Mining E-Commerce Query Relations using Customer Interaction Networks

Bijaya Adhikari[*,†], Parikshit Sondhi[○,†], Wenke Zhang[○], Mohit Sharma[○], B. Aditya Prakash[*]
[*]Department of Computer Science, Virginia Tech
[○]WalmartLabs
[*]{bijaya, badityap}@cs.vt.edu, [○]parikshit@neulogic.ai, {wzhang1, Mohit.Sharma}@walmartlabs.com

ABSTRACT

Customer Interaction Networks (CINs) are a natural framework for representing and mining customer interactions with E-Commerce search engines. Customer interactions begin with the submission of a query formulated based on an initial product intent, followed by a sequence of product engagement and query reformulation actions. Engagement with a product (e.g. clicks) indicates its relevance to the customer's product intent. Reformulation to a new query indicates either dissatisfaction with current results, or an evolution in the customer's product intent. Analyzing such interactions within and across sessions, enables us to discover various query-query and query-product relationships.

In this work, we begin by studying the properties of CINs developed using Walmart.com's product search logs. We observe that the properties exhibited by CINs make it possible to mine intent relationships between queries based purely on their structural information. We show how these relations can be exploited for a) clustering queries based on intents, b) significantly improve search quality for poorly performing queries, and c) identify the most influential (aka. 'critical') queries whose performance have the highest impact on performance of other queries.

KEYWORDS

Customer Interaction Networks, Query Relation Mining, E-commerce

ACM Reference Format:
Bijaya Adhikari[*,†], Parikshit Sondhi[○,†], Wenke Zhang[○], Mohit Sharma[○], B. Aditya Prakash[*]. 2018. Mining E-Commerce Query Relations using Customer Interaction Networks. In *WWW 2018: The 2018 Web Conference, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3178876.3186174

1 INTRODUCTION

Search engine logs serve as an invaluable resource of customer interactions with a search engine. Each search session in the log, begins with the submission of a query formulated based on an initial intent, followed by a sequence of result engagement and query reformulation actions. Engagement with a result (e.g. clicks), signals its relevance to the customer's intent. Reformulation to a new query indicates either dissatisfaction with current results, or evolution of intent. Considerable attention is therefore paid toward mining meaningful information from search logs, and using it to improve various aspects of the system. In web-search domain, this is reflected in several prior works [1, 4, 14, 22], which propose novel search log representations, formulate various log mining tasks, and evaluate the utility of mined information in delivering measurable system improvements. Graph based representations such as click-graphs [23], cover graphs [1], query flow graphs [6], term graphs [35] etc. are frequently used. Popular mining tasks include identification of relationships (e.g. synonymy, generalization, specialization etc.) between query-query pairs, and relevance relationships between query-URL pairs. The mined information is then used for applications like related query recommendation, search quality improvement (via relevant URL retrieval) etc.

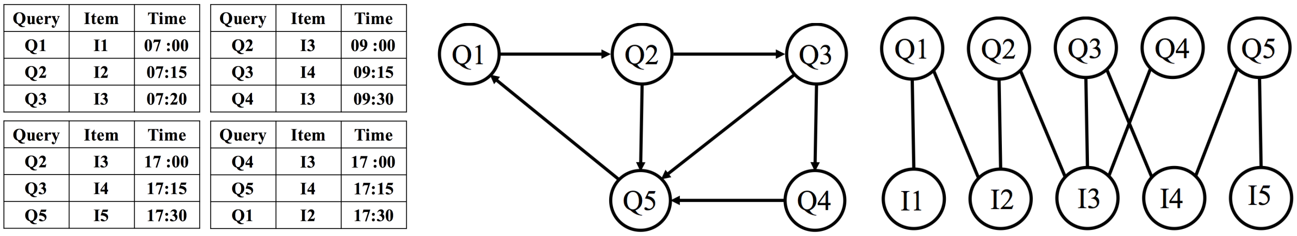
In context of E-Commerce search however, literature is sparse. There has been some work on analyzing E-Commerce search logs to study relationships between customers and products [13, 27]. Nevertheless, to the best of our knowledge, no formal study on the properties and utility of various query-query and query-item graph representations currently exists. Compared to web-search, the E-Commerce domain presents several unique characteristics, which make such a study interesting.

(1) **Precise Intent:** Since E-Commerce search is a type of entity search, the notion of query intent is more precise compared to web search. In E-Commerce, intent can typically be represented by a well defined set of product attribute-value pairs expected by the query.
(2) **Narrow Search Mission:** The goal of search is also narrow i.e. to buy a particular product, which makes sessions coherent. There are also clear task completion signals i.e. an item being added-to-cart or purchased.
(3) **Category Hierarchy:** Products in an E-Commerce catalog are usually organized into a well defined category hierarchy, which can serve as useful ground truth for intent mining tasks.

These characteristics allow us to define better metrics around various graph mining tasks, and conduct large scale evaluations without using human input. A review of prior papers in web search domains suggests this is a significant problem, since evaluations are typically manual and consequently small in scale. Also owing to these differences, we use the term Customer Interaction Networks (CINs) as an umbrella term to refer to various graphs constructed using E-Commerce search logs, distinguishing them from their web-search counterparts.

[†] The work was done when the second author was at WalmartLabs and the first author was a summer intern there.

This paper is published under the Creative Commons Attribution 4.0 International (CC BY 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.
WWW 2018, April 23–27, 2018, Lyon, France
© 2018 IW3C2 (International World Wide Web Conference Committee), published under Creative Commons CC BY 4.0 License.
ACM ISBN 978-1-4503-5639-8/18/04.
https://doi.org/10.1145/3178876.3186174



Query	Item	Time
Q1	I1	07 :00
Q2	I2	07:15
Q3	I3	07:20

Query	Item	Time
Q2	I3	09 :00
Q3	I4	09:15
Q4	I3	09:30

Query	Item	Time
Q2	I3	17 :00
Q3	I4	17:15
Q5	I5	17:30

Query	Item	Time
Q4	I3	17 :00
Q5	I4	17:15
Q1	I2	17:30

(a) Session logs (b) *Query Reformulation* Network (c) *Item Click* Network

Figure 1: Network Creation Process. Snapshots of four distinct session logs. Each entry in the log consists of a query, an item, the time of engagement among other data. (b) *Query Reformulation* **network and (c)** *Item Click* **networks constructed from the session logs.**

In this paper, we present a formal study of the properties of various graphs constructed using E-Commerce search logs, with a focus on their utility toward mining query-query and query-product relationships. Our proposed graph mining based techniques provide us a complementary source for discovering query and product relationships, without directly using their textual content, making our techniques language independent. We begin by studying the properties of real-world customer interaction networks developed using Walmart.com's product search logs. We observe that CINs exhibit significantly different properties compared to other real world networks (e.g. WWW, social networks etc.), making it possible to mine intent relationships between queries, based purely on their structural information. We then leverage CINs for three different query relations mining tasks. Our main contributions are:

- **Empirical Study:** We study structural properties of four different CINs, namely *Query Reformulation*, *Item Click*, *Composite Click*, and *Cover* networks.
- **Graph Theoretic Problem Formulation:** We formulate intent based query clustering and critical query mining problems as formal Query Clustering and Critical Queries problems.
- **Algorithms:** We propose carefully designed efficient HubQ-Expansion and Critical-Queries algorithms to solve Query Clustering and Critical Queries problems respectively.

We have omitted proofs for some of the lemmas due to lack of space.

2 RELATED WORK

Network Analysis. [10, 16] were among the first to study the macroscopic structure of large scale networks. Recently, Zhang et al. [39] studied the structure of expertise networks, Shen et al. [34] presented empirical study of E-commerce marketplace network, and Zlatic et al. [41] studied the network formed by hyperlinks among Wikipedia pages. Community detection is a well-studied problem in network setting [20, 31], including for bipartite networks [3] and heterogeneous networks [26]. Other important tasks related to our work include discovering influential nodes in networks [9, 24] and measuring node centrality [8, 33].

Query Graphs. Beeferman and Berger introduced click graphs [4] for clustering similar queries and URLs. Several subsequent works utilized click-graphs for other applications [12, 18, 19] like query-suggestion, document-search, relevance feedback and URL-annotation. Baeza-Yates [1] proposed various variants of query relation graphs including *Cover* graphs. Boldi et al. [6, 7] studied query-flow graphs and used them for related query suggestion. These are similar to our *Query Reformulation* networks (Section 3.1), except that they also use query content to decide the existence of query-query edges.

Query Relation mining. Many techniques have been employed in mining relations between queries. Query clustering has been exploited to group similar queries [2, 4, 37]. Some other works are based on association rules [17] and modeling users [40].

3 DATA AND APPLICATIONS

3.1 Our Networks

In this work, we used session level customer interaction data collected over a year's period from Walmart.com. The collected data consists of information including query string, clicked items, time of interaction etc. From these, we created four CINs, namely *Query Reformulation*, *Item Click*, *Composite Click*, and *Cover* networks (See Figure 1).

Query Reformulation Network. *Query Reformulation* network is a directed weighted network $G(Q, E, W)$, where each node $q \in Q$ is a query string. A directed edge (q_1, q_2) exists if query string q_2 was a consecutive reformulation of query string q_1 within a session. The weight $w(q_1, q_2) \in \mathrm{R}^+$ for edge (q_1, q_2) indicates frequency with which q_1 tends to get reformulated to q_2. To filter out noisy and insignificant data, we define some constraints. An edge (q_1, q_2) is added to the network $G(Q, E, W)$, only if the support of query q_1 is greater than δ_1 and the reformulation ratio from q_1 to q_2 is greater than δ_2 percentage. For our experiments both δ_1 and δ_1 were in the range $[0, 20]$[1]. Note that filtering out the noisy nodes and edges does not affect the performance of our algorithms as we are concerned only about significant reformulation relations. After clearing out insignificant edges and nodes, our final *Query Reformulation* network has 2.11 million nodes and 2.14 million edges.

Item Click Network. *Item Click* network is a bipartite weighted network $B(Q, I, E, W)$ where Q is the query partition and I is the item partition. A query node $q \in Q$ is a query string. An item node

[1] The exact values of the threshold is not disclosed due to confidentiality issues.

$i \in I$ is an item id. An edge $(q, i) \in E$ exits if a customer clicks on an item i after giving query q. The weight $w(q, i) \in \mathrm{R}^+$ for edge (q_1, q_2) indicates frequency with which i tends to get clicked for query q. Similar to *Query Reformulation* network, we filter out insignificant edges. Our final *Item Click* network has 5.4 million nodes and 18.4 million edges.

Composite Click Network. *Composite Click* network is a standard click network consisting of both query-to-query and query-to-item edges. We created *Composite Click* network by superimposing the *Query Reformulation* and *Item Click* networks. The resulting network has 6.3 million nodes and 20.5 million edges.

Cover Network. From our *Item Click* network, we inferred query-to-query *Cover* network. In the *Cover* network, two queries have an edge between them if they share an item neighbor in the *Item Click* network. Query to item click relations are clear indication of query intents. Hence, the edges in the the *Cover* network, inferred from click relations, naturally represent similar intents between two query nodes. These networks tend to get very dense. Hence we imposed additional threshold on edge-weight. The resulting graph has 785 thousand nodes and 71 million edges.

3.2 Applications

Since our CINs capture various facets of customer interaction data, they can be leveraged for various applications like query clustering, improving performance of queries with no engagement data, and so on. One can design methods based on query contents for these applications, however our goal here is to exploit the network structure to solve these problems. An advantage of leveraging the graph structure over a content-based approach is that graph based methods are language independent. Hence, our approach can be easily used for any E-commerce search system, regardless of the language it uses. Moreover, our methods are complementary to language/content-based approaches. Combining these two approaches is an interesting future work.

In this work, we focus on leveraging CINs for three different applications. Descriptions of the applications are as follows.

Intent Based Query Clustering: In E-commerce search, identification of query intent is crucial to returning relevant items. An intent of a query is a mapping to attribute-value pairs of the products. Ultimately, it is represented as a set of products.

Product Recommendation: In any E-commerce search system, one often encounters queries with no customer engagement data. In this application, we exploit query relations to recommend products for poorly performing queries.

Critical Queries: Critical queries are the queries which have the highest impact on the performance of other queries. In this application, we try to exploit structure of the *Query Reformulation* network to identify most critical queries. We formalize the notion of critical queries in a later section.

In the next sections, we first characterize various structural properties of our CINs. We then discuss how to exploit them for various applications.

4 CHARACTERIZING OUR NETWORKS

It is well-known that most real networks like WWW, social networks, the Internet, buyer-seller networks, etc. [10, 16, 34] demonstrate specific regular structural properties. In this section, we investigate the structural properties of our CINs and show how they differ from other networks. These differences have a major implications for our applications.

4.1 Degree Distribution

Many real networks are scale free in nature, i.e, the in-degree and the out-degree follow power law distributions [10, 16]. The probability of a node having a degree θ in a scale-free networks is given by the probability density function $P(\theta) \propto \theta^{-\alpha}$. Another distribution that is prevalent in real networks is the log-normal distribution where $P(\theta) = \frac{1}{\sqrt{2\pi}\sigma\theta} e^{-(\ln\theta - \mu)^2/2\sigma^2}$ [28]. Both distributions are heavy tailed, i.e. they have (near) linear log density.

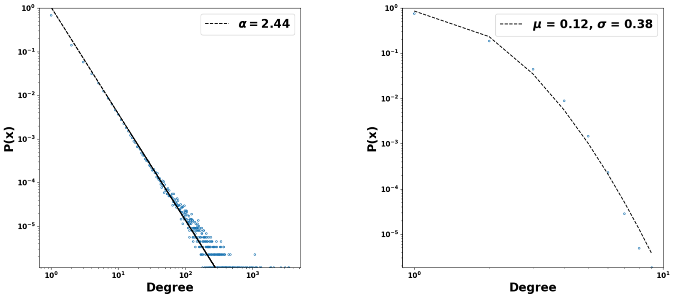
We first look at the degree distributions of the *Query Reformulation* network. The observed empirical pattern in the degree distribution of *Query Reformulation* network is summarized in the following observation.

Observation 1. ***Query-Query degree dist.*** *The in-degree distribution of the Query Reformulation network follows power law distribution with* $\alpha = 2.41$*, while the out-degree distribution follows log-normal distribution with* $\mu = 0.12$ *and* $\sigma = 0.38$*.*

The degree distribution plots for *Query Reformulation* network are presented in Figure 2. The in-degree follows power law distribution with multiple nodes having in-degree greater than 1000. On the other hand, the maximum out-degree is 9, which is negligible in comparison. Note that, our noise filtering process contributes in reducing the maximum value of out-degree in the *Query Reformulation* network to some extent. However, the exact value of the maximum out degree is much less than that warranted by our thresholds. This suggests that while it is probable that many queries get re-formulated into a single query consistently, it is not the case where one query repeatedly gets reformulated into many other queries. This observation is very different from other networks where both in and out degree tend to have similar power law distributions [10, 34]. We found the queries with highest in-degrees tend to be very general queries such as “sweatshirts”, “tablets”, “tv” etc. The in-neighbors of these queries tend to be more specialized queries such as “hooded fleece sweatshirts”, “infant sweatshirts”, “hp tablet”, “htc tablet” etc.

In the bipartite *Item Click* network $B(Q, I, E, W)$, we look at the degree distribution of the query partition Q and the item partition I individually. We observed that the degree distribution for both partitions follow log-normal distribution. Similarly, we also observed that the degree distribution for *Cover* network follows log-normal distribution while that for *Composite Click* network follows power law distribution.

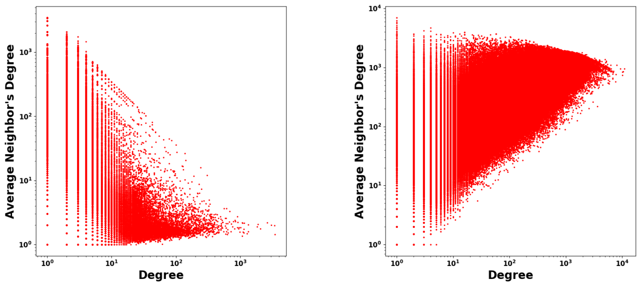
In summary, we found that degree distributions for all of our CINs follow heavy tailed degree distributions. The heavy tailed degree distributions indicate that while there exist some popular queries which connect with many other queries and items, most queries connect only to a few queries and items. Hence, the networks (due to sparse connections only between relevant nodes)



(a) In-degree Distribution

(b) Out-Degree Distribution

Figure 2: In and out degree distributions of *Query Reformulation* network.



(a) *Query Reformulation*

Cover

Figure 3: Assortativity Plots (degree vs average neighbor's degree) for *Query Reformulation* and *Cover* networks.

preserve the relationships between queries and items for the most part.

4.2 Assortativity and Degree Correlation

Degree assortativity, $r \in [-1, 1]$, is a measure of similarity between nodes and their neighbors in terms of degree [31]. Formally, degree assortativity is defined as the Pearson Correlation Coefficient of degrees between all pairs of connected nodes. The value $r = -1$ implies that the network is disassortative (negative correlation) and $r = 1$ implies that the network is assortative (positive correlation). Social networks are known to be assortative. However, other networks like protein-protein interaction network are known to be disassortative [30]. The observation regarding assortativity of our CINS networks is as follows:

Observation 2. ***Degree assortativity.*** *Query Reformulation, Item Click, and Composite Click networks are neither assortative nor disassortative, with $r = -0.02$, $r = -0.09$, and $r = -0.07$ respectively, while the Cover network is assortative with $r = 0.22$.*

The assortativity plot for the *Query Reformulation* and *Cover* networks are shown in Figure 3. For *Query Reformulation* network, we observe that the neighbors of high degree nodes have very low degrees. On the other hand, high degree nodes connect to each other predominantly in the *Cover* network. For *Item Click* and *Composite Click* networks, we do not observe any asymmetrical pattern. The positive assortativity of the *Cover* network implies that it is ill-suited for query intent mining, as general influential queries which typically have distinct intents, tend to connect to each other. The degree distribution and assortativity of the *Query Reformulation* network suggests the dominance of star-like structures in the network. It highlights that unpopular queries are typically reformulated to related popular queries, capturing the intent of the queries.

4.3 Connected Components, Diameter, and Clustering

Many real directed networks are known to have the "bow-tie" structure with a giant strongly connected component (SCC)[10, 39]. It is reported that the WWW has SCC consisting of 27.7% of the nodes [10], while community expertise network for Java forum has SCC consisting of 12.3% of the nodes [39]. In our only directed network, the *Query Reformulation* network, we do not find the "bow-tie" structure, with just 300 out of 2.11 million nodes in the largest strongly connected component. The reason for absence of "bow-tie" structure can be attributed to customer behavior. It is unlikely that customers reformulate a query with an specific intent to a query with drastically different intent repeatedly. Most significant reformulations are related, thus creating distinct partitions of graphs which are not reachable from each other in both directions. On the other hand, the web pages can arbitrarily link to one another in the WWW and people with different expertise may interact with each other in the Java Forum network, which leads to a formation of SCC in these networks.

Another common property exhibited by most real world networks is the "small world" phenomenon, commonly referred to as six degree of separation. Very large real networks like the WWW, social networks e.t.c are known to have small diameters [10, 36]. However, all three of *Query Reformulation*, *Item Click*, and *Composite Click* networks have relatively large diameters of 94, 37, and 36 respectively. This suggests that "weak links" are missing in these networks which implies customers do not typically search for unrelated queries one after another and do not click on arbitrary items for a given query significant number of times. On the other hand, the diameter of the *Cover* network is only 12, which suggests that *Cover* network does not consist of regions representing homogeneous intent.

Average Clustering Co-efficient, $ACC \in [0, 1]$, of a network measures how well the nodes are clustered together. The value of $ACC = 0$ indicates that the network is not clustered at all, whereas $ACC = 1$ indicates that the network is well clustered. For bipartite networks, clustering co-efficient is defined in terms of overlapping neighbors of nodes in the same partition [25]. We computed the average clustering co-efficient for all of our networks and observed that the *Query Reformulation*, *Item Click*, and *Composite Click* networks have very low clustering co-efficient of 0.05, 0.12, and 0.07, respectively, while the *Cover* network has very high clustering co-efficient of 0.76. The clustering co-efficient gives further validation of previous implication that *Query Reformulation*, *Item Click*, and *Composite Click* network are suitable for query intent mining, while the *Cover* network is not.

Table 1: Summary of properties of CINs. QQ stands for *Query Reformulation*, Qi for *Item Click*, QQI for *Composite Click* and C for *Cover* networks. ACC stands for average clustering coefficient.

Properties	QQ	QI	QQI	C
degree	power-law log-normal	log-normal	power-law	log-normal
assortativity	none	none	none	positive
diameter	94	37	36	12
ACC	0.05	0.12	0.07	0.76

4.4 Summary

In this section we explored various properties exhibited by our CINs. The properties of our networks indicate that they are different from common real world networks and that they preserve relevance between queries and items. Thus, our CINs can be leveraged for various query mining tasks. In the next three sections, we explore applications of CINs in query intent mining and product recommendation.

5 APPLICATION 1: INTENT BASED QUERY CLUSTERING

In E-commerce search, identification of query intent is crucial to returning relevant items. However, in practice, one encounters with many queries with ambiguous intent due to very little engagement data. An approach to identify intent of such queries is to cluster them with other queries whose intent is known and leverage the general intent of the cluster to recommend product for queries with low engagement data. Clustering queries based on intent is known to be useful in many potential applications like query recommendation, categorization etc. in both web and E-commerce search [2, 4].

Since our *Query Reformulation* network captures the significant reformulation relations, we propose to exploit the *Query Reformulation* network to cluster the queries with same intent.

5.1 Problem Formulation

Recall that the *Query Reformulation* network is a query-to-query reformulation network. Hence neighboring queries in the *Query Reformulation* network are similar to each other. Therefore, intuitively a community in the *Query Reformulation* network is expected to consist of queries with similar intent. Hence the problem of intent based query clustering in the *Query Reformulation* network is well-founded. The problem can be stated as follows:

Informal Problem 1. *Query Clustering*
Given: A Query Reformulation network $G(Q, E, W)$, and an integer $k \in \mathbb{Z}$.
Find: A k partition of Q, such that each partition contains queries with the same intent.

To formalize Informal Problem 1, two questions must be addressed (i) How is intent defined in terms of graph structure? (ii) How to measure 'closeness' between two queries in terms of intent?

To address the first question, we rely on the empirical study. As mentioned in Section 4, nodes with high in-degree tend to be general queries with broad intent like 'tv', 'phone', 'sweater' etc. Majority of specific queries reformulated to these general queries tend to have similar intents. Hence, these general queries with high in-degree nodes in the *Query Reformulation* network are good candidates to represent the intent. To address the second question, we look at the edge relation in the *Query Reformulation* network. Each edge in the *Query Reformulation* network represents significant reformulation. Therefore, shorter reformulation paths from one query to another is a good indication of similar intents and vice-versa. Hence both questions (i) and (ii) can be answered in terms of the graph structure.

Next, we formalize Informal Problem 1 leveraging two graph properties (i) high in-degree nodes and (ii) shortest paths. Given a *Query Reformulation* network and the number of distinct intents k, our goal is to discover k disjoint partitions $\{C_1, C_2, \dots, C_k\}$. Since intents are well-represented by the high in-degree nodes, we formalize the problem by asking to find a set $\mathcal{S} = \{s_1, s_2, ..., s_k\}$ of such nodes and partitions $C = \{C_1, C_2, ..., C_k\}$, such that C_i is the partition with the intent represented by s_i. Moreover, since the short reformulation path indicates closer intent between queries, we require nodes in C_i to have a short distance to s_i.

Let $\theta^i(v)$ be the in-degrees of v; $d(a, b)$ be the shortest hop distance between two nodes a and b; and $s(v)$ be the node in $\mathcal{S}$ such that both the nodes v and $s(v)$ belong to the same partition (i.e. $s(v)$ is the seed node of the community v belongs to). Now, our formal problem, purely in terms of network structure, can be stated as follows:

Problem 1. *Given a query reformulation network $G(Q, E)$ and an integer k, identify a set $\mathcal{S}^* = \{S_1, S_2, ..., S_k\}$ of the general query nodes and set of partitions $C^* = \{C_1, C_2, ..., C_k\}$, such that $S_i \in C_i$ and*

$$S^*, C^* = \arg\min_{S,C} J(S, C) = \arg\min_{S,C} \left[\left(\sum_{v \in V} d(v, s(v)) \right) \left(\sum_{s \in S} \frac{1}{\theta^i(s)} \right) \right]$$

5.2 Methods

Since our original problem requires partitions, traditional community detection methods are natural baselines for our problem. Hence we use an existing community detection method based on modularity [32], an overlapping community detection method and a heuristic specifically designed for Problem 1 as baseline methods. Brief descriptions are as follows:

- Louvian: We used the popular Louvian method to maximize modularity in *Query Reformulation* network [5].
- BigClam: It is an overlapping community detection method based on bipartite affiliation model [38].
- LouvianSmall: Most queries share intent with few other queries. Hence, we modified the Louvian to generate smaller communities by defining threshold on the first stage of the Louvian algorithm.
- Star: Since *Query Reformulation* network is dominated by star-like structures, we generate star shaped communities by clustering high in-degree nodes with their neighbors. This approach is designed to choose high degree nodes as the community center. Hence, it is a heuristic for Problem 1.

We run Louvian to cluster the *Cover* and the *Composite Click* networks as well. Since, the *Composite Click* network is a heterogeneous network, we also used modified version of Louvian to maximize the composite modularity [26] defined on heterogeneous networks. We name this method ComLouvian.

While traditional community detection methods are natural baselines for Problem 1, they would be sub-optimal as they do not directly optimize the given objective. Our main idea is to leverage the structural properties of the *Query Reformulation* network instead, to solve Problem 1. We exploit the following properties: (a) over half of the nodes in the *Query Reformulation* network lie outside the giant weakly connected component, (b) the assortativity plots (see Figure 3) shows that the high in-degree nodes are very unlikely to have an edge between them, and finally, (c) the *Query Reformulation* network has low clustering co-efficient and long diameter which indicates that the queries with distinct intents are well separated. Based on these observations, we propose our algorithm HubQExpansion (Hub-Query Expansion) for clustering queries with similar intents in E-Commerce *Query Reformulation* network.

Property (a) indicates that the significant number of queries exist outside the giant connected component, hence we cluster queries in each connected components. We distribute the number of communities to be found in each connected component proportionally to their size, i.e., for each connected component $G_i(Q_i, E_i, W_i)$ in G, the number of community to be found is set to $k_i = \frac{|Q_i|}{|Q|}$. Following the property (b), we assign k_i nodes with highest in-degree in the component G_i, to their own community. Assigning high degree nodes to to their own community is justified as they tend to be general queries and it is intuitive that general queries like ‘tv’ and ‘sweater’ have distinct intents. Finally, (c) suggests that queries with distinct intents are well separated. Hence, we expand the communities using breadth-first search. We continue community expansion until all the nodes in the connected component are assigned to a community. The complete pseudocode is in Algorithm 1.

The objective in Problem 1, involves two terms $\sum_{v \in V} [d(v, s(v))]$ and $\sum_{s \in S} \left[\frac{1}{\theta^i(s)}\right]$. Intuitively, Algorithm 1 tries to optimize the second term of the objective by assigning high in-degree nodes as the cluster centers and the first term by assigning nodes to the same community as the closest (shortest-path) cluster centers. Since we observe that the high in-degree nodes tend to have short paths to many queries and also are well-separated with each other, we expect the solution obtained from Algorithm 1 to minimize both terms in the objective and result in a good solution to Problem 1.

Lemma 5.1. *Algorithm 1 has linear time complexity of $O(m + n)$, where m is the number of edges and n is the number of nodes.*

5.3 Experiments

Metrics. Measuring how well the methods minimize the objective in Problem 1 demonstrates their ability in solving the problem. However, it does not indicate how well the communities are clustered in terms of their intents. Since sets of relevant items were not available for most queries, we treat product category learned from an accurate tagger as the proxy for query intents. Intuitively, if two queries have associated items in common, they should also have product categories in common. Hence, product categories are good proxy for intent. We obtained categories for 267K queries, which we use to evaluate all the methods.

Algorithm 1 HubQExpansion

```
Require: Query Reformulation network $G(Q, E, W)$ , number of communities $k$
Ensure: $k$ disjoint partitions of $Q$
Partition $P = \emptyset$
for each connected component $G_i(Q_i, E_i, W_i)$ in $G$ do
    $k_i = \frac{|Q_i|}{|Q|}$
    Temp set $S = \emptyset$
    for node $v$ in $k_i$ nodes in $Q_i$ with highest in-degree do
        $S = S \cup \{v\}$
        Assign nodes in $Q_i$ to nodes in $S$ using BFS
    $P = P \cup S$
return $P$
```

Table 2: Performance of Louvian on *Query Reformulation*, *Composite Click*, and *Cover* networks. The table shows *AIH*, *AIS*, and $F1$ based on categories. The performance of Louvian on *Query Reformulation* network is the best.

Networks	AIH_{cat}	AIS_{cat}	$F1_{cat}$
Query Reformulation	0.26	0.11	**0.15**
Composite Click	0.07	0.31	0.11
Cover	0.05	0.54	0.09

Table 3: Performance of various methods for Query Clustering in *Query Reformulation* network. The table shows *AIH*, *AIS*, and $F1_{cat}$. The final objective value J is also shown. HubQExpansion outperforms all the baselines.

Method	AIH_{cat}	AIS_{cat}	$F1_{cat}$	$J(\times 10^6)$
Louvian	0.26	0.11	0.15	19.7
ComLouvian	0.07	0.33	0.12	118.7
LouvianSmall	0.39	0.08	0.13	0.73
Star	0.38	0.12	0.18	3.01
BigClam	0.14	0.21	0.17	17.7
HubQExpansion	0.37	0.14	**0.20**	**0.54**

A measure of cluster goodness is the categorical homogeneity of each community. To that end, for a community C, we define its Community Intent Homogeneity *CIH* as the fraction of node pairs which share a category, i.e., $CIH(C) = 2 * \frac{\sum_{q_i, q_j \in C} \delta(PC(q_i), PC(q_j))}{|C| \times |C-1|}$, where $PC(q_i)$ represents the category associated with node q_i and $\delta(a, b) = 1$ if $a = b$, 0 otherwise. Note that for *CIH*, we only include the nodes for which category information is available. We then compute the Average Intent Homogeneity AIH_{cat} for a partition P as $AIH_{cat} = \frac{\sum_{C \in P} CIH(C)}{|P|}$. The AIH_{cat} score of 0 represents that the communities in the partition are heterogeneous, while the AIH_{cat} score of 1 represents that the communities are perfectly homogeneous.

AIH_{cat} has a drawback as the smaller communities tend to get higher score. Hence, to overcome this, we also measure the number of communities in which a category is represented. Ideally, we would want each category to be represented in a single community. Hence, we measure average inverse spread AIS_{cat} of a category as $\frac{1}{Spread}$, where *Spread* is defined as the average number of communities the categories are represented in. The AIS_{cat} score of 1 represents that each category is represented in a single community, while a score close to 0 represents that the categories are spread across communities. Finally, we compute $F1_{cat}$ as the harmonic mean of AIH_{cat} and AIS_{cat}.

Performance. First, we ran LOUVIAN on the *Query Reformulation*, the *Composite Click*, and the *Cover* networks. The results are summarized in Table 2. The results show that the clusters obtained from the *Query Reformulation* perform the best, indicating that it is the most suitable network for query clustering.

Next, we ran all the methods in the *Query Reformulation* network and COMLOUVIAN in the *Composite Click* network. First, we computed performance of the methods with respect to the objective of Problem 1. The result is presented in Table 3. As expected HUBQEXPANSION outperforms all the baselines in terms of the Problem 1 objective. Next, we computed the intent based metrics described above. The results are summarized in Table 3. As observed, HUBQEXPANSION outperforms all the baselines. LOUVIAN performs decently indicating that traditional community detection methods are indeed suitable for Problem 1. Poor performance of LOUVIANSMALL indicates that artificially creating smaller clusters prevents good clusters from forming. Naive STAR heuristic performs well due to the fact that communities in the *Query Reformulation* network are centered around the 'popular' queries. However, HUBQEXPANSION outperforms all the methods, since it exploits the unique structure of the *Query Reformulation* network to find communities with the same intent.

6 APPLICATION 2: PRODUCT RECOMMENDATION

Improving search quality for queries with no customer engagement data is a challenging task. In this section, we propose to leverage the *Composite Click* network to associate items with poorly performing queries and evaluate whether such a method does in fact improve search quality for queries with no engagement data.

6.1 Problem and Method

In this task, we explore whether product recommendations made based on the *Composite Click* network could help improve search quality for poorly performing queries. We used the current Walmart.com product search engine as the baseline, and identified poorly performing queries. The criteria for selection was queries in the lowest $10th$ percentile in terms of click through rate, and a conversion rate, defined as $(\frac{\#Querieswithorders}{\#Queries})$, of 0. For variation, we used a random walk based method similar to [12], except that our network is directed (we treat query-to-item links as directed here), and there are no out going edges from the items, making them sink nodes. The unnormalized edge weights for our *Composite Click* network were computed as follows:

- **Query q_1 to q_2 edge:** $w(q_1, q_2) = \frac{c(q_1,q_2)}{c(q_1)}$
- **Query q to product i edge:** $w(q, i) = \frac{c(q,i)}{c(q)}$,

where $c(q_1, q_2)$ represents the number of times q_1 is formulated to q_2, $c(q_1)$ represents the number of times q_1 occurs, and $c(q, i)$ represents the number of times product i is clicked for query q. We further normalized the weight of each directed edge (q, x), where x may be a query or product, by the sum of all out-going edges from the base node q.

In order to make product recommendations for some query q, we started with a weight of 1 at node q, and spread it across the graph by executing random walk iterations. After several iterations (50 were usually enough), a portion of the weight settled on the product nodes. Top 5 highest weighted products were then used as recommendations for q. Our variation ranking method injected these recommended products into the top 10 search results for query q demoting some of the original results below 10th position.

6.2 Experiments

For evaluation, we identified a random sample of 136 poorly performing queries[2]. A dataset of query-product pairs was created by obtaining top 10 results for each query, from both control and variation. Expert E-Commerce analysts were then asked to assign a relevance rating between $0-4$ for each query document pair, 4 being extremely relevant and 0 being irrelevant. We observed that our recommendation based variation performed significantly better achieving a 34% improvement in average NDCG@10[21] (baseline NDCG10: 0.439, NDCG10: 0.588). Out of 136, 99 queries were improved, while 31 were degraded. From a practical point of view the degradations are not harmful, since the queries already have poor conversion rates. This observation was further backed up by our online A/B test evaluation which showed a statistically significant 5.8% lift in click through rate and 6.9% lift in conversion.

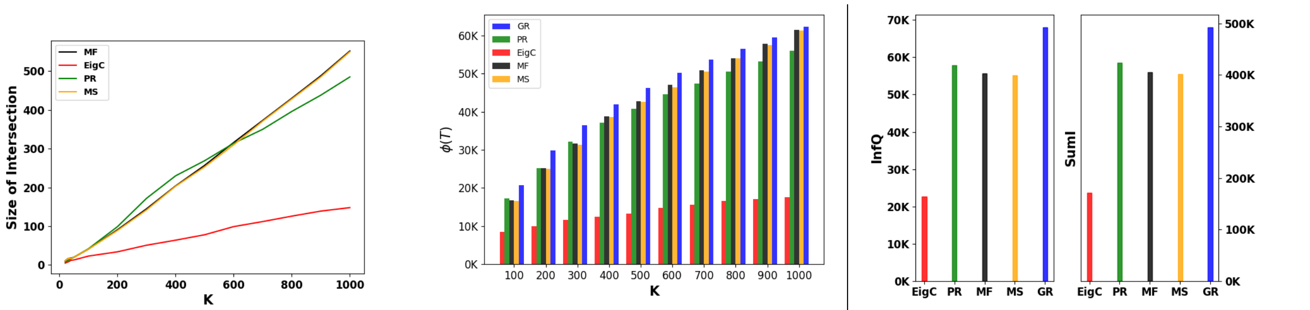
7 APPLICATION 3: CRITICAL QUERIES

In the previous section (Section 6), we showed that the engagement data from one query can be leveraged to improve the performance of other related (via reformulation relation) queries. A natural question that arises in this setting is which queries have the highest cumulative impact on the performance of other related queries? We refer to these as the 'Critical Queries'.

Mining the critical queries is important for improving the overall performance of the search system. Typically in processes like manual curation, which aims at improving search quality by manually improving the search results, the queries that appear most frequently in the search log are selected. However, this yields a sub-optimal improvement in the overall performance as these most-frequent queries do not necessarily improve the performance of other queries. On the other hand, by definition, critical queries have the highest impact on the performance of the related queries. Hence, correctly identifying the critical queries for the curation process would yield a better improvement in performance. Note that the critical queries can be leveraged for many other tasks in E-commerce like measuring performance of a search system, identifying broad search categories and so on.

Since our *Query Reformulation* network captures the reformulation relation well, we propose to mine the critical queries by leveraging the *Query Reformulation* network.

[2] Number of queries was obtained based on the available crowdsourcing budget.



(a) Intersection

(b) $\phi(\mathcal{T})$

(c) Item relevance based performance

Figure 4: (a) Intersection between top-k critical queries returned by Algorithm 2 and other baselines. (b) Our method outperforms all the baselines in terms of $\phi(\mathcal{T})$. (c) Our method performs the best based on usability metrics.

7.1 Problem Formulation

To formalize the problem of choosing the most critical queries, we need to correctly model customer behavior during the query reformulation process. We model customer behavior in query reformulation as a discrete-time dynamical process that occurs over the *Query Reformulation* network. We call it the RANDOMIZED USER NAVIGATION (RUN) model.

RUN Model: In an E-Commerce search system, a customer (user) submits an arbitrary query to the search system. A list of items relevant to the query are displayed. The customer, depending on many factors like satisfaction with the search result, relevance of products etc. may decide to submit another query or exit the search system.

This process can actually be viewed as a discrete-time probabilistic dynamical process over the *Query Reformulation* network $G(Q, E, W)$. Given a current node v, we proceed as follows:

(1) With probability p_t, the process terminates
(2) With probability $1 - p_t$, we continue the process and jump from current node v to a query node u, such that $(v, u) \in E$, with probability $p_j = \frac{w(v,u)}{\sum_{(v,a)\in E} w(v,a)}$

The process starts from an arbitrary query node sampled from Q uniformly at random. Note that, an instance of RUN model produces a sequence of queries which we call 'reformulation logs'.

Now, let $\mathcal{T}$ be a set of nodes. We define $\phi(\mathcal{T})$ as the probability that an arbitrary instance of RUN model goes through at least one node in $\mathcal{T}$. Empirically, $\phi(\mathcal{T})$ is the fraction of times at least one node in $\mathcal{T}$ appears in reformulation log produced by repetitions of the RUN model.

Remark. Since both PAGERANK's Random Surfer model [33] and our RUN model simulate a random walker over a network, they appear to be similar. However, PAGERANK's Random Surfer model is distinct from our RUN model as it has no notion of a termination probability and the walker can teleport to any node in the network. Given enough time, every node $v \in Q$ is visited and hence, $\phi(\mathcal{T})$ for any set $\mathcal{T}$ is always 1 under the Random Surfer model—which is not the case for the RUN model. The RUN model is also distinct from the cascade style models (like IC [24]) as only a single node is visited at a time in the RUN model, whereas the 'contagion' spreading in the cascade models can infect multiple nodes at once (depending on who else was 'infected' in the previous time-step)

Having defined the RUN model and $\phi(\cdot)$, we can state our Critical Queries identification problem formally as follows:

PROBLEM 2. *CRITICAL QUERIES*
GIVEN: A Query Reformulation network $G(Q, E, W)$, and budget $k \in Z$.
FIND: a set of nodes $\mathcal{T}^* = \{q | q \in Q\}$, such that $|\mathcal{T}^*| = k$ and

$$\mathcal{T}^* = \arg\max_{\mathcal{T}} \phi(\mathcal{T})$$

.

7.2 Methods

Problem 2 is NP-hard (we can reduce from the SETCOVER problem; proof omitted due to space). Although it is challenging to solve Problem 2 optimally in an efficient manner, one can use various centrality measures or query logs based heuristics to identify critical queries. Some of these methods can be the following:

- MOSTFREQ (MF): In this method, we select the queries that had the highest frequency from the same data from which the *Query Reformulation* network was created.
- SESSIONFREQ (MS): In this method, we select the queries that appeared in most sessions.
- PAGERANK (PR): We pick the nodes with highest page rank [33] on the *Query Reformulation* network.
- EIGCENTRALITY (EIGC): We pick the nodes with highest eigenvector centrality [8] on the *Query Reformulation* network.

None of methods mentioned above solve Problem 2 directly. Hence, we seek for a fast algorithm with a performance guarantee. It turns out that $\phi(\cdot)$ is sub-modular [24]. A function $f(\cdot)$, which maps a set to a real number, is sub-modular if it satisfies the diminishing return property i.e. $f(\mathcal{A} \cup \{v\}) - f(\mathcal{A}) \geq f(\mathcal{B} \cup \{v\}) - f(\mathcal{B})$, for any element v, and sets $\mathcal{A} \subset \mathcal{B}$. Next, we prove that $\phi(\cdot)$ is sub-modular and monotonous.

LEMMA 7.1. *$\phi(\cdot)$ is sub-modular and monotonous.*

PROOF. In the RUN model, a customer at node q_i makes a decision to whether make a random jump to one of q_i's out-neighbors and which node to jump to. Suppose the customer makes the random decision before the process, i.e, decides on the number of RUN model iterations l, the starting nodes, and the node jumps beforehand. Let the set D be the set of outcomes of the random decisions. Note that given D, these k iterations of RUN are a deterministic processes, which produce l reformulation logs.

Now, let $\mathcal{A}$ and $\mathcal{B}$ be two sets of nodes of *Query Reformulation* network and let $\mathcal{A} \subset \mathcal{B}$. Consider a node v such that $v \notin \mathcal{B}$. Also let $\phi_D(\mathcal{A})$ be the fraction of logs that go through any node is $\mathcal{A}$ given D. Now, $\phi_D(\mathcal{A} \cup \{v\}) - \phi_D(\{v\})$ is the fraction of logs that go though v, which does not already go through any node in $\mathcal{A}$. This value is definitively larger than or equal to $\phi_D(\mathcal{B} \cup \{v\}) - \phi_D(\{v\})$, since $\mathcal{A} \subset \mathcal{B}$. Hence, $\phi_D()$ is sub-modular.

Now, $\phi(\mathcal{T})$ equals $\sum_{\text{Decision } D} \text{Prob}(D)\phi_D(\mathcal{T})$.

Since, any linear combination of sub-modular functions is also sub-modular, $\phi(\cdot)$ is sub-modular. Since, $\phi(\cdot)$ is a non-decreasing function, it is also monotonous. □

Speeding up. Due to Lemma 7.1, a simple greedy algorithm will give $1 - 1/e$ approximation [29]. However, such a method is expensive due to repetitive simulations. Leveraging the idea in [11], we propose sampling based greedy algorithm Critical-Queries. First of all, we initialize set $\mathcal{T}$ to an empty set. We then sample a graph G' from G based on the number of iterations l and termination probability p_t. Given the deterministic network G', we compute $\phi_{G'}(\mathcal{T})$ and $\phi_{G'}(\{v\})$ for every v in G'. We repeat such process R times and choose v with the highest average gain in $\phi(\cdot)$. We repeat the entire process until $|\mathcal{T}| = k$. Note that there exist other techniques from related problems to speed up our algorithm [9, 15]. However, we chose this method for its simplicity. The complete pseudocode is presented in Algorithm 2.

Algorithm 2 Critical-Queries (GR)

```
Require: Query Reformulation network $G(Q, E, W)$ , termination probability $p_t$, number of iterations of RUN $l$, and budget $k$
Ensure: Best set of nodes $\mathcal{T}$
$\mathcal{T} = \emptyset$
for $i = 1$ to $k$ do
    $g_v = 0$ for all $v \in V \setminus \mathcal{T}$
    for $i = 1$ to $R$ do
        Sample $G'$ based on $p_t$ and $l$
        compute $\phi_{G'}(\mathcal{T})$
        for $v \in V \setminus \mathcal{T}$ do
            $g_v += \phi_{G'}(\{v\})$
        $g_v = g_v/R$ for all $v \in V \setminus \mathcal{T}$
        $\mathcal{T} = \mathcal{T} \cup \{\arg\max_v(g_v)\}$
return $\mathcal{T}$
```

As shown by the next two lemmas, Algorithm 2 gives a provable approximation guarantee and has near linear time complexity.

Lemma 7.2. *Algorithm 2 provides a $(1 - 1/e)$ approximation to Problem 2.*

Lemma 7.3. *The time complexity of Algorithm 2 is $O(kR(n + m))$.*

7.3 Experiments

Metrics. While the value of $\phi(\mathcal{T})$ for various methods indicates the quality of $\mathcal{T}$, it does not highlight usefulness of queries in $\mathcal{T}$. Hence we define two additional metrics to measure usability of $\mathcal{T}$.

To improve state of search system by curating the $\mathcal{T}$, the ancestor nodes in *Query Reformulation* network, i.e. queries leading up to $\mathcal{T}$ must actually be related to the queries in $\mathcal{T}$. Hence, we measure whether the ancestor queries of $\mathcal{T}$ are actually relevant to it or not. To this end, we determined the set of items related to each query in $\mathcal{T}$ and computed the number of ancestor queries, $InfQ$, that had at least one common relevant items with the queries in $\mathcal{T}$. Formally, let $\mathcal{A}^d$ be the set of ancestors within distance d of all query q in $\mathcal{T}$. Let, the set of items relevant to set of queries $\mathcal{T}$, be $\mathcal{I}(\mathcal{T})$. Now, we calculate $InfQ$, influenced queries, as $InfQ = \sum_{q \in \mathcal{A}^5} \mathbb{1}(|\mathcal{I}(\mathcal{T}) \cap \mathcal{I}(\{q\})| \geq 1)$. Similarly, another metric of interest is how close the ancestors are related to the critical queries. To capture the notion of overall relation between queries and their ancestors, we also compute the size of set of items, *SumI*, which are relevant for both the nodes in $\mathcal{T}$ and their ancestors as $SumI = |\mathcal{I}(\mathcal{T}) \cap \mathcal{I}(\mathrm{A}^5)|$.

Performance. For Critical-Queries, we set p_t as 0.7 and l as the number of nodes in the network. We ran all the methods on the *Query Reformulation* network. First of all, we check the whether the set $\mathcal{T}$ returned by Critical-Queries is distinct from the ones returned by the baselines. In Figure 4 (a), we plot the size of intersection of $\mathcal{T}$ returned by various methods and $\mathcal{T}$ obtained from Critical-Queries against, k, the size of $\mathcal{T}$. At least 45 % of nodes returned by Critical-Queries are not present in sets returned by any other method. Hence, Critical-Queries returns the critical queries which other methods fail to discover.

The performance of all the methods in terms of $\phi(\mathcal{T})$ is shown in Figure 4 (b). As we can see, Critical-Queries consistently outperforms all the baselines for multiple values of k in terms of $\phi(\mathcal{T})$. PageRank (PR) and EigCentrality (EigC) and have poor performance as they tend to choose nodes which are close to each other and return a set of similar queries. The MostFreq (MF) and SessionFreq (MS) heuristics perform better than other baselines, however, they too suffers from the same problem especially for lower values of k. The results for $InfQ$ and *SumI* are shown in Figure 4 (c). Our method has higher values for both $InfQ$ and *SumI* compared to all the baselines. The results reveal that the queries we find using Critical-Queries are closely related to their ancestor queries showcasing their usability.

8 CONCLUSIONS

In this work, we studied various structural properties of CINs constructed from customer interaction with E-Commerce search engine. Our results show that these networks are significantly distinct from other real world networks. We also observed that the structural properties of CINs, the *Query Reformulation* and the *Composite Click* networks in particular, make them useful for mining query relations. We demonstrated usability of these networks, by leveraging them to cluster queries based on their intents, improve performance of poorly performing queries, and mine critical queries. To cluster queries based on intent, we proposed efficient HubQExpansion algorithm, carefully designed to exploit special structure of the *Query Reformulation* network. Similarly, we modeled user interactions in E-Commerce search system as the RUN model, formulated Critical Queries problem and proposed efficient Critical-Queries algorithm to identify critical queries. Our extensive experiments demonstrate that the *Query Reformulation* network is useful and our methods are successful in mining query relations.

Acknowledgments. This paper is based on work partially supported by the NEH (HG-229283-15), NSF CAREER (IIS-1750407), ORNL (Task Order 4000143330), and a Facebook faculty gift.

REFERENCES

[1] Ricardo Baeza-Yates. 2007. Graphs from search engine queries. *SOFSEM 2007: Theory and Practice of Computer Science* (2007), 1–8.

[2] Ricardo A Baeza-Yates, Carlos A Hurtado, Marcelo Mendoza, et al. 2004. Query Recommendation Using Query Logs in Search Engines.. In *EDBT workshops*, Vol. 3268. Springer, 588–596.

[3] Michael J Barber. 2007. Modularity and community detection in bipartite networks. *Physical Review E* 76, 6 (2007), 066102.

[4] Doug Beeferman and Adam Berger. 2000. Agglomerative clustering of a search engine query log. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 407–416.

[5] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment* 2008, 10 (2008), P10008.

[6] Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, Aristides Gionis, and Sebastiano Vigna. 2008. The query-flow graph: model and applications. In *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 609–618.

[7] Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, and Sebastiano Vigna. 2009. Query suggestions using query-flow graphs. In *Proceedings of the 2009 workshop on Web Search Click Data*. ACM, 56–63.

[8] Phillip Bonacich. 2007. Some unique properties of eigenvector centrality. *Social networks* 29, 4 (2007), 555–564.

[9] Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. 2014. Maximizing social influence in nearly optimal time. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 946–957.

[10] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. 2000. Graph structure in the web. *Computer networks* 33, 1 (2000), 309–320.

[11] Wei Chen, Yajun Wang, and Siyu Yang. 2009. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 199–208.

[12] Nick Craswell and Martin Szummer. 2007. Random Walks on the Click Graph. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07)*. ACM, New York, NY, USA, 239–246. https://doi.org/10.1145/1277741.1277784

[13] Atish Das Sarma, Nish Parikh, and Neel Sundaresan. 2014. E-commerce product search: personalization, diversification, and beyond. In *Proceedings of the 23rd International Conference on World Wide Web*. ACM, 189–190.

[14] Michelangelo Diligenti, Marco Gori, and Marco Maggini. 2011. A unified representation of web logs for mining applications. *Information Retrieval* 14, 3 (2011), 215–236.

[15] Nan Du, Le Song, Manuel Gomez Rodriguez, and Hongyuan Zha. 2013. Scalable influence estimation in continuous-time diffusion networks. In *Advances in neural information processing systems*. 3147–3155.

[16] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. 1999. On power-law relationships of the internet topology. In *ACM SIGCOMM computer communication review*, Vol. 29. ACM, 251–262.

[17] Bruno M Fonseca, Paulo Braz Golgher, Edleno Silva de Moura, and Nivio Ziviani. 2003. Using association rules to discover search engines related queries. In *Web Congress, 2003. Proceedings. First Latin American*. IEEE, 66–71.

[18] Alexandre P Francisco, Ricardo Baeza-Yates, and Arlindo L Oliveira. 2012. Mining query log graphs towards a query folksonomy. *Concurrency and Computation: Practice and Experience* 24, 17 (2012), 2179–2192.

[19] Alexandre P Francisco, Ricardo A Baeza-Yates, and Arlindo L Oliveira. 2008. Clique Analysis of Query Log Graphs.. In *SPIRE*, Vol. 5280. Springer, 188–199.

[20] Michelle Girvan and Mark EJ Newman. 2002. Community structure in social and biological networks. *Proceedings of the national academy of sciences* 99, 12 (2002), 7821–7826.

[21] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.* 20, 4 (Oct. 2002), 422–446. https://doi.org/10.1145/582415.582418

[22] Daxin Jiang, Jian Pei, and Hang Li. 2013. Mining search and browse logs for web search: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)* 4, 4 (2013), 57.

[23] Shan Jiang, Yuening Hu, Changsung Kang, Tim Daly Jr, Dawei Yin, Yi Chang, and Chengxiang Zhai. 2016. Learning Query and Document Relevance from a Web-scale Click Graph. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 185–194.

[24] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 137–146.

[25] Matthieu Latapy, Clémence Magnien, and Nathalie Del Vecchio. 2008. Basic notions for the analysis of large two-mode networks. *Social networks* 30, 1 (2008), 31–48.

[26] Xin Liu, Weichu Liu, Tsuyoshi Murata, and Ken Wakita. 2014. A framework for community detection in heterogeneous multi-relational networks. *Advances in Complex Systems* 17, 06 (2014), 1450018.

[27] Zitao Liu, Gyanit Singh, Nish Parikh, and Neel Sundaresan. 2014. A large scale query logs analysis for assessing personalization opportunities in e-commerce sites. *WSCD ?2014 New York, New York USA, ACM-2014* (2014).

[28] Michael Mitzenmacher. 2004. A brief history of generative models for power law and lognormal distributions. *Internet mathematics* 1, 2 (2004), 226–251.

[29] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. 1978. An analysis of approximations for maximizing submodular set functions?I. *Mathematical Programming* 14, 1 (1978), 265–294.

[30] Mark EJ Newman. 2002. Assortative mixing in networks. *Physical review letters* 89, 20 (2002), 208701.

[31] Mark EJ Newman. 2003. Mixing patterns in networks. *Physical Review E* 67, 2 (2003), 026126.

[32] Mark EJ Newman. 2006. Modularity and community structure in networks. *Proceedings of the national academy of sciences* 103, 23 (2006), 8577–8582.

[33] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web.* Technical Report. Stanford InfoLab.

[34] Zeqian Shen and Neel Sundaresan. 2011. eBay: an E-commerce marketplace as a complex network. In *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM, 655–664.

[35] Yang Song, Dengyong Zhou, and Li-wei He. 2012. Query suggestion by constructing term-transition graphs. In *Proceedings of the fifth ACM international conference on Web search and data mining*. ACM, 353–362.

[36] Jeffrey Travers and Stanley Milgram. 1967. The small world problem. *Phychology Today* 1 (1967), 61–67.

[37] Ji-Rong Wen, Jian-Yun Nie, and Hong-Jiang Zhang. 2001. Clustering user queries of a search engine. In *Proceedings of the 10th international conference on World Wide Web*. acm, 162–168.

[38] Jaewon Yang and Jure Leskovec. 2013. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM, 587–596.

[39] Jun Zhang, Mark S Ackerman, and Lada Adamic. 2007. Expertise networks in online communities: structure and algorithms. In *Proceedings of the 16th international conference on World Wide Web*. ACM, 221–230.

[40] Zhiyong Zhang and Olfa Nasraoui. 2006. Mining search engine query logs for query recommendation. In *Proceedings of the 15th international conference on World Wide Web*. ACM, 1039–1040.

[41] Vinko Zlatić, Miran Božičević, Hrvoje Štefančić, and Mladen Domazet. 2006. Wikipedias: Collaborative web-based encyclopedias as complex networks. *Physical Review E* 74, 1 (2006), 016115.