Shopping for Products You Don't Know You Need

Srikanth Jagabathula*
EECS, MIT
Cambridge, MA
jskanth@mit.edu

Nina Mishra
Microsoft Research
Mountain View, CA
ninam@microsoft.com

Sreenivas Gollapudi
Microsoft Research
Mountain View, CA
sreenig@microsoft.com

ABSTRACT

Recommendation engines today suggest one product to another, e.g., an accessory to a product. However, intent to buy often precedes a user's appearance in a commerce vertical: someone interested in buying a skateboard may have earlier searched for {varial heelflip}, a trick performed on a skateboard. This paper considers how a search engine can provide early warning of commercial intent. The naive algorithm of counting how often an interest precedes a commercial query is not sufficient due to the number of related ways of expressing an interest. Thus, methods are needed for finding sets of queries where all pairs are related, what we call a *query community*, and this is the technical contribution of the paper. We describe a random model by which we obtain relationships between search queries and then prove general conditions under which we can reconstruct query communities. We propose two complementary approaches for inferring recommendations that utilize query communities in order to magnify the recommendation signal beyond what an individual query can provide. An extensive series of experiments on real search logs shows that the query communities found by our algorithm are more interesting and unexpected than a baseline of clustering the query-click graph. Also, whereas existing query suggestion algorithms are not designed for making commercial recommendations, we show that our algorithms do succeed in forecasting commercial intent. Query communities increase both the quantity and quality of recommendations.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Search Process

General Terms

Experimentation, Algorithms

Keywords

query communities, long-range recommendations

1. INTRODUCTION

Recommendation engines have been widely adopted in many commerce verticals with great success. At first launch, Amazon's recommendation engine resulted in a large increase in revenue [17]. Recommendations today typically take the form of complements and substitutes. For instance, people who buy cameras tend to also buy a case or battery (complement) and people who shop for fuji cameras also shop for canon cameras (substitute). In a sense these recommendations serve an immediate, subsequent commercial need.

Looking over the recommendation engines available today, all seem to provide suggestions within a particular commerce domain. In contrast, our hypothesis is that the intent to buy precedes a user's appearance in a shopping vertical – that a user's interests is a good predictor of what products they will purchase. This paper investigates whether user interests, as captured in search data, can be exploited to generate longer-range commercial recommendations. Note that our objective is not to recommend/advertise a specific brand and/or model number of a product, rather our goal is to help a user find a class of products related to their interests. The task of pinning down an exact product is left to a commerce vertical.

As a running example we refer to a sequence of search queries inspired by a real user shown in Table 1. The user begins with a sequence of queries related to cast iron cookware, then searches for vacation ideas in Europe in the winter, looks at the weather in Europe in the winter, then seeks a vacation in a warmer destination in the mayan riviera, hunts for tickets to fly there, and then searches for products related to the trip, i.e., sunblock and an underwater camera. The main recommendation to infer from such a user is that people who vacation to the Mayan Riviera may need an underwater camera and sunblock. However, there are many inferences we would not like to draw: (1) people who go the Mayan Riviera need cast iron cookware (2) people flying to Europe need an underwater camera. The challenge is to find the correct recommendations without introducing too many incorrect ones. To compound matters, user queries are a very noisy source of data, not all interests are necessarily expressed in the search box, and intent rapidly changes during search.

Given a search query, the problem considered in this paper is to find related commercial queries, assuming such relationships exist. The problem seems simple enough: Why not just recommend the product that most often follows an interest? One problem is that there are many ways to pose the query Mayan Riviera, e.g., vacations in the Yucatan, coastal and reef destinations in the Caribbean. Similarly, there are many queries that one could pose to buy an underwater camera. Another problem is that large co-occurrence is not necessarily an indication of relationship, the two queries could just both be popular.

To overcome the first difficulty that a user has multiple ways to express the same intent, we define a new problem, finding query communities. A *query community* is a subset of queries where all

*Work done while author was an intern at MSR Search Labs

WSDM'11, February 9–12, 2011, Hong Kong, China.

le creuset cookware	Dec 12	10:00am
cast iron reviews	Dec 12	10:01am
cast iron cookbook	Dec 12	10:02am
winter vacations	Dec 13	07:00pm
european vacation ideas in winter	Dec 13	07:01pm
weather map of europe in winter	Dec 13	07:02pm
mayan riviera trip advisor	Dec 13	07:30pm
best excursions in mayan riviera	Dec 13	07:33pm
best resort to stay in mayan riviera	Dec 13	07:45pm
mexicana airlines flights	Dec 14	07:00pm
buy water resistant sunblock	Dec 20	08:05pm
where to buy an underwater camera	Dec 21	08:15pm

Table 1: Search queries posed by a hypothetical user

pairs of queries are related. Note the key difference between a query community and a query suggestion, where in the latter the goal is to find a set of suggestions that are all related to a single query, but where there is no requirement that the suggestions themselves be related. Note also the distinction with synonymous queries, where the goal is to find queries where each is a substitute of the other. While synonymous queries form a subset of a community, they may not be rich enough to form a complete community. For example, {enchilada} and {enchiladas} are synonyms, but a community of Mexican food should ideally also include {burrito}, {tostada}, etc.

To overcome the second problem, that large co-occurrence does not necessarily imply that queries are related, we propose two methods for finding recommendations, which turn out to be complementary. One is still based on co-occurrence, but uses time to select co-occurrences – concretely, a recommendation $q \rightarrow r$ is good if it has strong support, many users pose q before r, and if q precedes r more often than r precedes q. The second method views a user's queries as a possible explanation for a commercial query. Aggregating over all users who pose a commercial query, the method seeks a succinct justification for why the commercial query was posed – modeled as a classic hitting set problem.

Contributions: The technical contribution of this paper is a method for identifying query communities in search data. We represent queries and relationships through a graph where the vertices are queries and there is an edge between queries if they are related. If the graph contained all possible relationships, then a query community is a clique in this graph. However, it turns out that any graph created from available data sources is woefully incomplete. Thus, we propose a simple random model from which our observed graph is derived from an ideal graph containing all possible relationships. The random process uniformly keeps each edge with probability p. We then propose a method that involves densifying the graph by adding edges between vertices with probability proportional to the number of neighbors they share. The resulting densified graph is then clustered using a known algorithm.

Our analysis of the method suggests: (1) as the probability p that an edge is kept increases, the more likely our method will succeed in reconstructing query communities and (2) as cluster overlap decreases, the more likely we can reconstruct communities. Specifically, as cluster overlap decreases, the more likely our method is to introduce a right edge over a wrong edge. We prove conditions under which we can reconstruct the community: the more the clusters overlap, the larger p has to be to ensure recovery. If p is too small, the cluster cannot be recovered at all.

To recommend products, we identify queries with commercial intent using [9] and then propose two methods for making a connection between search queries and commercial queries, one is based on co-occurrences and the other on hitting set, as mentioned above.

To validate our methods for finding query communities and recommendations, we conduct an extensive series of experiments on real search logs. For query communities, our experiments show that many of the communities we find are of high quality – only 2.6% are bad. Also, our densification procedure enables us to find four times as many clusters as without densification. Whereas a baseline of clustering the query-click graph produces communities where 75% are minor misspelling/word additions, our techniques find communities where 69% are interesting and unexpected. Next, we evaluate recommendations. We show that existing query suggestion algorithms are not designed for the problem of recommending commercial queries. We show that the two recommendations methods are complementary in the sense that each can find recommendations that the other cannot. Our most striking finding is that query-communities increase not only the number, but also the quality of recommendations.

2. RELATED WORK

We discuss work related to finding query communities as well as work on recommendations. While there is rich literature in these areas, our problem formulation differs and hence these methods do not directly apply to our problem.

Query Communities: The problem of finding "concepts" or "query clusters" has received attention in the literature [30, 5]. Some approaches involve clustering the query-click graph. In our experience with such techniques (Section 6.1), each cluster found usually contains very tightly related queries. The queries in a cluster tend to contain minor word additions/misspellings of each other – it is rare to find a surprising cluster. Note that if two queries are related, but they do not share a URL in the top 10, a user cannot demonstrate that they are connected with a click. Thus the performance of an algorithm that clusters the query-click graph is upper bounded by the relevance of a search engine, we seek to go beyond that. Our work uses the sequence of queries a user poses as a signal for relatedness. This signal is less limited by the relevance of the search engine.

In another related work, the analysis of our algorithm has connections to the analysis performed by [7] in the context of clustering with constraint graphs. The authors consider a classical problem of partitioning a set of points with the additional goal of respecting certain constraints on which points must and must not be included in the same partition. The authors consider the scenario when there is noise in the given constraints. Under two general noise models, they show that even a 'small amount' of noise can severely affect the process of partitioning in an adverse manner. While one of their noise models (*noisy edges* model) is similar to the random graph models we consider in this paper, the results derived there are not applicable to our problem. Specifically, the authors consider the problem of finding disjoint clusters, whereas our clusters overlap. In addition, the authors consider general clustering criteria and prove negative results on the noise levels that lead to the entire graph being grouped into one partition. In contrast, our work focuses on a specific clustering criterion, but demonstrates a positive result when a particular algorithm is used to discover query communities.

Recommendations: In the area of query suggestions, there is a plethora of methods including random walks on click graphs [6, 19, 10], snippets of search results [26], query reformulation graphs [13, 4], and clustering the query-click graph [5]. A recent paper by [25] considers the problem of clustering query refinements that represent different information needs into different groups in order to diversify query suggestions and improve them based on user session intent. At first glance, the work on query suggestions seems appli-

cable to both finding query communities and inferring commercial recommendations. However, query suggestions do not solve the community problem as we define it: query suggestions identify a collection of queries that are related to a reference query; however, all queries related to the reference query may not be related to each other. In terms of applicability to making commercial recommendations, in principle, query suggestions can do so by simply filtering out the suggestions that are not commercial. However, in our experience (Section 6.2) with a query suggestion method based on a random walk on the query-click graph [10], we find very few commercial suggestions. These techniques, while valuable for the query suggestion problem, are not designed to solve our problem of early warning of commercial intent.

Our work may have relevance in the advertising context, though we have not explored advertising in this paper. Broad-match expansion is a method for broadening the scope of a keyword to related keywords so as to match an ad to more queries than an advertiser's bids [11]. In addition to suggesting keywords to advertisers, the techniques presented in our paper can also be used to forecast future commercial intent of search users to advertisers. Such information can be extremely useful and timely to advertisers in designing more effective ad campaigns.

The discovery of longer-term relationships can be viewed as an association rule mining problem [2] wherein relationships between queries are extracted based on co-occurrence. In addition, there has been work in uncovering longer-term relationships in search queries. The work by [22] is an important step in that direction. Given a reference query, a pattern of queries distinct from a baseline aggregate is shown to uncover useful relationships. However, a given interest can be expressed via a variety of search queries. Indeed, surprising connections tend to be rarer co-occurrences, and consequently may be difficult to find, absent grouping queries into user interests/query communities first.

Finally, there is rich literature in the context of recommendation systems, which includes content-based approaches, collaborative-filtering based approaches [27, 21, 12, 15, 1], and hybrid approaches [28]. In addition, popular commercial examples include Amazon.com recommendation system [17] and Netflix movie recommendation system [3]. These methods assume access to clean data about users, e.g., feature vectors over items (books, movies, news articles, etc.), a sample of ratings. This data is then used to either predict ratings for unrated items or rank unrated items for recommendation purposes. The challenge presented in this paper is how to recommend with search queries alone, i.e., without any ratings or purchase data. With search data, one does not even know if a particular interest is related to a product. Collaborative filtering approaches do not apply in our context. An interesting approach proposed by Zheng et. al. [31] studies a different problem from the one considered in this paper, namely extracting substitutes and complements from user browsing data so as to enrich recommender systems with this information.

3. QUERY COMMUNITY

In this section, we define a query community and describe our algorithm for recovering communities.

Defining a query community: We assume the existence of a relation R that describes which queries are related, i.e., $R(q, q') = 1$ if queries q and q' are related and $R(q, q') = 0$ otherwise. We define a *query community* C as a maximal set of queries such that *every* pair of queries is related to each other, i.e., for each q, q' in C, $R(q, q') = 1$. In practice, note that we do not have access to R for an arbitrary pair of queries. We return to this issue shortly.

Representing a class of query communities: We now consider a convenient graph representation of the relation R. Specifically, we represent R as an undirected graph $G = (V, E)$ where the vertices correspond to queries and an edge connects two queries that are related. We think of this graph as the 'ground truth'. Note that each community forms a clique in this graph.

This graph representation, however, is not rich enough to describe all classes of query communities. Specifically, since each community is defined as a maximal set of queries, it is tempting to say that there is a one-to-one correspondence between communities and maximal cliques. This statement, however, is false: for three communities C_1, C_2 and C_3 such that $C_3 = (C_1 \setminus C_2) \cup (C_2 \setminus C_1)$, $C_1 \cup C_2 \cup C_3$ forms a maximal clique. Consequently, we assume that each community has at least one query that is unique to it, i.e., for each community C there is a $q \in C$ such that for all other communities C', $q \notin C'$. It can be shown any set of communities satisfying such an assumption can be represented as maximal cliques in a graph (proved in the appendix). We believe the assumption is quite natural as there should be at least one unambiguous search query that only belongs to C.

No Relationship Graph: If we had the relationship graph G, then we could efficiently identify all communities by finding maximal cliques via existing algorithms [29, 14]; however, in practice, we are not given G, but rather a substantially sparser version $\hat{G}$ of G (a simple generative model for $\hat{G}$ will be provided soon). Specifically, depending on the application, one can use combinations of various signals that are present in the data to infer some of the edges present in graph G. In search logs, if many users issue two queries within a small window of time, then the data provides a strong signal of an edge between the two queries. Using this signal, one can construct a graph $\hat{G}$. Since the data is noisy and fundamentally incomplete, it follows that $\hat{G}$ cannot contain all the edges in G. Further, if we remain conservative in adding edges, it may not be unreasonable to assume that every edge in $\hat{G}$ is present in G. Thus, we assume that $\hat{G}$ is a (very) sparse subgraph of G. Note that other signals of relatedness may also be helpful such as document clicks, snippet text and ad bids.

Since $\hat{G}$ is unlikely to have many true cliques, we hope to find (a subset of) communities by finding dense subgraphs in the graph; this is reasonable if the structure of most of the cliques in G is essentially retained in $\hat{G}$ – a precise analysis follows. While numerous definitions of a dense subgraph exist in the literature, for our purposes it is crucial that some vertices belong to multiple clusters since there are many search queries with ambiguous intent, such as jaguar. Hence, we adopt the definition of a dense subgraph given in [20]: A subset of queries is an (α, β)-*cluster* if each query in the cluster is related to at least a β-fraction of queries in the cluster, and any query outside of the cluster is related to at most an α-fraction of the queries in the cluster:

Definition 1. *Given a graph, $G = (V, E)$, where every vertex has a self-loop*[1]*, let $E(c, C)$ denote the edge set $\{(c, v) : \forall v \in C\}$. Then, $C \subset V$ is an* (α, β)-cluster *if*

1. ***Internally Dense:*** $\forall v \in C, |E(v, C)| \geq \beta |C|$
2. ***Externally Sparse:*** $\forall u \in V \setminus C, |E(u, C)| \leq \alpha |C|$

Given $0 \leq \alpha < \beta \leq 1$, the (α, β)-clustering problem *is to find all* (α, β)-*clusters.*

Observe that two (α, β)-clusters can overlap, e.g., two 4-cliques that overlap in one vertex are each $(\frac{1}{4}, 1)$-clusters. A maximal clique

[1]This is a technical assumption needed to ensure that $\beta = 1$ clusters are possible.

is an $(\alpha, 1)$-cluster, where α measures the extent to which cliques overlap. This definition of α turns out to be important in our analysis: as community overlap or α increases, it becomes harder to find the cluster.

Random Deletion Model: We now translate the above intuition of how graph $\hat{G}$ is obtained into a formal generative model that describes the relationship between $\hat{G}$ and G. We seek to describe such a process so that we can design algorithms for reconstructing an approximation of G from $\hat{G}$. We consider a simple model in which each edge is kept with probability p.

Definition 2. *Given a relationship graph $G = (V, E)$, a p-random graph $\hat{G}$ is obtained from G by including each edge $e \in E$ with probability p and not including the edge with probability $1-p$.*

One can consider more sophisticated models for describing the process by which $\hat{G}$ is obtained from G. Indeed, the precise process depends on the type of data that is being used to construct $\hat{G}$. Thus, in the spirit of keeping the analysis general, we consider the uniform model. Further, while being a simple, yet nontrivial, model to analyze, it may provide insight into more complicated models.

The Recovery Algorithm: We now propose an algorithm to recover communities from $\hat{G}$ by finding dense subgraphs. Our algorithm involves two steps, densification followed by clustering. The densification step simply adds edges between pairs of vertices with probability proportional to the number of neighbors they share, concretely the dice coefficient. Then the clustering algorithm described in [20] is used to identify communities. A formal description is given in Algorithm 1).

Intuitively, the densification step aids in finding clusters by making it easier to identify subgraphs that are close to being dense. Note that in practice, the densification step is absolutely crucial for identifying communities – we are not aware of prior work that describes the value of densification. The downside is that it also introduces wrong edges that could potentially lead to merging of dense subgraphs. Our analysis in the next section identifies the conditions under which carrying out the densification step is beneficial.

Algorithm 1 Finding Query Communities in Sparse Graphs

Input: Graph $\hat{G}$, size K, internal density β, external sparsity α
Output: Set of Communities $\mathcal{C}$ each of size K
1: **for all** pairs of vertices $(u, v) \notin E$ **do**
2: $d(u, v) = (|\Gamma(u) \cap \Gamma(v)|)/(|\Gamma(u) \cup \Gamma(v)|)$
3: Add edge (u, v) to $\hat{G}$ with probability $d(u, v)$.
4: **end for**
5: **for** each $c \in V$ **do**
6: $C = \emptyset$
7: **for** each $v \in \Gamma(c) \cup \Gamma(\Gamma(c))$ **do**
8: if $|\Gamma(v) \cap \Gamma(c)| \geq (2\beta - 1)K$ then add v to C.
9: **end for**
10: if C is an (α, β)-cluster then output C.
11: **end for**

4. ANALYTIC EVALUATION OF THE PROCEDURE

The success of our proposed procedure in terms of recovering the underlying cliques depends on the value of the deletion probability $1 - p$ and the topology of graph G. Our goal for this section is to determine the relationship between p and graph topology for which our procedure produces a good approximation to G from $\hat{G}$. Particularly, since $\hat{G}$ is a random graph, we find conditions under which our procedure recovers most of the cliques in G.

Before we give the details of the conditions, we first describe the problem setup. We assume that all the cliques of G are of the same size K. This is a reasonable assumption because the algorithm proceeds in rounds where, for a given round, the algorithm assumes that all of the clusters are of the same size. In addition, our analysis shows that recovery of cliques depends on three quantities related to the underlying graph topology: the size of the maximum overlap of cliques, the maximum degree of a vertex, and the maximum number of neighbors of a clique. Thus, we introduce three parameters (f, d, h) to capture the underlying graph topology. Specifically, we say that G is an (f, d, h) graph if for any two cliques, C_1, C_2 in G, $|C_1 \cap C_2| \leq fK$, the degree of any vertex is at most dK, and $\Gamma(C_1) \setminus C_1 \leq hK$, where for any clique C, $\Gamma(C)$ denotes the set of neighbors of C: $\bigcup_{v \in C} \Gamma(v)$. Clearly, $0 \leq f \leq 1$ and $d \geq 1$. Further, let $\tilde{G}$ denote the graph we obtain after densification of $\hat{G}$. Since the densification procedure as well as $\hat{G}$ are random, graph $\tilde{G}$ is random.

The condition under which we can recover query-communities depends on the clustering algorithm used to find (α, β) clusters. The authors of [20] prove that the clustering algorithm, described in steps 5-9 of Algorithm 1, successfully finds all (α, β) clusters if there is at least one ρ champion per cluster and

$$\beta > 1/2 + (\alpha + \rho)/2, \tag{1}$$

where a vertex v in cluster C is called a ρ-champion if $|\Gamma(v) \cap V \setminus C| \leq \rho|C|$. In other words, roughly speaking, the underlying clusters can be recovered provided the gap between α and β is 'large enough'. To be concise, we term an (α, β)-cluster with a ρ-champion as an (α, β, ρ)-cluster. In our case, G is a graph with $\beta = 1, \alpha = f$ and $\rho = 0$. Since $0 < f < 1$, condition (1) is readily satisfied for G.

Since we run the clustering algorithm on graph $\tilde{G}$, we derive the relationship between the parameters p, f, d, h for recovery by computing the values of α, β, ρ for random graph $\tilde{G}$ and imposing condition (1). However, since $\tilde{G}$ is random, the parameters α, β and ρ are all random. Hence, we assume K is sufficiently large that α, β, ρ concentrate around their respective expected values. Thus, our derivations involve computing the expected values of α, β, ρ and imposing condition (1). Our goal is to understand (a) for what values of p, f, d and h can we recover the underlying cliques, and (b) for what values of p, f, d and h does it make sense to carryout densification.

We now describe our results in two steps. First, we describe the conditions under which we can recover the underlying cliques by clustering graph $\hat{G}$ directly without any densification. Then, we describe the conditions under which densification is beneficial.

In order to describe the results for the case when we cluster $\hat{G}$ without densification, first consider the simpler case when G consists of just two overlapping cliques: C_1, C_2. Utilizing the notation established above, $|C_1 \cap C_2| = fK, d = 2 - f$ and $h = 1 - f$. Assume that $f < 1$, so that each clique has at least one vertex that is unique to it. Because of this reason, and our model that $\hat{G}$ is obtained from G by deleting edges independently with probability $1 - p$, note that in both G and $\hat{G}$, each clique has at least one vertex that is unique to it; therefore, $\rho = 0$ for both G and $\hat{G}$. Further, note that in $\hat{G}$, each vertex in expectation is adjacent to pK vertices of the cluster. This implies that $\beta \approx p$. Now suppose that the two cliques are disjoint i.e., $f = 0$. In this case, α must be zero and hence (1) is satisfied provided $p \approx \beta > 1/2$. Similarly, in the case when $0 < f < 1$, it can be seen that $\alpha \approx fp$. (1) now implies

that $p > 1/2 + fp/2 \implies p > 1/(2-f)$. These results can be formally stated as the following theorem.

Theorem 1. *If we cluster $\hat{G}$ without carrying out densification, then in expectation we can recover $1-\delta$ fraction of cliques provided*

$$p > \frac{1}{2-f-3\varepsilon},$$

where $\delta = (h+1)K\exp(-\varepsilon^2 fp/3)$.

Thus, as expected as the overlap fraction f increases, we need a higher value of p for recovery.

Now consider the case when we densify graph $\hat{G}$ before clustering it. For this case, let's again start with the simple example when G consists of just two overlapping cliques. Densification is beneficial provided the increase in β more than compensates for the increase in $\alpha + \rho$. In the case when there is no overlap between the two cliques, it is clear that even after densification, α and ρ both remain 0. Hence, densification can only be beneficial. In the case when there is non-zero overlap, however, we can show that densification is beneficial only when $p > 1 - 2f(1-f)$. It follows from the bound that the worst case is when $f = 1/2$ in which case p must be at most $1/2$. For f low, the overlap will be low and hence the probability of adding a wrong edge will be small. For f high, while the probability of adding a wrong edge is high, the number of wrong edges will be low since there are very few vertices that are non-overlapping. These results extend to a general graph. More precisely, we have the following theorem.

Theorem 2. *If we cluster $\hat{G}$ after carrying out one densification step, then in expectation we can recover $1-\delta$ fraction of cliques provided*

$$p > \frac{1}{(1+\Delta)(2-f-3\varepsilon)} \tag{2}$$

where

$$\Delta = \frac{1-p}{d} - \frac{f(1-f+h)}{2-f}$$

and $\delta = (h+1)K\exp(-\varepsilon^2 Kt/3) + 2(h+1)K^2\exp(-\varepsilon^2 Kfp^2/3)$ with $t = \min\{fr + (1-f)w, hw\}$, where $r = p + (1-p)p/d$ and $w = fp$.

The proofs of both the theorems are given in the appendix. Note that in the above theorem, the value of Δ depends on the value of p. Therefore, a natural question is whether the conditions $\Delta > 0$ and (2) can be simultaneously met. In order to see this, suppose p is such that $\Delta > 0$ but $\Delta \approx 0$. Then, we must have $p \approx 1 - df(1-f+h)/(2-f)$. This implies that for $f \approx 0$, we must have $p \approx 1$. On the other hand, for $\Delta \approx 0$ and $f \approx 0$, the RHS of (2) will be $\approx 1/2$. This implies that under these conditions, (2) will be satisfied. Thus, there exist reasonable values of p, f, d, and h for which both $\Delta > 0$ and (2) are simultaneously satisfied. Using similar arguments, one can compute precise bounds for which the conditions $\Delta > 0$ and (2) are satisfied.

5. RECOMMENDATION METHODS

In this section we discuss two methods for finding recommendations. Both are motivated by the fact that users tend to express interests related to products as they search. The methods are evaluated in our experiments.

The first method given in Algorithm 2 essentially counts co-occurrences. For each commercial query r and search query q, the method outputs $q \to r$ provided q is frequently posed before r and provided more people pose the query q before r than r before q. The justification for using the ordering between q and r comes from the fact that there is an inherent sequentiality in commerce search. For example, if a user poses the query {hdmi cable}, we would not want to recommend an {hdtv} – presumably the TV is purchased before the cable.

Note that the method can be suitably modified to utilize query communities. Specifically, the counts would then reflect the number of users who posed any query in the community prior to a commercial query, and vice versa. Note that all queries are in some community, possibly a community of size one.

Algorithm 2 Co-Occurrence Recommendation Method

Input: Search Log, threshold θ_1, θ_2
Output: Set of Recommendations $\{q \to r\}$
1: $R = \{$Commercial queries in the Search Log$\}$
2: $Q = \{$Queries in the Search Log$\}$
3: **for** each $r \in R$ and $q \in Q$ **do**
4: $n_{q\to r}$ = number of users who posed query q before commercial query r
5: $n_{r\to r}$ = number of users who posed commercial query r before query q
6: **end for**
7: Output all $n_{q\to r}$ where $n_{q\to r} > \theta_1$ and $n_{q\to r} > \theta_2 n_{r\to q}$

The above approach of finding recommendations through co-occurrence counts is very natural and, in some sense, conservative: essentially, it finds recommendations with high co-occurrence counts. The co-occurrence counts exhibit a power-law distribution with heavy tails. Hence, we can increase the number of recommendations by designing a procedure that can dig through the tail (low co-occurrence counts) and carefully pick relevant recommendations. Since the tail is very noisy, we note that finding the relevant connections in the tail is a hard problem.

Next, we propose a method that finds recommendations through a novel way of viewing connections between queries as 'explanations' rather than as co-occurrence counts. This method is described in Algorithm 3. Specifically, given a commercial query r, this method seeks to find a shortest explanation for why r was posed. For instance, among users who purchase underwater cameras, we may find that the interest {mayan riviera} covers the largest set of users and the interest {ice fishing} covers the remaining users. We view the problem as a classic hitting set, one for each commercial query. In other words, for each commercial query r, we create an instance: $Q_1, \ldots Q_k$ where each Q_i is the set of queries posed by a user who posed the query r and where k is the number of users who posed the query r. We seek a minimum hitting set, i.e., a minimum sized set of queries T such that $T \cap Q_i$ is non-empty. We use the well-known greedy algorithm for finding T that repeatedly adds the query q to T that covers the most uncovered users. The algorithm is known to have a $\log \max_i |Q_i|$ approximation. The method can be suitably modified to utilize query communities, where again all queries are in some community, possibly a community of size one.

6. EXPERIMENTS

The goal of our experiments is two-fold: to demonstrate that our method can successfully find high quality query communities in real search data, and to show that query communities increase both the number and the quality of recommendations.

For the query community problem, as a baseline, we cluster the query-click graph. We find that more than 75% of such clusters are Obvious in the sense that they contain minor word additions/spelling

Algorithm 3 Hitting Set Recommendation Method

Input: Search Log, threshold θ
Output: Set of Recommendations $\{q \rightarrow r\}$
1: $R = \{$Commercial queries in the Search Log$\}$
2: $Q = \{$Queries in the Search Log$\}$
3: U_r =Set of subsets of search queries posed by users who also posed the commercial query r
4: **repeat**
5: Find query $q \in Q$ that most frequently occurs in U_r
6: Output $q \rightarrow r$ if frequency > θ
7: Remove all subsets of queries in U_r that contain the query q
8: **until** U_r is empty

variations of each other. In contrast, the communities produced by our Algorithm 1 uncover more interesting and unexpected relationships. Also, densification yields a four-fold increase in the number of query communities.

For the recommendation problem, we begin by showing that an existing query suggestion algorithm [10] is not designed to produce commercial recommendations. In comparison, both the co-occurrence and hitting set method unearth a large quantity of interesting and surprising recommendations. Also, query communities increase the number of recommendations found for both methods. The increase is accompanied by an increase in the quality of recommendations. Further, there is small overlap in co-occurrence and hitting set recommendations, thus showing that the two approaches are complementary.

Data Our experiments are based on searches collected from 1.7 million users over a one month time period. The logs are restricted to queries of one or two words so as to limit the data to a manageable size – queries reported below that are longer than two words contain underscores which we omit for clarity. We utilize searches in different ways depending on whether we seek query communities or recommendations. Specifically, communities are based on queries posed in close temporal proximity while recommendations are based on queries that may or may not be close in time.

6.1 Query Communities

As a baseline for finding query communities, we cluster the query-click graph [30]. For the same time period and queries, we construct a bipartite query click graph, where an edge connects a query to a URL if the query led to at least three clicks on the URL. The graph was clustered using [20] where the goal is to find dense bipartite subgraphs – specifically, of size at least four queries and four URLs, where each query is adjacent to at least three URLs and each URL is adjacent to at least three queries. The method produced 51K clusters.

Our Method: Algorithm 1: To discover communities using our method, we utilize queries posed in rapid succession. The vertices of the graph correspond to queries while the edges connect queries posed in rapid succession, set to within five minutes in our study. The weight of an edge in this graph reflects the number of users who posed the queries in rapid succession. We kept edges only if a minimum number of users posed both queries in a short time window (minimum set to two users in our experiments). In addition, we removed head queries, i.e., vertices of degree greater than 100. Head queries were removed because otherwise queries such as {facebook} belong to more clusters than they should. The resulting graph contained 680K vertices and 632K edges. The graph is extremely sparse, containing 195K connected components. The largest connected component contains 29K queries and there are 131K components with two queries. Clustering the graph without densification yields 17.7K query communities.

Category	Baseline: Cluster Query-Click Graph	Algorithm 1
Surprising	0%	21%
Good	14%	48%
Obvious	76%	10%
Bad	1%	3%
Foreign	6%	16%
Adult	3%	2%

Table 2: Evaluation of Query Communities

Once the graph was constructed, we employed the densification step described in Algorithm 1, i.e., adding edges between two vertices with probability proportional to the number of neighbors they share. The densification steps added 336K edges to the graph – increasing the number of edges by over 50%. Then we ran the clustering algorithm described in Algorithm 1, from [20] – seeking clusters of at least four, six and eight queries, each with density at least $3/4, 2/3$, and $3/4$, respectively. For example, for clusters of four vertices, each query in the cluster has to be adjacent to at least two other queries in the cluster, with an implied third self-loop edge, yielding $\beta = 3/4$. The algorithm identified a total of 72K clusters and the average number of queries per cluster was 4.5. Note that the densification step increased the number of discovered query communities by a factor of four.

Evaluating Query Communities: To evaluate the clusters, we randomly selected a sample and conducted a manual investigation. We cannot use automatic evaluation schemes because the relationships we find do not already exist in the data and the underlying relationship is not always evident, for example, it is not immediately apparent that the set {love & monsters}, {the impossible planet}, {the satan pit}, {the idiot's lantern} is a community of episodes of Dr. Who. We also believe that the task of characterizing the clusters would be too complex and time-consuming for a mechanical turk.

Our evaluation is limited to queries present in the community, and not the queries missing from the community — for any cluster we find, we can easily imagine additional queries that could belong to the cluster. Fundamentally, our ability to find clusters is limited by the queries in the logs: if we based our study on more users over a longer period of time, we would be able to generate more queries per community. Another possibility is to design methods that generate more queries from a seed collection. For instance, techniques from [24, 23] can be used to infer probabilistic automata that can generate more similar queries. Our evaluation is focused on whether all pairs of queries are related.

Each community was categorized into one of seven categories: Surprising, Good, Obvious, Bad, Foreign and Adult. A community was "Surprising" if an unexpected set of queries appeared in the same community, "Good" if the set was expected but nevertheless interesting, "Obvious" if the set contained minor spelling variations, "Bad" if we could not find a connection between the queries, "Foreign" if the query was not in English and "Adult" if the query contained adult content. We manually labeled about 80 randomly sampled query communities found by the baseline method and Algorithm 1. A summary of our findings appears in Table 2.

While clustering the query-click graph produces many clusters, the overwhelming majority are Obvious, e.g., {all recipe.com}, {all reciepes}, {all recipes}, {all receipe}, {all recipe website}. The main reason is that users can only click on what they see, and the clusters can only be as good as the search engine. If the search

engine does not know that two queries fall in the same community then it cannot display the same search results for the two queries. We seek to go beyond what the search engine already knows.

Among the clusters found by Algorithm 1, nearly half of the communities are rated "Good". Two examples of Good communities with the theme of Mexican food and shopping include: (1) {enchilada recipe}, {chimichanga}, {enchilada}, {tostada} (2) {dillards dresses}, {macys dresses}, {macy's dresses}, {macy's locations}

The next largest category was "Surprising" communities and these were quite unexpected: (1) {kira-kira}, {the higher power of lucky}, {newberry award}, {newberry medal}, {criss cross} (2) {kktv},{krdo},{kvor},{kccy} (3) {sri lanka lion}, {cave lion}, {steppe wisent}, {cebu warty pig} (4) {love & monsters}, {the impossible planet}, {the satan pit}, {the idiot's lantern} The examples above are tied by the themes: (1) Books that won the Newberry Award (2) Channels serving Colorado (3) Extinct species (4) Episodes of Dr. Who.

In summary, clustering the query click graph produces largely obvious clusters, while Algorithm 1 produces more Good and Surprising clusters.

6.2 Recommendations

Next we evaluate recommendations generated by the two methods described in Section 5. Our recommendation experiments are on the same data used to find communities. The data is used differently in that now we make connections between queries posed over an arbitrarily long period of time, since it may take time for a related product to follow an expressed interest.

Query Suggestion: We are not aware of any existing algorithms that specifically recommend commercial queries. One may hypothesize that query suggestion algorithms already recommend decent commercial queries. To answer, we adapt a query suggestion algorithm [10] as follows: Given a seed query, a random walk on the query-click graph is performed, treating the seed query as an absorbing node. Queries that have a sufficiently high probability of ending at that absorbing node are viewed as suggestions to the seed query. We keep only those suggestions that are commercial using [9]. One complication is how best to select seed nodes. We tried two methods. In the first, we randomly selected frequently posed queries as seed queries, 1.6K out of the top 100K most frequently posed queries. However, this method of selecting seed queries did not work: for 95% of the queries, no commercial recommendations were generated. In the second method, we selected a set of 100 queries that we knew possessed future commercial intent from our own algorithms. Again, this method did not work because for two-thirds of the queries, no commercial recommendation was produced. Since this algorithm was designed for a different purpose, i.e., suggesting keywords to advertisers, we would not expect it to solve our problem. What we observe is that longer-range commercial recommendations are rare, in fact in many cases there does not seem to be a path from a search query to a commercial query in the query-click graph.

Co-Occurrence: To evaluate the co-occurrence method, for the set of all queries Q, we compute a subset $R \subset Q$ that possess commercial intent [9]. Then we compute two counts $n_{q \to r}$, the number of users who pose query q before commercial query r, and $n_{r \to q}$ the number of users that pose r before q. This simple method recommends r to q if $n_{q \to r}$ occurs more frequently than $n_{r \to q}$. In our experiments, we output (q, r) for which $n_{q \to r} > 5$ and $n_{q \to r} > 2 \cdot n_{r \to q}$.

To assess the impact of communities, we compute $n_{T \to q}$, the number of users that posed at least one query q in the community T that later pose the commercial query r. The same heuristic is used to select (T, r) combinations. The method found 9K recommendations without communities, and 11K recommendations with communities. The intersection contained 7.7K recommendations.

Category	Co-Occurrence Recommendations w/o Communities	Co-Occurrence Recommendations with Communities
Surprising	2%	12%
Good	28%	49%
Obvious	54%	11%
Bad	7%	10%
¬Commercial	6%	13%
Adult	3%	3%
Foreign	0%	2%

Table 3: Evaluation of Co-Occurrence Recommendations

To evaluate, we manually categorized each recommendation into one of five types, "Surprising" if the connection was unexpected, "Good" if the recommendation was reasonable, "Obvious" if the suggestion was a spelling variation or just added a closely related word "Not Commercial" if the commercial intent classifier failed by asserting that the recommended query was commercial when in fact it was not, and "Bad" if we could not find an explanation for the recommendation. We randomly selected 100 recommendations without communities and we also selected another 100 recommendations found with communities that were not found without communities. The manual categorization is shown in the second and third columns of Table 3.

The majority of recommendations found without the benefit of communities are Obvious. Most are spelling corrections or minor variations such as {christmas backgrounds} $\to$ {christmas borders}. The next largest category are Good recommendations such as {itunes download} $\to$ {zune}. Not Commercial is a problem for about 6% of the queries. Some of these recommendations are good, but simply do not direct the user to a commercial space. For example, {wizards of waverly} $\to$ {suite life on deck}, while both are shows on the Disney channel, the latter need not have commercial intent. The Bad recommendations we could not justify such as {rice} $\to$ {oil} and {fireworks} $\to$ {fire truck}. There are very few surprising recommendations.

Query Communities improve our ability to find unexpected recommendations. The major shift is away from Obvious recommendations towards more Good and Surprising. A Surprising example is {engagement ring} $\to$ {promise ring} – a promise ring is a pre-engagement promise to be loyal ring, a social sign of less commitment. The increase in Good and Surprising recommendations comes at a slight increase in Bad recommendations. In some circumstances, communities create transient connections. For example, {presentation clipart} $\to$ {christmas lights} is a Bad recommendation created because the search logs are from December and {presentation clipart} is in a community with {christmas image}, {christmas images}, and {christmas clipart}.

Hitting Set Next we evaluate the hitting set method. The experiment was run on the same data as before. In our experiments, we output (q, r) if at least three uncovered users posed the queries q and r. The method found 36K recommendations when query communities were not used and 42K when query communities were used.

The results are shown in Table 4. As with co-occurrence, note the intriguing shift towards good and surprising recommendations that communities bring, at the cost of some increase in Bad recommendations. Some of the surprising recommendations include:

Category	Hitting Set Recommendations w/o Communities	Hitting Set Recommendations with Communities
Surprising	2%	12%
Good	24%	24%
Obvious	52%	42%
Bad	2%	4%
¬Commercial	16%	8%
Adult	4%	4%
Foreign	0%	6%

Table 4: Evaluation of Hitting Set Recommendations

{hard flip} → {zero skateboards} and {double backflip} → {trampoline}, where {hard flip} is in a community with {varial heelflip}, {laser flip}, {pressure flip}, {360 hardflip} – all moves that can be executed on a zero skateboard. In the second case, {double backflip} is in a community with other flips that can be performed on a trampoline.

Finally, there is a noticeable fraction of recommendations that are not commercial. The commercial intent identification method that we are using [9] produces a score between 0 and 1 indicating how likely it is the query has commercial intent. In our experiments, we set this threshold to 0.6. Our results could improve if we increased this threshold. One example of such a non-commercial recommendation is {david paterson} → {blindness}. While some implication exists, blindness is not a commercial query.

What is the difference between the recommendations found by the co-occurrence and the hitting set algorithms? Hitting set increases the number of recommendations found with little loss of quality. One way to achieve this is to decrease the threshold of the co-occurrence frequency. We observed that decreasing the threshold below five results in a high number of "Bad" recommendations. Further, note that hitting set fails to find recommendations with high co-occurrence frequency – algorithmically, it removes covered users resulting in the loss of some high co-occurrence recommendations. Therefore, the two algorithms are complimentary, and we recommend using both to find an extensive set of high quality recommendations.

In summary, prior query suggestion methods largely do not generate commercial recommendations, nor are they designed for that purpose. Further, both co-occurrence and hitting set recommendation methods produce interesting and unexpected recommendations. Query communities are the key to increasing the number and quality of recommendations. The two methods can be used in tandem to produce a large collection of high quality recommendations.

7. CONCLUSIONS & FUTURE WORK

We presented an approach to forecast future commercial intent of search users, thereby enabling product recommendation early in the search process. A crucial feature of the approach is the use of query communities which amplify forecasting signals beyond what each individual query can provide.

We leave many interesting directions for future work. One avenue is the investigation of richer random deletion models. In addition to the uniform deletion model we studied, one can consider other types of models that are tailored to the type of data used. For example, we could imagine models where users initially pose queries in the overlap of communities, i.e., ambiguous queries, and over time make their queries less ambiguous. The deletion process in this case may be different from uniform, and such structure may be exploited to improve recovery of communities.

It would be beneficial to understand how signals from other sources impact the quality of query communities. In this paper, we identified interests via only search queries. One may augment or replace the graph with edges between queries if they share clicks, if they are both bid on by the same advertiser, etc.

In addition, one may consider the problem of assembling a diversified portfolio of product recommendations for each query, so as to maximize the probability that a user clicks on at least one, i.e., minimize the probability the portfolio defaults. In this context, work on creating a diversified portfolio of investments may be relevant [8].

Next, an ideal recommendation system would explain to the user why the product was recommended. In some cases, there is a chain of reasoning involved and methods that provide justification can help the user. At the moment, we have statistics such as x people who query q later query r. It may be desirable to also provide a sequence of queries $q \rightarrow q_1 \rightarrow q_2 \rightarrow r$ to help the user understand the recommendation.

Finally, we have not touched on the privacy angle and our recommendations are based on quite private user searches. In the case of hitting set, we find more surprising recommendations when the signal is based on a smaller number of users. From a privacy perspective, if a query is in the tail it is intuitively more unique to a user and hence more private [16]. We leave the question of private recommendations [18] as an interesting direction for future work.

8. REFERENCES

[1] D. Achlioptas and F. McSherry. Fast computation of low rank matrix approximations. In *STOC*, pages 611–618, 2001.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases*, pages 487–499, 1994.

[3] R. Bell, Y. Koren, and C. Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *KDD*, pages 95–104, 2007.

[4] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna. The query-flow graph: model and applications. In *CIKM*, pages 609–618, 2008.

[5] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. In *KDD*, pages 875–883, 2008.

[6] N. Craswell and M. Szummer. Random walks on the click graph. In *SIGIR*, pages 239–246, 2007.

[7] A. Freund, D. Pelleg, and Y. Richter. Clustering from constraint graphs. In *SIAM International Conference on Data Mining*, 2008.

[8] R. Frey, A. McNeil, and M. Nyfeler. Copulas and credit models. *Risk*, 14(10):111–114, 2001.

[9] A. Fuxman, A. Kannan, A. Goldberg, R. Agrawal, P. Tsaparas, and J. Shafer. Improving classification accuracy using automatically extracted training data. In *KDD*, pages 1145–1154, 2009.

[10] A. Fuxman, P. Tsaparas, K. Achan, and R. Agrawal. Using the wisdom of the crowds for keyword generation. In *WWW*, pages 61–70, 2008.

[11] S. Gupta, M. Bilenko, and M. Richardson. Catching the drift: Learning broad matches from clickthrough data. In *KDD*, 2009.

[12] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *IJCAI*, pages 688–693, 1999.

[13] T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Trans. Inf. Syst.*, 25(2), 2007.

[14] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. On generating all maximal independent sets. *Information Processing Letters*, 27(3):119–123, 1988.

[15] J. Kleinberg and M. Sandler. Using mixture models for collaborative filtering. In *STOC*, pages 569–578, 2004.

[16] A. Korolova, K. Kenthapadi, N. Mishra, and A. Ntoulas. Releasing search queries and clicks privately. In *WWW*, pages 171–180, 2009.

[17] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[18] F. McSherry and I. Mironov. Differentially private recommender systems: building privacy into the net. In *KDD*, pages 627–636, 2009.

[19] Q. Mei, D. Zhou, and K. Church. Query suggestion using hitting time. In *CIKM*, pages 469–478, 2008.

[20] N. Mishra, R. Schreiber, I. Stanton, and R. Tarjan. Finding strongly knit clusters in social networks. *Internet Mathematics*, 5(1):155–174, 2009.

[21] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an

open architecture for collaborative filtering of netnews. In *CSCW*, pages 175–186, 1994.
[22] M. Richardson. Learning about the world through long-term query logs. *TWEB*, 2(4):1–27, 2008.
[23] D. Ron, Y. Singer, and N. Tishby. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, 25(2-3):117–149, 1996.
[24] D. Ron, Y. Singer, and N. Tishby. On the learnability and usage of acyclic probabilistic finite automata. *J. Comput. Syst. Sci.*, 56(2):133–152, 1998.
[25] E. Sadikov, J. Madhavan, L. Wang, and A. Halevy. Clustering query refinements by user intent. In *WWW*, pages 841–850, 2010.
[26] M. Sahami and T. Heilman. A web-based kernel function for measuring the similarity of short text snippets. In *WWW*, pages 377–386, 2006.
[27] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW*, pages 285–295, 2001.
[28] D. Stern, R. Herbrich, and T. Graepel. Matchbox: large scale online bayesian recommendations. In *WWW*, pages 111–120, 2009.
[29] S. Tsukiyama, M. Ide, H. Ariyoshi, and I. Shirakawa. A new algorithm for generating all the maximal independent sets. *SIAM J. Comput*, 6(3):505–517, 1977.
[30] J. Yi and F. Maghoul. Query clustering using click-through graph. In *WWW*, pages 1055–1056, 2009.
[31] J. Zheng, X. Wu, J. Niu, and A. Bolivar. Substitutes or complements: another step forward in recommendations. In *EC*, pages 139–146, 2009.

APPENDIX

A. REPRESENTING A CLASS OF QUERY-COMMUNITIES

We prove that if each community has a query that is unique to it then a set of queries forms a community iff the set forms a maximal clique.

Theorem 3. *Let $\mathcal{C}$ be a class of query communities such that each community has at least one query that is unique to it. Let G be the graph representation of $\mathcal{C}$. Then $C \in \mathcal{C}$ is a community if and only if the set of vertices in C forms a maximal clique in G.*

Proof. We first show that each community forms a maximal clique. Let the set of communities be denoted by $\mathcal{C} = C_1, \ldots, C_k$ where q_i is the query that is unique to C_i for $i = 1, \ldots, k$. Each C_i is a clique in the graph representation of $\mathcal{C}$ by construction. Assume by way of contradiction that C_i is not maximal. Then there exists a vertex v that can be added to C_i that neighbors all of C_i. This in turn implies that v is adjacent to q_i, which contradicts that q_i is unique to C_i.

We now show that each maximal clique forms a community. Consider a maximal clique V' in G. Note that the set of vertices in V' must contain some community $C \in \mathcal{C}$, by construction of G. Let q be the query that is unique to C. Assume by way of contradiction that there exists a vertex $v \in V'$ that is not in C. Observe that v must be adjacent to each vertex in C since v is in a maximal clique V' containing the community/clique C. In particular, v must be adjacent to q, contradicting the fact that q is unique to C. Thus, V' is identical to C. Combining the above, the theorem follows. □

B. PROOFS OF THEOREMS

Before we prove Theorems 1 and 2, we state the lemmas we need to prove the theorems.

Lemma 1. *Let $G = (V, E)$ be the underlying ground truth that is a collection of overlapping cliques and fix a clique C in G. Suppose we construct a random graph G' with vertex set V by putting an edge between vertices u and v with 'right' probability r if $(u, v) \in E$ and 'wrong' probability w if $v \in \Gamma(C) \setminus C$, where C is the clique containing u. Then, with probability at least $1 - (h+1)K\exp(-\varepsilon^2 Kt/3)$, C forms an α, β, ρ cluster with $\alpha \leq (1+\varepsilon)(fr + (1-f)w)$, $\beta \geq (1-\varepsilon)r$, and $\rho \leq (1+\varepsilon)hw$, where $t = \min\{r, w + f(r-w), hw\}$, if $w \neq 0$ and $t = fr$ if $w = 0$.*

Lemma 2. *Consider graph $\hat{G}$ that is generated from G from the deletion model. Fix a clique C. Then, for two vertices $u, v \in C$, we must have the dice coefficient $d(u, v) \geq (1 - \varepsilon_L)p/d$ with probability $1 - 2\exp(-\varepsilon^2 Kp^2/3)$, where $\varepsilon_L = 2\varepsilon/(1+\varepsilon)$. Similarly, when $u \in C$ and $v \notin C$, then we must have $d(u, v) \leq (1 + \varepsilon_U)fp$ with probability at least $1 - 2\exp(-\varepsilon^2 fKp^2/3)$, where $\varepsilon_U = 2\varepsilon/(1-\varepsilon)$.*

Lemma 3. *Consider a random variable $Z = nr/dr$. Let $\mu = \mathbb{E}[\mathrm{nr}]/\mathbb{E}[\mathrm{dr}]$. Then, with probability at least $1-\delta$, $Z \leq (1+\varepsilon_U)\mu$ and with probability at least $1 - \delta$, $Z \geq (1 - \varepsilon_L)\mu$, where $\delta = \exp(-\varepsilon^2\mathbb{E}[\mathrm{nr}]/3) + \exp(-\varepsilon^2\mathbb{E}[\mathrm{dr}]/3)$, $\varepsilon_U = 2\varepsilon/(1-\varepsilon)$ and $\varepsilon_L = 2\varepsilon/(1+\varepsilon)$.*

Proof of Theorem 1. We invoke Lemma 1 with appropriate values of r and w to derive the relationship between p and graph topology for which recovery of cliques is possible. It follows from our deletion model that all clusters in $\hat{G}$ have $\rho = 0$. In order to invoke Lemma 1, we consider the following dual interpretation of the deletion model: we obtain $\hat{G}$ by starting with an empty graph and add edges independently between two vertices u, v with probability p if and only if $(u, v) \in G$. In the language of Lemma 1, it now follows that the probability of adding a right edge between two vertices in same clique $r = p$ and than of adding a wrong edge between two vertices in different cliques $w = 0$. Therefore, a given clique C forms an (α, β)-cluster with $\alpha \leq (1+\varepsilon)fp$ and $\beta \geq (1-\varepsilon)p$ with probability at least $1 - \delta$, where $\delta = (h+1)K\exp(-\varepsilon^2 fp/3)$. Clique C will be recovered if $\beta > 1/2 + (\alpha + \rho)/2$. Thus, C will be recovered if

$$(1-\varepsilon)p > \frac{1}{2} + \frac{(1+\varepsilon)fp}{2} \iff p > \frac{1}{2 - f - \varepsilon(2+f)}$$
$$\impliedby \quad p > \frac{1}{2 - f - 3\varepsilon}.$$

Since each clique is recovered with a probability at least $1 - \delta$, on an average a fraction of $1 - \delta$ cliques will be recovered provided $p > 1/(2 - f - 3\varepsilon)$. This finishes the proof of the theorem. □

Proof of Theorem 2. First, we fix a clique C in G. Let $\tilde{G}$ denote the random graph we obtain upon densification of $\hat{G}$; $\tilde{G}$ is random not only because $\hat{G}$ is random, but also because the densification step is random. Because of this reason, the probabilities of adding right and wrong edges in $\tilde{G}$ are both random. Specifically, consider a pair of vertices u, v in clique C. The probability that there is an edge between u and v in $\tilde{G}$ is $r = p + (1-p)d(u, v)$ since the edge could either be present in $\hat{G}$ or have been added in the densification step. The dice coefficient $d(u, v)$ is a random quantity and hence r is random. It follows from Lemma 2 that with a probability at least $1 - 2\exp(-\varepsilon^2 Kp^2/3)$, $d(u, v) \geq (1 - \varepsilon_L)p/d$, where $\varepsilon_L = 2\varepsilon/(1-\varepsilon)$. Ignoring ε to simplify expressions we can write that for *all* vertices $u, v \in C$ (using a union bound),

$$r \geq p + (1-p)p/d \text{ with probability } \geq 1 - 2K^2\exp(-\varepsilon^2 Kp^2/3).$$

Coming to the wrong edges, note that the densification step only adds edges between $u \in C$ and $v \notin C$ if $v \in \Gamma(C) \setminus C$. Thus, using the same argument as above, it can be shown using Lemma 2 (and ignoring ε_U) that for *all* vertices $u \in C$, $v \in \Gamma(C) \setminus C$ (again by union bound),

$$w \leq fp \text{ with probability } \geq 1 - 2hK^2\exp(-\varepsilon^2 Kfp^2/3).$$

Invoking lemma 1, it now follows that clique C will be recovered with probability at least $1-\delta$, where $\delta = (h+1)K\exp(-\varepsilon^2 Kt/3) +$

$2(h+1)K^2\exp(-\varepsilon^2 Kfp^2/3)$ with
$t = \min\{fr + (1-f)w, hw\}$ (since $r - w > 0$) if

$$(1-\varepsilon)r > \frac{1}{2} + \frac{(1+\varepsilon)(fr + (1-f)w + hw)}{2}$$

Re-arranging the terms in the above equation, it can be shown that recovery is possible for $p > 1/((2-f-3\varepsilon)(1+\Delta))$ where

$$\Delta = \frac{1-p}{d} - \frac{f(1-f+h)(1+\varepsilon)}{2-f-\varepsilon(2+f)}$$

Assuming ε is small, we ignore it in the expression for Δ in order to simplify the expressions. Therefore, since each clique C can be recovered with a probability at least $1-\delta$, on an average a fraction of $1-\delta$ cliques will be recovered provided $p > 1/((2-f-3\varepsilon)(1+\Delta))$. This finishes the proof of the theorem. □

Proof of Lemma 1. Let $u \in C$ and $v \in \Gamma(C) \setminus C$. Further let C' denote the clique that contains v. It now follows from our definitions that

$$\alpha = \frac{1}{K}\max_{v\in\Gamma(C)\setminus C} |\Gamma(v)\cap C|, \quad \beta = \frac{1}{K}\min_{u\in C} |\Gamma(u)\cap C|$$
$$\rho \leq \frac{1}{K}|\Gamma(u)\setminus C|.$$

Since G' is random, all the quantities on the right in the above expressions are random. Hence, we compute their expected values, and argue that with a high probability they concentrate around their expected values. To this end, let $X(u,v)$ denote the indicator random variable for adding an edge between vertices u and v. It follows from our definitions that for any $v' \in C$, $\mathbb{E}[X(u,v')] = r$ if $v' \in C$ and $\mathbb{E}[X(v,v')] = r$ if $v' \in C \cap C'$ and $\mathbb{E}[X(v',v)] = w$ if $v' \in C \setminus C'$. Letting $|C \cap C'| = \eta K$, we can now write

$$\mathbb{E}[|\Gamma(u)\cap C|] = \sum_{v'\in C}\mathbb{E}\left[X(u,v')\right] = Kr$$
$$\mathbb{E}[|\Gamma(u)\setminus C|] = \sum_{v'\in\Gamma(C)\setminus C}\mathbb{E}\left[X(u,v')\right] \leq hKw,$$

and

$$\mathbb{E}[|\Gamma(v)\cap C|] = \sum_{v'\in C\setminus C'}\mathbb{E}\left[X(v,v')\right] + \sum_{v'\in C\cap C'}\mathbb{E}\left[X(v,v')\right]$$
$$= Kw + K\eta(r-w) \leq Kw + Kf(r-w),$$

where the last inequality follows from the fact that $\eta \leq f$ and $r - w > 0$. It follows from Chernoff bound that $\Pr(Z \geq (1-\varepsilon)\mathbb{E}[Z]) \leq 1 - \exp(-\varepsilon^2\mathbb{E}[Z]/3)$ for any random variable Z that is a sum of independent indicator random variables. It thus follows that each of the above quantities concentrate around their expectations with probabilities that can easily derived from the Chernoff bound. Moreover, since the expressions for α and β involve taking maximization and minimization over a set of vertices, we need to apply union bound. Particularly, letting $Z_1(v)$ denote $|\Gamma(v)\cap C|$, $Z_2(u)$ denote $|\Gamma(u)\cap C|$, and $Z_3(u)$ denote $|\Gamma(u)\setminus C|$, we can write

$$Z_2(u) \geq (1-\varepsilon)Kr \;\forall u \in C \quad \text{w.p. } 1 - K\exp(-\varepsilon^2 Kt/3)$$
$$Z_3(u) \leq (1+\varepsilon)hKw \quad \text{w.p. } 1 - \exp(-\varepsilon^2 Kt/3),$$

and

$$Z_1(v) \leq (1+\varepsilon)(Kw + Kf(r-w)) \;\forall v \in \Gamma(C)\setminus C$$
$$\text{w.p. } 1 - hK\exp(-\varepsilon^2 Kt/3)$$

where $t = \min\{r, w + f(r-w), hw\}$. Putting everything together, we can now write $\alpha \leq (1+\varepsilon)(fr+(1-f)w)$, $\beta \geq (1-\varepsilon)r$, and $\rho \leq (1+\varepsilon)hw$ with probability $1-(h+1)K\exp(-\varepsilon^2 Kt/3)$. If $w = 0$, then it is easy to see that $\rho = 0$ and, hence, we only need to consider α, β. Thus, for $w = 0$, the above expressions carry with $t = \min\{r, fr + (1-f)w\} = fr$. This finishes the proof of this lemma. □

Proof of Lemma 2. If follows from our definitions that $d(u,v) = \mathrm{nr}/\mathrm{dr}$, where $\mathrm{nr} = 2|\Gamma(u)\cap\Gamma(v)|$ and $\mathrm{dr} = |\Gamma(u)| + |\Gamma(v)|$. Let $X(u,v)$ denote the indicator random variable of there being an edge between u and v; it follows from our deletion model that $\mathbb{E}[X(u,v)] = p$ if u, v are in the same clique and 0 otherwise. For $u, v \in C$, it follows that

$$\mathbb{E}[\mathrm{nr}] = 2\sum_{v'\in\Gamma(u)}\mathbb{E}\left[X(u,v')X(v',v)\right] \geq 2Kp^2,$$

where the last inequality follows because the random variables $X(\cdot,\cdot)$ are independent and $|\Gamma(u)| \geq K$. Similarly,

$$\mathbb{E}[|\Gamma(u)|] = \sum_{v'\in\Gamma(C)}\mathbb{E}\left[X(u,v')\right] \leq dKp,$$

since by definition of d, it follows that $|\Gamma(u)| \leq dK$ for any vertex v. Putting the above two together, we get

$$\mu_r = \frac{\mathbb{E}[\mathrm{nr}]}{\mathbb{E}[\mathrm{dr}]} \geq \frac{2Kp^2}{2dKp} = p/d.$$

It follows from lemma 3 that $d(u,v) \geq (1-\varepsilon_L)\mu_r$ with probability at least $1-\delta$ where

$$\delta \leq \exp(-2\varepsilon^2 Kp^2) + \exp(-\varepsilon^2 Kp) \leq 2\exp(-\varepsilon^2 Kp^2).$$

Similarly, when u, v belong to different clusters C and C', we have

$$\mathbb{E}[\mathrm{nr}] = \sum_{v'\in C\cap C'}\mathbb{E}\left[X(u,v')X(v',v)\right] = |C\cap C'|p^2 \leq fKp^2$$

where the last inequality follows from the fact that overlap fraction is at most f. Since $\mathbb{E}[\mathrm{dr}] \geq \mathbb{E}[|\Gamma(u)|] \geq Kp$, it follows that

$$\mu_w = \mathbb{E}[\mathrm{nr}]/\mathbb{E}[\mathrm{dr}] \leq \frac{fKp^2}{Kp} = fp$$

It now follows from lemma 3 that $d(u,v) \leq (1+\varepsilon_U)\mu_w$ with probability at least $1-\delta$ where

$$\delta \leq \exp(-\varepsilon^2 Kfp^2) + \exp(-\varepsilon^2 Kp) \leq 2\exp(-\varepsilon^2 Kfp^2).$$

This finishes the proof of the lemma. □

Proof of Lemma 3. This lemma can be proved by a direct application of Chernoff bound. We prove the upper bound. The proof for lower bound is similar.

It follows from Chernoff bound that $\mathrm{nr} \leq (1+\varepsilon)\mathbb{E}[\mathrm{nr}]$ with probability at least $1 - \exp(-\varepsilon^2\mathbb{E}[\mathrm{nr}]/3)$ and $\mathbb{E}[\mathrm{dr}] \geq (1-\varepsilon)\mathbb{E}[\mathrm{dr}]$ with probability at least $1 - \exp(-\varepsilon^2\mathbb{E}[\mathrm{dr}]/3)$. Thus, it follows that with probability at least $1 - \exp(-\varepsilon^2\mathbb{E}[\mathrm{nr}]/3) - \exp(-\varepsilon^2\mathbb{E}[\mathrm{dr}]/3)$, we have

$$Z = \frac{\mathrm{nr}}{\mathrm{dr}} \leq \frac{(1+\varepsilon)\mathbb{E}[\mathrm{nr}]}{(1-\varepsilon)\mathbb{E}[\mathrm{dr}]} = (1 + \frac{2\varepsilon}{1-\varepsilon})\mu.$$

The result of the lemma now follows. □