Generating Preview Tables for Entity Graphs

Ning Yan, Abolfazl Asudeh, Chengkai Li
Department of Computer Science and Engineering
The University of Texas at Arlington
ning.yan@mavs.uta.edu, a.asudeh@gmail.com, cli@uta.edu

ABSTRACT

We witness an unprecedented proliferation of entity graphs that capture entities and their relationships. Users and developers are tapping into entity graphs for numerous applications. It can be a challenging task to select entity graphs for a particular need, given abundant datasets from many sources and the oftentimes scarce information available for the datasets. Given an entity graph with many types of entities and relationships, we propose methods to automatically produce a set of preview tables, for compact presentation of important entity types and relationships. The preview tables assist users in attaining a quick and rough preview of the data. They can be shown in a limited display space for a user to browse and explore, before she decides to spend time and resources to fetch the complete dataset and investigate it in more detail.

We propose scoring functions for measuring the goodness of previews. Based on the scoring measures, we formulate several optimization problems that look for previews with the highest scores, under various constraints on preview size and distance between preview tables. We prove that the optimization problem under distance constraint is **NP**-hard. We design a dynamic-programming algorithm and an Apriori-style algorithm for finding optimal previews. We conducted experiments and user studies using Freebase data. The results demonstrated both the preview scoring measures' accuracy and the preview discovery algorithms' efficiency.

1. INTRODUCTION

We witness in many domains an unprecedented proliferation of *entity graphs* that capture entities (e.g., persons, products, organizations) and their relationships. Figure 1 is a tiny excerpt of an entity graph, in which the edge labeled *Actor* between nodes Will Smith and Men in Black captures the fact that the person is an actor in the film. Real-world entity graphs include knowledge bases (e.g., DBpedia [2], YAGO [13], Probase [15] and Freebase [3], which powers Google's knowledge graph), [1] social graphs, drug and disease databases, gene and protein databases, and program analysis graphs, to name just a few. Users and developers are tapping into entity graphs for numerous applications, including search, recommendation systems, business intelligence and health informatics.

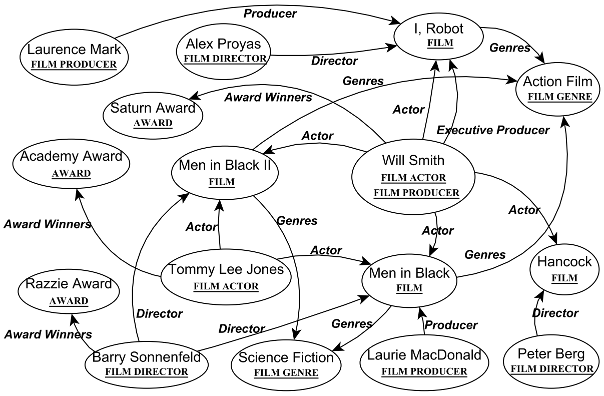

Figure 1: An Excerpt of an Entity Graph.

Entity graphs are often represented as RDF triples, due to heterogeneity of entities and the often lacking schema in data. The Linking Open Data community has interlinked billions of RDF triples spanning over several hundred datasets. [2] Many other entity graph datasets are also available from various data repositories such as Amazon's Public Data Sets, [3] Data.gov [4] and NCBI's databases. [5].

It can be a challenging task to select entity graphs for a particular need, given abundant datasets from many sources and the oftentimes scarce information available for the datasets. While sources such as the aforementioned data repositories often provide dataset descriptions, typically users cannot get a direct look at an entity graph itself before fetching the data. In this paper, we propose methods to automatically produce *preview tables* for entity graphs. Given an entity graph with many types of entities and relationships, we generate a set of tables, each of which for an important entity type. Each table comprises a set of attributes, each of which corresponds to a relationship associated with the entity type. Each tuple in the table consists of an entity belonging to the entity type and its related entities for the table attributes.

Figure 2 is a conceivable preview of the entity graph in Figure 1. It consists of two preview tables—the upper table has attributes FILM, *Director* and *Genres*, and the lower table has attributes FILM ACTOR and *Award Winners*. In this preview, entities of types FILM and FILM ACTOR are deemed of central importance in the entity graph.

[1] http://www.google.com/insidesearch/features/search/knowledge.html

[2] http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

[3] http://aws.amazon.com/publicdatasets/

[4] http://www.data.gov/

[5] http://www.ncbi.nlm.nih.gov/

	Film	*Director*	*Genres*
t_1	Men in Black	Barry Sonnenfeld	{Action Film, Science Fiction}
t_2	Men in Black II	Barry Sonnenfeld	{Action Film, Science Fiction}
t_3	Hancock	Peter Berg	-
t_4	I, Robot	Alex Proyas	{Action Film}

	Film Actor	*Award Winners*
t_5	Will Smith	Saturn Award
t_6	Tommy Lee Jones	Academy Award

Figure 2: A Two-Table Preview of the Entity Graph in Figure 1. (The upper and lower tables are for the subgraphs #1 and #2 in Figure 3, respectively.)

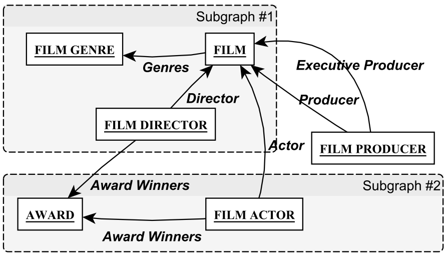

Figure 3: The Schema Graph for the Entity Graph in Figure 1.

Hence, Film and Film Actor are the *key attributes* of the two tables, respectively, marked by the underlines beneath them. Attributes *Director* and *Genres* in the upper table are considered highly related to Film entities. Similarly, *Award Winners* in the lower table is highly related to Film Actor entities. The two tables contain 4 and 2 tuples, respectively. For instance, the first tuple of the upper table is $t_1 = \langle$Men in Black, Barry Sonnenfeld, {Action Film, Science Fiction}$\rangle$. The tuple indicates that entity Men in Black belongs to type Film and has a relationship *Director* from Barry Sonnenfeld and has relationship *Genres* to both Action Film and Science Fiction.

The proposed preview tables are for compact presentation of important types of entities and their relationships in an entity graph. They assist users in attaining a quick and rough preview of the schema of the data. The tuples in the tables further give the users an intuitive understanding of the data. (Note that it is only necessary to show a few sample tuples instead of all.) The preview tables can be shown in a limited display space for a user to browse and explore, before the user decides to spend more time and resources (which may be monetary) to investigate the entity graph in more detail and fetch the complete entity graph.

To this end, several other approaches are arguably less adequate for gaining a quick overview of a data graph.

(1) The first solution is to visualize a data graph [8]. The whole graph can be large. For instance, in a September 2012 snapshot of the "film" domain of Freebase, there are 190K vertices (i.e., entities) and 1.6M edges (i.e., relationships). Given the sheer size and complexity of such data, a visualization tool is more effective for showing either the macro structure of the data graph or the local details surrounding individual nodes.

(2) The second solution is to show a schema graph corresponding to the data graph. Figure 3 is the schema graph for the entity graph in Figure 1. While its definition is given in Section 2, we note that it is generated by merging entity graph vertices of the same entity type and merging edges of the same relationship type. Although a schema graph is much smaller than the corresponding entity graph, it is not small enough for easy presentation and quick preview. For instance, the aforementioned "film" domain of Freebase consists of 50 entity types and 136 relationship types.

(3) The third approach is to present a summary of the schema graph. Schema summarization has been investigated for relational databases [16, 17, 18], XML [18] and graphs [14, 19]. While Section 7 discusses these works in more detail, we note that the preview tables proposed in this paper are different in several significant ways. It is unclear how to apply these methods on an entity graph or its schema graph, due to differences in data models. Some of these methods [16, 17, 18] work on relational and semi-structured data, instead of graph data. Some [18, 14, 19] produce trees or graphs as output instead of flat tables. Although it is plausible that some of these approaches can be adapted for entity graphs, there are more profound reasons that can render them ineffective. *First*, schema summary can still be quite large. For instance, the method in [16, 17] clusters the tables in a database but does not reduce the number of tables or the complexity of database schema. If we treat each entity type as a table and its neighboring entity types in the schema graph as the table attributes, the number of tables would equal the number of entity types. For the aforementioned "film" domain in Freebase, it means the users would need to understand the result of clustering 50 tables. *Second*, schema summarization is for helping database administrators and programmers in gaining a detailed understanding of a database in order to form queries. Our goal is to assist users in attaining a quick and rough understanding, before they decide to investigate the entity graph in more detail and fetch the complete dataset. Therefore we look for a structure much smaller than the schema summaries in the aforementioned works.

In our definition (details in Section 2), a *preview* is a set of preview tables, each of which has a *key attribute* (corresponding to an entity type) and a set of *non-key attributes* (each corresponding to a relationship type). Given an entity graph and its schema graph, there is thus a large space of possible previews. Our goal is to find an "optimal" preview in the space. To this end, we tackle several challenges: (1) We discern what factors contribute to the goodness of a preview and propose several scoring functions for key and non-key attributes as well as preview tables. The scoring functions are based on several intuitions related to how much information a preview conveys and how helpful it is to users. (2) Based on the scoring measures, a preview's score is maximized when it includes as many tables and attributes as possible. However, the purpose of having a preview is to help users attain a quick understanding of data and thus a preview must fit into a limited display space. Considering the tradeoff, we enforce a constraint on preview size. Furthermore, we consider enforcing an additional constraint on the pairwise distance between preview tables. Given the spaces of all possible previews, we formulate the optimization problem of finding an preview with the highest score among those satisfying the constraints. The optimization is non-trivial, as we prove that it is **NP**-hard under distance constraint. (3) The search space of previews grows exponentially by data size and the constraints. A brute-force approach is thus too costly. For efficiently finding optimal previews, we designed a dynamic programming algorithm and an Apriori [1]-style algorithm.

In summary, this paper makes the following contributions:

- We motivated the novel problem of generating preview tables for entity graphs.
- We proposed ideas for measuring the goodness of previews based on several intuitions. (Section 3)
- We formulated optimal preview discovery problem, and proved its **NP**-hardness under distance constraint. (Section 4)
- We developed a dynamic-programming algorithm and an Apriori-style algorithm for finding optimal previews. (Section 5)
- Extensive experiments and user study verified the accuracy of the scoring measures, the efficiency of the algorithms, and the overall effectiveness of discovered previews. (Section 6)

$G_d(V_d, E_d)$	an entity graph
$v \in V_d$	an entity
$e(v, v') \in E_d$	a directed relationship from entity v to entity v'
$G_s(V_s, E_s)$	a schema graph
$\tau \in V_s$	an entity type
$\gamma(\tau, \tau') \in E_s$	a relationship type from entity type τ to entity type τ'
T	a preview table
$T.key$	the key attribute of T
$T.nonkey$	the non-key attributes of T
$T.\tau$	the set of entities of type τ, which is the key attribute of T
$t \in T$	a tuple t in preview table T
$t.\tau$	t's value on τ which is the key attribute of T
$t.\gamma$	t's value on non-key attribute γ
$\mathcal{P} = \{\mathcal{P}[1], ..., \mathcal{P}[k]\}$	a preview, which consists of k preview tables
$\mathcal{P}_{opt}$	an optimal preview
$S(\mathcal{P})$	the score of preview $\mathcal{P}$
$S(T)$	the score of preview table T
$S_{cov}(\tau), S_{walk}(\tau)$	score of key attribute τ based on coverage and random walk
$S^{\tau}_{cov}(\gamma), S^{\tau}_{ent}(\gamma)$	score of non-key attribute γ based on coverage and entropy
$\mathbb{T}$	the space of all possible preview tables
$\mathbb{P}$	the space of all possible previews
$dist(\tau, \tau')$	distance between τ and τ' in schema graph G_s

Table 1: Notations

2. PREVIEW DISCOVERY PROBLEM

An *entity graph* is a directed graph $G_d(V_d, E_d)$ with vertex set V_d and edge set E_d. Each vertex $v \in V_d$ represents an entity and each edge $e(v, v') \in E_d$ represents a directed relationship from entity v to v'. The entity graph G_d is actually a multigraph since there can be multiple edges between two vertices. (E.g., in Figure 1, there are two edges *Actor* and *Executive Producer* from entity Will Smith to entity I, Robot.)

Each entity is labeled by a name. For simplicity and intuitiveness of presentation, we shall mention entities by their names, assuming all entities have distinct names, although in reality they are distinguished by unique identifiers. Each entity belongs to one or more *entity types*, underlined in Figure 1. (E.g., Will Smith belongs to types Film Actor and File Producer and I, Robot belongs to type Film.) Each relationship belongs to a *relationship type*. (E.g., the edge from Will Smith to Men in Black has type *Actor*.) The type of a relationship determines the types of its two end entities. For instance, an edge of type *Actor* is always from an entity belonging to File Actor to an entity belonging to Film. We will mention edges by the surface names of their relationship types. Two different relationship types may have the same surface name for intuitively expressing their meanings, although underlyingly they have different identifiers. For instance, the *Award Winners* edge from Will Smith to Saturn Award and the *Award Winners* edge from Barry Sonnenfeld to Razzie Award belong to two different relationship types. The former is for relationships from Film Actor to Award, while the latter is for relationships from Film Director to Award.

Given an entity graph $G_d(V_d, E_d)$, its *schema graph* is a directed graph $G_s(V_s, E_s)$, where each vertex $\tau \in V_s$ represents an entity type and each directed edge $\gamma(\tau, \tau') \in E_s$ represents a relationship type from entity type τ to τ'. An edge $\gamma(\tau, \tau') \in E_s$ if and only if there exists an edge $e(v, v') \in E_d$ where e has type γ, v has type τ and v' has type τ'. Figure 3 shows the schema graph corresponding to the entity graph in Figure 1. Note that a schema graph is also a multigraph as there can be multiple relationship types between two entity types. For example, from entity type Film Producer to Film there are two relationship types—*Producer* and *Executive Producer*. It is clear from the above definitions that, given a data graph, the corresponding schema graph is uniquely determined.

Definition 1 (Preview Table and Preview)**.** Given an entity graph $G_d(V_d, E_d)$ and its schema graph $G_s(V_s, E_s)$, a *preview table* T is a table with a mandatory *key attribute* (denoted $T.key$) and at least one *non-key attributes* (denoted $T.nonkey$). T corresponds to a star-shape subgraph of the schema graph $G_s(V_s, E_s)$. The key attribute corresponds to an entity type $\tau \in V_s$, and each non-key attribute corresponds to a relationship type $\gamma(\tau, \tau') \in E_s$ or $\gamma(\tau', \tau) \in E_s$. Note that the edges from and to an entity are both important. Hence, the non-key attributes of T include both $\gamma(\tau, \tau')$ and $\gamma(\tau', \tau)$.

The preview table T consists of a set of tuples. The number of tuples in T equals the number of entities of type τ, which is the key attribute of T, i.e., $|T| = |T.\tau|$ and $T.\tau = \{v | v \in V_d \wedge v \text{ has type } \tau\}$.

Given an arbitrary tuple $t \in T$, we denote t's key attribute value by $t.\tau$. We denote its values on a non-key attribute γ by $t.\gamma$.

Each tuple $t \in T$ thus attains a distinct value on the key attribute τ. Its value on a non-key attribute $\gamma(\tau, \tau')$ is a set—the set of entities in entity graph G_d incident from $t.\tau$ through an edge of type $\gamma(\tau, \tau')$. More formally, $t.\gamma(\tau, \tau') = \{u | u \in V_d \wedge e(t.\tau, u) \in E_d \wedge u \text{ belongs to type } \tau'\}$. Symmetrically, its value on a non-key attribute $\gamma(\tau', \tau)$ is the set of entities in G_d incident to $t.\tau$ through an edge of type $\gamma(\tau', \tau)$. More formally, $t.\gamma(\tau', \tau) = \{u | u \in V_d \wedge e(u, t.\tau) \in E_d \wedge u \text{ belongs to type } \tau'\}$.

A *preview* $\mathcal{P}$ is a set of preview tables, i.e., $\mathcal{P} = \{\mathcal{P}[1], ..., \mathcal{P}[k]\}$, where $\forall i \neq j, \mathcal{P}[i].key \neq \mathcal{P}[j].key$, $k \leqslant |V_s|$ is the total number of preview tables. Note that $|V_s|$ is the number of vertices in G_s, i.e., the number of entity types in G_d. ■

According to Definition 1, the upper and lower tables in Figure 2 correspond to the star-shape subgraphs #1 and #2 in Figure 3, respectively. The key attribute in the upper table is Film with its non-key attributes are *Director*, *Genres*. Similarly, the key attribute in the lower table is Film Actor with its non-key attributes are *Award Winners*. Due to the aforementioned symmetric relation, if there exists a preview table with key attribute Director, it may have *Film* as one of its non-key attributes. It is worth noting that, although each tuple's value on the key attribute is non-empty, unique and single-valued, its value on a non-key attribute can be empty (e.g., t_3.*Genres* in Figure 2), duplicate (e.g., t_1.*Director* and t_2.*Director* in Figure 2) and multi-valued (e.g., t_1.*Genres* and t_2.*Genres* in Figure 2). It also follows that a preview table is not a relational table.

By Definition 1, every vertex τ in a schema graph can serve as the key attribute of a candidate preview table, which also includes at least one non-key attribute—an edge incident on τ. We use $\mathbb{T}$ to denote the space of all possible preview tables. A preview is a set of preview tables. We use $\mathbb{P}$ to denote the space of all possible previews. Note that $\mathbb{P} \subset 2^{\mathbb{T}}$, i.e., not every member of the power set $2^{\mathbb{T}}$ is a valid preview, because by Definition 1 preview tables in a preview cannot have the same key attribute.

Problem Statement: Given an entity graph $G_d(V_d, E_d)$ and its corresponding schema graph $G_s(V_s, E_s)$, the *preview discovery problem* is to find $\mathcal{P}_{opt}$—the optimal preview among all possible previews. We shall develop the notion of goodness for a preview and define its measures in Section 4.

3. SCORING MEASURES FOR PREVIEWS

In this section, we discuss the scoring functions for measuring the goodness of previews for entity graphs. While it is possible to propose many conceivable scoring measures, we present measures based on two intuitions: 1) a good preview should relate to as many entities and relationships as possible; and 2) a good preview should be helpful for users to understand or to browse the entity graph. The first intuition is obvious, as a preview relating to only a small number of entities or relationships will inevitably lose lots of information thus leads to poor comprehensibility of the original graph. The second intuition tries to model the goodness of previews based on users' behaviors of browsing the entity graph and the preview tables using the same ideas behind PageRank algorithm [4] and decision tree learning [11].

3.1 Preview Scoring

We measure the score of a preview by aggregating the scores of each individual preview tables, and we measure the score of a preview table by aggregating the scores of its key attribute and non-key attributes. We further elaborate the scoring measures for key and non-key attributes in Sections 3.2 and 3.3.

The aggregated score of a preview $\mathcal{P} = \{\mathcal{P}[1], ..., \mathcal{P}[k]\}$ is simply given by the summation of individual preview tables' scores:

$$S(\mathcal{P}) = \sum_{i=1}^{k} S(\mathcal{P}[i]), \qquad (1)$$

where $S(\mathcal{P}[i])$ is the score of a preview table $\mathcal{P}[i]$, defined as:

$$S(\mathcal{P}[i]) = S(\tau) \times \sum_{\gamma \in \mathcal{P}[i].nonkey} S^{\tau}(\gamma), \qquad (2)$$

where $S(\tau)$ is the score of the key attribute of $\mathcal{P}[i]$ (i.e., $\mathcal{P}[i].key$=τ) and $S^{\tau}(\gamma)$ is the score of a non-key attribute γ with regard to the key attribute τ.

In the above definition, the score of a preview table equals the product of its key attribute's score and the summation of its non-key attributes' scores. The definition gives the key attribute τ much higher importance than any individual non-key attribute, because the preview table centers around the entities of type τ and describes their non-key attributes, i.e., their relationships with other entities.

3.2 Key Attribute Scoring

Coverage-based scoring measure:

Given an entity graph $G_d(V_d, E_d)$ and its corresponding schema graph $G_s(V_s, E_s)$, the key attribute τ of a candidate preview table T corresponds to an entity type, i.e., $\tau \in V_s$. If the entity graph consists of many entities of type τ, including T in the preview makes the preview relevant to all those entities. The coverage-based scoring measure thus defines the score of τ as the number of entities bearing that type:

$$S_{cov}(\tau) = |\{v | v \in V_d \wedge v \text{ has type } \tau\}|$$

For example, given the entity graph in Figure 1 and the corresponding schema graph in Figure 3, the coverage-based score of the key attribute FILM is $S_{cov}(\text{FILM}) = 4$.

Random-walk based scoring measure:

We consider a *random walk process* over a graph G converted from the schema graph $G_s(V_s, E_s)$, inspired by the PageRank algorithm for Web page ranking. In G, the vertices are entity types and the edges are undirected. The edge between τ_i and τ_j in G is weighted by the number of relationships (i.e., the number of edges) in the entity graph between entities of types τ_i and τ_j. We denote the weight by w_{ij}, defined as follows.

$$w_{ij} = w_{ji} = \sum_{\gamma(\tau_i,\tau_j) \in E_s} |\{e | e \in E_d \wedge e \text{ has type } \gamma(\tau_i, \tau_j)\}|$$
$$+ \sum_{\gamma(\tau_j,\tau_i) \in E_s} |\{e | e \in E_d \wedge e \text{ has type } \gamma(\tau_j, \tau_i)\}|$$

The *transition matrix* M is a $|V_s| \times |V_s|$ matrix where an element M_{ij} corresponds to the *transition probability* from τ_i to τ_j in G. M_{ij} equals the ratio of w_{ij} to the total weight of all edges incident on τ_i in G:

$$M_{ij} = \frac{w_{ij}}{\sum_k w_{ik}}$$

For example, the transition probability from FILM to FILM GENRE is $M_{\text{FILM},\text{FILM GENRE}} = w_{\text{FILM},\text{FILM GENRE}} / (w_{\text{FILM},\text{FILM GENRE}} + w_{\text{FILM},\text{FILM ACTOR}} + w_{\text{FILM},\text{FILM DIRECTOR}} + w_{\text{FILM},\text{FILM PRODUCER}}) = 5/(5+6+4+3) = 0.28$. The transition probability from FILM to FILM PRODUCER is $M_{\text{FILM},\text{FILM PRODUCER}} = w_{\text{FILM},\text{FILM PRODUCER}} / (w_{\text{FILM},\text{FILM GENRE}} + w_{\text{FILM},\text{FILM ACTOR}} + w_{\text{FILM},\text{FILM DIRECTOR}} + w_{\text{FILM},\text{FILM PRODUCER}}) = 3/(5+6+4+3) = 0.17$.

Suppose a user traverses in G, either by going from an entity type τ_i to another entity type τ_j through the edge between them with probability M_{ij} or by jumping to a random entity type. Entity types that are more likely to be visited by the user are of higher importance. The random walk process will converge to a stationary distribution which represents the chances of entity types being visited. The stationary distribution π of the random walk process is given as follows. Note that a similar idea was applied in [16] for ranking relational tables by importance.

$$\pi = \pi M$$

The random-walk based score of a candidate key attribute τ_i is: $S_{walk}(\tau_i) = \pi_i$, where π_i is the stationary probability of τ_i.

3.3 Non-Key Attribute Scoring

Coverage-based scoring measure:

The coverage-based scoring measure for non-key attribute is similar to that for key attribute. Given an entity graph $G_d(V_d, E_d)$ and its schema graph $G_s(V_s, E_s)$, consider a candidate preview table T with key attribute τ. A non-key attribute γ of T corresponds to a relationship type, i.e., $\gamma \in E_s$. If the entity graph contains many edges (i.e., relationships) belonging to the type γ, incorporating such a relationship type into the table T makes it relevant to all those relationships and their corresponding entities. The coverage-based scoring measure thus defines the score of γ as the number of relationships bearing that type:

$$S^{\tau}_{cov}(\gamma) = |\{e | e \in E_d \wedge e \text{ has type } \gamma\}|$$

For example, given the entity graph in Figure 1 and the corresponding schema graph in Figure 3, the coverage-based scores of non-key attributes *Director* and *Genres* are $S^{\text{FILM}}_{cov}(\textit{Director}) = 4$ and $S^{\text{FILM}}_{cov}(\textit{Genres}) = 5$.

The coverage-based scoring measure for non-key attribute is symmetric, i.e., given $\gamma(\tau, \tau')$ (or $\gamma(\tau', \tau)$) $\in T.nonkey$, $S^{\tau}_{cov}(\gamma) \equiv S^{\tau'}_{cov}(\gamma)$. Both τ and τ' can be the key attribute of a different preview table, in which γ is a non-key attribute. The scores of γ in the two tables are equal.

Entropy-based scoring measure:

For a preview table T with key attribute τ, we measure the goodness of a non-key attribute $\gamma(\tau, \tau')$ (or $\gamma(\tau', \tau)$) by how much information it provides to T, for which the *entropy* of γ ($H(\gamma)$) is a natural choice of measure:

$$S^{\tau}_{ent}(\gamma) = H(\gamma) = \sum_{j=1} \frac{n_j}{|t.\gamma|} \log(\frac{|t.\gamma|}{n_j}),$$

where n_j is the number of tuples in T that attain the same jth attribute value u on non-key attribute $\gamma(\tau, \tau')$ (or $\gamma(\tau', \tau)$), i.e., $u \in V_d \wedge u \text{ has type } \tau'$ and $n_j = |\{v | v \in T.\tau \wedge e(v, u) \in E_d$ (or $e(u, v) \in E_d) \wedge e \text{ has type } \gamma\}|$. $|t.\gamma|$ is the number of distinct non-key attribute values of $\gamma(\tau, \tau')$ (or $\gamma(\tau', \tau)$). Continue with the example above, the entropy-based scores of non-key attributes *Director* and *Genres* are $S^{\text{FILM}}_{ent}(\textit{Director}) = (2/4)\log(4/2) + (1/4)\log(4/1) + (1/4)\log(4/1) = 0.45$, and $S^{\text{FILM}}_{ent}(\textit{Genres}) = (2/3)\log(3/2) + (1/3)\log(3/1) = 0.28$. Note that for two values on a multi-valued attribute (e.g., {Action Film, Science Fiction} and {Action Film} on FILM.*Genres* in Figure 2), we consider them equivalent if and only if they have the same set of component values. It is easy to see that, by definition, the entropy-based scoring measure for non-key attribute is asymmetric, i.e., given $\gamma(\tau, \tau')$ (or $\gamma(\tau', \tau)$) $\in T.nonkey$, $S^{\tau}_{ent}(\gamma) \not\equiv S^{\tau'}_{ent}(\gamma)$.

4. OPTIMAL PREVIEWS UNDER SIZE AND DISTANCE CONSTRAINTS

In this section, based on the scoring measures defined in Section 4, we formulate several optimization problems that look for the optimal previews with best scores under various constraints on preview size and distance between preview tables. We prove that some of these optimization problems are **NP**-hard.

By Equation 1 (or any other monotonic aggregate function), the score of a preview monotonically increases by its member preview tables—the more preview tables in a preview, the higher its score. Similarly by Equation 2, the score of a preview table monotonically increases by its non-key attributes. The properties are formally stated in the following two propositions. Recall that $\mathbb{P}$ and $\mathbb{T}$ denote the space of all possible previews and all possible preview tables.

Proposition 1. Given previews $\mathcal{P}_1, \mathcal{P}_2 \in \mathbb{P}$, if $\mathcal{P}_1 \supseteq \mathcal{P}_2$, then $S(\mathcal{P}_1) \geq S(\mathcal{P}_2)$.

Proposition 2. Given preview tables $T_1, T_2 \in \mathbb{T}$, if $T_1.key = T_2.key$ and $T_1.nonkey \supseteq T_2.nonkey$, then $S(T_1) \geq S(T_2)$.

By the above propositions, a preview's score is maximized when it includes as many tables and attributes as possible. However, the purpose of having a preview is to help users attain a quick understanding of data and thus a preview must fit into a limited display space. Therefore the size and the goodness score of a preview present a tradeoff. Considering the tradeoff, we enforce a constraint on preview size, given by a pair of integers (k, n), where k is the number of allowed preview tables and n is the number of allowed attributes in the tables. The previews satisfying the size constraint are called *concise previews*.

Furthermore, we consider enforcing an additional constraint on the pairwise distance between preview tables. The distance between two preview tables T_1 and T_2 (denoted $dist(T_1, T_2)$) is the length of the shortest undirected path[6] between their key attributes $T_1.key$ and $T_2.key$ in schema graph G_s. Recall that the key attributes are vertices (i.e., entity types) in G_s. For example, the distance between the two tables in Figure 2 is 1, which is the shortest path length for entity types Film and Film Actor in schema graph in Figure 3. Similarly, for two tables whose key attributes are Film and Award, their distance would be 2.

Based on the above notion of distance, the constraint on table distance is given by an integer d, which is the maximum (minimum) distance between preview tables. The previews satisfying the distance constraint are called *tight (diverse) previews*. Intuitively speaking, the preview tables in a tight preview are highly related to each other due to their short pairwise distance, while the preview tables in a diverse preview are not tightly related to each other and cover different types of concepts. Arguably, both types of previews are useful for understanding an entity graph. We shall compare them empirically in Section 6.

Given the spaces of all possible concise, tight and diverse previews, we formulate three optimization problems of finding an *optimal preview*—a preview with the highest score among the corresponding space of previews. Below we formally define the three types of previews and the corresponding optimization problems.

Definition 2 (Concise, Tight and Diverse Previews)**.** Given the size constraint (k, n), a *concise preview* has k preview tables (i.e., key attributes) and no more than n non-key attributes in the tables. [7] The space of all concise previews is

$$\mathbb{P}_{k,n} = \{\mathcal{P} \mid \mathcal{P} \in \mathbb{P}, |\mathcal{P}| = k, \sum_{i=1}^{k} |\mathcal{P}[i].nonkey| \leq n\}.$$

Given the size constraint (k, n) and the distance constraint d, a *tight preview* (*diverse preview*) is a concise preview in which the distance between any pair of preview tables is smaller (greater) than or equal to d. The space of all tight previews is

$$\mathbb{P}_{k,n,\leq d} = \{\mathcal{P} \mid \mathcal{P} \in \mathbb{P}_{k,n}, \forall T_1, T_2 \in \mathcal{P}, dist(T_1, T_2) \leq d\}.$$

The space of all diverse previews is

$$\mathbb{P}_{k,n,\geq d} = \{\mathcal{P} \mid \mathcal{P} \in \mathbb{P}_{k,n}, \forall T_1, T_2 \in \mathcal{P}, dist(T_1, T_2) \geq d\}.$$

Definition 3 (Optimal Preview Discovery Problems)**.** The optimization problem of finding an *optimal preview* is defined as follows, where $\mathbb{P}$ can be any of the aforementioned three spaces—$\mathbb{P}_{k,n}$, $\mathbb{P}_{k,n,\leq d}$ and $\mathbb{P}_{k,n,\geq d}$. Note that the arg max function may return a set of optimal previews due to ties in scores.

$$\mathcal{P}_{opt} \in \arg\max_{\mathcal{P} \in \mathbb{P}} S(\mathcal{P}) \qquad (3)$$

The optimal preview discovery problems are non-trivial. Particularly, we prove that the optimal preview discovery problem in the spaces of both tight previews ($\mathbb{P}_{k,n,\leq d}$) and diverse previews ($\mathbb{P}_{k,n,\geq d}$) is **NP**-hard.

Theorem 1. The optimal tight preview discovery problem is **NP**-hard.

Proof. The decision version of the optimal tight preview discovery problem is $TightPreview(G_s, k, n, d, s)$—Given a schema graph G_s, decide whether there exists such a preview $\mathcal{P}$ that (1) $\mathcal{P}$ has k tables and no more than n non-key attributes; (2) the distance between every pair of preview tables is not greater than d; and (3) the preview's score is at least s, i.e., $S(\mathcal{P}) \geq s$.

We construct a reduction, in polynomial-time, from the **NP**-hard Clique problem to $TightPreview(G_s, k, n, d, s)$. Recall that the decision version of $Clique(G, k)$ is to, given a graph $G(V, E)$, decide whether there exists a clique in G with k vertices. The reduction is by constructing a schema graph G_s from G. For simplicity of exposition, in both this proof and the proof of Theorem 2, we assume the schema graph G_s is undirected and every edge γ in G_s corresponds to the same relationship type. This assumption is made without loss of generality. Note that our following proof casts no requirement on the score of a preview (i.e., $s = 0$) and thus no requirement on the scores of key and non-key attributes in G_s. Hence, edge orientation and its corresponding relationship type bears no significance in the proof.

In detail, we construct a schema graph $G_s(V_s, E_s)$ from G through a vertex bijection $f : V \to V_s$:

- $\forall e(v, v') \in E$, there exists an edge (i.e., relationship type) $\gamma(\tau, \tau') \in E_s$, where $\tau = f(v)$ and $\tau' = f(v')$.
- $\forall \gamma(\tau, \tau') \in E_s$, there exists an edge $e(v, v') \in E$, where $v = f^{-1}(\tau)$ and $v' = f^{-1}(\tau')$.

$Clique(G, k)$ is thus reduced to $TightPreview(G_s, k, k, 1, 0)$ by the above bijections. ■

The **NP**-hardness of the optimal diverse preview discovery problem is also based on a reduction from the Clique problem, although the proof is more complex.

[6] An undirected path in a directed graph is a path in which the edges are not all oriented in the same direction.

[7] A preview with less than n non-key attributes may outscore another preview with exactly n non-key attributes. Further, a set of k entity types may have only less than n edges in the schema graph. Hence, the condition $|\mathcal{P}[i].nonkey| \leq n$ instead of $|\mathcal{P}[i].nonkey| = n$. On the other hand, it is safe to assume that an entity graph with practical significance always has more than k entity types under any reasonably small k. Therefore an optimal preview always should have exactly k preview tables, given the monotonic scoring function (cf. Equation 1).

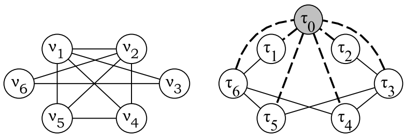

Figure 4: Construction of G_s (Right) from G (Left), for Reduction from the Clique Problem to the Optimal Diverse Preview Discovery Problem.

Theorem 2. The optimal diverse preview discovery problem is **NP**-hard.

Proof. The decision version of the optimal diverse preview discovery problem is $DiversePreview(G_s, k, n, d, s)$—Given a schema graph G_s, decide whether there exists such a preview $\mathcal{P}$ that (1) $\mathcal{P}$ has k tables and no more than n non-key attributes; (2) the distance between every pair of preview tables is not smaller than d; and (3) the preview's score is at least s, i.e., $S(\mathcal{P}) \geq s$.

We construct a reduction, in polynomial-time, from the **NP**-hard $Clique(G, k)$ to $DiversePreview(G_s, k, n, d, s)$. The reduction is also by constructing a schema graph $G_s(V_s, E_s)$ from G. It is similar to the reduction for $TightPreview(G_s, k, n, d, s)$ in Theorem 1, but also bears two important differences. (1) G_s contains a special vertex, denoted τ_0, that is directly connected to every other vertex in G_s. (2) Barring τ_0 and all its incident edges, G_s is the complement graph of G—There is still a vertex bijection $f : V \rightarrow V_s$, but an edge exists between two vertices in G_s if and only if there is no edge between the corresponding vertices in G. In detail, the construction of G_s from G is as follows:

- $\forall \tau, \tau' \in V_s \backslash \{\tau_0\}$, $\gamma(\tau, \tau') \in E_s$ if and only if $\nexists e(v, v') \in E$, where $v = f^{-1}(\tau)$ and $v' = f^{-1}(\tau')$.
- $\forall \tau \in V_s \backslash \{\tau_0\}$, $\gamma(\tau_0, \tau) \in E_s$.

$Clique(G, k)$ is thus reduced to $DiversePreview(G_s, k, k, 2, 0)$ by the above construction of G_s. ■

To understand why $Clique(G, k)$ is reduced to $DiversePreview$ $(G_s, k, k, 2, 0)$ by the construction of G_s from G in the proof of Theorem 2, consider Figure 4. The figure shows an example with G (left) and the constructed schema graph G_s (right), where the gray vertex in G_s is τ_0. Consider an arbitrary pair of vertices (v, v') in G and their corresponding vertices (τ, τ') in G_s. On the one hand, if v and v' are not directly connected in G (e.g., v_1 and v_6), an edge between τ and τ' (i.e., τ_1 and τ_6) is included into G_s. When finding a diverse preview where pairwise table distance must be at least 2, τ and τ' will never be chosen as the key attributes of two tables in the preview. Correspondingly, this means a clique must not include both v and v'. On the other hand, if v and v' are directly connected in G (e.g., v_1 and v_2), there must not be a direct edge between τ and τ' (i.e., τ_1 and τ_2) in G_s. The distance between τ and τ' is exactly 2, since they are only indirectly connected through τ_0. They will thus be considered in choosing the key attributes of two tables in a diverse preview where pairwise table distance must be at least 2. Correspondingly, the directly connected v and v' are thus considered together in forming a clique.

5. ALGORITHMS

In this section we discuss algorithms for solving the optimal preview discovery problem. As given in Equation 3, the problem is to find a preview with the highest score among candidate previews, where the space of candidates can be concise previews ($\mathbb{P}_{k,n}$), tight previews ($\mathbb{P}_{k,n,\leq d}$) or diverse previews ($\mathbb{P}_{k,n,\geq d}$). Recall that we use $S(\tau)$ to denote the score of a candidate key attribute τ for a preview table T and $S^\tau(\gamma)$ to denote the score of a candidate non-key attribute $\gamma(\tau, \tau')$ (or $\gamma(\tau', \tau)$) for T whose key attribute is τ.

Algorithm 1: Brute-Force Algorithm for Optimal Preview Discovery

```
Input  : schema graph $G_s$, size constraint $(k, n)$
Output: an optimal preview $\mathcal{P}_{opt}$
1  foreach $\tau \in V_s$ do
2      $\langle \gamma_1^\tau, \gamma_2^\tau, \ldots \rangle \leftarrow$ sort the candidate non-key attributes $\gamma_j^\tau \in \Gamma^\tau$
       by their scores $S^\tau(\gamma_j^\tau)$;
3  $max\_score \leftarrow 0$; $\mathcal{P}_{opt} \leftarrow \varnothing$;
4  foreach $k$-subset of $V_s$ (denoted $V$) do
5      $score \leftarrow 0$; $\mathcal{P} \leftarrow \varnothing$; $i \leftarrow 1$;
6      foreach $\tau \in V$ do
7          $\mathcal{P}[i].key = \tau$;
8          $\mathcal{P}[i].nonkey = \{\gamma_1^\tau\}$;
9          $score = score + S(\tau) \times S^\tau(\gamma_1^\tau)$;
10         $i \leftarrow i + 1$;
11     $\Gamma \leftarrow$ top-$(n-k)$ candidate non-key attributes from all $\tau \in V$ in
       descending order of $S(\tau) \times S^\tau(\gamma_j^\tau)$;
12     foreach $\gamma_j^\tau \in \Gamma$, where $\tau = \mathcal{P}[x].key$ do
13         $score \leftarrow score + S(\tau) \times S^\tau(\gamma_j^\tau)$;
14         $\mathcal{P}[x].nonkey \leftarrow \mathcal{P}[x].nonkey \bigcup \{\gamma_j^\tau\}$;
15     if $score > max\_score$ then
16         $max\_score \leftarrow score$;
17         $\mathcal{P}_{opt} \leftarrow \mathcal{P}$;
18 return $\mathcal{P}_{opt}$;
```

Before we present the algorithms, consider the space of all possible previews. Every entity type τ can be the key attribute of a preview table T. Suppose Γ^τ denotes the set of all edges (i.e., relationship types) incident on τ in schema graph G_s. Any $\gamma \in \Gamma^\tau$ can be a candidate for the non-key attributes of T. By the scoring function in Equation 2 and the problem formulation in Equation 3, the non-key attributes of T must have the highest scores among the candidates in Γ^τ. This is formally stated in Theorem 3, which is an important property used by our algorithms.

Theorem 3. Suppose an optimal (concise, tight or diverse) preview $\mathcal{P}_{opt}$ contains a preview table $T \in \mathbb{T}$ with key attribute τ. If T has m non-key attributes, they must be the top-m non-key attributes by scores, i.e., $\forall \gamma, \gamma' \in \Gamma^\tau$, if $\gamma \in T.nonkey$ and $\gamma' \notin T.nonkey$, then $S^\tau(\gamma) \geq S^\tau(\gamma)$.

5.1 A Brute-Force Algorithm

This section presents a brute-force algorithm for the optimal preview discovery problem, shown in Algorithm 1. It enumerates all possible k-subsets of entity types, as the k entity types in each subset form the key attributes of k preview tables in a preview $\mathcal{P}$ (Line 4). For a candidate key attribute τ, the elements in the set of its candidate non-key attributes Γ^τ are ordered by their scores. We denote these candidates in descending order of scores by γ_1^τ, γ_2^τ, and so on (Line 2). Suppose preview table T uses τ as its key attribute. Each table must contain at least one non-key attribute, according to Definition 1. Hence, γ_1^τ (i.e., the candidate non-key attribute with the highest score) must be included into $T.nonkey$ (Line 8), by Theorem 3. Further, among the remaining candidate non-key attributes for the k entity types, the top-$(n-k)$ candidates by scores must be included into $\mathcal{P}$ (Lines 11–14), by Theorem 3. Note that, since the sorted list of candidate non-key attributes for each τ is already created (Line 2), it is unnecessary to do a full sorting in order to determine the top-$(n-k)$ candidates Γ. Instead, a simple merge operation on the k sorted lists will get Γ.

The complexity of this algorithm is $O(KN \log N + \binom{K}{k}(k+n))$, where $K = |V_s|$ is the number of candidate key attributes, $N = 2|E_s|$ is the number of candidate non-key attributes for all candidate key attributes, $\binom{K}{k}$ is the number of k-subsets, and $KN \log N$

is for sorting individual lists of candidates (Line 2), in which each list contains at most N elements.

Algorithm 1 is for finding one of the optimal previews. To find all optimal previews, it needs simple extension to deal with ties in scores, which we will not further discuss.

The same brute-force algorithm is applicable for optimal preview discovery in all three types of spaces—concise, tight and diverse previews. The pseudo code in Algorithm 1 is for concise previews and does not enforce distance constraint, for simplicity of presentation. Enforcing distance constraint for tight/diverse previews is straightforward, by performing distance check on every pair of preview tables in each k-subset of entity types.

5.2 A Dynamic-Programming Algorithm for Concise Preview Discovery Problem

As the combinatorial number of k-subsets grows exponentially, the performance of the above brute-force algorithm becomes unacceptable for finding an optimal preview under modest size constraints. We thus developed a dynamic-programming algorithm to more efficiently discover optimal concise previews.

Consider an arbitrary order on all K entity types—τ_1, ..., τ_K. We use $\mathcal{P}_{opt}(k, n, x)$ to denote an optimal concise preview among the first x entity types $\tau_1, \ldots, \tau_x$. Thus the optimal concise preview discovery problem is to find $\mathcal{P}_{opt}(k, n, K)$. $\mathcal{P}_{opt}(k, n, x)$ can be constructed from the solutions to smaller problems, in two ways. (1) It can be equal to $\mathcal{P}_{opt}(k, n, x-1)$, i.e., its k tables and n non-key attributes are from the first $x-1$ entity types and the x-th entity type τ_x does not contribute anything. (2) It can be the union of $\mathcal{P}_{opt}(k-1, n-m, x-1)$ and a table T_x^m. $\mathcal{P}_{opt}(k-1, n-m, x-1)$ is an optimal preview with $k-1$ tables and $n-m$ non-key attributes among the first $x-1$ entity types. T_x^m is the table whose key attribute is τ_x and whose non-key attributes are the top-m elements in Γ^{τ_x}—the sorted list of candidate non-key attributes for τ_x. The number m is between 1 and $n-(k-1)$ (or less if there are less than $n-(k-1)$ elements in Γ^{τ_x}), since each of the $k-1$ tables in $\mathcal{P}_{opt}(k-1, n-m, x-1)$ must contribute at least one non-key attribute. The optimal substructure of the problem is formally given as follows. (We do not discuss boundary cases (i.e., $k = 1$ or $x = 1$ or $n = k$) due to space limitations.)

$$\mathcal{P}_{opt}(k, n, x) = \underset{\mathcal{P} \in \mathbb{P}(k,n,x)}{\arg\max}\ S(\mathcal{P})$$

$$\mathbb{P}(k, n, x) = \left\{ \begin{array}{l} \mathcal{P}_{opt}(k, n, x-1), \\ \mathcal{P}_{opt}(k-1, n-1, x-1) \bigcup \{T_x^1\}, \\ \mathcal{P}_{opt}(k-1, n-2, x-1) \bigcup \{T_x^2\}, \\ \ldots \\ \mathcal{P}_{opt}(k-1, k-1, x-1) \bigcup \{T_x^{n-(k-1)}\} \end{array} \right\},$$

where $T_x^m.key = \tau_x$ and $T_x^m.nonkey$ = top-m candidate non-key attributes in Γ^{τ_x}. Note that the optimal substructure is inapplicable when previews must satisfy distance constraint in addition to size constraint (details omitted). Therefore the dynamic-programming algorithm is for concise previews but not tight/diverse previews.

The pseudo code of the dynamic-programming algorithm is shown in Algorithm 2. Its complexity is $O(KN \log N + Kkn^2)$. Similar to Algorithm 1, Algorithm 2 is for finding one optimal preview. Finding all optimal previews requires simple extension to deal with ties in scores, which we will not further discuss.

Both Algorithm 1 and 2 assume that, given any k entity types (key attributes), they always together have at least n non-key attributes. That may not be true in reality. In fact, for two previews with the same number of tables, the preview with less non-key attributes may have the higher score than the other preview. Note that, in Equation 3, the optimal preview is not required to have exactly n non-key attributes. It is simple to extend Algorithm 1 and 2 to fully comply with the definition. Given any entity type τ, if it has less than n candidate non-key attributes, we can simply pad the sorted list Γ^{τ} by pseudo non-key attributes with zero scores.

Algorithm 2: Dynamic-Programming Algorithm for Optimal Concise Preview Discovery

```
Input  : schema graph $G_s$, size constraint $(k, n)$
Output: an optimal concise preview $\mathcal{P}_{opt}$
1  foreach $x \leftarrow 1$ to $K$ do
2      $\langle \gamma_1^{\tau_x}, \gamma_2^{\tau_x}, \ldots \rangle \leftarrow$ sort the candidate non-key attributes
       $\gamma_j^{\tau_x} \in \Gamma^{\tau_x}$ by their scores $S^{\tau_x}(\gamma_j^{\tau_x})$;
3  for $x \leftarrow 1$ to $K$ do
4      for $i \leftarrow 1$ to $\min(k, x)$ do
5          for $j \leftarrow i$ to $n$ do
6              $\mathcal{P}_{opt}(i, j, x) \leftarrow \mathcal{P}_{opt}(i, j, x-1)$;
7              for $m \leftarrow 1$ to $min(j-i+1, |\Gamma^{\tau_x}|)$ do
8                  $T_x^m.key \leftarrow \tau_x$;
9                  $T_x^m.nonkey \leftarrow$ top-$m$ candidate non-key
                   attributes in $\Gamma^{\tau_x}$;
10                 $\mathcal{P} \leftarrow \mathcal{P}_{opt}(i-1, j-m, x-1) \bigcup \{T_x^m\}$;
11                 if $S(\mathcal{P}) > S(\mathcal{P}_{opt}(i, j, x))$ then
12                     $\mathcal{P}_{opt}(i, j, x) \leftarrow \mathcal{P}$;
13 $\mathcal{P}_{opt} \leftarrow \mathcal{P}_{opt}(k, n, K)$;
14 return $\mathcal{P}_{opt}$;
```

5.3 An Apriori-style Algorithm for Tight / Diverse Preview Discovery Problem

Since the dynamic-programming algorithm is inapplicable when previews must satisfy distance constraint, we propose an efficient algorithm for optimal tight/diverse preview discovery, shown in Algorithm 3. It consists of two steps—(1) finding k-subsets of entity types (i.e., vertices in G_s) satisfying the distance constraint (Lines 1– 14) and (2) for each qualifying k-subset of entity types, forming a preview under the size constraint, computing its score and choosing a preview with the highest score (Lines 15– 20).

The first step is essentially finding k-cliques in a graph converted from the schema graph G_s, in which vertices are considered adjacent if they are within distance d (for tight previews) or apart by at least distance d (for diverse previews). The k-clique problem is well-studied and many efficient algorithms have been designed in the past. Our method is inspired by the well-known Apriori algorithm [1] for frequent itemset mining. In [9], an algorithm was proposed for finding k-cliques (where edges correspond to metabolite correlations) by similar ideas, although the connection to Apriori was not made. Their experimental results demonstrated superior efficiency in comparison with the more well-known Bron-Kerbosch algorithm [5]. Nevertheless, the two broad steps of our optimal tight/diverse preview discovery algorithm are independent from each other, and thus any more efficient or even approximate algorithm for finding k-cliques can be plugged into it to further improve its execution efficiency.

In more details, the first step of Algorithm 3 iteratively generates a k-subset of entity types by merging two $(k-1)$-subsets. Entity types are arbitrarily ordered as $\tau_1, \ldots, \tau_K$. In the i-th iteration of the algorithm, if two $(i-1)$-subsets A and B only differ by their last entity types $\tau_{A[i-1]}$ and $\tau_{B[i-1]}$, and the distance between their last entity types satisfies the distance constraint, a candidate i-subset is generated by appending $\tau_{B[i-1]}$ to the end of A.

In the second step, for each candidate k-subset of entity types, a preview is computed ($ComputePreview(A)$ in Line 17 of Algorithm 3). The details of function $ComputePreview$ are omitted. It follows Theorem 3 and is essentially the same as Lines 5– 14 in

Algorithm 3: Apriori-style Algorithm for Optimal Tight/Diverse Preview Discovery

```
Input  : schema graph $G_s$, size constraint$(k, n)$, distance constraint $d$
Output: an optimal tight/diverse preview $\mathcal{P}_{opt}$
$\mathcal{L}_2 \leftarrow \varnothing$;
foreach $i \leftarrow 1$ to $K$ do
    foreach $j \leftarrow i+1$ to $K$ do
        if $dist(\tau_i, \tau_j) \leq d$ then    /* $\geq d$ for diverse preview */
            $\mathcal{L}_2 \leftarrow \mathcal{L}_2 \cup \{\langle i\ j\rangle\}$;
$i \leftarrow 3$;
while $i \leq k$ and $\mathcal{L}_{i-1} \neq \varnothing$ do
    $\mathcal{L}_i \leftarrow \varnothing$;
    foreach $A, B \in \mathcal{L}_{i-1}$ s.t. $(\forall j < i-1 : A[j] = B[j])$ and $(A[i-1] < B[i-1])$ do
        /* $\geq d$ for diverse preview */
        if $dist(\tau_{A[i-1]}, \tau_{B[i-1]}) \leq d$ then
            $\mathcal{L}_i \leftarrow \mathcal{L}_i \cup \{\langle A[1] \ldots A[i-1]\ B[i-1]\rangle\}$;
    $i \leftarrow i+1$;
if $\mathcal{L}_k = \varnothing$ then
    return $\varnothing$;
$max\_score \leftarrow 0$;
foreach $A \in \mathcal{L}_k$ do
    $\mathcal{P} \leftarrow ComputePreview(A)$;
    if $score(\mathcal{P}) > max\_score$ then
        $max\_score \leftarrow score(\mathcal{P})$;
        $\mathcal{P}_{opt} \leftarrow \mathcal{P}$;
return $\mathcal{P}_{opt}$;
```

Algorithm 1, which is described in Section 5.1. The score of each preview is computed (details also the same as in Lines 5– 14 of Algorithm 1) and a preview with the highest score is returned.

6. EVALUATION

We conducted experiments to evaluate the preview scoring measures' accuracy (Section 6.1), the preview discovery algorithms' efficiency (Section 6.2) as well as the overall quality of discovered previews (Section 6.3). All experiments were conducted on a DELL T100 server running Ubuntu 8.10. The server has Dual Core Xeon E3120 processors, 6MB cache, 4GB RAM, and two 250GB RAID1 SATA hard drivers. The entity graph used in our experiment is a dump of Freebase at September 28, 2012.[8] The dataset is imported into a MySQL database. All algorithms are implemented in C++ and compiled with '-O2' optimization in GCC-4.3.2.

In Freebase, the entire entity graph is partitioned into many domains. We pre-computed the schema graphs as well as all scoring measures in Section 3 for several domains of different sizes and used these schema graphs in our evaluation. Both a schema graph and the scoring measures of its vertices and edges can be incrementally updated (details omitted). Our work currently is limited to named entities, thus all numeric attribute values from the data dump have been removed. Note that a schema graph may be disconnected. To ensure the convergence of random walk process in such a graph, we added a small transition probability 10^{-5} to every pair of entity types.

6.1 Accuracy of Preview Scoring Measures

We conducted two experiments to evaluate the accuracy of the scoring measures for both key and non-key attributes presented in Section 3. One experiment compares the ranking orders of candidate key (non-key) attributes by the scoring measures with the gold standard ranking orders obtained from Freebase.com. The other experiment calculates the correlation between the pairwise order between candidate key (non-key) attributes by the scoring measures and the pairwise order collected from user study using crowd-sourcing service.

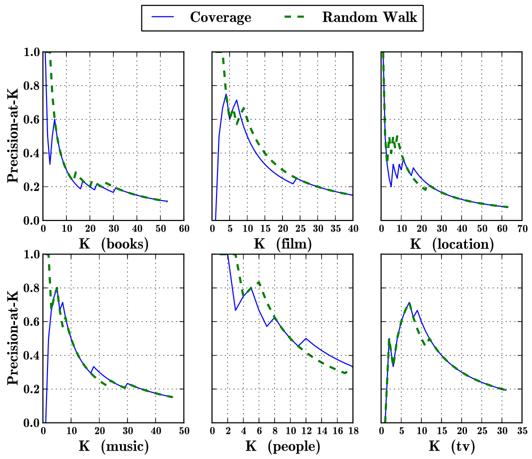

Figure 5: Precision-at-K of Key Attribute Scoring.

Domain	Coverage	Entropy	Domain	Coverage	Entropy
books	0.8	0.786	music	0.528	0.589
film	0.2	0.25	people	0.708	0.606
location	0.55	0.592	tv	0.622	0.379

Table 2: Mean Reciprocal Rank of Non-Key Attribute Scoring.

6.1.1 *Comparison with Gold Standard*

We collected gold standard data for the 6 largest entity domains in Freebase—"books", "film", "location", "music", "people" and "tv". For each domain, Freebase offers an entrance page showing 6 major entity types in that domain. A user can choose to browse entities in any of the 6 types. [9] As such entrance pages were manually created by Freebase, our conjecture is that they are of high quality and reflect the most popular entity types. We thus treated the 6 entity types listed in the entrance page of a domain as the gold standard for top-6 key attributes in that domain.

For both the coverage-based and the random-walk based scoring measures in Section 3.2, we ranked all candidate key attributes by their scores. We calculated the accuracy of a scoring measure by several widely-used IR evaluation measures, including Precision-at-K (P@K), Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (nDCG) [10]. Since they demonstrate similar results, we only report P@K due to space limitations. For a scoring measure for key attributes, P@K is the percentage of its top-K results that belong to the aforementioned gold standard top-6 key attributes. The results are shown in Figure 5. For both the coverage-based and the random-walk based scoring measures, P@10 is above 0.4 in 5 out of the 6 domains, which means the top-10 results contain at least 4 of the 6 gold standard key attributes.

For each entity type, Freebase offers a table for users to browse and query the entities belonging to that type. [10] Regardless of the entity type, that table always has 3 common columns for recording names, types and article contents of entities. The table also has 3 or less type-dependent non-key attributes manually selected by Freebase editors. Although Freebase allows users to add more at-

[8] https://developers.google.com/freebase/data

[9] As of September 28, 2012, the entrance pages to these 6 domains were all under "Featured Data" on Freebase.com. For instance, http://www.freebase.com/view/film was the entrance page for domain "film". These pages have become unavailable.

[10] http://www.freebase.com/music/artist?instances=, for instance, would display a table for type ARTIST in "music" domain.

Domain	Coverage	Random Walk	Domain	Coverage	Random Walk
books	0.55	0.43	music	0.33	0.46
film	0.48	0.25	people	0.31	0.29
location	-0.17	-0.08	tv	0.69	0.65

Domain	Coverage	Entropy	Domain	Coverage	Entropy
books	0.43	0.43	music	0.42	0.41
film	0.35	0.35	people	0.43	0.43
location	0.20	0.21	tv	0.47	0.47

Table 3: Pearson Correlation Coefficient for Key Attribute Scoring (Upper) and Non-Key Attribute Scoring (Lower).

tributes into this table, we believe that the original 3 type-dependent attributes in general bear higher quality. We thus treated these attributes as the gold standard for top non-key attributes for that entity type.

For both the coverage-based and the entropy-based scoring measures in Section 3.3, we ranked all candidate non-key attributes by their scores. We calculated the accuracy of a scoring measure by Mean Reciprocal Rank (MRR) [10] instead of P@K as there are only 3 or less gold standard answers for top non-key attributes in each entity type. For a scoring measure for non-key attributes, the reciprocal rank is the multiplicative inverse of the rank of the first gold standard non-key attribute among its ranking results. MRR is the average reciprocal rank across all entity types with at least 5 candidate non-key attributes. (If an entity type has only less than 5 candidates, the gold standard answers are ranked deceptively high. Thus we exclude such entity types, to obtain more accurate evaluation.) The results are shown in Table 2. In every domain except "film" and for both the coverage-based and the entropy-based measures, MRR is above 0.5. This means in average a gold standard non-key attribute appeared in the top-2 ranked results. The lower MRR for "film" domain is from only one entity type and thus is not truly indicative, since only one entity type in that domain has at least 5 candidate non-key attributes.

6.1.2 User Study

We conducted an extensive user study in Amazon Mechanical Turk (AMT)[11]—a popular crowdsourcing service—and measured the correlation between our scoring measures and users' opinions with regard to key and non-key attributes ranking. We explain the user study procedure for evaluating key attribute ranking in one domain, since the procedure is repeated for all 6 gold standard domains and is the same for both key and non-key attribute ranking.

Given a domain, we randomly generated 50 pairs of entity types, i.e., candidate key attributes. Each pair was presented to 20 AMT workers. The workers were asked which of the 2 entity types in the pair is more important. Hence, we collected $1,000$ opinions in total. We then constructed two lists—X and Y, each of which contains 50 values corresponding to the 50 pairs. A value in X represents the difference in the ranking positions (by our scoring measures) of the two entity types in the corresponding pair. A value in Y represents the difference in the numbers of AMT workers favoring the two entity types. The correlation between X and Y is measured by Pearson Correlation Coefficient (PCC) [6] as follows.

$$PCC = \frac{\mathrm{E}(XY) - \mathrm{E}(X)\mathrm{E}(Y)}{\sqrt{\mathrm{E}(X^2) - (\mathrm{E}(X))^2}\sqrt{\mathrm{E}(Y^2) - (\mathrm{E}(Y))^2}} \qquad (4)$$

The PCC value ranging from -1 to 1 indicates the degree of correlation between the pairwise ranking orders produced by our scoring methods and the pairwise preferences given by AMT workers. A PCC value in the ranges of [0.5,1.0], [0.3,0.5) and [0.1,0.3) indicates a strong, medium and small positive correlation, respectively. The PCC values for the 6 gold standard domains are shown in Tables 3. For 5 out of the 6 domains, the results show at least a medium positive correlation between our scoring measures and AMT workers. For domain "location", small positive correlations are shown for non-key attribute scoring and even a negative correlation was obtained for key-attribute scoring. This appears to be largely due to AMT workers' unfamiliarity with many entity types in this domain such as country-specific geographical concepts (e.g., JAPANESE SUBPREFECTURE).

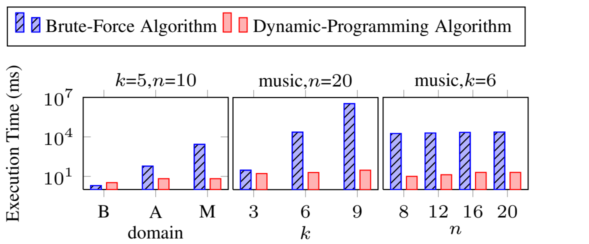

Figure 6: Efficiency Evaluation of Optimal Concise Preview Discovery Algorithms.

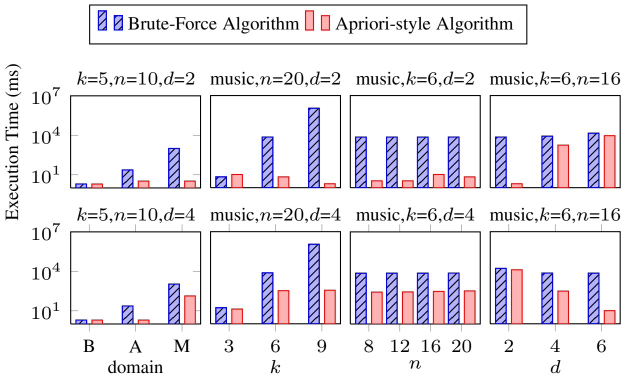

Figure 7: Efficiency Evaluation of Optimal Tight (Upper) and Diverse (Lower) Preview Discovery Algorithms.

6.2 Efficiency of Optimal Preview Discovery Algorithms

This section presents results on the efficiency of the optimal preview discovery algorithms in Section 5. On optimal concise preview discovery, we compared the Brute-Force Algorithm 1 and the Dynamic-Programming Algorithm 2. Specifically, we compared their execution times by varying: (1) size of schema graph (i.e., number of candidate key attributes (K) and number of candidate non-key attributes (N)); (2) number of preview tables (i.e., key attributes) in a preview (k); and (3) maximum number of non-key attributes in a preview (n). For (1), we fixed k=5, n=10 and experimented with 3 domains—"basketball" (B), "architecture" (A), and "music" (M). They differ greatly in the sizes of their schema graphs (B: K=6, N=21; A: K=23, N=48; M: K=46, N=133). For (2), we varied k from 3 to 9, fixed n=20 and used "music" domain. For (3), we varied n from 8 to 20, fixed k=6 and used "music" domain.

On optimal tight/diverse preview discovery, we compared the Brute-Force Algorithm 1 and the Apriori-style Algorithm 3, by varying not only the aforementioned 3 parameters but also the distance constraint on d. When we varied other parameters, d is fixed at 2 and 4 for tight and diverse previews, respectively. When we fixed other parameters, d was varied from 2 to 6.

The results are shown in Figure 6 and Figure 7. In all results, the execution time is averaged across 3 runs, and execution time less than 1 millisecond is rounded to 1 millisecond. The results show that both the Dynamic-Programming and the Apriori-style algorithms outperformed the Brute-Force algorithm in execution time by orders of magnitude in most cases. The exceptions are

[11] https://www.mturk.com/mturk/

Key attributes	Non-key attributes (Target entity types)
Domain="film", KS=Coverage, NKS=Coverage, k=5, n=10	
Film Character	*Portrayed in films* (Film, Film Actor)
Film Actor	*Film performances* (Film, Film Character)
Film	*Performances* (Film actor, Film Character), *Genres* (Film Genre), *Runtime* (Film Cut), *Country of origin* (Country), *Directed by* (Film Director), *Languages* (Human Language)
Film Director	*Films directed* (Film)
Film Crewmember	*Films crewed* (Film, Film crew role)
Domain="music", KS=Random Walk, NKS=Coverage, k=5, n=10	
Musical Recording	*Releases* (Musical Release), *Tracks* (Release Track), *Recorded by* (Musical Artist)
Musical Release	*Tracks* (Musical Recording), *Track list* (Release Track)
Release Track	*Release* (Musical Release), *Recording* (Musical Recording)
Musical Artist	*Tracks recorded* (Musical Recording)
Musical Album	*Releases* (Musical Release), *Release type* (Musical Album Type)
Domain="tv", KS=Random Walk, NKS=Entropy, k=5, n=10	
TV Episode	*Previous episode* (TV Episode), *Next episode* (TV Episode), *Performances* (TV Actor, TV Character), *Season* (TV Season), *Series* (TV Program) , *Personal appearances* (Person, Personal Appearance Role)
TV Program	*Regular acting performances* (TV Actor, TV Character, TV Season)
TV Season	*Episodes* (TV Episode)
TV Actor	*TV episode performances* (TV Episode, TV Character)
TV Director	*TV episodes directed* (TV Episode)

Table 4: Samples of Optimal Concise Previews.

the smallest domain "basketball" and when the number of requested preview tables is small (k=3). In these cases, the overheads of more complex data structures and calculations in the advanced algorithms outweighed their benefits.

Figure 7 shows that the Apriori-style algorithm did not perform well for d=6 in tight preview discovery and d=2 in diverse preview discovery. It is due to the excessive number of candidate k-subsets that satisfy the distance constraint in such cases. For instance, the diameter of a schema graph typically is not large. In the schema graph of "film" domain, the longest path length is 7 and the average path length is around 3–4. Setting distance constraint d=6 in finding tight previews will make most previews "tight". It is unnecessary to enforce such a distance constraint.

6.3 Sample Optimal Previews

To demonstrate the combined effectiveness of both scoring measures and preview discovery algorithms, Table 4 presents the optimal concise previews in 3 selected domains by 3 different combinations of key attribute scoring (KS) and non-key attribute scoring (NKS) measures. The size constraint is set as k=5 and n=10. All result previews show that the selected key and non-key attributes have covered important entity types and their important relationship types. Further, Table 5 shows the optimal tight (d=2) and diverse (d=4) previews in "film" domain by one particular choice of key and non-key attribute scoring measures. We see that, in the tight preview result, the chosen key attributes are all highly related to one entity type Film. In the diverse preview result, the chosen key attributes are far less related to each other. Both verify the effectiveness of the concepts of tight/diverse previews.

Note that in the generated previews, certain non-key attributes represent relationship types involving more than two entity types. An example in Table 4 is *Portrayed in films*, which is a non-key attribute of entity type Film Character. Different from other non-key attribute such as *Films directed*, it represents a 3-way relationship among Film Character, Film and Film Actor. For instance, Agent J is a Film Character played by Film Actor Will Smith in Film Men in Black. To present the values of such a *multi-way* non-key attribute in a preview table, we employ a simple approach of presenting values for all participating entity types in this relationship. It is arguable that this approach widens the preview table, which to some extent violates a given size constraint. An alternative solution is to use separate preview tables for all multi-way relationships. These pose interesting directions for our future work.

Key attributes	Non-key attributes (Target entity types)
Domain="film", KS=Coverage, NKS=Coverage, k=5, n=10, d=2	
Film	*Performances* (Film Character, Film Actor), *Genres* (Film Genre), *Runtime* (Film Cut), *Country of origin* (Country), *Directed by* (Film Director), *Languages* (Human Language)
Film Director	*Films directed* (Film)
Film Producer	*Films produced* (Film)
Film Writer	*Film writing credits* (Film)
Film Editor	*Films edited* (Film)
Domain="film", KS=Coverage, NKS=Coverage, k=5, n=10, d=4	
Film Character	*Portrayed in films* (Film, Film Actor), *Portrayed in films (dubbed)* (Film, Film Actor)
Film Crewmember	*Films crewed* (Film, Film Crew Role)
Person or Entity appearing in Film	*Films appeared in* (Film, Type of Appearance)
Film Festival	*Individual festivals* (Film Festival Event), *Location* (Location), *Focus* (Film Festival Focus), *Sponsoring organization* (Sponser)
Film Company	*Films* (Film)

Table 5: Samples of Optimal Tight (Upper) and Diverse Previews (Lower).

7. RELATED WORK

DataGuide [7] is among the earliest approach to constructing structural summaries for semi-structured databases. Such summary helps query formulation and query processing on semi-structured data. Our work is also closely related to schema summarization ideas for relational databases [16, 17, 18], XML [18] and general graph data [14, 19]. [18] produces schema summarization for relational databases and XML data. Its summary is in the form of a condensed schema tree where a node may correspond to multiple nodes or a trunk in the original schema tree. The notion of summary in [16, 17] refers to clustering the tables in a database by their semantic roles and similarities as well as identifying direct join relationships and indirect join paths between the tables. The graph summarization in [14, 19] groups graph nodes based on their attribute similarity and allows users to browse the summary from different grouping granularities. In Section 1, we explained why these methods are inapplicable or ineffective for producing preview tables from entity graphs, due to different data models in input and output as well as different goals.

There are many works on graph clustering [12]. They are not effective for generating preview tables, since clustering focuses on partitioning but does not present a concise structure. On the contrary, preview tables only contain a small number of key attributes (vertices) and non-key attributes (edges) in a schema graph.

8. CONCLUSION

This paper studies how to generate preview tables for entity graphs. The problem is challenging due to the scale and complexity of such graphs. We proposed effective scoring measures for preview tables. We proved that the optimal preview discovery problem under distance constraint is **NP**-hard. We designed efficient algorithms for discovering optimal previews. Our experiments on Freebase data verified the accuracy and efficiency of our methods.

9. REFERENCES

[1] R. Agarwal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of the 20th VLDB Conference*, pages 487–499, 1994.

[2] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, , and Z. Ives. DBpedia: A nucleus for a Web of open data. In *ISWC*, 2007.

[3] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pages 1247–1250, 2008.

[4] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the seventh international conference on World Wide Web*, pages 107–117, 1998.

[5] C. Bron and J. Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM*, 16(9):575–577, Sept. 1973.

[6] J. Cohen. *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Erlbaum Associates, 1988.

[7] R. Goldman and J. Widom. Dataguides: Enabling query formulation and optimization in semistructured databases. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, VLDB '97, pages 436–445, San Francisco, CA, USA, 1997.

[8] I. Herman, G. Melançon, and M. S. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, Jan. 2000.

[9] F. Kose, W. Weckwerth, T. Linke, and O. Fiehn. Visualizing plant metabolomic correlation networks using clique-metabolite matrices. *Bioinformatics*, 17(12):1198–1208.

[10] C. D. Manning, P. Raghavan, and H. Schtze. *Introduction to Information Retrieval*. Cambridge University Press, NY, USA, 2008.

[11] J. R. Quinlan. Induction of decision trees. *Mach. Learn.*, 1(1):81–106, Mar. 1986.

[12] S. E. Schaeffer. Survey: Graph clustering. *Comput. Sci. Rev.*, 1(1):27–64, Aug. 2007.

[13] F. M. Suchanek, G. Kasneci, and G. Weikum. YAGO: a core of semantic knowledge unifying WordNet and Wikipedia. In *WWW*, pages 697–706, 2007.

[14] Y. Tian, R. A. Hankins, and J. M. Patel. Efficient aggregation for graph summarization. In *SIGMOD Conference*, pages 567–580, 2008.

[15] W. Wu, H. Li, H. Wang, and K. Q. Zhu. Probase: a probabilistic taxonomy for text understanding. In *SIGMOD*, pages 481–492, 2012.

[16] X. Yang, C. M. Procopiuc, and D. Srivastava. Summarizing relational databases. *PVLDB*, 2(1):634–645, 2009.

[17] X. Yang, C. M. Procopiuc, and D. Srivastava. Summary graphs for relational database schemas. *PVLDB*, 4(11):899–910, 2011.

[18] C. Yu and H. V. Jagadish. Schema summarization. In *VLDB*, pages 319–330, 2006.

[19] N. Zhang, Y. Tian, and J. M. Patel. Discovery-driven graph summarization. In *ICDE*, pages 880–891, 2010.