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

Ullas Gargi
Google, Inc.
Mountain View, CA
ullas@google.com

Wenjun Lu
University of Maryland
College Park, MD
wenjunlu@umd.edu

Vahab Mirrokni
Google, Inc.
New York, NY
mirrokni@google.com

Sangho Yoon
Google, Inc.
Mountain View, CA
shyoon@google.com

Abstract

Detecting coherent, well-connected communities in large graphs provides insight into the graph structure and can serve as the basis for content discovery. Clustering is a popular technique for community detection but global algorithms that examine the entire graph do not scale. Local algorithms are highly parallelizable but perform sub-optimally, especially in applications where we need to optimize multiple metrics. We present a multi-stage algorithm based on local-clustering that is highly scalable, combining a pre-processing stage, a local clustering stage, and a post-processing stage. We apply this to the YouTube video graph to generate named clusters of videos with coherent content. We formalize coverage, coherence, and connectivity metrics and evaluate the quality of the algorithm for large YouTube graphs. 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

Many real-world graphs decompose naturally into communities where nodes are densely connected within the community and have much sparser connection between communities. Communities typically correspond to behavioral 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 as well as provide better discovery and recommendation tools for very large collections. 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 linking outside it 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 divided into global and local algorithms. A more extensive survey on the large body of community detection work can be found at (Lancichinetti and Fortunato 2009; Schaeffer 2007).

For large graphs such as the YouTube video graph we operate on, global approaches that require the entire graph to be processed simultaneously do not scale well. Examples are methods based on minimum-cut maximum-flow (Flake, Lawrence, and Giles 2000), modularity-based clustering (Newman 2006) and spectral clustering methods (Chung 1997). A more scalable approach is to use local clustering algorithms (Bagrow 2008; Clauset 2005; Andersen and Lang 2006) that do not require the full knowledge of the graph and examine only a subset of the graph at each step. 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. We adapt two such local algorithms (Section 4).

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. Single-stage local clustering may not optimize application metrics. Our goal in this paper is to design scalable multi-stage clustering methods that take into account multiple metrics and generate clusters with good coverage—important if the resulting clustering is to be used in exploration and discovery. We post-process the results to optimize application metrics in addition to graph metrics. In particular, we study community detection for the YouTube online video community with the objective of generating named video clusters which each have a coherent topic and content. We adapt local algorithms for efficient parallel implementation and 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. 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.

2 Graph clustering framework

We consider the YouTube graph, where each video is a vertex and the edge between vertices captures their similarity, which could be defined in many ways. We use the graph induced by co-watching of videos by users in anonymized

YouTube user sessions. Two videos that have a high co-watch value will be considered 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.

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. 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 a text-based coherence metric and then a global merging step combines duplicate sub-clusters from different clusters. These merged clusters have coherent topics and can be labeled or named by entity extraction on the constituent video titles. We present formal definitions for the coverage, connectivity and coherence metrics used in evaluating the quality of clusters. Consider a similarity graph $G(V, E)$ 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., $\text{conductance}(C) = \frac{\text{vol}(C) - 2|E(C)|}{\text{vol}(C)}$. In addition to graph metrics, we also define the text coherence of the cluster $\mathcal{C}$ as $\text{coherence}(t) = \frac{|T_1 \cup T_2 \cup \cdots \cup T_t|}{|\mathcal{C}|}$ 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, and high coherence individually; and low average conductance and high ratio of $\text{coverage}(\mathcal{C})/\text{size}(\mathcal{C})$ overall, where $\text{size}(\mathcal{C}) = \sum_{1 \le i \le k} |C_i|$.

3 Pre-processing

To compute clusters that cover a majority of the graph, a pre-processing step can be used to select an optimum set of seed videos to apply local partitioning. Using every video in the graph as a seed is computationally expensive and will generate a large number of duplicate clusters. The objective of the pre-processing step is to find a set of seed videos such that the clusters C generated around these seed videos cover the graph well but have low overlap between 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. We hypothesize that the local algorithm is robust to the precise seed node selected. Therefore, instead of attempting to formulate an optimization problem, we take the following heuristic approach to seed video selection, and use a post-processing step to improve the quality of clustering. We rank videos by viewcount and iteratively add the next most popular video in the list that is not in the neighborhood of a previously added seed. This simple greedy algorithm gives us both coverage (since popular videos are well connected) and lower overlap in the resulting clusters. We stop when we have an empirically determined number of seeds (e.g. 50,000).

In addition to seed selection, we also compute auxiliary features for videos – text terms extracted from the title and descriptions. These text features are used to compute the text coherence of the cluster during its growth process and determine the proper termination condition.

4 Graph clustering

We run local partitioning algorithms on each of the selected seed videos in a parallel fashion using MapReduce. We adapt 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.

The first algorithm by Andersen (2008) (Dense Partitioning, abbreviated DP) exploits the close relationship between the densest subgraph and the largest eigenvalue of the graph's adjacency matrix. A deterministic pruned growth process is used to generate a sequence of vectors by successively multiplying a vector of cluster nodes (initially the seed) with the adjacency matrix followed by pruning. 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 the vector. Therefore, after pruning, only nodes with high degrees are retained in the resulting dense subgraph. During the growth process, we monitor the cluster quality 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.

Conductance is another important metric for cluster quality. A small conductance indicates more edges are within the cluster than leaving it. The second algorithm (Andersen and Peres 2009) (Evolving Set, abbreviated ES)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 $p(v, \mathcal{C}) = \frac{1}{2}(\frac{e(v,\mathcal{C})}{d(v)} + \mathbf{1}(v \in \mathcal{C}))$, 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.

5 Post-processing

The local partitioning algorithms in the previous section produce clusters of high density or low conductance around seed videos. There is a possibility that two videos on different topics (but still linked by user co-visitation) are added to the cluster in the early stage of cluster growth; these two videos will attract their neighbors into the cluster to form two sub-clusters that may have few edges connecting them, which can be detected by auxiliary features such as text. For example, a table with the top 8 most frequent text terms and their frequency for a sample cluster is shown 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 certain themes, there are clearly several sub-topics that would be desirable to separate. We post-process the clusters from the local partitioning step to obtain more coherent clusters by first applying a refinement process using text coherence to potentially split a cluster into sub-clusters and then applying an iterative global merging to merge sub-clusters (split off from different clusters).

Cluster refinement using text coherence To identify sub-clusters, we compute cluster text statistics. Specifically, we extract the most representative text terms for each video based on title and descriptions, then compute the occurrence frequency for each of the text terms over the entire cluster. Denoting 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 coherent clusters each related to a certain topic and serve as a good first-step partitioning. However, considering only single text terms has the limitation that it might ignore bigrams or semantically similar terms. To identify bigrams, we iteratively compare every pair of sets S_i and S_j to compute their overlap; a large overlap indicates the two text terms t_i and t_j are correlated, therefore they are likely bigrams, in which case we combine S_i and S_j. 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 sub-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 sub-clusters for bigrams and similar terms, we obtain the final set of sub-clusters that have high text coherence.

Global cluster merging Given that each cluster $\mathcal{C}_i$ has been divided into a set of coherent sub-clusters $\tilde{C}_{i1}, \tilde{C}_{i2}, \cdots, \tilde{C}_{ik}$, the last step in our post-processing is to combine duplicate sub-clusters from different clusters, i.e., comparing $\tilde{\mathcal{C}}_{i*}$ and $\tilde{\mathcal{C}}_{j*}$ for $i \neq j$, and merging them if their overlap is larger than a threshold. The final set of clusters will have higher text coherence, smaller overlap, and high coverage of the whole YouTube graph.

6 Experiments

Experiment Setup We clustered a co-watch graph of tens of millions of YouTube videos, using 50,000 popular videos as seeds. Each local clustering process terminates whenever a cluster reaches a specified density or conductance or exceeds a maximum cluster size, empirically set to 30,000 (cluster splitting and global merging make our method robust to precise values).

Cluster statistics Fig. 1 shows the number of clusters obtained after local clustering, post-processing splitting, and the final merging step. 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. ES produces many more clusters after splitting than the dense partitioning algorithm. This could be an indication that ES tends to generate more diverse clusters in the first place. However, compared to DP, such a high number of clusters after splitting may also indicate that there are many overlapping clusters that need to be merged. Therefore the need for a global merging step. After merging, the number of clusters is greatly reduced to a few tens of thousands, a suitable number for topic modeling or discovery applications.

Comparison of local clustering algorithms 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.

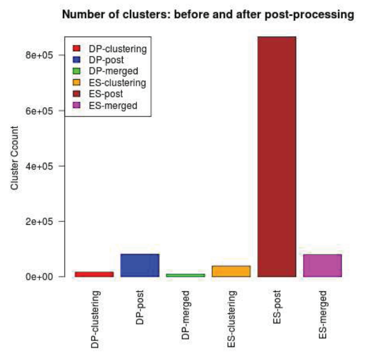

Figure 1: Number of clusters after each step of the algorithm

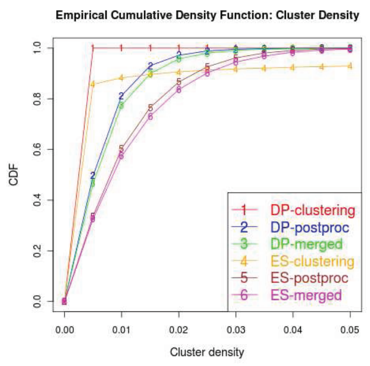

Figure 2: Density of clusters

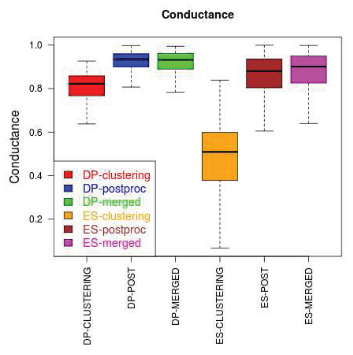

Figure 3: Conductance of clusters

Higher density and lower conductance indicate better clustering. Comparing the medians, 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 topic.

Figures 2 and 3 show 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, and increases both the average and median density for DP. The overall effect of post-processing is 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 a multi-stage algorithm over single-stage ones.

Cluster naming

Size	Sample titles	Annotation Name
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,

Assigning meaningful names to clusters allows use for content discovery and gives us confidence in the value of the clustering. We use entities defined in the Freebase structured data repository (Metaweb Inc.) and extracted from the titles of the videos in a cluster to name the cluster. The table below shows sample clusters with their size, sample video titles, and the assigned cluster name. This approach does not work as well for clusters that, while thematically coherent, do not correspond to something as easily identifiable as an entity.

7 Conclusions

We have presented a scalable multi–stage graph clustering algorithm and applied it to 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 and produce very coherent clusters with good coverage. These clusters can be aptly labeled by entity annotation and are useful for content discovery.

Avenues for future work include: using clusters for personalization; clustering audio-visual content-similarity video graphs; better naming and representation of clusters; and topic clustering over the graph of clusters.

References

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

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

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.

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

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

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

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