TableView: A Visual Interface for Generating Preview Tables of Entity Graphs

Sona Hasani
University of Texas at Arlington
sona.hasani@mavs.uta.edu

Ning Yan*
Huawei U.S. R&D Center
ning.yan.uta@gmail.com

Chengkai Li
University of Texas at Arlington
cli@uta.edu

***Abstract*—This paper introduces TableView, a tool that helps generate preview tables of massive heterogeneous entity graphs. Given the large number of datasets from many different sources, it is challenging to select entity graphs for a particular need. TableView produces preview tables to offer a compact presentation of important entity types and relationships in an entity graph. Users can quickly browse the preview in a limited space instead of spending precious time or resources to fetch and explore the complete dataset that turns out to be inappropriate for their needs. In this paper we present: 1) a detailed description of TableView's graphical user interface and its various features, 2) a brief description of TableView's preview scoring measures and algorithms, and 3) the demonstration scenarios we intend to present to the audience.**

***Keywords*-entity graph; knowledge graph; usability; summarization**

I. Introduction

There is a revolutionary increase of gigantic, heterogeneous *entity graphs* that represent entities and their relationships in various domains. An *entity graph* is a directed multi-graph where each vertex represents an entity and each edge represents a directed relationship from its source entity to its destination entity. Entity graphs are often represented as Resource Description Framework (RDF) triples. Some of the most salient real-world entity graphs are knowledge bases such as DBpedia [1], YAGO [2], Probase [3], Freebase [4], and Google's Knowledge Vault [5]. Numerous applications use entity graphs in areas such as search, recommendation systems, business intelligence, and health informatics. An example of a very small entity graph is displayed in Fig. 1.

Due to the increasing number of entity graphs from many sources and oftentimes inadequate information available about them, it is extremely challenging to select the one that satisfies a particular need. A data worker may need to tackle many challenges before she can start any real work on an entity graph. While some descriptions may be provided by the data sources, the data worker is not able to get a direct look at an entity graph before fetching it. Moreover, downloading a dataset and loading it into a database can be a daunting task.

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, there are 190K vertices and 1.6M edges in a snapshot of the film domain of Freebase, and the corresponding schema graph consists of 50 entity types (vertices) and 136 relationship types (edges). Several studies have suggested to generate a summary of the schema graph, by schema summarization techniques. Some of these methods work on relational and semi-structured data instead of graph data [6], [7], [8], while others produce trees or graphs as output instead of flat tables [8], [9], [10]. The most closely related work to our approach clusters the tables in a database but does not reduce the complexity of database schema [6], [7].

Yan et al. [11] introduced *preview tables* to assist data workers in obtaining a quick preview of a given entity graph before they decide to spend more time and resources to fetch and investigate the complete graph. They show a few randomly selected sample tuples in each preview table to facilitate a better understanding of the data graph. In order to select the best preview, they proposed a scoring system, including coverage-based and random-walk based scoring methods for entity types and coverage-based and entropy-based scoring methods for relationships. In this paper we present TableView, a tool based on [11] that automatically produces preview tables for entity graphs.

Fig. 2 is a possible preview of the entity graph in Fig. 1. It consists of two preview tables. The upper table includes attributes Film, Director and Genres, and the lower table has attributes Film Actor and Award Winners. Film and Film Actor are the *key attributes* of the generated preview tables, marked by the underlines beneath them. Attributes Director and Genres in the upper table are considered highly related to Film entities. The two tables contain four and two sample tuples, respectively.

II. User Interface

In this section we describe various features implemented in TableView. Fig. 3 presents its graphical user interface which consists of four main segments: configuration panel, preview panel, information panel, and manually generated preview tables. Below we describe these components.

Configuration Panel: This panel is designed for setting the parameters of the preview. The user can customize the following fields. *Domain*: The user can select among several domains provided by the dataset or pre-processed for the dataset. *Key attributes*: The user will specify the desired number of key attributes which determines the number of tables in the preview. *Non-key attributes*: The user will specify the number of non-

*The work was performed while the author was with UT-Arlington.

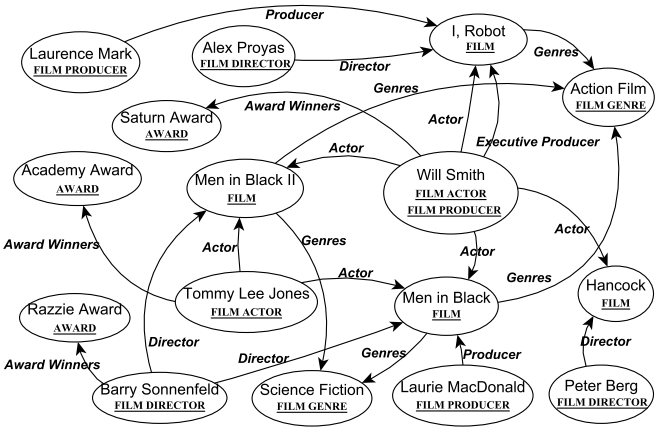

Fig. 1. An excerpt of an entity graph.

	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

Fig. 2. A preview of the entity graph in Fig. 1.

key attributes in the preview which indicates the total number of non-key attribute columns of the preview tables. *Preview type*: The user can choose their desired type of preview from a drop-down list of three options. A *concise* preview is the optimal preview with the specified number of key and non-key attributes. A *tight* preview is a concise preview where the distance between every two key attributes of the preview is at most equal to some specific value specified by the user. On the other hand, a *diverse* preview is a concise preview where the distance between every two key attributes in the preview is at least equal to a distance specified by the user. *Distance*: If the user selects tight or diverse preview as the type, this field will show up and the user can specify the desired distance between key attributes in the preview. *Number of the records*: The user can specify how many sample records she would like to see in each preview table. *Attribute scoring*: For key attribute scoring the user can choose between coverage-based scoring and random-walk based scoring, while for non-key attribute scoring she can select coverage-based scoring and entropy-based scoring.

Preview Panel: When the configuration parameters are set, the user needs to click the Show Preview button to see the automatically generated preview in the preview panel. This panel has three segments, including preview tables, schema graph, and summarized schema graph. The user can choose to see the preview tables and the schema graph side by side. If the user clicks on any column's header, the corresponding node or edge in the graph is highlighted and the user can have a better understanding of the surrounding entities of that particular item. The user can hide, display or delete the sample records in each preview table. Moreover, each column can be renamed and non-key attribute columns can be reordered. The user can also rearrange the preview tables by dragging each table to a desired location. In this panel, the user can choose to see the complete schema graph, or the summarized schema graph which is the sub graph of the schema graph corresponding to the generated preview.

Information Panel: When the user clicks on any item in the preview panel, additional information for that item is displayed in the information panel. For example, when the user clicks on a key attribute in a preview table, or a node in a schema graph, such additional information includes entity type, ID, and number of the entities in the dataset that belong to the marked entity type. Similarly, if the user clicks on a non-key attribute of a preview table or an edge of a schema graph, additional information such as relationship type, ID, source and destination entity types is presented.

Manually Generated Preview Tables: TableView enables data experts to handcraft their own preview tables and customize them. Fig. 4 shows the graphical user interface for handcrafting preview tables. After selecting the dataset, domain, number of key attributes, and number of non-key attributes, the user can manually design her own preview. The schema graph of the selected domain and the URL to the dataset are provided to the user. Based on the selected dataset and domain, a list of key attributes under the selected domain are shown to the user. When the user chooses a key attribute, the list of relationships to or from that key attribute are presented to her. Each preview table must contain one key attribute and at least one non-key attribute. Once the user selects a non-key attribute for the chosen key attribute, a new table is added to the preview with the selected key and non-key attributes. The user can add more non-key attributes to any of generated preview tables as long as the total number of the non-key attributed is not over the specified limit. The numbers of remaining key and non-key attributes are updated at each step. The user is able to delete the key and non-key attributes from the partially generated preview. When the preview is ready, the user can export the preview in PDF format.

III. System Overview

A *preview* is a set of preview tables, where each table has a *key attribute*, corresponding to an entity type, and a set of *non-key attributes*, each corresponding to a relationship between the key attribute and another entity in the entity graph. There is thus a large space of possible previews for a given schema graph. The preview is designed to help the users attain a quick understanding of the entity graph, therefore, it must fit into a limited display space. This desideratum is captured by a constraint on the size of the preview, in terms of the numbers of key and non-key attributes. Furthermore, in order to control how tight or diverse the preview can be an additional constraint on the pairwise distance between preview

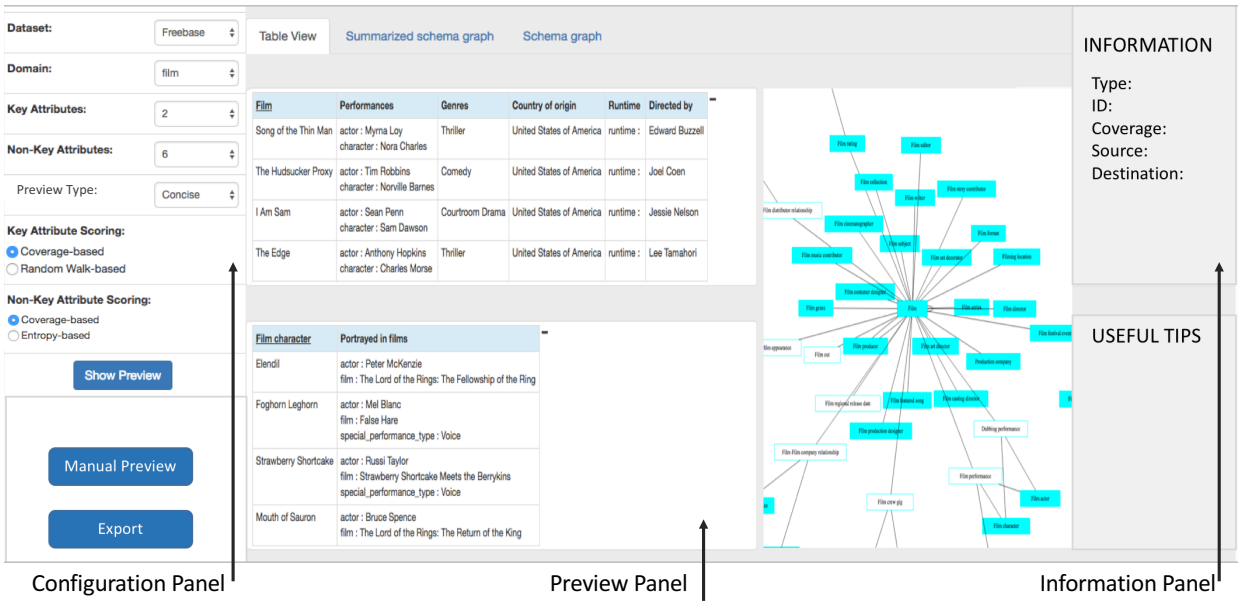

Fig. 3. User interface of TableView.

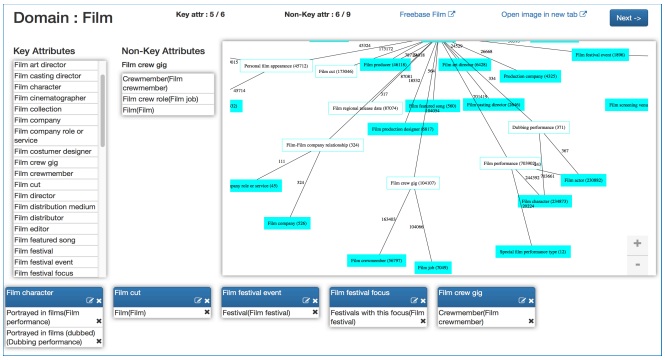

Fig. 4. User interface for handcrafted preview tables.

tables is considered. We formulate the problem as finding a preview with the highest score among those possible previews satisfying the size and distance constraints. In the following we briefly explain the scoring measures used in TableView.

A. Scoring Measures and Algorithms

The score of a preview is simply aggregated from individual preview tables' scores, by summation. The score of each preview table is calculated from the product of its key attribute's score and the summation of its non-key attributes' scores.

For key attribute scoring we implemented coverage-based and random-walk based methods. 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., $\tau \in V_s$. The coverage-based scoring measure defines the score of τ as the number of entities that belong to that type:

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

In order to calculate the random-walk based score of a key attribute, 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 [12]. The edge between τ_i and τ_j in G is weighted by the number of relationships in the entity graph between entities of types τ_i and τ_j.

For non-key attributes scoring we implemented coverage-based and entropy-based methods. The coverage-based scoring measure defines the score of a non-key attribute γ as the number of relationships that belong to that type:

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

To calculate the entropy-based score of a non-key attribute, we measure the value of a non-key attribute $\gamma(\tau, \tau')$ (or $\gamma(\tau', \tau)$)—the edge between entity types τ and τ' in the schema graph—by the *entropy* of γ ($H(\gamma)$) to estimate the amount of information it provides to T.

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

where n_j is the number of tuples in T that have the same jth attribute value u on non-key attribute $\gamma(\tau, \tau')$ (or $\gamma(\tau', \tau)$). $|t.\gamma|$ is the number of tuples in T with non-empty values on $\gamma(\tau, \tau')$ (or $\gamma(\tau', \tau)$).

We developed a dynamic programming algorithm for finding the optimal concise preview and an Apriori-style algorithm for finding optimal tight/diverse previews based on these scoring measures. Interested readers can find more details in Yan et al. [11].

B. System Implementation

The algorithms in TableView are implemented in C++ and Python 2.7. The system is hosted on a Dell T100 server with

TABLE I
SIZES OF ENTITY/SCHEMA GRAPHS.

Domain	# of vertices	# of edges
book	6M / 91	15M / 201
film	2M / 63	18M / 136
music	27M / 69	187M / 176
TV	2M / 59	17M / 177
people	3M / 45	17M / 78

Dual Core Xeon E3120 processor, 6MB cache, 4GB RAM, and two 250GB RAID1 SATA hard drives running Ubuntu 8.10. We used Django for implementing the web interface and Javascript, HTML, and CSS for parsing the results on the client side. For graph visualization, we used D3.js.

The entity graph used in our demonstration is a dump of Freebase on September 28, 2012. The dataset is imported into a MySQL database. We can use the same approach for generating the preview tables for any entity graph that provides the type information for entities and relationship. In Freebase, the entire entity graph is partitioned into several domains. We pre-processed five of its domains, including film, book, music, TV, and people. Table I presents the size of each domain.

IV. DEMONSTRATION PLAN

In this section we describe a few scenarios that we intend to present to the audience.

A: Automatically generating preview tables
Using the following steps, the user will be able to automatically generate a preview. A1) Select Freebase from the dataset drop-down list. A2) Select one of the domains from the domain drop-down list. A3) Select the number of the key-attribute. A4) Select the number of the non-key attributes. A5) Select the key-attribute scoring method. A6) Select the non-key attribute scoring method. A7) Click the Show Preview button to see the automatically generated preview in the preview panel.

B: Interacting with the generated preview
The following steps will show the user how to interact with the generated preview in scenario A. B1) Delete a preview table. B2) Delete one of the non-key attributes of a preview table. B3) Refresh the preview to replace the deleted key and non-key attributes. B4) Delete a sample record from one of the tables in the preview. B5) Hide and show the sample records in a table. B6) Rearrange the tables. B7) Rearrange the non-key attributes of the tables. B8) Rename non-key attributes.

C: Additional information
The following steps will show the user how to find the preview tables in the schema graph and look up the additional information about key attributes and non-key attributes in the schema. C1) Click on the key attribute or non-key attributes of a preview table and check the highlighted nodes and/or edges in the schema graph. C2) Click on any column on the preview tables and check the additional information in the information panel. C3) Click the Export button to generate a PDF file of the preview.

D: Generating a preview manually
The following steps will guide the user towards handcrafting a preview for a given schema graph. D1) Select one of the domains in the domain drop-down list. D2) Select the numbers of key and non-key attributes. D3) Click the Manual Preview button. D4) Select a key attribute from the key attribute list. D5) Select non-key attributes from the non-key attribute list. D6) Check the schema graph and the numbers of remaining key attributes and non-key attributes. D7) Delete one of the non-key attributes from the table. D8) Delete the key attribute of a table. D9) Design and customize the complete preview by selecting the key attributes and corresponding non-key attributes. D10) Click on each table and check the corresponding sub-graph in the schema graph. D11) Export the preview in PDF format.

In addition to TableView, we implemented the summarization algorithm from Yang et al. [6]. We will demonstrate the schema summary of Freebase generated by their algorithm and compare it with the preview tables produced by TableView.

ACKNOWLEDGMENTS

The authors have been partially supported by NSF grants 1408928, 1719054 and NSF-China grant 61370019. Any opinions, findings, and conclusions in this publication are those of the authors and do not necessarily reflect the views of the funding agencies. We also thank Rudresh Ajgaonkar and Aaditya Kulkarni for their contributions.

REFERENCES

[1] 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, pp. 722–735.
[2] F. M. Suchanek, G. Kasneci, and G. Weikum, "YAGO: a core of semantic knowledge unifying WordNet and Wikipedia," in *WWW*, 2007, pp. 697–706.
[3] W. Wu, H. Li, H. Wang, and K. Q. Zhu, "Probase: a probabilistic taxonomy for text understanding," in *SIGMOD*, 2012, pp. 481–492.
[4] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *SIGMOD*, 2008, pp. 1247–1250.
[5] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang, "Knowledge vault: A web-scale approach to probabilistic knowledge fusion," in *KDD*, 2014, pp. 601–610.
[6] X. Yang, C. M. Procopiuc, and D. Srivastava, "Summarizing relational databases," *PVLDB*, vol. 2, no. 1, pp. 634–645, 2009.
[7] X. Yang, C. M. Procopiuc, and D. Srivastava, "Summary graphs for relational database schemas," *PVLDB*, vol. 4, no. 11, pp. 899–910, 2011.
[8] C. Yu and H. V. Jagadish, "Schema summarization," in *VLDB*, 2006, pp. 319–330.
[9] Y. Tian, R. A. Hankins, and J. M. Patel, "Efficient aggregation for graph summarization," in *SIGMOD*, 2008, pp. 567–580.
[10] N. Zhang, Y. Tian, and J. M. Patel, "Discovery-driven graph summarization," in *ICDE*, 2010, pp. 880–891.
[11] N. Yan, S. Hasani, A. Asudeh, and C. Li, "Generating preview tables for entity graphs," in *SIGMOD*, 2016, pp. 1797–1811.
[12] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," in *WWW*, 1998, pp. 107–117.