Query-Based Outlier Detection in Heterogeneous Information Networks

Jonathan Kuck*†, Honglei Zhuang*†, Xifeng Yan‡, Hasan Cam§ and Jiawei Han†
†Department of Computer Science, University of Illinois at Urbana-Champaign
‡Computer Science Department, University of California at Santa Barbara
§US Army Research Lab
{jkuck, hzhuang3, hanj}@illinois.edu, xyan@cs.ucsb.edu, hasan.cam.civ@mail.mil

ABSTRACT

Outlier or anomaly detection in large data sets is a fundamental task in data science, with broad applications. However, in real data sets with high-dimensional space, most outliers are hidden in certain dimensional combinations and are relative to a user's search space and interest. It is often more effective to give power to users and allow them to specify outlier queries flexibly, and the system will then process such mining queries efficiently. In this study, we introduce the concept of query-based outlier in heterogeneous information networks, design a query language to facilitate users to specify such queries flexibly, define a good outlier measure in heterogeneous networks, and study how to process outlier queries efficiently in large data sets. Our experiments on real data sets show that following such a methodology, interesting outliers can be defined and uncovered flexibly and effectively in large heterogeneous networks.

1. INTRODUCTION

Heterogeneous networks are the networks composed of multi-typed, interconnected vertices and links. Since the real world information entities are interconnected, forming numerous, gigantic networks, heterogeneous information networks are ubiquitous and form the basic semantic structure of interconnected data. Thus, detecting anomalies or finding outliers in such networks becomes an important task in network analysis. Although outlier detection has been studied extensively in data mining and various application fields [5, 14], outlier detection in heterogeneous information networks poses several unique challenges:

1. Unlike many outlier analysis methods that work on homogeneous datasets (*e.g.*, find anomalous communications in a communication network), this new endeavor needs fundamental changes on the definition and detection of outliers since it involves heterogeneously typed vertices and links.
2. In a gigantic network, and particularly in a heterogeneous network, it is unrealistic to discover all outliers using "global" techniques. The variety of vertex types, edge types, and paths connecting particular vertices creates many potential viewpoints from which outliers may be classified. These views are difficult to compare and possibly conflicting.
3. Data analysts (as users) need to obtain results promptly to react to outliers or further elaborate their queries. This creates a big challenge to efficiently process outlier queries in heterogeneous information networks.

Based on the above observations, we propose a new outlier detection task, called *query-based outlier detection in heterogeneous information networks*, by facilitating users to compose various kinds of outlier queries flexibly in heterogeneous networks via a novel query language; defining a new outlier measure, called NetOut, to measure the outlierness in such heterogeneous networks; and developing an efficient network detection algorithm for this task. The following is an example that illustrates our ideas.

Motivating example. The DBLP network is a network generated from the computer science bibliographic publication database[1] that consists of a set of vertex types: paper (P), venue (V), author (A), and term (T). A research publication entry essentially generates a set of links of the types $P-V$, $P-A$, and $P-T$, each connecting in the network a paper with its publication venue, set of authors and set of terms, respectively.

It is unrealistic and meaningless to find outliers with respect to all types of the vertices in the entire heterogeneous network. However, it is more interesting to give users freedom to specify what they want. For example, a user can confine the outliers to be among the coauthors of Christos Faloutsos (*i.e.*, all the authors connected with Christos via at least one joint paper).

Even for this author set, it is still unclear what aspect the outliers should be judged by: should the outliers be judged based on their publication venues or their collaborators? The former may lead to finding those who publish multiple papers in rather different venues than the majority of Christos' coauthors; whereas the latter may find those who have rather different collaboration behavior than the majority of his coauthors. Different judgment criteria lead to rather different results, which makes it essential to

*The first two authors made equal contributions.

[1] `http://www.informatik.uni-trier.de/~ley/db`

10.5441/002/edbt.2015.29

ask users to specify the criteria explicitly. Furthermore, a user may like to find outliers among Christos' coauthors, not compared within this coauthor set itself but compared with another explicitly specified set, such as prolific EDBT authors (*e.g.*, those who have published at least 10 papers in EDBT).

From this example, one can see that it is necessary to provide an outlier query language with which a user can specify the candidate set (*e.g.*, Christos' coauthors), the aspect (*e.g.*, publishing venues) by which the outliers will be judged, and sometimes the reference set (*e.g.*, prolific EDBT authors). With such primitives, a user can flexibly and unambiguously specify the outliers to be mined in a heterogeneous network.

Besides providing flexible ways for users to interact with the system to specify outlier queries in networks, another important issue is how to define the outlier measures for heterogeneous networks. Taking the query, "finding outlying co-authors of Christos in terms of their publishing venues", it is important to work out a good outlier measure in heterogeneous networks so that one can readily identify the outliers among Christos' co-authors who published multiple papers at rather different venues than that by the majority coauthors of Christos. Such intuition may help us work out a new definition of outlier measure in heterogeneous networks.

In this study, we work on this interesting problem and have made the following contributions.

1. We introduce the concept of query-based outlier in heterogeneous information networks, formalize different components for an outlier query in such networks, and develop a user-friendly, meta-path based outlier query language that allows users to interact with the outlier detection system using their intuition;

2. We introduce a new outlier measure, NetOut, which defines query-based outlierness in heterogeneous information networks, with respect to the queries specified by users;

3. We develop an efficient computation method to find query-based outliers in heterogeneous information networks and analyze our performance improvement in the efficiency study.

The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 introduces the basic concepts. Section 4 introduces the formal definition of outlier query, and designs a query language as an interface for users to specify outlier queries. Section 5 develops a new outlier measure, NetOut, and shows its effectiveness. Section 6 outlines the implementation of the proposed outlier detection system as well as several query optimization techniques. The performance study of the comparative methods, as well as the efficiency study are reported in Section 7. We present an overall discussion in Section 8 and provide concluding remarks in Section 9.

2. RELATED WORK

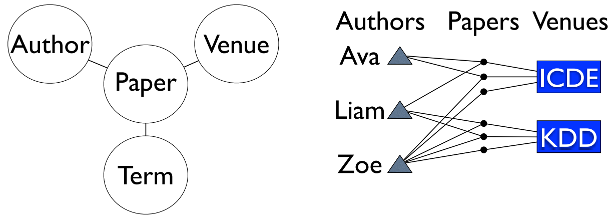
Outlier detection. The field of outlier detection has been explored for years. A good overview of outlier detection techniques can be found in surveys [5, 14].

Our work is most related to the thread of research on networked data outlier detection. There are some early explorations on network outlier detection on bipartite graphs [23]. But most existing studies are confined to homogeneous networks. For example, Gao *et al.* [6] studied contextual outliers in (homogeneous) networks, *i.e.* outliers deviating from their closely connected peers; Akoglu *et al.* [1] proposed OddBall, which takes several network structural properties as features to identify outliers in weighted graphs; Gupta *et al.* [8] studied outliers in terms of their abnormal dynamics among communities; Perozzi *et al.* [19] and Li *et al.* [18] explored outlier detection in attributed graphs. Zong *et al.* [30] studied how to detect abnormal network events and their possible sources. However, these methods are not applicable to heterogeneous information networks. For heterogeneous networks, Gupta *et al.* [7] proposed to measure outlierness based on community distribution of each vertex in the network; Gupta *et al.* [9] also studied outlier detection based on assumption of association-based cliques in networks.

Although a great variety of research has been done on outlier detection given a data set, few of them really consider doing outlier detection in a query-based fashion. Gupta *et al.* [11] proposed using a template subgraph as a query for finding outlier subgraphs, but the definition of a query in this work is not general enough to be extended to most common use cases. Efficient mining of top-k outliers in large databases has been studied in early research. For example, Ramaswamy *et al.* [20] proposed a partition-based algorithm to mine top-k outliers in very large databases, using a distance-based outlier detection algorithm [17]; Jin *et al.* [15] presented an algorithm based on "micro-clusters" to find top-k outliers using the local outlier factor (LOF) measure [4]. Nevertheless, they only optimize for a certain type of outlier definition on the entire data set. Schubert *et al.* [22] proposed a generalized point of view for local outlier detection, but it does not explicitly consider the query-based scenario. In this work we generalize the outlier detection framework and give users flexibility to specify their own definition of an outlier.

Query languages for heterogeneous networks. Managing data organized in heterogeneous information networks is a challenging problem. Compared to a traditional relational database, data is organized in an arbitrary and potentially more complicated graphical structure. There are several different threads of research on developing graph query languages and optimizing queries. A comparison of different graph database models can be found in [2, 3]. Research related to the semantic web usually organizes information into machine-readable web information represented in a language called Resource Description Framework (RDF) [12]. Optimization of the RDF query language SPARQL is studied in [21]. Cypher[2] is another query language utilized by open-source graph database Neo4j; while GraphQL [13] is another query language which supports querying by graph structure. For graph query optimization, Yan *et al.* [26, 27] and Zhao *et al.* [29] proposed indexing strategies to efficiently process graph queries. As knowledge graphs have attracted more studies in recent years, there is also some recent research on querying schema-less graphs. Kasneci *et al.* [16] studies keyword search on knowledge graphs; Yang *et al.* [28] propose an SQL framework where users do not need to specify the querying graph schema/structure precisely. Gupta *et al.* [10] proposed a method to efficiently find the top-k most inter-

[2] http://docs.neo4j.org/chunked/stable/cypher-introduction.html



(a) Schema (b) Instantiated network

Figure 1: Bibliographic network schema and instantiated network.

esting subgraphs in a heterogeneous network. However, as far as we know, none of the current graph query languages explicitly support queries for outlier detection in graphs.

3. PRELIMINARIES

Real-world informational or abstract entities are often interconnected, forming multiple gigantic networks. When such networks can be structured around a small number of entity/link types, many interesting properties can be explored systematically. Here we introduce a few related concepts.

Definition 1 (Heterogeneous information network). *A heterogeneous information network [25] is an information network with multiple types of vertices. Without loss of generality, it can be defined as a directed network $G = (\mathcal{V}, \mathcal{E}; \phi, \mathcal{T})$ where $\mathcal{V}$ is the set of vertices, and $\mathcal{E}$ is the set of edges. There is a vertex type mapping function $\phi : \mathcal{V} \rightarrow \mathcal{T}$ where $\mathcal{T}$ is the set of types, i.e., each vertex $v \in \mathcal{V}$ belongs to a particular type $T \in \mathcal{T}$. For undirected cases, an undirected edge can be viewed as two symmetric directed edges. When there exists only one vertex type, the network reduces to a homogeneous information network.*

A bibliographic network, such as DBLP, is a heterogeneous information network where there are four types of vertices: paper (P), venue (V), author (A), and term (T), and an edge type represents a type of link between two vertex types (*e.g.*, $P-V$ represents that paper P was published in venue V).

To formalize the relationships between two vertices in a heterogeneous network, meta-paths [24] have been used to represent semantic links at the schema level.

Definition 2 (Meta-path). *In a heterogeneous network G, a meta-path is an ordered sequence of vertex types, denoted as $\mathcal{P} = (T_0 - T_1 - \ldots - T_l)$, or $\mathcal{P} = (T_0 T_1 \ldots T_l)$, where $T_x \in \mathcal{T}$.*

In a bibliographic network, $\mathcal{P}_{ca} = (APA)$ is a meta-path, representing the coauthorship between two authors.

We also introduce two basic operators for meta-paths.

Definition 3 (Reversal of a meta-path). *A meta-path $\mathcal{P} = (T_0 T_1 \ldots T_l)$ can be reversed, where the reversed path is denoted as $\mathcal{P}^{-1} = (T_l T_{l-1} \ldots T_0)$.*

As an example, if $\mathcal{P} = (APV)$, then its reversal $\mathcal{P}^{-1} = (VPA)$.

Definition 4 (Concatenation of meta-paths). *Given two meta-paths $\mathcal{P}_1 = (T_{1,0} \ldots T_{1,l})$ and $\mathcal{P}_2 = (T_{2,0} \ldots T_{2,l'})$. If $T_{1,l} = T_{2,0}$, then $\mathcal{P}_1$ can be concatenated by $\mathcal{P}_2$, where the concatenated meta-path is denoted as $(\mathcal{P}_1\mathcal{P}_2) = (T_{1,0} \ldots T_{1,l} T_{2,1} \ldots T_{2,l'})$.*

For example, if we have two meta-paths $\mathcal{P}_1 = (APV)$ and $\mathcal{P}_2 = (VPT)$, $\mathcal{P}_1$ can be concatenated by $\mathcal{P}_2$, and the concatenated meta-path $(\mathcal{P}_1\mathcal{P}_2) = (APVPT)$.

Meta-paths provide the schema to instantiate actual paths in a heterogeneous network.

Definition 5 (Meta-path instantiation). *We say an instantiation of $\mathcal{P}$ is a path in G, denoted as $p = (v_0 v_1 \ldots v_l)$, satisfying $\phi(v_x) = T_x, \forall x = 0, 1, \ldots, l$. A meta-path can be instantiated by different paths. We represent the set of all path instances of meta-path $\mathcal{P}$ between vertices v_i and v_j by $\pi_{\mathcal{P}}(v_i, v_j)$.*

As an example, for authors Ava and Liam in Figure 1(b), the number of instantiations of meta-path $\mathcal{P}_{ca} = (APA)$ connecting them represents the number of papers they have coauthored, denoted as $|\pi_{\mathcal{P}_{ca}}(\text{Ava, Liam})| = 1$. Similarly, for authors Liam and Zoe, the number of instantiations of meta-path $\mathcal{P}_{ca}$ connecting them is $|\pi_{\mathcal{P}_{ca}}(\text{Liam, Zoe})| = 2$.

Based on the definition of a meta-path, we can define the "neighbors" of a vertex in a heterogeneous network. Different from traditional homogeneous networks, immediate neighbors of a certain vertex could be of different types and therefore have different semantics. Also, vertices that are multiple hops away from the given vertex can be meaningful "neighbors". For the sake of generality, we define the neighborhood of a vertex with respect to a given meta-path. Formally,

Definition 6 (Neighborhood). *In a heterogeneous network G, we define the neighborhood of a certain vertex v_i with regard to a given meta-path $\mathcal{P}$ as $N_{\mathcal{P}}(v_i) = \{v_j | \pi_{\mathcal{P}}(v_i, v_j) \neq \emptyset\}$.*

For example, the set of coauthors of author Zoe in Figure 1(b) can be represented by $N_{\mathcal{P}_{ca}}(\text{Zoe}) = \{\text{Ava, Liam}\}$.

Every vertex v_j, in the neighborhood of vertex v_i, is connected to v_i by at least one instantiation of the specified meta-path. However, multiple instantiations may exist. To better characterize the neighborhood of a vertex, we further define a vector representation of the neighborhood as a "neighbor vector".

Definition 7 (Neighbor vector). *We define a function $\sigma_{\mathcal{P}} : V \mapsto \mathbb{N}^{|V|}$ as the neighbor vector function. With regard to a given meta-path $\mathcal{P}$, it returns the neighbor vector given a certain vertex as input, where the j-th dimension is the count of paths instantiated by $\mathcal{P}$ between v_i and v_j. More precisely,*

$$\sigma_{\mathcal{P}}(v_i) = \Big[|\pi_{\mathcal{P}}(v_i, v_1)|, \ldots, |\pi_{\mathcal{P}}(v_i, v_n)| \Big]$$

For example, given meta-path $\mathcal{P}_{ca}$, Zoe's neighbor vector contains the count of papers co-authored with each of her coauthors, $\sigma_{\mathcal{P}_{ca}}(\text{Zoe}) = [\text{Ava} : 1, \text{Liam} : 2, \text{Zoe} : 5]$. Alternatively, Zoe's neighbor vector given meta-path $\mathcal{P}_v = (APV)$ is the count of papers that she has published in each venue, $\sigma_{\mathcal{P}_v}(\text{Zoe}) = [\text{ICDE} : 2, \text{KDD} : 3]$.

4. OUTLIER QUERIES

In this section, we formalize the definition of a query in the context of outlier detection in heterogeneous information networks. We also design a query language for users to specify queries.

4.1 General Formalization

Generally, a declarative query for outliers consists of two parts, a candidate set containing all the candidates that are potentially meaningful outliers, and a reference set providing a reference for outliers to be compared. In most outlier detection frameworks, the candidate set and the reference set are both assumed to be the entire data set. In our query-based outlier detection framework, users are provided with the flexibility to specify the candidate and reference sets of their interest, which enables our framework to be applicable to a broader range of scenarios.

Another important part of a user query is how the vertices should be compared. In a heterogeneous network, vertices can be compared in many different ways. For example, a pair of authors in a bibliographic information network can be compared based on how much their coauthors overlap, or how many common conferences they attend. Users should be given the flexibility to determine how they would like to compare two vertices.

There are two alternative ways to formulate the comparison method in a query. One is to directly ask the user to define a comparison function $\kappa : S_c \times S_r \mapsto \mathbb{R}$ to compare vertices in the candidate and reference sets; another is to ask the user to declare how a vertex should be characterized, and leave the implementation of the comparison method to the system. In most cases users are clear about the semantics of the desired outliers (*e.g.* comparing two authors based on their coauthors), but do not necessarily understand how to formulate a comparison function accordingly (*e.g.* comparing two authors by the number of their common coauthors), so we adopt the second query formulation where users specify how to characterize vertices using a meta-path based language.

Based on the principles above, we assemble a query for outlier detection with the following modules: the candidate set, the reference set, feature meta-path(s) to specify how a vertex is characterized in the context of outlier detection, and an optional vector used to weight feature meta-paths. To be precise,

Definition 8 (General outlier query). *An outlier query in a heterogeneous network G is denoted by $Q = (S_c, S_r, \mathbf{P}, \mathbf{w})$, where $S_c \subset V$ is the candidate set of vertices, from which the outliers will be chosen; $S_r \subset V$ is the reference set of vertices, serving as the reference of normal vertices; $\mathbf{P} = (\mathcal{P}_1, \ldots, \mathcal{P}_m)$ is a collection of feature meta-paths, describing the user's intuition of which aspects should characterize candidate vertices; $\mathbf{w} \in \mathbb{R}^m$ is a weighting vector for feature meta-paths, and by default is an all-one vector if not specified by users. The outlier detection algorithm should return outliers as a subset of the candidate set,* i.e. *$\Omega \subset S_c$, that are significantly different from vertices in S_r, in terms of the given meta-paths and weighting vector.*

As an example, if we want to find outliers among Christos Faloutsos' coauthors, then S_c should be defined as all of Christos' coauthors. In the most intuitive scenario the reference set S_r will be the same as S_c. A more complicated query could consist of finding outliers among Christos Faloutsos' coauthors who are unusual with respect to all computer science authors. In this case S_c should still be all of Christos' coauthors, but S_r should be all authors in computer science.

We also need to explicitly state how we are going to define outliers. For example, if we want to find outliers among Christos' coauthors who publish papers in substantially different venues, then it would be appropriate to define a single feature meta-path (APV) to extract all the publishing venues of each author.

Notice that although not explicitly pointed out in the definition, in this paper, we are assuming that all the vertices in $S_c \bigcup S_r$ are of the same type, which is a more intuitive scenario. Also, we require all the meta-paths $\mathcal{P}_1, \ldots, \mathcal{P}_m$ has their first element in the same vertex type as vertices in S_c and S_r, otherwise we cannot extract meaningful features from the given feature meta-paths.

To bring this general framework to real-world use cases in heterogeneous information networks, we need a powerful query language for users to specify the query.

4.2 Outlier Query Language

In this subsection, we present an outlier query language. It is capable of effectively supporting most outlier queries in a heterogeneous information network. Outlier detection is not part of the basic functionality supported by traditional SQL languages and relational databases. Due to the complexity of heterogeneous information networks, writing in SQL to specify an outlier detection query can be extremely awkward. Therefore, we define a query language for our outlier detection problem. Notice that although our query language is designed specifically for outlier detection queries in heterogeneous information networks, with minor modification it can also be applied to other types of data sets such as relational databases.

General Formulation. The general structure of a statement for an outlier query is:

```
FIND OUTLIERS FROM ... //Candidate set
COMPARED TO ... //Reference set
JUDGED BY ...  //Feature meta-paths
TOP ...; //Number of outliers to return
```

In the `FROM` or `COMPARED TO` clauses, users can specify a set of vertices. For the `FROM` clause, users specify the candidate set S_c, namely the set of vertices from which outliers are selected. The `COMPARED TO` clause is used to specify the reference set S_r, namely the set of vertices used as a reference. Notice that the `COMPARED TO` clause is optional. If it is not specified, the reference set S_r will be the same as the candidate set S_c.

In the `JUDGED BY` clause, users are required to give a single feature meta-path $\mathcal{P}$ or a collection of feature meta-paths $\mathbf{P}$. The weights of different feature meta-paths may optionally be provided. Vertices in S_r and S_c are compared based on the feature meta-paths and their weights. The top-k outliers, where k is specified in the `TOP` clause, are returned as results.

In the next part of this subsection, we introduce how to actually specify a set of vertices, and how to specify a collections of (weighted) feature meta-paths. Then we give several examples.

Specifying candidate/reference set. In the simplest case, users can refer to a certain vertex by its type and name:

```
venue{"EDBT"}
```

which returns all the venue-typed vertices with exactly the name "EDBT".

In many cases, users are interested in outliers in a certain local area in the network. Therefore, we allow the user to specify the neighborhood of a certain vertex, with regard to the definition in Section 3. Recall that to define a neighborhood requires a specific vertex v_i and meta-path $\mathcal{P}$. We use the dot operator to concatenate different types and represent a meta-path, where the first element is a specified vertex. As an example:

```
venue{"EDBT"}.paper.author
```

returns the neighborhood of venue-typed vertex EDBT with respect to meta-path (VPA). More formally it returns $N_{\mathcal{P}}(v_i)$, where v_i is the vertex that represents the venue EDBT and $\mathcal{P} = (VPA)$. Semantically, it is the set of all the authors with papers published in the venue EDBT.

We also allow users to specify additional conditions in a `WHERE` clause, to further restrict the vertices selected in the candidate or reference set. For example, the set of authors who have published in the conference EDBT and who have published more than 10 papers can be specified as:

```
venue{"EDBT"}.paper.author AS A
WHERE COUNT(A.paper) > 10
```

Multiple SQL-style operations can be applied to extract each vertex set. For instance, a user can generate the union of multiple sets using the `UNION` operator:

```
venue{"EDBT"}.paper.author
UNION
venue{"ICDE"}.paper.author
```

which will return the set of authors who have published in EDBT or ICDE.

Alternatively, a user can generate the intersection of several sets using the `INTERSECT` operator:

```
venue{"EDBT"}.paper.author
INTERSECT
venue{"ICDE"}.paper.author
```

which will return the set of authors who have published in both EDBT and ICDE.

Specifying feature meta-paths. A meta-path in our query language can simply be represented as an ordered list of types separated by dots. For example, in a bibliographic network, we can specify the feature meta-path (APV) to compare authors with respect to their publishing venues as:

```
author.paper.venue
```

in the `JUDGED BY` clause.

If there are multiple aspects, namely multiple feature meta-paths, that a user would like to use when classifying outliers, we separate different meta-paths by commas "`,`". For example, a user may judge outlier authors based on both their publishing venues and their coauthors. Meta-paths (APV) and (APA) are given as feature meta-paths and we write in the `JUDGED BY` clause:

```
author.paper.venue, author.paper.author
```

We have shown that our query language can support the specification of a collection of feature meta-paths $\mathbf{P}$. When users want to define different weights for different meta-paths, we also allow users to specify the weights in this query language, by writing the weight after a colon following the corresponding meta-path. As an example, suppose the user wants to judge outliers based on both their publishing venues and their coauthors, weighting the venues with twice the importance of coauthors. We can write in the `JUDGED BY` clause

```
author.paper.venue: 2.0, author.paper.author
```

where as `author.paper.author` is explicitly assigned a weight, it is by default weighted as 1.

Notice, we require that all specified feature meta-paths have the same type in their first element as vertices in S_c and S_r.

4.3 Example queries

Example 1. To find the top-10 outliers among Christos' coauthors in terms of venues they publish (*i.e.*, find 10 authors in Christos' coauthor who publish in the weirdest venues), we write the query:

```
FIND OUTLIERS
FROM author{"Christos Faloutsos"}.paper.author
JUDGED BY author.paper.venue
TOP 10;
```

Since no reference set is specified, the outliers are determined by comparing with others in the candidate set A.

Example 2. Alternatively, a user might want to find outliers in Christos' coauthors who are significantly different from the authors in the KDD community, in terms of the venues they publish in and their coauthors. We can write this query as:

```
FIND OUTLIERS
FROM
  author{"Christos Faloutsos"}.paper.author
COMPARED TO
  venue{"KDD"}.paper.author
JUDGED BY
  author.paper.venue,
  author.paper.author
TOP 10;
```

Example 3. To find the top-50 outliers among SIGMOD authors, who have published at least 5 papers, with respect to their coauthors (weight 1) and the vocabulary used in their paper titles (weight 3), we write the query:

```
FIND OUTLIERS
FROM venue{"SIGMOD"}.paper.author AS A
     WHERE COUNT(A.paper) >= 5
JUDGED BY
     author.paper.author,
     author.paper.term : 3.0
TOP 50;
```

5. NETWORK-BASED OUTLIER MEASURE: NETOUT

There have been many outlierness measures for numerical and categorical data. However, defining a good outlierness measure for use in heterogeneous information networks is still a challenging problem. The major challenge is the ambiguity of outlier semantics, as there are multiple types of paths connecting vertices.

Basic principle. In this section we define the properties of an outlier in a heterogeneous information network given a specific query. We address the problem for the query of finding outlier vertices among a set of candidate vertices with respect to a set of reference vertices, when judged by a specific feature meta-path. The feature meta-path and sets of candidate and reference vertices are given in the query formulation. The definition should be intuitive while utilizing the rich information provided by the network.

In general, an outlier among a group is an object that differs substantially from the rest of the group. In the context of finding outliers in a network, we look for vertices that are least connected to the group, but it is also important to consider each vertex's maximum potential for connectivity when comparing its group connectivity with that of other vertices. In the context of our specific problem, we follow the basic principle that outlying vertices should be most structurally disconnected from the reference set, with respect to their expected potential for connectivity.

5.1 Normalized Connectivity

We begin by presenting a measure to express the connectivity between two individual vertices with respect to their potential connectivity. Later we will apply it between individual candidate vertices and all vertices in the reference set to determine an outlier score for each candidate vertex.

In our query language we allow the user to specify a collection of feature meta-paths $\mathbf{P}$. In this section we only consider queries where $\mathbf{P}$ consists of a single feature meta-path $\mathcal{P}$. Finding outliers given a collection of weighted feature meta-paths can be done in a number of ways. The connectivity between vertices can be redefined, or independent outlier scores can be computed considering each feature meta-path independently and then averaged. We leave the problem of determining the best method to a future study.

The meta-path $\mathcal{P}$ can be viewed by the user as specifying a traditional feature type which will be used to judge the outlierness of each candidate vertex. We are interested in finding outliers that are most structurally disconnected, so we construct the symmetric meta-path linking the candidate type to itself, $\mathcal{P}_{sym} = (\mathcal{P}\mathcal{P}^{-1})$.

We can now define the connectivity κ between two vertices, v_a and v_b, as the number of path instantiations of $\mathcal{P}_{sym}$ between the two vertices, $\kappa(v_a, v_b) = |\pi_{\mathcal{P}_{sym}}(v_a, v_b)|$. The visibility of vertex v_a is the connectivity between v_a and itself, $\kappa(v_a, v_a)$, which is a measure of a vertex's potential connectivity with other vertices. We define the normalized connectivity between vertices v_a and v_b as the ratio of their connectivity to v_a's visibility:

Definition 9 (Normalized Connectivity). *Given heterogeneous network G containing two vertices v_a and v_b of type $T \in \mathcal{T}$ and symmetric meta-path $\mathcal{P}_{sym} = (\mathcal{P}\mathcal{P}^{-1}) = (T \dots T)$, the normalized connectivity between v_a and v_b is defined as* $\tilde{\kappa}(v_a, v_b) = \frac{\kappa(v_a,v_b)}{\kappa(v_a,v_a)} = \frac{|\pi_{\mathcal{P}_{sym}}(v_a,v_b)|}{|\pi_{\mathcal{P}_{sym}}(v_a,v_a)|}$

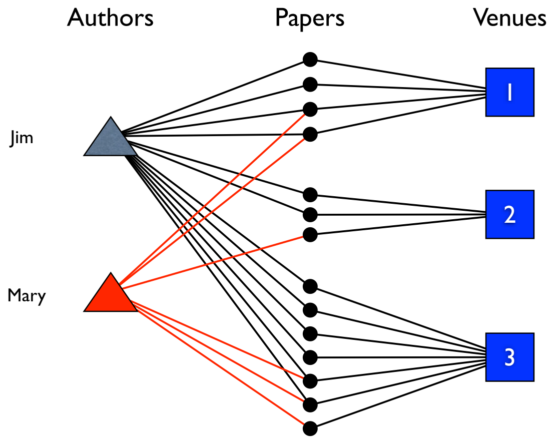


Figure 2: Path instantiations of the meta-path $(APVPA)$ connecting authors Jim and Mary

Note that $\tilde{\kappa}(v_a, v_b) \neq \tilde{\kappa}(v_b, v_a)$ when $\kappa(v_a, v_a) \neq \kappa(v_b, v_b)$. Normalized connectivity can be interpreted in terms of a random walk beginning at v_a along meta-path $\mathcal{P}_{sym}$. The probability of ending at v_b is $\frac{|\pi_{\mathcal{P}_{sym}}(v_a,v_b)|}{||\sigma_{\mathcal{P}_{sym}}(v_i)||_1}$, where σ is the neigbor vector function defined in Section 3. The probability of returning to v_a is $\frac{|\pi_{\mathcal{P}_{sym}}(v_a,v_a)|}{||\sigma_{\mathcal{P}_{sym}}(v_i)||_1}$. The probability of ending at v_b divided by the probability of returning to v_a is then $\frac{|\pi_{\mathcal{P}}(v_a,v_b)|}{|\pi_{\mathcal{P}}(v_a,v_a)|}$, which is the normalized connectivity $\tilde{\kappa}(v_a, v_b)$. This fits with our intuition well. The probability of returning to v_a acts as a normalization constant such that the normalized connectivity between v_a and itself will always be 1. When v_a is more connected to v_b than itself, $\tilde{\kappa}(v_a, v_b) > 1$. When v_a is less connected to v_b than itself, $\tilde{\kappa}(v_a, v_b) < 1$. Comparing $\tilde{\kappa}(v_a, v_c)$ with $\tilde{\kappa}(v_b, v_c)$ shows whether it is more likely to arrive at v_c in a random walk beginning at v_a or v_b (normalized by the likelihood of returning to the original vertex).

Example 4. We use a concrete example (Cf. Figure 2) to illustrate the behavior of normalized connectivity. We examine two authors Jim and Mary in a bibliographic network G given feature meta-path $\mathcal{P} = (APV)$.

The connectivity (path count) between Jim and Mary is $2\times4+1\times2+3\times6 = 28$. The normalized connectivities in this example are $\tilde{\kappa}(Jim, Mary) = 0.5$ and $\tilde{\kappa}(Mary, Jim) = 2$. This reflects that Jim's connectivity with Mary is half his connectivity with himself, while Mary's connectivity with Jim is twice that with herself.

5.2 Outlier Measure: NetOut

To measure a certain vertex v_i's outlierness with regard to a given reference set S_r we sum the normalized connectivity between v_i and all vertices in S_r. This gives v_i's connectivity to the reference set as a whole, normalized by its potential connectivity. The lower this normalized group connectivity, the more likely that v_i is an outlier. We define the outlierness measure NetOut as:

Definition 10 (Outlierness in Heterogeneous Networks: NetOut). *In a heterogeneous network G, given a query Q, for any $v_i \in S_c$, the outlierness can be measured by*

$$\Omega_{NetOut}(v_i; Q) = \sum_{v_j \in S_r} \tilde{\kappa}(v_i, v_j)$$

where smaller Ω values correspond to greater likelihood of being an outlier. We refer to $\Omega_{NetOut}(v_i; Q)$ as simply $\Omega(v_i; Q)$ outside this section when there is no potential ambiguity.

Rather than summing v_i's normalized connectivity with every vertex in the reference set, we could find the minimum or maximum normalized connectivity between v_i and any vertex in the reference set. In many cases finding the minimum normalized connectivity is not meaningful because many vertices in the candidate set are completely disconnected from at least one vertex in the reference set. To evaluate the usefulness of finding the maximum normalized connectivity consider two vertices v_i and v_j. Vertex v_i is moderately connected to one vertex in the reference set but completely disconnected from every other context vertex. Vertex v_j is weakly connected to every vertex in the reference set. In most cases it is hard to justify that v_j should be a stronger outlier than v_i.

Summing the normalized connectivities has the additional advantage of computational efficiency. Computing NetOut for every vertex in the candidate set can be reduced to an $\mathcal{O}(|S_r| + |S_c|)$ operation. In comparison, using the minimum or maximum normalized connectivity instead would always require $\mathcal{O}(|S_r| \times |S_c|)$ time.

Next we justify our use of normalized connectivity when defining NetOut by comparing with the similarity measures PathSim and cosine similarity.

PathSim. Superficially it may appear that a similarity measure could be used instead of normalized connectivity in our outlier detection problem. The normalized connectivity between two vertices is not a true similarity measure because it lacks symmetry. In this section we introduce the similarity measure PathSim for comparison, to justify the need for normalized connectivity.

In a previous study of similarity search in heterogenous information networks [24], PathSim was introduced as an interesting measure to define network-based structural similarity. The PathSim measure between two vertices v_i and v_j following a meta-path $\mathcal{P}$ in a heterogeneous information network is defined as,

$$PathSim_{\mathcal{P}_{sym}}(v_i, v_j) = \frac{|\pi_{\mathcal{P}_{sym}}(v_i, v_j)|}{\left(|\pi_{\mathcal{P}_{sym}}(v_i, v_i)| + |\pi_{\mathcal{P}_{sym}}(v_j, v_j)|\right)/2}$$

For comparison purposes we define:

$$\Omega_{PathSim}(v_i; Q) \quad = \quad \sum_{v_j \in S_r} PathSim_{\mathcal{P}_{sym}}(v_i, v_j)$$

$PathSim_{\mathcal{P}_{sym}}(v_i, v_j)$ is defined by the connectivity between v_i and v_j divided by the average of v_i and v_j's visibility. Based on this formula, PathSim assigns high similarity values to the vertices that are strongly connected (*i.e.*, there are many paths between v_i and v_j following the meta-path) but having low average visibility (*i.e.*, there are not many other paths from v_i or v_j reaching v_i or v_j itself).

PathSim has demonstrated its promise at similarity search in heterogeneous information networks. Comparing to SimRank or Personalized PageRank, PathSim assigns lower similarity to vertices whose connectivity is high but whose visibilities differ.

Cosine Similarity. We define a comparable version of NetOut using the cosine similarity instead of normalized connectivity:

$$\Omega_{CosSim}(v_i; Q) \quad = \quad \sum_{v_j \in S_r} \frac{\sigma_{\mathcal{P}}(v_i) \cdot \sigma_{\mathcal{P}}(v_j)}{||\sigma_{\mathcal{P}}(v_i)||_2 \times ||\sigma_{\mathcal{P}}(v_j)||_2}$$

Where $\sigma_{\mathcal{P}}(v_i)$ is the neighbor vector function defined in Section 3.

NetOut Example. We consider a toy example to demonstrate NetOut's properties. Table 1 shows the publication records of candidate authors. In this example we consider a query giving the reference set composed of 100 authors with publication records identical to Sarah's and feature meta-path $\mathcal{P} = (APV)$. Table 2 shows NetOut scores for each candidate author. We compare with outlier scores computed using PathSim and the cosine similarity in place of normalized connectivity in the NetOut formula.

Sarah is clearly not an outlier, with $\Omega(Sarah; Q) = 100$ (normalized connectivity with identical vertices of 1 multiplied by the size of the reference set). Rob has an unusual publication record and a low NetOut score, signifying he is a potential outlier. Lucy's publication differs from authors in the reference set, but is more similar than Robs, giving her a higher NetOut score than Rob.

Next we compare NetOut scores with outlier scores computed using PathSim and the cosine similarity in place of normalized connectivity. PathSim is computed using the meta-path $(APVPA)$. The cosine similarity is computed using each author's neighbor vector (defined in Section 3). All three measures find the same outlier ordering for Sarah, Rob, and Lucy.

Joe has published only two papers in the venue SIGGRAPH. While SIGGRAPH is an unusual venue, Joe's publication record is currently unstable and likely to change over the course of his life. It is possible that his first publications are simply noise. NetOut does not classify Joe as an outlier. While his connectivity with authors in the reference set is low, this is expected because of his low visibility. In a random walk beginning at Joe following the meta-path $(APVPA)$, the probability of reaching an author in the reference set is the same as the probability of returning to Joe. However, the PathSim and cosine similarity versions both classify Joe as an outlier with very low scores.

Emma is clearly a very unusual author, and this is apparent in her NetOut score. She has only published in the unusual venue SIGGRAPH and she has published more papers than the authors in the reference set, so we can assume her publication record is stable at this point. Her outlier score computed using PathSim is actually higher than Joe's, because her visibility is more similar to the visibility of authors in the reference set. Emma's outlier score computed using the cosine similarity is the same as Joe's. Both have neighbor vectors with the same direction, so their cosine similarity with other authors is identical. NetOut computed using nor-

Table 1: Publication records of candidate and reference vertices. The reference set contains 100 authors with identical publication records, given by the reference author.

	VLDB	KDD	STOC	SIGGRAPH
Reference Author	10	10	1	1
Sarah	10	10	1	1
Rob	0	1	20	20
Lucy	0	5	10	10
Joe	0	0	0	2
Emma	0	0	0	30

Table 2: NetOut outlier scores of select candidate vertices given a query whose feature meta-path is $\mathcal{P} = (APV)$ and reference set is given in Table 1, compared with scores computed using PathSim and the cosine similarity in place of normalized connectivity.

	Ω_{NetOut}	$\Omega_{PathSim}$	Ω_{CosSim}
Sarah	100	100	100
Rob	6.24	9.97	12.43
Lucy	31.11	32.79	32.83
Joe	50	1.94	7.04
Emma	3.33	5.44	7.04

malized connectivity finds outliers without bias towards any particular visibility, while PathSim and the cosine similarity are biased towards authors with low visibility.

NetOut Experimental Comparison. To further explain our use of normalized connectivity, rather than a symmetric similarity measure such as PathSim or CosSim, we employ a concrete example on DBLP data set to compare the results returned by different methods.

We construct a query, to find top-5 outliers among all the coauthors of Christos Faloutsos, in terms of their publishing venues. The context and candidate sets are specified as Faloutsos' co-authors and the feature meta-path is given by $\mathcal{P} = (APV)$.

The comparison results are shown in Table 3. The top outliers found by NetOut defined using normalized connectivity are active in fields besides data mining, which is Christos' primary focus, and have a wide range of visibilities. Adam Wright has published roughly 30 papers, while Katia P. Sycara has published roughly 300 papers. In contrast, all the top-5 outliers found by PathSim or CosSim are authors who have published less than 2 papers, which makes them uninteresting as outliers. This further demonstrates the inherent bias towards candidate vertices with low visibility when using PathSim or the cosine similarity.

6. IMPLEMENTATION

In this section we briefly introduce some technical details regarding the implementation of our query-based outlier detection system. We first introduce a basic baseline implementation and then optimizations to improve the efficiency of query execution.

6.1 Baseline

There are two basic steps to execute an outlier query: retrieve the candidate and reference sets S_c and S_r and calculate the outlierness of each vertex in the reference set based on the given feature meta-paths.

For retrieval of S_c and S_r, the basic operations are finding a vertex v_i given its name and type and then traversing the network from v_i while counting the instantiations of the given meta-path. The first operation can be naively implemented by a hash table, or a trie, which is relatively efficient. The second basic operation, materializing a meta-path $\mathcal{P}$, has time complexity exponential to the length of $\mathcal{P}$.

A naïve way to calculate the outlierness measure would be to first calculate the normalized connectivity $\tilde{\kappa}(\cdot,\cdot)$ between each vertex in the candidate set S_c and each vertex in the references set S_r, then sum up all the $\tilde{\kappa}(v_i,\cdot)$ for each vertex in S_c to obtain the outlierness. However, this has time complexity of $O(|S_r| \times |S_c|)$.

Recall the definition of connectivity, $\kappa(\cdot,\cdot)$:

$$\begin{aligned}\kappa(v_i, v_j) &= |\pi_{\mathcal{P}_{sym}}(v_i, v_j)| \\ &= \sigma_{\mathcal{P}}(v_i) \cdot \sigma_{\mathcal{P}}(v_j)\end{aligned}$$

The calculation of NetOut can be re-written as:

$$\begin{aligned}\Omega(v_i; Q) &= \sum_{v_j \in S_r} \tilde{\kappa}(v_i, v_j) \\ &= \sum_{v_j \in S_r} \frac{\kappa(v_i, v_j)}{\kappa(v_i, v_i)} \\ &= \sum_{v_j \in S_r} \frac{\sigma_{\mathcal{P}}(v_i) \cdot \sigma_{\mathcal{P}}(v_j)}{\sigma_{\mathcal{P}}(v_i) \cdot \sigma_{\mathcal{P}}(v_i)} \\ &= \frac{1}{||\sigma_{\mathcal{P}}(v_i)||_2^2} \left(\sigma_{\mathcal{P}}(v_i) \cdot \left(\sum_{v_j \in S_r} \sigma_{\mathcal{P}}(v_j) \right) \right) \quad (1)\end{aligned}$$

Notice that the term $\sum_{v_j \in S_r} \sigma_{\mathcal{P}}(v_j)$ remains the same for all $v_i \in S_c$. Therefore we can first calculate it, then calculate the outlierness value NetOut for all vertices $v_i \in S_c$. Therefore, the time complexity of calculating NetOut for every candidate vertex is only $O(|S_r| + |S_c|)$.

However, even if the calculation of NetOut is efficient, it is still relatively slow compared to actually obtaining the neighbor vector $\sigma_{\mathcal{P}}(v_i)$ for a given v_i and meta-path $\mathcal{P}$. Materializing this neighbor vector requires traversal of the heterogeneous network, which can be time-consuming when the specified meta-path is long or the degree of the vertex of interest is high. Therefore we aim to optimize the query processing time by reducing the materialization time of meta-paths.

6.2 Optimization

Pre-materialization. To accelerate the materialization of meta-paths, we can pre-compute the materialization of length-2 meta-paths. Depending on the pattern of user queries we may compute all length-2 paths or only a subset. To be more precise, for each vertex $v_i \in V$, and all possible $\mathcal{P}$ such that $|\mathcal{P}| = 2$, we can calculate and store the vector $\sigma_P(v_i)$.

In the execution of a query, it may be necessary to calculate $\sigma_P(v_i)$ for an arbitrary meta-path $\mathcal{P}$. We can always decompose $\mathcal{P}$ into several length-2 meta-paths as

Table 3: Comparing different outlierness measure, with query $S_c = S_r$ =author("Christos Faloutsos").paper.author and feature meta-path $\mathcal{P} = (APV)$ Outliers found by normalized connectivity are interesting outliers, while outliers found by PathSim or CosSim are authors with very few papers, which are not interesting.

Method	Ω_{NetOut}		$\Omega_{PathSim}$		Ω_{CosSim}	
Ranking	Name	Ω-value	Name	Ω-value	Name	Ω-value
1	Adam Wright	2.54	Wenyao Ho	1.07	John Chien-Han Tseng	0.0022
2	Philip Koopman	2.55	Fernanda Balem	1.12	Fernanda Balem	0.0038
3	Nicholas D. Sidiropoulos	3.29	Rebecca B. Buchheit	1.31	Guoqiang Shan	0.0046
4	Katia P. Sycara	3.64	John Chien-Han Tseng	1.41	Wenyao Ho	0.0066
5	David S. Doermann	3.65	Chi-Dong Chen	1.47	Chi-Dong Chen	0.0077

$\mathcal{P} = (\mathcal{P}_1 \cdots \mathcal{P}_k)$, where $|\mathcal{P}_1| = \cdots = |\mathcal{P}_{k-1}| = 2$. If the original meta-path $\mathcal{P}$ is even-length, then $|\mathcal{P}_k| = 2$.

Notice that for any $\mathcal{P} = (\mathcal{P}_1\mathcal{P}_2)$, we have

$$\begin{aligned}\sigma_{\mathcal{P}}(v_i) &= \sum_{v_j} |\pi_{\mathcal{P}_1}(v_i, v_j)| \sigma_{\mathcal{P}_2}(v_j) \\ &= [\sigma_{\mathcal{P}_2}(v_1), \ldots, \sigma_{\mathcal{P}_2}(v_n)]\, \sigma_{\mathcal{P}_1}(v_i)\end{aligned}$$

which implies that by decomposing an arbitrary meta-path $\mathcal{P}$ into several length-2 meta-paths, we can calculate $\sigma_{\mathcal{P}}(v_i)$ by multiplication of indexed vectors. Even if the original meta-path is odd-length, we only need to traverse the network for a single hop. Retrieving an index can be of $O(1)$ by storing the vectors in a hash table and the time complexity of multiplication is affordable when the vectors are sparse.

By efficiently retrieving $\sigma_{\mathcal{P}}(v_i)$, multiple steps in the query processing benefit, including the retrieval of candidate set S_c and reference set S_r, and the calculation of connectivity functions.

Selective pre-materialization. The aforementioned indexing strategy pre-calculates the indexed vectors for all vertices with regard to all length-2 meta-paths. This exhaustive indexing strategy guarantees efficiency improvement, but can also result in a large index table. To achieve reasonable efficiency while conserving memory, we may only want to construct length-2 meta-paths starting from a certain set of vertices.

To this end, a strategy is to count the frequency with which different vertices appear in queries. The query set used for selecting vertices for building indices is referred to as "initialization query set" for SPM. The initialization query set can be existing query logs, or else synthetic queries when query logs are not available. A certain absolute or relative threshold is set, and length-2 meta-paths are only computed beginning at vertices that appear in queries with frequency above the set threshold.

7. EXPERIMENTAL RESULTS

In this section we evaluate experimental performance.

7.1 Experiment Setup

Data set. We employ a bibliographic data set from ArnetMiner[3] to construct a heterogeneous information network. The data set consists of 2,244,018 publications and 1,274,360 authors in the field of computer science. The heterogeneous network contains 4 types of vertices: paper, venue, author and term. Possible type of edges include paper-author (written-by), paper-venue (published in) and paper-term (title contains).

Table 4: Query templates used to construct query sets for efficiency experiments. 10,000 random authors are selected and substituted where indicated by "·" in each query template.

Number	Query Templates

$\mathcal{Q}_1$	`FIND OUTLIERS FROM author{·}.paper.author`
	`JUDGED BY author.paper.venue`
	`TOP 10;`

$\mathcal{Q}_2$	`FIND OUTLIERS IN author{·}.paper.venue`
	`JUDGED BY venue.paper.term`
	`TOP 10;`

$\mathcal{Q}_3$	`FIND OUTLIERS IN author{·}.paper.term`
	`JUDGED BY term.paper.venue`
	`TOP 10;`

Query sets. In order to check the efficiency performance of our algorithm, we randomly select 10,000 author-typed vertices from the heterogeneous information networks. Three different types of queries are shown in Table 4, which are referred to as "query templates". For each template, we substitute the randomly selected vertices into the position indicated by the dot "·", to generate 10,000 queries. We refer to each set of queries as $\mathcal{Q}_i$. These randomly generated query sets are used in efficiency studies.

Comparison methods. In efficiency studies, we compare the following implementations.

- *Baseline.* The baseline implementation without pre-materialization (Cf. Equation (1)).
- *Pre-Materialization (PM).* All length-2 meta-path instantiations are pre-computed and stored.
- *Selective Pre-Materialization (SPM).* A subset of all length-2 meta-path instantiations are pre-computed and stored, for selected vertices that frequently appear in S_c given a set of specified queries, where the relative frequency threshold is set to 0.01.

We use the set of all possible queries for the given query template as the initialization query set in SPM.

[3] `http://arnetminer.org/AMinerNetwork`

7.2 Case Study

We examine the effectiveness of our proposed outlierness measure by checking the experimental results of several typical queries. The results are summarized in Table 5.

In our first two experiments we use Christos Faloutsos' coauthors as the candidate and reference sets. We use `author.paper.venue` in the first experiment as the single feature meta-path and `author.paper.author` in the second.

The first query we try is to find outliers with regard to their publishing venues. The returned top-10 outliers of Christos' coauthors are actually quite deviated from his research field (with one exception), which is data mining. For example, Adam Wright works on biomedical informatics; Philip Koopman is in the area of embedded systems. Interestingly, Nicholas D. Sidiropoulos publishes most of his work in the community of signal processing. However, one of his research interests is tensor analytics and mining, which is closely related to Christos Faloutsos' research interests. As we are judging outliers based on publishing communities, Nicholas D. Sidiropoulos is still listed as one of the top outliers. Although most of the aforementioned outliers are relatively established authors in their own fields, John Chien-Han Tseng is a student who has published only one paper in the venue KDIR (a very rare venue for authors in the reference set to publish in). Tseng's appearance demonstrates that our method does not discriminate against candidate outliers based on their visibility.

In the second query, we still search for outliers among Christos' coauthors, but judged by their coauthors. The results are substantially different from the first query, with only one overlapping author (Katia P. Sycara). This is evidence that in a heterogeneous information network outliers can be reasonably defined in multiple ways, resulting in totally different outcomes. Without user-specified queries, mining outliers can be an ill-defined problem leading to semantic ambiguity. The top outliers are still mainly in fields other than data mining, with an interesting exception: Ee-Peng Lim is a researcher who also focuses on social network analysis and mining [4], with a significant number of papers published in data mining venues. Lim is still listed as an outlier among Christos' coauthors, as his collaborator network does not overlap much with Christos' collaborators. This is a typical example of the importance of providing a specific outlier definition. Outliers under one definition could be totally normal given another definition.

In the third query, we attempt to find outliers among KDD authors, with respect to their publishing venues. The top outlier turns out to be "NULL" which represents missing data. Other top outliers are also interesting: Wolfgang Glänzel is a professor of economics and business, with the majority of his papers published in economic related venues; Paul M. Thompson has published most papers in medical or neuroscience venues.

7.3 Efficiency Studies

We also examine the efficiency performance of our different query optimization strategies. In this experiment, we process the query sets generated from the query template in Table 4 and measure the system performance by query processing time.

Table 5: Case study of NetOut results on several queries.

$S_c = S_r =$ author("Christos Faloutsos").paper.author
$\mathcal{P} =$ author.paper.venue

Ranking	Name	Ω-value
1	Adam Wright	2.54
2	Philip Koopman	2.55
3	Nicholas D. Sidiropoulos	3.29
4	Katia P. Sycara	3.64
5	David S. Doermann	3.65
6	Asim Smailagic	3.69
7	John Chien-Han Tseng	4.00
8	Daniel P. Siewiorek	4.22
9	Jessica K. Hodgins	4.52
10	Dimitris N. Metaxas	4.57

$S_c = S_r =$ author("Christos Faloutsos").paper.author
$\mathcal{P} =$ author.paper.author

Ranking	Name	Ω-value
1	Dimitris N. Metaxas	1.06
2	Bin Zhang	1.06
3	Hui Zhang	1.07
4	Lionel M. Ni	1.07
5	Bin Liu	1.08
6	Joel H. Saltz	1.08
7	Yang Wang	1.08
8	Hao Wang	1.08
9	Ee-Peng Lim	1.12
10	Katia P. Sycara	1.13

$S_c = S_r =$ venue("KDD").paper.author
$\mathcal{P} =$ author.paper.venue

1	*NULL*	1.27
2	Wolfgang Glänzel	4.99
3	Paul M. Thompson	6.46
4	Yehuda Lindell	9.21
5	Kwan-Liu Ma	12.2
6	Dhabaleswar K. Panda	13.23
7	Christos Davatzikos	13.95
8	Andrzej Skowron	14.62
9	Anil K. Jain	15.75
10	Fillia Makedon	15.95

Improved efficiency with pre-materialization. In Figure 3 we compare the performance of the baseline implementation, the implementation with all length-2 meta-paths pre-materialized (PM), and the selective pre-materialized version with relative frequency threshold 0.01 (SPM). With pre-materialization the efficiency can always be improved significantly, 5-100 times faster than the baseline implementation. This verifies the effectiveness of the indexing strategy. The performance of SPM is generally worse than the fully materialized version PM, but is more than 10 times faster than the baseline in query set $\mathcal{Q}_3$.

In-depth efficiency analysis of SPM. For the SPM strategy, we conduct a study to look into the processing time spent on different parts. As shown in Figure 4, For almost all query sets, most of the processing time is spent on materializing feature meta-paths of vertices without pre-materialization. Loading pre-stored instantiations of feature meta-paths for vertices with materialization is the least time

[4] `https://sites.google.com/site/aseplim/`

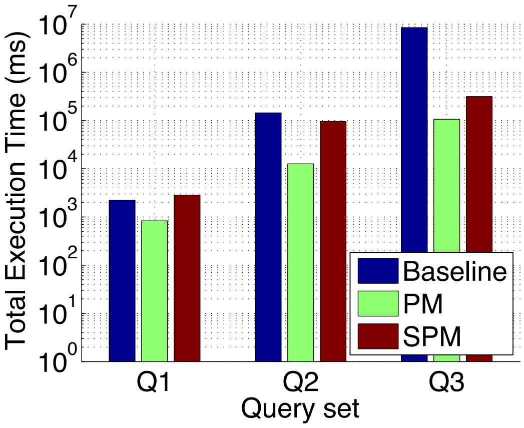


Figure 3: Comparing total execution time for 10,000 randomly generated queries between the baseline implementation and the implementation with pre-materialization.

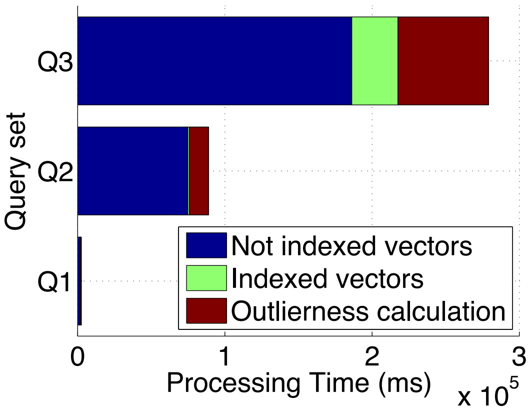
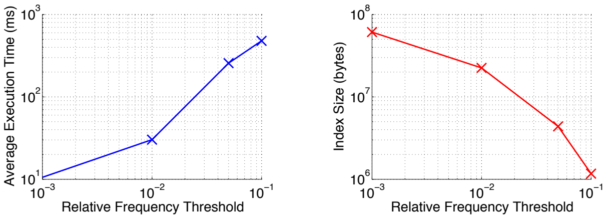


Figure 4: In-depth analysis of query processing time using selective pre-materialization strategies with the relative frequency threshold set to 0.01. "Not indexed vectors" indicates processing time spent on meta-path materialization from vertices without pre-materialization; "Indexed vectors" indicates time spent looking up pre-materialized meta-paths from materialized vertices; "Outlierness calculation" indicates calculation time of NetOut.

consuming part, while calculating NetOut can be slower. Calculating inner products between vectors is potentially more expensive than retrieving vectors from indices.

Threshold studies for SPM. We check the performance of SPM strategies with different relative frequency threshold. We construct indices with the relative frequency threshold set at 0.001, 0.01, 0.05, and 0.1 respectively, and compare both the processing time and index size, as shown in Figure 5. Not surprisingly, the index size decreases as the threshold rises, while the average query processing time also increases. A relatively optimal threshold is likely to be found between 0.01 and 0.05, considering both factors.



(a) Average execution time vs. relative frequency threshold

(b) Index size vs. relative frequency threshold

Figure 5: Comparison of efficiency performance with different relative frequency threshold in selective pre-materialization indexing strategy.

8. DISCUSSION

Alternative query language design. There are other ways to define the query language with more generality. It is possible to allow users to specify functions that are not meta-path based for measuring the similarity between two vertices, or to allow users to define their own outlierness measure, *etc.*. However, maximizing the generality will require users to have more expertise knowledge, which violates our principle to provide users with a more declarative language. In comparison, our language design is simple and satisfies most needs for data analysis.

Outlierness measure. The outlierness measure NetOut we defined in this paper is easy to compute compared to many state of the art outlier detection algorithms. It is still possible to substitute other outlier detection algorithms based on our query-based outlier detection framework, as long as they support the input specified by our queries. However, most of them are not efficient enough to be suited for users' exploratory query behavior. Our experiments comparing with other outlier detection algorithms (*e.g.* LOF [4]) suggest that they cannont produce better results than NetOut.

Extensions. Although we frame our query-based outlier detection study in a closed-schema heterogeneous information network data set, our framework can easily be extended to a broader range of data sets. For example, our query language can be applied to open-schema networks such as a knowledge graph, and the baseline implementation of NetOut should also be applicable. It is also possible to apply our query-based outlier detection idea on traditional relational databases, with a structure similar to our defined outlier query language, but changing the meta-path-based language into SQL. It would be interesting to develop a query-based outlier detection system for different types of data sets, based on our defined framework and query language, while exploring the implementation challenges.

There are additional directions to further facilitate users' exploratory interaction with the system. For example, instead of returning the top-k outliers after the user specifies the query, it might be helpful to visualize outliers to provide more insight. Alternatively, the system could find the approximate top-k outliers, with confidences, while the query is being processed so that users can determine whether to continue processing the query. The system might even be able to suggest how the users can modify their queries to get more interesting, or more unusual, outliers.

9. CONCLUSION

In this paper, we propose a query-based outlier detection framework. We design a query language for outlier detection in heterogeneous information networks, which gives users flexibility to mine various types of outliers based on their intuition. We also propose NetOut, a novel meta-path based outlierness measure for mining outliers in heterogeneous networks, and show its effectiveness compared to other outlierness measures. We finally present implementation details, where we utilize pre-materialization and selective pre-materialization to optimize query processing time. Experimental results show that our proposed query-based outlier detection framework can efficiently return meaningful results for a range of queries.

Acknowledgements. Research was sponsored in part by the Army Research Lab. under Cooperative Agreement No. W911NF-09-2-0053 (NSCTA) and W911NF-11-2-0086, the Army Research Office under Cooperative Agreement No. W911NF-13-1-0193, National Science Foundation IIS-1017362, IIS-1320617, and IIS-1354329, HDTRA1-10-1-0120, NIH Big Data to Knowledge (BD2K) (U54), and MIAS, a DHS-IDS Center for Multimodal Information Access and Synthesis at UIUC.

10. REFERENCES

[1] L. Akoglu, M. McGlohon, and C. Faloutsos. Oddball: Spotting anomalies in weighted graphs. In *PAKDD*, pages 410–421. Springer, 2010.

[2] R. Angles. A comparison of current graph database models. In *ICDE Workshops*, pages 171–177. IEEE, 2012.

[3] R. Angles and C. Gutierrez. Survey of graph database models. *ACM Computing Surveys (CSUR)*, 40(1):1, 2008.

[4] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: identifying density-based local outliers. In *SIGMOD*, pages 93–104. ACM, 2000.

[5] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):15:1–15:58, 2009.

[6] J. Gao, F. Liang, W. Fan, C. Wang, Y. Sun, and J. Han. On community outliers and their efficient detection in information networks. In *KDD*, pages 813–822. ACM, 2010.

[7] M. Gupta, J. Gao, and J. Han. Community distribution outlier detection in heterogeneous information networks. In *ECML/PKDD*, pages 557–573. Springer, 2013.

[8] M. Gupta, J. Gao, Y. Sun, and J. Han. Integrating community matching and outlier detection for mining evolutionary community outliers. In *KDD*, pages 859–867. ACM, 2012.

[9] M. Gupta, J. Gao, X. Yan, H. Cam, and J. Han. On detecting association-based clique outliers in heterogeneous information networks. In *ASONAM*, pages 108–115. IEEE, 2013.

[10] M. Gupta, J. Gao, X. Yan, H. Cam, and J. Han. Top-k interesting subgraph discovery in information networks. In *ICDE*, pages 820–831. IEEE, 2014.

[11] M. Gupta, A. Mallya, S. Roy, J. H. Cho, and J. Han. Local learning for mining outlier subgraphs from network datasets. In *SDM*, 2014.

[12] C. Gutierrez, C. Hurtado, and A. O. Mendelzon. Foundations of semantic web databases. In *PODS*, pages 95–106. ACM, 2004.

[13] H. He and A. K. Singh. Graphs-at-a-time: query language and access methods for graph databases. In *SIGMOD*, pages 405–418. ACM, 2008.

[14] V. J. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.

[15] W. Jin, A. K. Tung, and J. Han. Mining top-n local outliers in large databases. In *KDD*, pages 293–298. ACM, 2001.

[16] G. Kasneci, F. M. Suchanek, G. Ifrim, M. Ramanath, and G. Weikum. Naga: Searching and ranking knowledge. In *ICDE*, pages 953–962. IEEE, 2008.

[17] E. M. Knox and R. T. Ng. Algorithms for mining distancebased outliers in large datasets. In *VLDB*, pages 392–403, 1998.

[18] N. Li, H. Sun, K. Chipman, J. George, and X. Yan. A probabilistic approach to uncovering attributed graph anomalies. In *SDM*, 2014.

[19] B. Perozzi, L. Akoglu, P. Iglesias Sánchez, and E. Müller. Focused clustering and outlier detection in large attributed graphs. In *KDD*, pages 1346–1355. ACM, 2014.

[20] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *SIGMOD*, volume 29, pages 427–438. ACM, 2000.

[21] M. Schmidt, M. Meier, and G. Lausen. Foundations of sparql query optimization. In *ICDT*, pages 4–33. ACM, 2010.

[22] E. Schubert, A. Zimek, and H.-P. Kriegel. Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection. *Data Mining and Knowledge Discovery*, 28(1):190–237, 2014.

[23] J. Sun, H. Qu, D. Chakrabarti, and C. Faloutsos. Neighborhood formation and anomaly detection in bipartite graphs. In *ICDM*, pages 418–425, 2005.

[24] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment*, 4(11), 2011.

[25] Y. Sun, B. Norick, J. Han, X. Yan, P. S. Yu, and X. Yu. Integrating meta-path selection with user-guided object clustering in heterogeneous information networks. In *KDD*, pages 1348–1356. ACM, 2012.

[26] X. Yan, P. S. Yu, and J. Han. Graph indexing: a frequent structure-based approach. In *SIGMOD*, pages 335–346. ACM, 2004.

[27] X. Yan, P. S. Yu, and J. Han. Substructure similarity search in graph databases. In *SIGMOD*, 2005.

[28] S. Yang, Y. Wu, H. Sun, and X. Yan. Schemaless and structureless graph querying. *Proceedings of the VLDB Endowment*, 7(7), 2014.

[29] P. Zhao and J. Han. On graph query optimization in large networks. *Proceedings of the VLDB Endowment*, 3(1-2):340–351, 2010.

[30] B. Zong, Y. Wu, J. Song, A. K. Singh, H. Cam, J. Han, and X. Yan. Towards scalable critical alert mining. In *KDD*, pages 1057–1066. ACM, 2014.