in P2P systems essentially consists of two steps, i.e., *local clustering* and *cluster assembly*. While local clustering can be done by employing existing clustering techniques, cluster assembly is a nontrivial issue, which concerns *representation model* (what should be communicated among peers) and *communication model* (how peers communicate with each other).

In this chapter, we review three representation models (including two *approximate representation models* and an *exact representation model*) and three communication models (including *flooding-based communication model*, *centralized communication model*, and *hierarchical communication model*).

The rest of this chapter is organized as follows. In next section, we provide some background knowledge on P2P systems and clustering techniques. The details of representation models and communication models are presented in Section 3. We discuss future trend and draw the conclusion in Section 4 and Section 5, respectively.

BACKGROUND

P2P Systems

Different from traditional client-server computing model, P2P systems have no central control. Each participant (peer) has equal functionality in P2P systems. Peers are autonomous and can join and leave the system at any time, which makes the systems highly dynamic. In addition, the number of peers in P2P sys-

tems is normally very large (in the range of thousands or even millions).

P2P systems display the following two nice features. First, they do not have performance bottlenecks and single points of failure. Second, P2P systems incur low deployment cost and have excellent scalability. Therefore, P2P systems have become a popular media for sharing voluminous amount of information among millions of users.

Current works in P2P systems have been focusing on efficient search. As a result, various proposals have emerged. Depending on whether some structures are enforced in the systems, existing proposals can be classified into two groups: *unstructured overlays* and *structured overlays*.

Unstructured Overlays

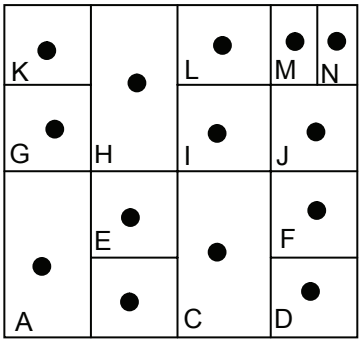
In unstructured overlays, a peer does not maintain any information about data objects stored at other peers, e.g., Gnutella. To search for a specific data object in unstructured overlays, the search message is flooded (with some constrains) to other peers in the system. While unstructured overlays are simple, they are not efficient in terms of search.

Structured Overlays

In structured overlays, e.g., CAN (Ratnasamy, 2001), CHORD (Stoica, 2001), SSW (Li, 2004), peers collaboratively maintain a distributed index structure, recording the location information of data objects shared in the system. Besides maintaining location information for some data objects, a peer also maintains a routing table with pointers pointing to a subset of peers in the system following some topology constraints. In the following, we give more details on one representative structured overlay, content addressable network (CAN).

Content Addressable Network (CAN): CAN organizes the logical data space as a k-dimensional Cartesian space and partitions the space into *zones*, each of which is taken charge of by a peer, called as *zone owner*. Data objects are mapped as points in the k-dimensional space, and the index of a data object is stored at the peer whose zone covers the corresponding point. In addition to indexing data objects, peers maintain routing tables, which consist of pointers pointing to neighboring subspaces along each dimension. Figure 1 shows one example of CAN, where data objects and peers are mapped to a 2-dimensional Cartesian space. The space is partitioned to 14 zones, and each has one peer as the zone owner.

Figure 1. Illustrative example of CAN



Clustering Algorithms

In the following, we first give a brief overview on existing clustering algorithms that are designed for centralized systems. We then provide more details on one representative density-based clustering algorithm, i.e., DBSCAN (Ester, 1996), since it is well studied in distributed environments. Nevertheless, the issues and solutions to be discussed are expected to be applicable to other clustering algorithms as well.

Overview

The existing clustering algorithms proposed for centralized systems can be classified into five classes: partition-based clustering, hierarchical clustering, grid-based clustering, density-based clustering, and model-based clustering. In the following, we provide a brief overview on these algorithms. Partition-based clustering algorithms (e.g., k-mean, MacQueen, 1967) partition n data objects into k partitions, which optimize some predefined objective function (e.g., sum of Euclidean distances to centroids). These algorithms iteratively reassign data objects to partitions and terminate when the objective function can not be improved further. Hierarchical clustering algorithms (Duda, 1973; Zhang, 1996) create a hierarchical decomposition of the data set, represented by a tree structure called dendrogram. Grid-based clustering algorithms (Agrawal, 1998; Sheikholeslami, 1998) divide the data