Large-Scale Community Detection on YouTube for Topic Discovery and Exploration

Ullas Gargi
Google, Inc.
ullas@google.com

Wenjun Lu
University of Maryland
wenjunlu@umd.edu

Vahab Mirrokni
Google, Inc.
mirrokni@google.com

Sangho Yoon
Google, Inc.
shyoon@google.com

Abstract

Detecting coherent and well-connected communities inside large-scale graphs is an interesting problem that can provide useful insight into the graph structure and individual communities. It can also serve as the basis for content exploration and discovery within the graph. Clustering is a popular technique for community detection, however, the two main categories of clustering algorithms, i.e, global and local algorithms, have either scalability or usability issues, e.g, global algorithms do not scale well, and local algorithms may cover only a portion of the graph. Such one-stage algorithms typically optimize one objective function and do not work well in settings where we need to optimize various coverage, coherence and connectivity metrics. In this paper, we study large-scale community detection over a real-world graph composed of millions of YouTube videos. In particular, we present a multi-stage scalable clustering algorithm, combining a pre-processing stage, a local clustering stage, and a post-processing stage to generate clusters of YouTube videos with coherent content. We formalize coverage, coherence, and connectivity metrics and evaluate the quality of the proposed multi-stage clustering algorithms for YouTube videos. We also use extracted entities to attach meaningful labels to our clusters. Our use of local algorithms for global clustering, and its implementation and practical evaluation on such a large scale is a first of its kind.

Keywords: Community detection; Graph partitioning; YouTube; Content discovery.

1 Introduction

Detecting communities or clusters in real-world graphs such as social networks, web graphs, and biological networks is an important problem that has been attracting a great deal of attention in recent years (Clauset, Newman, and Moore 2004; Fortunato 2009; Girvan and Newman 2002; Karrer, Levina, and Newman 2008; Lancichinetti and Fortunato 2009). Many real-world graphs decompose naturally into communities where nodes are densely connected within the community and have much sparser connection between the communities. The communities from large networks carry great scientific and practical value because they typically correspond to behavior or functional units of the network, such as social groups in a social network. Community detection provides us a valuable tool to analyze network structure and better understand complex networks as well as provide better exploration and browsing tools for very large collections. In this paper, we perform community detection on the YouTube online video community and address challenging issues of working with such a very large real-world graph.

By modeling complex networks as graphs, the community detection problem is typically modeled as a graph partitioning problem, where a community or cluster is a set of nodes in the graph that have more edges linking among its members than edges linked to the rest of the graph. Depending on whether every node in the graph or only a subset of the nodes are assigned to a cluster at the end, graph partitioning algorithms can be roughly divided into two categories: global and local algorithms. We review some representative methods below. More extensive survey on the large body of community detection work can be found at (Fortunato 2009; Lancichinetti and Fortunato 2009; Schaeffer 2007).

In global clustering, each node of the graph is assigned a cluster in the output of the method. One intuitive approach is based on the minimum-cut maximum-flow theorem. A graph can be split into two by identifying and removing the minimum cut, and a full clustering can be achieved by applying the procedure recursively. Flake et al. (2000; 2002) have used this idea to identify communities in the graph of world wide web. The work by Girvan and Newman (2002) used the concept of betweenness centrality, which measures the importance of an edge in connecting different parts of the network. Edges with highest betweenness are gradually removed to divide the graph into clusters. Another popular class of methods (Gkantsidis, Mihail, and Zegura 2003; Newman 2006; Richardson, Mucha, and Porter 2009) are based on spectral graph theory (Chung 1997; McSherry 2004), where the eigenvectors of the Laplacian matrix are used as similarity measure to perform clustering.

For large graphs such as social networks and web content graph, global approaches that require the entire graph to be accessible simultaneously do not scale well. In such settings, a more desirable approach is to use local clustering algorithms (Johnson et al. 1989; Bagrow and Bollt 2005;

Bagrow 2008; Clauset 2005; Andersen, Chung, and Lang 2006; Andersen and Lang 2006) that do not require the full knowledge of the graph and examine only a subset of the graph at a time. Local clustering algorithms typically start from one or a set of seed nodes and examine only the adjacency list of the seed nodes at a time. Anderson (2008) proposed a local algorithm for finding dense subgraphs, which is an approximation to the spectral algorithm by Kannan and Vinay (1999). Given a starting vertex, a pruned growth process quickly reveals a dense subgraph around the starting vertex, and the algorithm has complexity independent of the graph size. Recent work by Andersen and Peres (2009) improves the computational complexity over previous algorithms by simulating an evolving set process to achieve balanced cut with small conductance.

A clustering algorithm can generate either overlapping or non-overlapping clusters. The advantage of overlapping clustering over non-overlapping clustering has been studied in various applications like social network analysis (Mishra et al. 2007; Ahn, Bagrow, and Lehmann 2009), and inherent multi-assignment clustering (Streich et al.). In some settings, like discovering communities in social networks, the clusters are naturally overlapping and by restricting our attention to non-overlapping clustering, we may lose valuable information about the structure of communities in a social network. The advantage of overlapping clustering has also been observed in optimizing metrics such as density and conductance, and in light of hardness results, polylogarithmic approximation algorithms have been developed for these problems (Khandekar, Kortsarz, and Mirrokni 2010). Similar graph partitioning problems are already well-studied in the context of approximation algorithms, and several approximation algorithms have been developed for them (Spielman and Teng 1996; Arora, Rao, and Vazirani 2009; Leighton and Rao 1999; Andersen 2008; Andersen, Chung, and Lang 2006).

In spite of the rich literature on graph clustering, selecting the appropriate algorithm for community detection on a real-world graph is not straightforward and requires careful examination of the specific application. The global clustering methods do not scale well for huge data sets, and the local clustering methods only focus on local neighborhood in the graph or finds a good cluster somewhere in the graph ignoring the rest of the graph. Another drawback of one-stage clustering is that it may not optimize multiple desired metrics at the same time. Our goal in this paper is to design scalable multi-stage clustering algorithms that takes into account various metrics and also take advantage of scalability of local algorithms and, at the same time, output clusters that cover majority of the graph (not only a local neighborhood). This is important if the resulting clustering is to be used in exploration and discovery within the graph. In particular, we study community detection for the YouTube online video community with the objective of generating named video clusters such that the videos in the same cluster correspond to a same topic and have similar content.

The large number of videos on YouTube rules out the possibility of global clustering methods, which require knowledge of the whole graph and are not easily parallelizable. To perform clustering over large graphs such as YouTube, local algorithms need to be properly adapted to allow efficient parallel implementation. In particular, we design a multi-stage clustering algorithm by pre-processing the graph, running local clustering algorithms in parallel on different parts of the graph, and then post-processing the output clusters in order to get more useful results. We will formally define the multi-stage algorithm and study the algorithmic problem in each stage separately and combined. In order to evaluate our results, we consider various metrics capturing coverage, coherence, and connectivity of those clusters. Using these metrics, we compare different local clustering algorithms and design pre-processing and post-processing strategies to get coherent video clusters. These auto-generated cluster of videos can be used to better organize YouTube videos and help users better discover and browse interesting topics on YouTube. Our main contribution includes using existing state-of-the-art clustering as a building block to design a practical and efficient multi-stage clustering system to detect communities on a very large real-world graph with specific challenges and produce useful results. In particular, we apply the idea of large-scale local algorithms for the purpose of global clustering. To do so, we find a set of seed nodes in a careful way, and run the local clustering algorithm in parallel from all those seed nodes. Finally, we prune the set of clusters computed in the algorithm in the post-processing step. This approach is appropriate for clustering of large graphs like this, and this paper is the first to use such a multi-stage algorithm.

The rest of the paper is organized as follows. In Section 2, we present the overall framework of our system. Section 3 presents preliminaries on various graph metrics. Different clustering stages, such as pre-processing, local clustering, and post-processing are discussed in Sections 4, 5, and 6, respectively. We describe our experiments and present the results in Section 7. We end with conclusions and avenues for future work in Section 8.

2 Overall framework

We consider the YouTube graph, where each video is a vertex and the edge between vertices captures their similarity. There are different ways to define the similarity of videos. One approach is to use content-similarity to create the edge, where either or both of text or audio-visual content can be used. In this paper we use the graph induced by co-watching of videos by users in anonymous user sessions. Two videos that have a higher co-watch value will be considered more similar. We limit the number of co-watched videos to keep the graph sparse. Note that we construct the YouTube video graph based on co-watch statistics but also use text features to refine the clustering.

The framework for community detection is illustrated in Fig. 1. Due to the large size of the graph that we are considering, it is necessary to apply local partitioning algorithms with efficient parallel implementation. A pre-processing step is employed to select seed videos from which the local cluster will be grown. The seed videos are selected to be further apart in the sense that seed videos should not appear as a close neighbor of other seed videos. Such a pre-processing

step ensures that local clusters are grown with small overlap and thus reduce the amount of overall work.

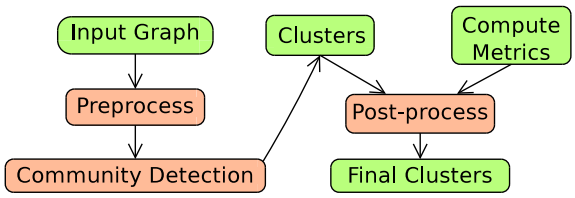

Figure 1: Framework of community detection on YouTube

After the pre-processing step, local partitioning algorithms are run on the selected seed videos in parallel. Each seed video is grown into a local cluster independently which permits overlap among clusters. It is desirable to allow overlapping clusters because the rich content of videos allow them to be related to several topics simultaneously. This is another advantage of using local clustering algorithms over global algorithms. To the best of our knowledge, our implementation of finding seed nodes and then local clustering algorithms is a first combined implementation of this type for the purpose of global clustering at this scale.

Local algorithms have the advantage of being scalable, but may not achieve global optima because the algorithm may not have the knowledge of the whole graph during clustering. For example, the local clustering algorithms used in this paper generate clusters that have either high density or small local conductance, but the average conductance and content coherence of all the clusters may not be optimum. Therefore, we apply a post-processing step to further refine the clustering result. In post-processing, each cluster from the local clustering step is further divided into smaller sets to optimize text-based coherence metric and then globally merged to combine duplicate clusters. These newly-formed clusters have more coherent content and form the final output of our community detection framework. A short name is suggested for each cluster based on its text features and used to assist browsing.

3 Preliminaries

In this section, we present formal definitions for the coverage, connectivity and coherence metrics used in evaluating the quality of clusters. Consider a similarity graph[1] $G(V, E)$ over the set of videos. Given a cluster $C \subseteq V$, let $E(C) = \{(v, u) \in E(G) | v, u \in C\}$, and volume of C be $\text{vol}(C) = \sum_{v \in C} \text{degree}(v)$. The density of cluster C is

$$\text{density}(C) = \frac{|E(C)|}{|C| \cdot (|C| - 1)/2},$$

and the conductance of this cluster is the ratio between the size of the cut outgoing from C and the volume of C, i.e.,

$$\text{conductance}(C) = \frac{\text{vol}(C) - 2|E(C)|}{\text{vol}(C)}.$$

[1]This could be the co-watched-video graph or similarity graph based on similarity of features associated with videos.

In addition to graph metrics, we also define the text coherence of the cluster $\mathcal{C}$ as

$$\text{coherence}(t) = \frac{|T_1 \cup T_2 \cup \cdots \cup T_t|}{|\mathcal{C}|}. \quad (1)$$

Here, T_i is the subset of $\mathcal{C}$ whose videos contain the ith most frequent text term of the cluster. A coherent cluster will have high coherence(t) for small t, i.e., a few text terms cover majority of the videos in the cluster. This text coherence can be considered as an application level metric to evaluate cluster quality for real-world applications.

The above metrics are defined for each cluster. Now given a set of clusters $\mathcal{C} = (C_1, C_2, \ldots, C_k)$, one can define connectivity metrics for the set of overlapping clusters based on the above connectivity metrics. The average-conductance of clusters in $\mathcal{C}$ is

$$\text{avg-conductance}(\mathcal{C}) = \frac{\sum_{i=1}^{k} \text{conductance}(C_i)}{k}.$$

Also, the average density of clusters in $\mathcal{C}$ is

$$\text{avg-density}(\mathcal{C}) = \frac{\sum_{i=1}^{k} \text{density}(C_i)}{k}.$$

In addition, the total coverage of the clusters is the total number of nodes covered by these clusters, i.e.,

$$\text{coverage}(\mathcal{C}) = |\cup_{i=1}^{k} C_i|.$$

Naturally, our goal is to find clusters that are internally well-connected and externally less connected, so we would like to find clusters with high density, low conductance, high coherence and overall result low average conductance and high ratio of coverage($\mathcal{C}$)/size($\mathcal{C}$), where size($\mathcal{C}$) $= \sum_{1 \le i \le k} |C_i|$,.

Below, we will describe the three steps of pre-processing, local clustering, and post-processing in details.

4 Pre-processing

In order to obtain clusters that cover a majority of the entire graph, a pre-processing step can be used to select an optimum set of seed videos to apply the local partitioning algorithm. A naive approach is to take every video in the graph as a seed and grow a cluster around it. Such an approach is computationally expensive and will generate a large number of duplicate clusters. Therefore, the objective of the pre-processing step is to find a set of seed videos $\{s_1, s_2, \ldots, s_k\}$ such that the clusters $\{C_1, C_2, \ldots, C_k\}$ generated around these seed videos can cover majority of the graph but have small overlap among the clusters.

We can formally define the pre-processing step as selecting k nodes from the graph $G = (V, E)$ such that the ratio $\frac{\text{coverage}(\mathcal{C})}{\text{size}(\mathcal{C})}$ is maximized. we observe that even with simplification the problem of choosing a set of seed nodes to maximize the total coverage of $\mathcal{C}$ is NP-hard. In fact, this problem subsumes the maximum coverage problem which is not approximable within a factor better than $1 - \frac{1}{e}$ under reasonable complexity theoretic assumptions (Feige 1998). Furthermore, the local partitioning algorithm that grows a cluster around a seed node usually has some robustness on

the seed node selection, i.e., the clusters generated from a set of close-by nodes in the graph will be highly likely to be the same or have very large overlap. Therefore, instead of attempting to formulate an optimization problem and finding the optimum set of seed videos, we take a heuristic approach on seed video selection and use a post-processing step later on to improve the quality of clustering.

The criterion of seed selection is to select further apart nodes so that the clusters generated around those seeds will have small overlap. There is also a higher priority to cover videos that are more important in the graph. We evaluate the importance of a video by its popularity, which is computed based on statistics such as number of views, number of user comments, etc. In our heuristic approach, we rank the videos in the graph by its popularity and select the most popular videos as seed videos to expand. To ensure seed videos are well separated, we will examine the h-hop neighbors of the already selected seed videos and make sure that a newly added seed video will not appear in the neighbors of existing seed videos. Here, we proposed a simple greedy algorithm inspired by the greedy algorithm for the maximum coverage problem. However, the order at which we examine the videos in the greedy algorithm is based on a popularity measure (based on the number of video plays). To be more specific, the seed selection algorithm works as follows:

Heuristic seed selection algorithm

Input: The list of videos in the graph that are ranked by their popularities as $(v_1, v_2, \cdots, v_n)$.

Output: A set of seed videos $(s_1, s_2, \cdots, s_k)$.

1. **Initialize:** $s_1 = v_1,\ N = \mathcal{N}(s_1),\ S = (s_1),\ i = 1$
2. **While** $|S| < k$
3. $\quad i \leftarrow i + 1$
4. $\quad$ **If** $v_i \in N$
5. $\quad\quad$ Continue
6. $\quad$ **Else**
7. $\quad\quad S \leftarrow S \cup \{v_i\}$
8. $\quad\quad N \leftarrow N \cup \mathcal{N}(v_i)$

Here the function $\mathcal{N}(v_i)$ returns the list of videos that are in the neighborhood of v_i in the graph. The neighborhood might be immediate neighbors or h-hop away. The larger h we choose, the better separation of the seed videos but also higher computational complexity. One can also use other characteristics of the videos other than the co-watched graph to compute the neighborhood. There is also trade-off in choosing the parameter k, i.e., the number of seed videos. A rough estimate of k can be obtained by dividing the size of the graph by the average size of a cluster. However, it is natural for a video to belong to several clusters of related topics and therefore, it is desirable to allow some overlap among clusters. Choosing a larger k will give better coverage of the graph but also more overlap among clusters, while choosing a small k may risk not covering enough videos in the graph. We set the value k and h empirically in this paper.

In addition to seed selection, in the pre-processing step, we also compute auxiliary features for each video in the graph. We use a list of text terms as the auxiliary feature in this paper. These text terms are extracted from the title, user-provided tags, and user comments to represent the content of the video. These text features are used in the graph clustering step to evaluate the text coherence of the cluster during its growth process and determine the proper termination condition. Text coherence is defined in Section 3.

5 Graph clustering

After the set of seed videos are selected in the pre-processing step, we run local partitioning algorithms on each of the seed video in a parallel fashion. Local partitioning algorithm generates a local cluster of high density and/or small conductance around the seed video. From the computational complexity perspective, finding optimum partitioning problems are NP-hard, and despite a lot of research in this area no constant-factor approximation algorithm is known for these problems (Leighton and Rao 1999; Arora, Rao, and Vazirani 2009; Feige, Peleg, and Kortsarz 2001). In light of such hardness results, several heuristic and approximation algorithms have been developed for these problems (Spielman and Teng 1996; Feige, Peleg, and Kortsarz 2001; Leighton and Rao 1999; Andersen and Peres 2009). Among these heuristics and approximation algorithms, we chose two algorithms that are scalable for large data sets and are suitable for running in a MapReduce-style (Dean and Ghemawat 2004) distributed-processing infrastructure. In particular, we compare two local partitioning algorithms by R. Andersen (2008; 2009), which optimize two different graph metrics, i.e., the density and conductance of the clusters, respectively. Below, we briefly describe the two algorithms and their adaptations in our work.

The first algorithm by Andersen (2008) is built upon the spectral technique developed by Kannan and Vinay (1999), which exploits the close relationship between the densest subgraph and the largest eigenvalue of the graph's adjacency matrix. A deterministic process called 'pruned growth process' is used to generate a sequence of vectors by successively multiplying the vector with the adjacency matrix followed by pruning. For a graph with n nodes and a starting node v_i, the initial vector $x_0 \in R^n$ is set to be all 0 except the ith position. The pruned growth process is defined as

$$x_{t+1} = \text{prune}(x_t \cdot A), \tag{2}$$

where A is the adjacency matrix of the graph and prune(x) essentially sets smaller elements in x to be 0. Through iterations, neighbors of existing nodes in the cluster will be added to the cluster and nodes with more neighbors will accumulate higher values in the corresponding elements of x. Therefore, after pruning, only nodes with high degrees are retained in the resulting dense subgraph.

Given the large number of YouTube videos, we need to make some adaptations in order to efficiently compute the pruned growth process. First, in the cowatch video graph that we use, we consider the top 60 immediate neighbors of each video ranked by their cowatch frequency. These top cowatch neighbors of a video are expected to have similar content. Second, to avoid the exponential growth of the cluster, we expand a fixed number of top weighted nodes from the current cluster at each iteration. In addition to the above

adaptations of reducing computational complexity, we also bring some randomness into the growth process by selecting the top weighted nodes to expand with a probability proportional to its weight. During the growth process, we compute and monitor the quality of the cluster in terms of text coherence. Once the cluster quality exhibits a desreasing trend, we revert back to the previously-known best cluster and restart the growing process from there. Finally, a cluster is generated once it reaches the desired density or exceeds the maximum allowed iterations.

In addition to density, conductance is also an important metric for cluster quality. A small conductance value indicates more edges are within the cluster than going out the cluster. The second algorithm (Andersen and Peres 2009) that we use simulates a volume-biased evolving set process to produce clusters of low conductance. The evolving set process is a Markov chain on subsets of the vertex set V. Given the current state of the cluster $\mathcal{C}^t$, the next state of the cluster $\mathcal{C}^{t+1}$ will be updated by the following rule: a threshold U is uniformly chosen at random from the interval $[0, 1]$. Let the set $\mathcal{B}_1 = \{v \in \mathcal{C}^t : p(v, \mathcal{C}^t) \leq U\}$ and $\mathcal{B}_2 = \{v \notin \mathcal{C}^t : p(v, \mathcal{C}^t) \geq U\}$. The updated cluster will be $\mathcal{C}^{t+1} = (\mathcal{C}^t - B_1) \cup B_2$. The $p(v, \mathcal{C})$ denotes the transition probability of the node v to the cluster $\mathcal{C}$ and is defined as

$$p(v, \mathcal{C}) = \frac{1}{2}(\frac{e(v, \mathcal{C})}{d(v)} + \mathbf{1}(v \in \mathcal{C})), \quad (3)$$

where $e(v, \mathcal{C})$ denotes the number of edges between node v and cluster $\mathcal{C}$. $d(v)$ is the degree of node v. The cluster growth process statistically adds new nodes that have dense connection to the cluster and remove nodes with few edges from the cluster. A final cluster is generated if it reaches the desired conductance or the cluster size is too large. This local partitioning algorithm based on evolving set has improved computational complexity over the best previous local partitioning algorithm by Andersen et al. (2006).

Both the two algorithms grow a local cluster from a given vertex, which makes it possible for parallel implemenation. Since the set of seed videos are generated by a heuristic approach from the most popular videos, it is possible that some seed videos may produce highly overlapping clusters. To avoid generating largely duplicate clusters, we maintain a status table recording nodes that have already been included in some clusters. During cluster generation, if a seed video or its close neighbors are already included in other clusters, we stop growing a cluster around this seed video to avoid doing duplicate computations.

6 Post-processing

The local partitioning algorithms in the previous section produce clusters of high density or low conductance around the seed videos. The cluster generation process in both algorithms grow the cluster by adding neighbors of existing nodes and shrink the cluster by removing nodes with small degrees. The advantage of such a process is that it is computationally efficient and requires only local knowledge of the graph. The disadvantage is that there is no capability of splitting a cluster during its generation. This is a problem when two videos of different topics are added into the cluster in the early stage of cluster generation, these two videos will attract their neighbors into the cluster to form two subclusters that may have few edges connecting them. An example is given in Fig. 2, where the top 500 videos with highest degrees in one of the generated clusters are shown. In the figure, the nodes with the same color seem to form dense subclusters within the cluster. By further looking into the corresponding videos, we do observe that in many cases, videos within the same subcluster correspond to the same topic, while videos in different subclusters are related to different topics.

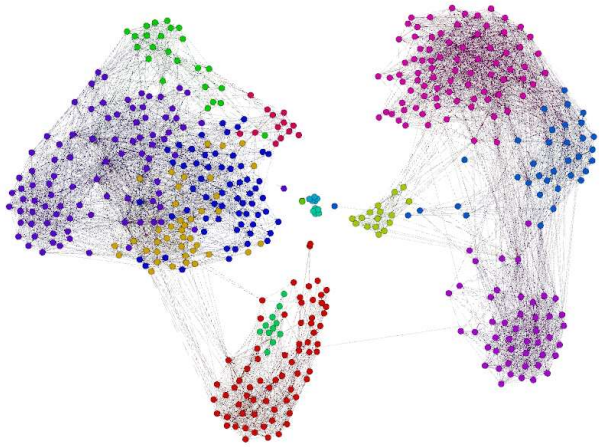

Figure 2: Top 500 videos with highest degrees in one of the generated clusters from the local partitioning algorithm. The colors represent different modularity classes of the cluster.

The existence of subclusters in the generated clusters implies that there might be several topics within the same cluster and therefore, the coherence of the cluster should be further improved. We take one of the generated clusters and compute its most popular text terms associated with the videos in the cluster. The table with the top 8 text terms and their occurrence frequency is show below.

Text term	Occurrence	Text term	Occurrence
Pocoyo	719	Dog	123
Baby	539	Charlie	122
Donald duck	493	Song	98
Mickey mouse	360	Discovery	94
Funny	161	Laughing	94

Although the cluster seems to be generally related to cartoon animation and babies, there are clearly several topics within the cluster and it is desirable to separate those topics to generate more coherent clusters.

Given the above observations, we apply post-processing on the clusters from the local partitioning step to obtain more coherent and useful clusters. The first step of post-processing is to split the cluster into subclusters that have dense connection within the same subcluster and fewer connections among different subclusters. At this step, we use the text coherence measure to guide the process of dividing

a potentially diverse cluster into several smaller but more coherent subclusters. Higher coherence makes it easier to suggest a good name for each cluster and better browsing experience. After splitting all the clusters into more coherent subclusters, we apply an iterative global algorithm to combine highly overlapping clusters and remove clusters that are too small to stand on their own. Below we describe in more detail these two steps of post-processing.

Cluster refinement using text coherence As discussed above, a cluster generated from the local partitioning algorithm might still contain diverse content due to the growing nature of the clustering. To identify denser subclusters within a cluster, we first compute text statistics over the cluster. More specifically, we extract the most representative text terms for each video based on its title and descriptions, then compute the occurrence frequency for each of the text terms over the entire cluster. Denote the top occurring text terms for the cluster $\mathcal{C}$ by $t_1, t_2, \cdots, t_k$, we can obtain k sets of videos $\{S_1, S_2, \cdots, S_k\}$ that contain each of the top k terms, i.e., S_i is a set of videos in the cluster that all contain the text term t_i. The sets $\{S_i\}$ can be considered as coherent clusters each related to a certain topic and serve as a good first-step partitioning of a potentially larger and diverse cluster. However, considering only single text term has the limitation that it might ignore bigrams or semantically similar terms. For example, a cluster of Micky Mouse is interesting but further dividing it into two subclusters containing Micky and Mouse separately is not desirable. To identify bigrams, we iteratively compare every two sets S_i and S_j to compute their overlap, a large overlap indicates the two text terms t_i and t_j typically appear together, therefore it is highly likely that they are bigrams. We then combine the two sets S_i and S_j into one cluster. To identify semantically similar terms such as Cars and Automobiles, we compute the semantic similarity between two text terms or two sets of text terms, and then merge two clusters if their text similarity is larger than a threshold (the text similarity is obtained from latent-topic modeling over a large corpus of text data and is not essential to this description). After merging bigrams and similar terms, we obtain the final set of subclusters that have high text coherence and each corresponds to a different topic.

Global cluster merging Given that each cluster $\mathcal{C}_i$ has been divided into a set of subclusters $\tilde{C}_{i1}, \tilde{C}_{i2}, \cdots, \tilde{C}_{ik}$, the last step in our post-processing is to combine duplicate subclusters from different clusters, i.e., comparing $\tilde{\mathcal{C}}_{i*}$ and $\tilde{\mathcal{C}}_{j*}$ for $i \neq j$, and remove subclusters that are too small to stand on their own. To combine duplicate clusters, we iteratively compare any two clusters $\tilde{\mathcal{C}}_{i*}$ and $\tilde{\mathcal{C}}_{j*}$. If their overlap is larger than certain threshold, i.e., $|\tilde{\mathcal{C}}_{i*} \cap \tilde{\mathcal{C}}_{j*}| / \min(|\tilde{\mathcal{C}}_{i*}|, |\tilde{\mathcal{C}}_{j*}|) > t$, these two clusters will be merged into one. The final set of clusters will have higher text coherence, smaller overlap, and high coverage of the whole YouTube graph. In the next section, we evaluate the quality of clustering using different metrics.

7 Experiments

Experiment Setup We carry out the multi-stage clustering algorithm on a co-watch graph of tens of millions of YouTube videos (the system can scale to larger graphs as well). In the pre-processing step, we sequentially select the top 50,000 popular videos that are not neighbors of existing seeds as seed videos. The local clustering algorithms grow clusters around each seed video in parallel. Each local clustering process terminates whenever a cluster has either reached a specified density or conductance value or exceeded a maximum allowed cluster size of 30,000.

Cluster statistics In Fig. 3, we show the number of clusters obtained after local clustering, post-processing splitting, and the final merging step. DP stands for the dense partition clustering and ES stands for the evolving set algorithm.

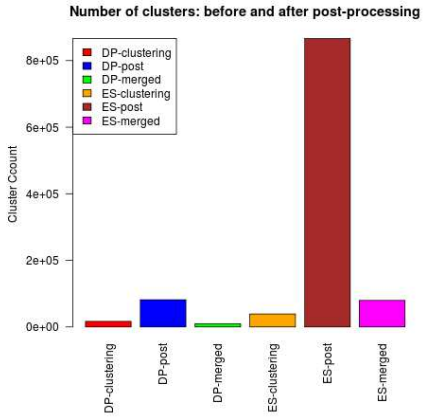

Figure 3: Number of clusters after each step of the algorithm

After local clustering, we have a relatively small number of clusters, most of which are large. After splitting each cluster based on text features, the cluster number is greatly increased and each cluster becomes smaller in size and more coherent in content. The evolving set algorithm produces many more clusters after splitting than the dense partitioning algorithm. This could be an indication that the ES algorithm tends to generate more diverse clusters in the first place. However, compared to the DP algorithm, such a high number of clusters after splitting may also indicate that there are many overlapping clusters that need to be merged. Therefore, the global merging step is important to combine overlapping clusters and related clusters that correspond to bi-grams or similar text terms. After merging, the number of clusters is greatly reduced to a few thousands. This number is a reasonable approximation to the number of popular topics on YouTube.

Comparison of local clustering algorithms We now compare the two local clustering algorithms more closely using different metrics. After the local clustering stage, the average cluster sizes for the DP and ES algorithms are 10190

and 33450, respectively. Each node appears in 8.2 and 11.9 clusters on average for DP and ES, respectively. From our observation, the ES algorithm runs faster than the DP algorithm, but it tends to generate clusters of larger size and more overlapping clusters. The average density is 0.0196 for ES and 0.00056 for DP, while the average conductance is 0.488 for ES and 0.813 for DP. Higher density and lower conductance indicate better clustering. Since average value may be dominated by a few large items, we also look at the median of both metrics. In this case, DP has better density than ES, while ES has better conductance than DP. At this stage, many clusters have large size and potentially diverse content, mixed from more than one topics. To improve cluster coherence, the post-processing step splits clusters based on text features and merges the resulted smaller clusters based on their overlaps.

In Figures 4 and 5, we report the distribution of density and conductance for DP and ES both before and after post-processing. ES outperforms DP in both density and conductance. After post-processing, the coherence of clusters is greatly improved. The average percentage of videos in each cluster covered by the top 50 terms is 94% for DP and 99.6% for ES after post-processing. While optimizing the text coherence measure, we observe that the post-processing step increases the average and median conductance for both ES and DP. In terms of density, post-processing decreases the average but increases the median density for ES, increases both the average and median density for DP. The overall effect of post-process tends to increase the conductance, but improves the density and text coherency. This comparison of cluster metrics before and after post-processing demonstrate the effect of post-processing and also the advantage of multi-stage algorithm over single-stage ones.

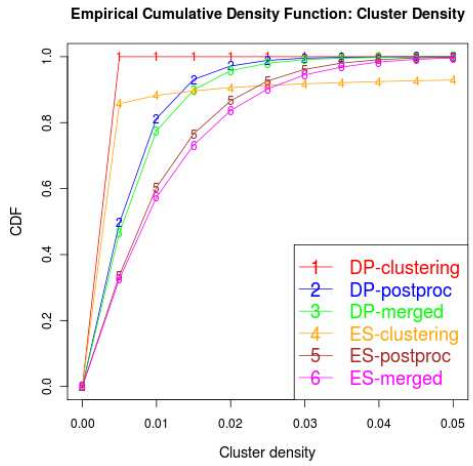

Figure 4: Density of clusters

Cluster naming Generating clusters of a large graph only aids in exploration and discovery if these clusters are suitably named. Finding a good name for a large group of videos is not trivial. One possible approach is to choose the most

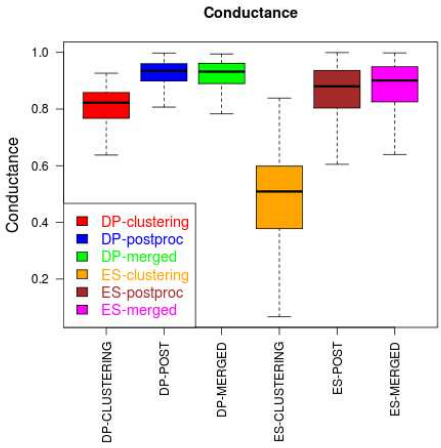

Figure 5: Conductance of clusters

frequently occurring tag in the cluster. This resulted in fairly poor names in our experiments, as the most frequent tag is often a very short word such as ”2L” in a cluster about small sporty (2-liter) cars. As an alternative approach we use entities extracted from the titles of the videos in a cluster to name the cluster. The entities used are based on the Freebase structured data repository (Metaweb Inc.). We extract entities and use the top few entities in a cluster by occurrence to name the cluster. This worked well for many clusters. The table below shows some sample clusters with their size, some sample video titles, and the name assigned based on entity annotation. This approach does not work as well for clusters that, while thematically coherent, do not correspond to something as easily identifiable as an entity. An example is the last line in the table where the cluster has videos showcasing different places of the world at different times, which is a concept not easily captured by an entity.

Size	Sample titles	Annotation Name
3823	16V Turbo 4Motion, Renault Clio 1.6 16V 110hp, megane 2.0 16v vs civic vti.avi, Golf V 1,4 16v 80 PS 0-100	Renault Megane-Coupe-Volkswagen Golf Mk2-Volkswagen Golf,
434	J.S Bach prelude from suite, BWV 1007, Bach - Cello Suite BWV 1007 on Bass	Sebastian-Johann Sebastian Bach
1383	1968 Red Camaro Big Block 4spd Fully restored, 1968 Camaro RS/SS, 1971 Plymouth Cuda Convertible Burnouts	Plymouth-Chevrolet Camaro,
716	Salsa Aerobic, Dance Special Rdesheim with schweppy!!!, Dance Aerobic - Choreography - Latino, Aerobic - Mambo	Aerobic exercise-Aerobics,
4727	The WORLD LIVE - 08:00 GMT on September 23, 2008, The WORLD LIVE - 21:00 GMT on January 31, 2010	Germany-United States of America-Greenwich Mean Time-Europe

More results and sample clusters can be found on the web at http://sites.google.com/site/ytcommunity. We have built a cluster browser to allow us to explore YouTube using the named clusters. Both the name and the top weighted videos of each cluster are sufficient to glean the content of a cluster for browsing purposes. Subjective evaluation of the resulting clusters indicates that they have good content coherence.

8 Conclusions

In this paper, we propose a multi–stage community detection algorithm for large–scale YouTube video graphs. Local partitioning algorithms implemented in a parallel fashion are used to efficiently generate clusters that cover large portions of the graph. Pre-processing and post-processing steps are used to optimize multiple graph–connectivity and coherence metrics, such as conductance, coverage, and a new text coherence measure. We perform clustering over tens of millions of YouTube videos, scalable to larger graphs, and produce very coherent clusters with good coverage. We label clusters using entities extracted from the titles of constituent videos. These named clusters can be used to improve content discovery on YouTube.

Avenues for future work include: clustering content-similarity video graphs; better naming and representation of clusters—while adequate, entity-based annotation has some limitations; and topic clustering—during clustering and post-processing, we obtain relations between different clusters in terms of the number of edges between related clusters. If we assume each cluster corresponds to a topic, such relations form a super-graph with topics as nodes and their relations as edges. Given the relatively small size of this super-graph, a global clustering algorithm could be used to discover topic structure.

References

Ahn, Y.-Y.; Bagrow, J. P.; and Lehmann, S. 2009. Link communities reveal multiscale complexity in networks. *Nature* 466:761764.

Andersen, R., and Lang, K. 2006. Communities from seed sets. In *WWW'06*, 223–232.

Andersen, R., and Peres, Y. 2009. Finding sparse cuts locally using evolving sets. In *STOC '09*, 235–244.

Andersen, R.; Chung, R.; and Lang, K. 2006. Local graph partitioning using pagerank vectors. In *FOCS'06*, 475–486.

Andersen, R. 2008. A local algorithm for finding dense subgraphs. In *SODA '08: Proc. of the 19th annual ACM-SIAM symposium on Discrete algorithms*, 1003–1009.

Arora, S.; Rao, S.; and Vazirani, U. V. 2009. Expander flows, geometric embeddings and graph partitioning. *J. ACM* 56(2).

Bagrow, J. P., and Bollt, E. M. 2005. A local method for detecting communities. *Physical Rview E* 72:046108.

Bagrow, J. P. 2008. Evaluating local community methods in networks. *Journal of Statistical Mechanics: Theory and Experiment* P05001.

Chung, F. 1997. *Spectral graph theory*. American mathematical society.

Clauset, A.; Newman, M.; and Moore, C. 2004. Finding community structure in very large networks. *Physical Review E* 70:066111.

Clauset, A. 2005. Finding local community structure in networks. *Physical Review E* 72:026132.

Dean, J., and Ghemawat, S. 2004. Mapreduce: simiplified data processing on large clusters. In *OSDI'04: 6th symposium on operating system design and implementation*.

Feige, U.; Peleg, D.; and Kortsarz, G. 2001. The dense -subgraph problem. *Algorithmica* 29(3):410–421.

Feige, U. 1998. A threshold of ln for approximating set cover. *J. ACM* 45(4):634–652.

Flake, G.; Lawrence, S.; Giles, C.; and Coetzee, F. 2002. Self-organization and identification of web communities. *IEEE Computer* 35(3):66–71.

Flake, G.; Lawrence, S.; and Giles, C. 2000. Efficient identification of web communities. In *Proc. of the Intl. Conf. on Knowledge Discovery and Data Mining*, 150–160.

Fortunato, S. 2009. Community detection in graphs. *arXiv:0906.0612*.

Girvan, M., and Newman, M. 2002. Community structure in social and biological networks. In *Proc. of the National Academy of Sciences of the United States of America*, 7821–7826.

Gkantsidis, C.; Mihail, M.; and Zegura, E. 2003. Spectral analysis of internet topologies. In *Proc. of the 22nd joint conf. of IEEE computer and communication societies, INFOCOM*, volume 1.

Johnson, D. S.; Aragon, C. R.; McGeoch, L. A.; and Schevon, C. 1989. Optimization by simulated annealing: an experimental evaluation. Part I, graph partitioning. *Operations Research* 37(6):865–892.

Kannan, R., and Vinay, V. 1999. Analyzing the structure of large graphs. In *Manuscript*.

Karrer, B.; Levina, E.; and Newman, M. 2008. Robustness of community structure in networks. *Physical Review E* 77:046119.

Khandekar, R.; Kortsarz, G.; and Mirrokni, V. 2010. Overlapping vs. non-overlapping clustering: Conductance and density.

Lancichinetti, A., and Fortunato, S. 2009. Community detection algorithms: a comparative analysis. *arXiv:0908.1062*.

Leighton, F. T., and Rao, S. 1999. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM* 46(6):787–832.

McSherry, F. 2004. Spectral methods for data analysis.

Metaweb Inc. The freebase open, creative-commons licensed repository of structured data. http://www.freebase.com.

Mishra, N.; Schreiber, R.; Stanton, I.; and Tarjan, R. E. 2007. Clustering social networks. In *WAW*, 56–67.

Newman, M. 2006. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E* 74:036104.

Richardson, T.; Mucha, P.; and Porter, M. 2009. Spectral tripartitioning of networks. *Physical Review E*.

Schaeffer, S. 2007. Graph clustering. *Computer Science Review* 1(1):27-64.

Spielman, D. A., and Teng, S.-H. 1996. Spectral partitioning works: Planar graphs and finite element meshes. In *FOCS*, 96–105.

Streich, A. P.; Frank, M.; Basin, D.; and Buhmann, J. M. Multi-assignment clustering for boolean data. In *ICML'09*, 969–976.