NOVEL METHODS FOR ENTITY-CENTRIC INFORMATION EXPLORATION

by

NING YAN

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2013

I leave no trace of wings in the air, but I am glad I have had my flight.

— Fireflies by Rabindranath Tagore (1928)

ACKNOWLEDGEMENTS

I would like to thank my supervising professor Dr. Chengkai Li for his continuous patience of motivating and discussing over various research topics. I feel benefited from not only his rigorous reasoning logic but also the correct attitude towards academic research. The concepts and habits accumulated these years would be a fortune for a lifetime.

I would also like to thank Dr. Gautam Das, Dr. Leonidas Fegaras, and Dr. Ramez Elmasri for serving in my dissertation committee and for providing invaluable advice throughout the progress of my doctoral studies.

I would like to thank my lab-mates in Innovative Database and Information Systems Research (IDIR) Lab — Xiaonan Li, Nandish Jayaram, Naeemul Hassan, Afroza Sultana, Gensheng Zhang, Abolfazl Asudeh, Wei Xiang, for collaborating on research projects and for gaming together.

Finally, I would like to thank my wife Lingjia Gong, who has been taken care of my life and has always been supportive for what I am doing.

November 21, 2013

ABSTRACT

NOVEL METHODS FOR ENTITY-CENTRIC INFORMATION EXPLORATION

NING YAN

The University of Texas at Arlington, 2013

Supervising Professor: Chengkai Li

We witness an unprecedented proliferation of entity graphs that capture entities (e.g., persons, products, organizations) and their relationships. Such entity data gradually evolves into the most comprehensive knowledge graph ever built by human beings. The flourish of entity data also boosts a growing demand for entity-centric information exploration tasks. Different from traditional information retrieval tasks that are based on statistics of text terms, entity-centric information exploration tasks are usually based on metrics over entities, their relationships, or graph structures.

In this work, we identified two essential tasks in entity-centric information exploration: 1) how to do faceted navigation over a set of homogeneous entities, utilizing entity relationships and concept hierarchies; 2) how to assist users in attaining a quick and rough preview of huge entity graphs.

As an approach for the first problem, we proposed a novel method that dynamically discovers a query-dependent faceted interface for a set of Wikipedia articles resulting from a keyword query. We further extended this method into a general framework for faceted interface discovery for Web documents. Our model leverages the collaborative vocabularies in Wikipedia, such as its category hierarchy and intensive internal hyperlinks, for building

faceted interfaces. We proposed metrics for ranking both individual facet hierarchies and faceted interfaces (each with k facet hierarchies). We then developed faceted interface discovery algorithms that optimize these ranking metrics.

As an approach for the second problem, we proposed a novel method that generates preview tables for entity graphs. The preview tables consist of important entities and relationships from entity graphs, which can help users quickly understand such graphs. We studied various scoring measures for the goodness of preview tables. Based on these scoring measures, we formulated several optimization problems that look for the optimal previews with the highest aggregated scores, under various constraints on preview size and distance between preview tables. We proved that the optimization problems under distance constraint are NP-complete. We then designed a dynamic-programming algorithm and an Apriori-style algorithm for finding optimal previews.

For both problems, we conducted experiments as well as user studies for evaluating our proposed models, ranking measures, and algorithms. The results demonstrated the efficiency and effectiveness of our proposed methods for discovering query-dependent faceted interfaces and generating preview tables.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iv

ABSTRACT v

LIST OF ILLUSTRATIONS x

LIST OF TABLES xii

Chapter Page

1. INTRODUCTION 1

2. DYNAMIC DISCOVERY OF QUERY-DEPENDENT FACETED INTERFACES FOR WEB DOCUMENTS 9

2.1 A Generic Model for Faceted Interfaces and its Instantiation for Faceted Search over Web Documents 17

2.1.1 A Generic Model of Faceted Interfaces 18

2.1.2 Instantiation of the Generic Model for Faceted Interfaces over Web Documents 20

2.1.3 Definitions of Concepts and Faceted Interface Discovery Problem . 23

2.2 Facet Ranking 26

2.2.1 Single-Facet Ranking 27

2.2.2 Multi-Facet Ranking 31

2.3 Algorithms 33

2.3.1 Relevant Category Hierarchy (Algorithm 1) 34

2.3.2 Ranking Single Facet (Algorithm 2 and 3) 37

2.3.3 Searching for k-Facet Interface (Algorithm 4) 40

2.4 System Implementation 41

2.4.1 Facetedpedia . . . 41
2.4.2 Facetednews . . . 44
2.5 Evaluation . . . 45
2.5.1 User Studies . . . 46
2.5.2 Characteristics of Generated Faceted Interfaces . . . 51
2.5.3 Efficiency Evaluation . . . 53
3. GENERATING PREVIEW TABLES FOR ENTITY GRAPHS . . . 55
3.1 Preview Discovery Problem . . . 57
3.2 Scoring Measures for Previews . . . 61
3.2.1 Preview Scoring . . . 62
3.2.2 Key Attribute Scoring . . . 62
3.2.3 Non-Key Attribute Scoring . . . 64
3.3 Optimal Previews under Size and Distance Constraints . . . 66
3.4 Algorithms . . . 72
3.4.1 A Brute-Force Algorithm . . . 72
3.4.2 A Dynamic-Programming Algorithm for Concise Preview Discovery Problem . . . 75
3.4.3 An Apriori-style Algorithm for Tight / Diverse Preview Discovery Problem . . . 77
3.5 Evaluation . . . 80
3.5.1 Accuracy of Preview Scoring Measures . . . 80
3.5.2 Efficiency of Optimal Preview Discovery Algorithms . . . 85
3.5.3 Sample Optimal Previews . . . 89
4. RELATED WORK . . . 91
4.1 Faceted Search Systems: A Comparative Study . . . 91
4.2 Other Related Work to Faceted Search . . . 94

4.3 Related Work to Generating Preview Tables for Entity Graphs 96
5. CONCLUSION 98
REFERENCES 99
BIOGRAPHICAL STATEMENT 106

LIST OF ILLUSTRATIONS

Figure Page

2.1 The faceted search interfaces of Newegg.com and NCSU library catalog. . . 10

2.2 The faceted search interface of Facetedpedia 11

2.3 Examples of exploring Facetedpedia. 12

2.4 The faceted search interface of Facetednews. 14

2.5 The generic model for faceted interfaces. 18

2.6 Instantiations of the generic faceted interface model for different scenarios. . 19

2.7 Instantiation of the generic model for Facetedpedia. 22

2.8 The concept of facet for documents. 24

2.9 The navigation on a 2-facet interface. 27

2.10 Navigational costs of facets. 30

2.11 The sequences of navigational steps. 32

2.12 The architecture of Facetedpedia. 41

2.13 The architecture of Facetednews. 44

2.14 Root Categories in Facetedpedia Sample Interfaces. 47

2.15 Root Categories in Facetednews Sample Interfaces 47

2.16 Average ratings of compared systems for 12 queries. 50

2.17 Average ratings of compared systems for 3 general questions. 51

2.18 Characteristics of faceted interfaces produced in Facetedpedia. 51

2.19 Characteristics of faceted interfaces produced in Facetednews. 52

2.20 Execution time of Facetedpedia and Facetednews. 53

3.1 An Excerpt of an Entity Graph. 56

3.2 A Two-Table Preview of the Entity Graph in Figure 3.1. 57
3.3 The Schema Graph for the Entity Graph in Figure 3.1. 57
3.4 Construction of Graphs for Reduction from the Clique Problem to the Optimal Diverse Preview Discovery Problem. 71
3.5 Precision-at-K of Key Attribute Scoring. 81
3.6 Efficiency Evaluation of Optimal Concise Preview Discovery Algorithms. . . 86
3.7 Efficiency Evaluation of Optimal Tight and Diverse Preview Discovery Algorithms. 86
4.1 Taxonomies of faceted search systems. 93

LIST OF TABLES

Table		Page
2.1	Characteristics of the Wikipedia dataset.	42
2.2	Querie Keywords for Evaluation.	48
3.1	Notations	58
3.2	Mean Reciprocal Rank of Non-Key Attribute Scoring.	81
3.3	Pearson Correlation Coefficient for Key Attribute Scoring and Non-Key Attribute Scoring.	84
3.4	Samples of Optimal Concise Previews.	88
3.5	Samples of Optimal Tight and Diverse Previews.	89

CHAPTER 1

INTRODUCTION

We witness in many domains an unprecedented proliferation of *entity graphs* that capture entities (e.g., persons, products, organizations) and their relationships. Sometimes there are also concept (or category) hierarchies over entity graphs describing type information of entities. Real-world entity graphs include knowledge bases (e.g., DBpedia [1], YAGO [2], Probase [3] and Freebase [4], 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. Entity graphs are often represented as RDF triples, due to heterogeneity of entities and the often lacking schema in data. The Linking Open Data [2] community has interlinked billions of RDF triples spanning over several hundred datasets. 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]. Users and developers are tapping into entity graphs for numerous applications, including search, recommendation systems, business intelligence and health informatics.

The flourish of entity graphs boost a growing demand for *entity-centric* information exploration tasks where entity graphs are involved. Compared with traditional information retrieval tasks that are based on statistics of text terms, entity-centric information exploration tasks are based on metrics over entities, entity relationships, or entity graph struc-

[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/

tures. On one hand, entity graphs can help information exploration tasks by serving as knowledge facilities for understanding or browsing external data sources. On the other hand, huge entity graphs themselves are hard for end users or developers to comprehend. Thus there is a pressing need to study how to improve the understanding process of entity graphs.

In this dissertation, we focus on two aspects of entity-centric information exploration, i.e., using entity graphs to browse external data sources and understanding entity graphs themselves. Specifically, we formulate two concrete goals: 1) given a set of Web documents , how to utilize entity graph including entities, relationships, and a concept hierarchy to navigate these documents; and 2) how to derive a set of preview tables to browse important entities and relationships in entity graphs. Towards these goals, we propose the following two innovative approaches:

- Faceted Interface Discovery.

  We propose innovative methods for dynamic discovery of query-dependent faceted interfaces over Wikipedia articles. We further extend the discovery method to a framework that could discover such an interface for general Web documents.

- Optimal Preview Discovery.

  We propose innovative methods for generating preview tables for entity graphs which help end users and developers for quick and better understanding of such graphs.

Below we briefly highlight these two approaches.

1. Dynamic Discovery of Query-Dependent Faceted Interfaces for Web Documents. (Chapter 2):

   We first study how to utilize entity graphs to browse Web documents. Faceted search [5] is a useful technique for information exploration, especially when a user needs to browse through a long list of articles or objects, which, without necessary auxiliary facility, could be time consuming and painstaking. A faceted interface for a

set of objects is a set of category hierarchies, where each hierarchy corresponds to an individual *facet* (dimension, attribute, property) of the objects. The user can navigate an individual facet through its hierarchy of categories and ultimately a specific facet value if necessary, thus reaching those objects associated with the chosen categories and value on that facet. The user navigates multiple facets and the intersection of the chosen objects on individual facets are brought to the user's attention. The procedure hence resembles repeated constructions of conjunctive queries with selection conditions on multiple dimensions.

Wikipedia has become the largest encyclopedia ever created, whose pages have grown over 3.5 million English articles. The prevalent manner in which web users access Wikipedia articles is keyword search. Keyword search has been effective in finding specific web pages matching the keywords. Therefore it may well satisfy users when they are casually interested in a single topic and use Wikipedia as a dictionary or encyclopedia for that topic. However, Wikipedia has now become a primary knowledge source for many users and even an integral component in the knowledge management systems of businesses for decision-making. It is thus typical for a user to explore a set of relevant articles, instead of targeting a particular article, for more sophisticated information discovery and exploratory tasks. With only keyword search, one would have to digest the potentially long list of search result articles, follow hyperlinks to connected articles, adjust the query, perform multiple searches, and synthesize information manually. This procedure is often time-consuming and error-prone. A faceted interface can facilitate the process of exploring articles in Wikipedia.

Different from previous approaches, we aim at developing methods that are fully automatic and dynamic in both facet dimension generation and category hierarchy construction. Toward this goal, we propose a general faceted search model and a general framework for faceted interface discovery which is instantiated into two prototype

systems, Facetedpedia and Facetednews, for exploring Wikipedia articles and news articles, respectively. Our model utilizes the collaborative vocabularies in Wikipedia, such as its category hierarchy and intensive internal hyperlinks, for building faceted interfaces. Given the sheer size and complexity of Wikipedia data, the search space of possible choices of faceted interfaces is prohibitively large. We propose metrics for ranking individual facet hierarchies by user navigational cost, and metrics for ranking interfaces (each with k facet hierarchies) by both average pair-wise facet similarities and average navigational costs. We thus develop faceted interface discovery algorithms that optimize for these ranking metrics. We conduct experimental evaluation and user studies to verify the effectiveness of the proposed metrics, the algorithms, and the prototype systems.

2. Generating Preview Tables for Entity Graphs (Chapter 3):

   We then study how to utilize entity graphs to gain a quick and better understanding of themselves. 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. To this end, 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. Such preview tables would be helpful for quick understanding of entity graphs.

Several other approaches are arguably less adequate for gaining a quick overview of entity graphs.

(1) The first solution is to visualize a data graph [6]. 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. The schema graph 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.

(3) The third approach is to present a summary of the schema graph. Schema summarization has been investigated for relational databases [7, 8, 9], XML [9] and graphs [10, 11]. While Section 4.3 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 [7, 8, 9] work on relational and semi-structured data, instead of graph data. Some [9, 10, 11] 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 [7, 8] 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 num-

ber of tables would equal the number of entity types. *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 3.1), 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 tackled several challenges: (1) We discerned what factors contribute to the goodness of a preview and proposed 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 enforced a constraint on preview size. Furthermore, we considered enforcing an additional constraint on the pairwise distance between preview tables. Given the spaces of all possible previews, we formulated the optimization problem of finding an preview with the highest score among those satisfying the constraints. The optimization problems are non-trivial, as we proved that they are NP-complete under distance constraints. (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 design a dynamic programming algorithm and

an Apriori [12]-style algorithm. We conduct experiments on the Freebase dataset to verify the accuracy and efficiency of our methods.

Overall, this dissertation makes the following contributions:

- Concepts

  We propose the concepts of query-dependent faceted interfaces for navigating Web documents and preview tables for previewing entity graphs. (Section 2.1)

- Systems

  We build systems Facetedpedia and Facetednews instantiated from the generic framework of faceted interface discovery. To the best of our knowledge, these systems are the first attempt of dynamic discovery of query-dependent faceted interface for text documents. We also build a system that generates preview tables for entity graphs.

- Metrics

  Based on a user navigation model, we propose metrics for measuring the "goodness" of facets, both individually and collectively. (Section 2.2) We propose methods for measuring the goodness of preview tables based on several intuitions. (Section 3.2) We thus formulate the problems of optimal faceted interface discovery and optimal preview discovery based on these measures.

- Algorithms

  We develop effective and efficient algorithms for discovering faceted interfaces in the large search space of possible interfaces. (Section 2.3) We develop a dynamic-programming algorithm and an Apriori-style algorithm for finding optimal previews. (Section 3.4)

- Evaluations

  For faceted interface discovery over Web documents, we conduct user study to evaluate the effectiveness of our prototype systems by comparing with alternative approaches. We also conduct experiments to quantitatively evaluate the quality and efficiency. (Sec-

tion 2.5) For generating preview tables for entity graphs, we conduct extensive experiments and user study to verify the accuracy of the scoring measures, the efficiency of the algorithms, and the overall effectiveness of preview tables discovered. (Section 3.5)

The rest of the dissertation is organized as follows. Chapter 2 studies the problem of dynamic faceted interface discovery for web documents. Chapter 3 studies the problem of generating preview tables for entity graphs. The survey of the literature related to this dissertation is provided in Chapter 4, and the conclusions are detailed in Chapter 5.

CHAPTER 2

DYNAMIC DISCOVERY OF QUERY-DEPENDENT FACETED INTERFACES FOR WEB DOCUMENTS

In this chapter, we investigate methods for dynamically discovering query-dependent faceted interfaces over Web documents. Given a set of result documents from a keyword search query, the objective is to produce a faceted interface for exploring the result documents.

We often experience faceted interfaces when shopping at E-commerce websites such as Amazon.com and Newegg.com. For example, when a customer is shopping for LCDs on Newegg.com, she can first issue a keyword query such as "Samsung LCD". The website will then return a list of LCDs. In addition, a faceted interface will be generated on the resulting web page, for exploring the LCDs (see Figure 2.1 (left)). The customer can narrow down her search results by facets on dimensions such as *price* and *screen size*. Today, many systems can generate faceted interfaces for relational data such as the product catalogs behind online stores similar to Newegg.com. Similar faceted interfaces are used in searching library catalogs by metadata fields such as *genre*, *subject*, etc. One such example is http://www.lib.ncsu.edu/catalog/ (Figure 2.1 (right)).

However, many document collections do not come with structured schema or metadata. Hence facets must be discovered from the text content of documents. There are only few such systems that discover faceted interface for document exploration. Furthermore, no prior system dynamically discovers query-dependent interfaces for the resulting documents of keyword search. In this section we propose a framework for dynamic discovery of query-dependent faceted interface from Web documents. The framework is instantiat-

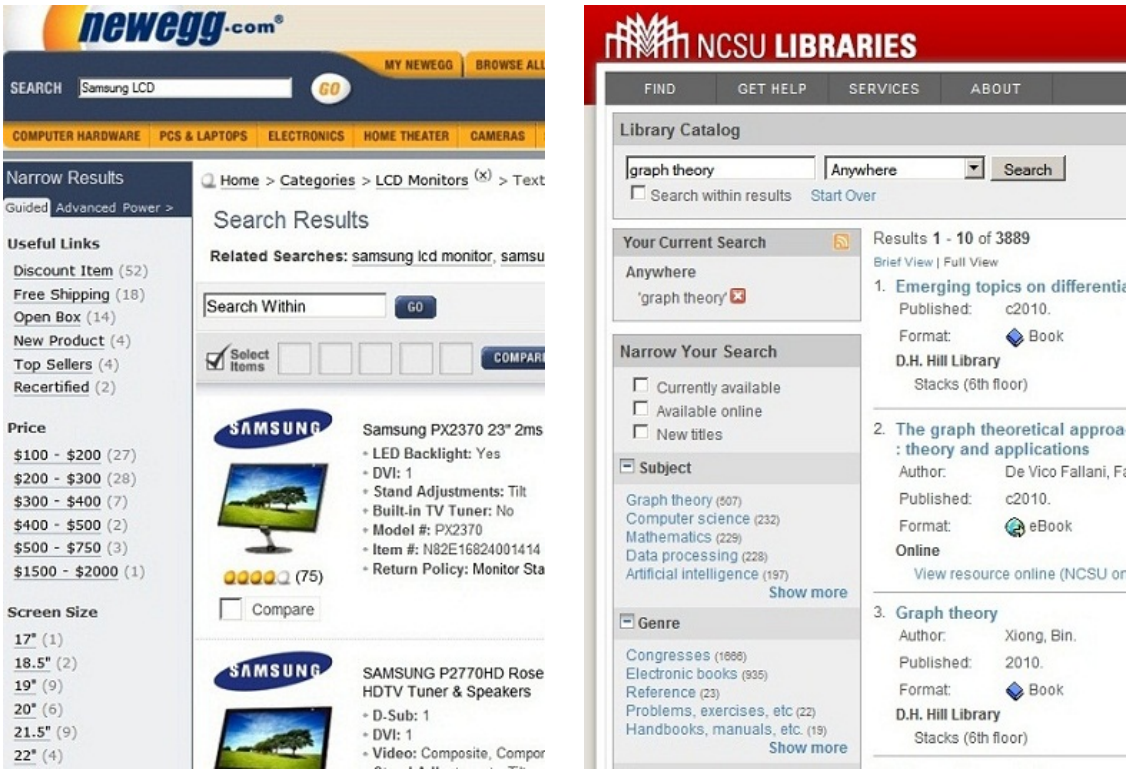


Figure 2.1: The faceted search interfaces of Newegg.com (left) and NCSU library catalog (right).

ed into two systems: Facetedpedia [1] [13, 14] and Facetednews [2], which are for exploring Wikipedia and news articles, respectively.

We use the following example as our motivating example throughout this chapter.

Example 1 (Motivating Example)**.** *Imagine that a user is exploring information about action films in Wikipedia. The Facetedpedia system takes a keyword query, say, "us action film", as the input and obtains a ranked list of Wikipedia articles from an external search engine. It will create a faceted interface, as shown in Figure 2.2, for navigating these articles. The system dynamically derives k facets (region (A)) for covering the top s result articles (region(C)) (s=234 in the example). Figure 2.2 only shows three*

[1] `http://idir.uta.edu/facetedpedia/`

[2] `http://idir.uta.edu/facetednews/`

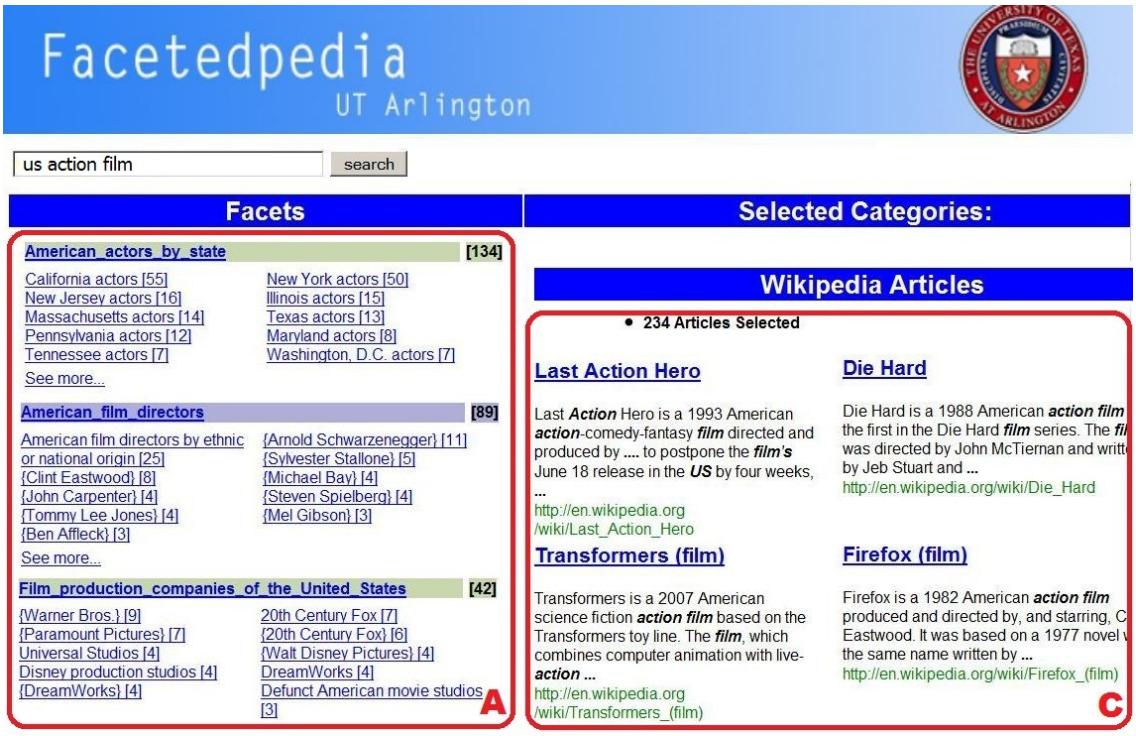


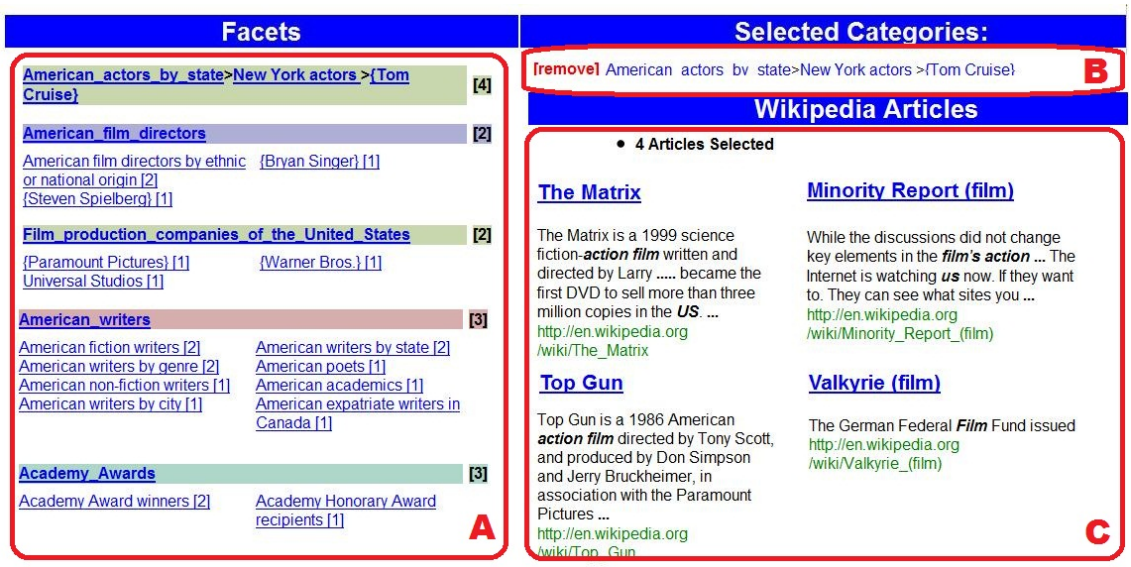
Figure 2.2: The faceted search interface of Facetedpedia generated for keywords "*us action film*".

of the generated facets: (1) American_actors_by_state; (2) American_film_directors; (3) Film_production_companies_of_the_United _States. Other facets include Academy_award_winners, American_film_screenwriters, and so on. Each facet is associated with a hierarchy of categories. Each article can be assigned to the nodes in these hierarchies, with an assignment representing an "attribute" value of the article.

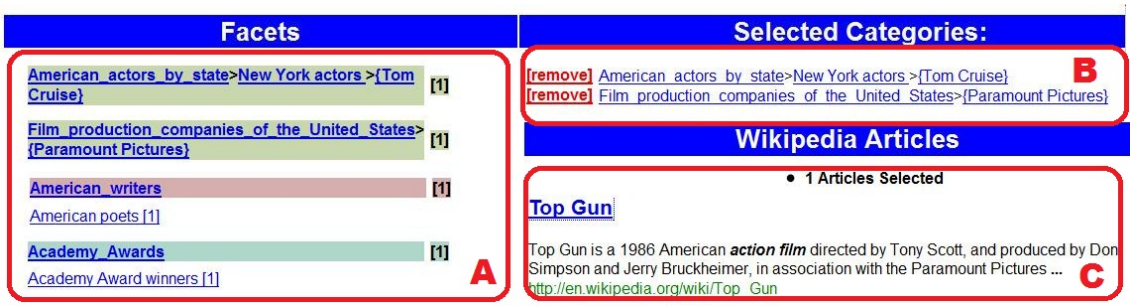
On each facet, the user can navigate through a category path which is formed by parent-child relationships between categories in the Wikipedia category system. [3] [4] *In Figure 2.3a, the user selects category New_York_actors under American_actors_by_state*

[3] A Wikipedia article may belong to one or more categories. These categories are listed at the bottom of the article.

[4] The Wikipedia category system is at `http://en.wikipedia.org/wiki/Wikipedia:Categorization`.



(a) Facetedpedia interface after selecting one navigational path: "*American_actors_by_state>New_York_actors>Tom_Cruise*".



(b) Facetedpedia interface after selecting two navigational paths: "*American_actors_by_state>New_York_actors>Tom_Cruise*" and "*Film_production_companies_of_Untied_States>Paramount_Pictures*".

(c) Facetedpedia interface after selecting two navigational paths: "*American_actors_by_state>New_York_actors>Tom_Cruise*" and "*American_film_directors>Steven_Spielberg*".

Figure 2.3: Examples of exploring Facetedpedia.

and she further selects attribute value Tom_Cruise under New_York_actors. A user navigational path is then added in region (B) of Figure 2.3a. There are four articles satisfying the chosen navigational path, and they are shown in region (C) of Figure 2.3a. The user could further add another facet condition Paramount_Pictures under category Film_production_companies_of_Untied_States. The result interface is shown in Figure 2.3b. Here another navigational path is added to region (B) of Figure 2.3b and the Wikipedia articles covered by both paths are shown in region (C) of Figure 2.3b.

If the user is not satisfied with current results, she could remove certain navigational path by clicking "[remove]" in front of that particular path. For example, she could remove the path Film_production_companies_of_Untied_States> Paramount_Pictures and add another path American_ film_directors>Steven_Spielberg. The result interface is shown in Figure 2.3c. In this way, the user filters the large number of result articles and finds those matching her interests. When the user clicks one particular article title in region (C), the corresponding Wikipedia article would be brought to the user by the system. (This part of the interface is omitted.)

Based on the same framework beneath Facetedpedia, Facetednews is a faceted search system for news articles (Figure 2.4). In a traditional news search engine, a user can search by keywords (e.g. "nba games") and then navigate through the result articles one by one. In Facetednews, several facets relevant to the news articles are generated to help the user narrow down her targets. If she wants to read articles related to Boston Celtics players, she can select category *Boston_Celtics_players* under facet *National_Basketball_Association_players_by_club*. If she is further interested in US players who have played in Olympic games, she can add another facet condition *Olympic_Basketball_play -ers_of_the_United_States*. This scenario is shown is Figure 2.4. Two navigational paths are added to region (B). She can proceed to read any article in region (C), which contains

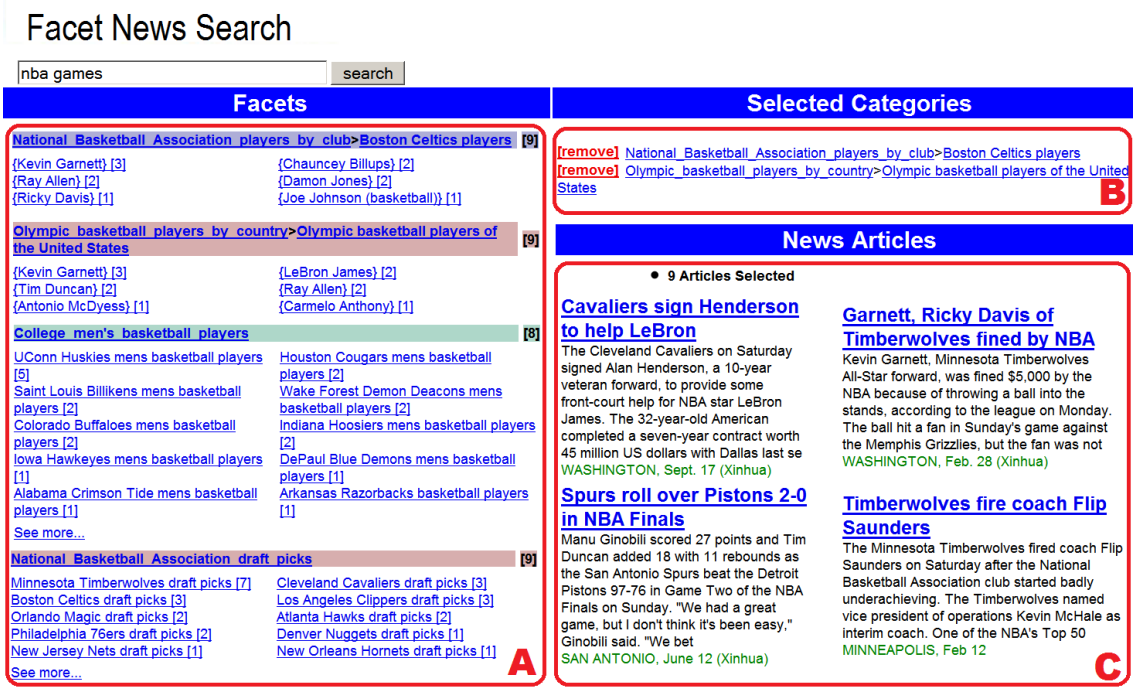


Figure 2.4: The faceted search interface of Facetednews.

articles at the intersection of the two chosen navigational paths. She can also add more paths to region (B).

Motivated by the examples in previous chapter, we study the problem of dynamic discovery of query-dependent faceted interfaces for Web documents. Given a set of top-s ranked articles as the search result from a keyword query, our goal is to produce an interface of multiple facets for exploring these result articles. Specifically, we focus on *automatic* and *dynamic* discovery of faceted interfaces. The facets could not be pre-computed due to the query-dependent nature of our proposed system. In applications where faceted interfaces are deployed for relational tuples or schema-available objects, the tuples/objects are captured by prescribed schemata with clearly defined dimensions (attributes), therefore a query-independent static faceted interface (either manually or automatically generated) may suffice. On the contrary, Web documents lack such pre-determined dimensions that could fit all possible dynamic query results. Therefore efforts on static facets would be fu-

tile. Even if the facets can be pre-computed for some popular queries, say, based on query logs, the computation must be automatic and dynamic. Given the sheer size and rapid growth of document corpora, the large number of attribute values that can be associated with documents, and the complexity of category systems such as the Wikipedia category system, a manual approach would be prohibitively time-consuming and cannot scale to stay up-to-date.

Dynamic discovery of query-dependent faceted interfaces for Web documents is a non-trivial undertaking. Below we briefly summarize the main challenges in realizing a faceted search system with such capability and our main ideas:

Challenge 1: The facets and their category hierarchies are not readily available.

Our concept of faceted interface is built upon two pillars: facets (i.e., dimensions or attributes) and the category hierarchy associated with each facet. The definition of "facet" itself for documents does not arise automatically, leaving alone the discovery of a faceted interface. Therefore we must answer two questions: (1) *facet identification* – What are the facets of Web documents? and (2) *hierarchy construction* – Where does the category hierarchy of a facet come from?

In this chapter we propose a generic model for faceted interfaces, which can be instantiated into different particular systems. Specifically, we instantiate the model into two prototype systems, Facetedpedia and Facetednews, for faceted search over Wikipedia and news articles, respectively. In instantiating the generic model, various sources may be used for identifying facet attributes from Web documents. For each document, the named entities appearing in it represent important attributes of that document. Various named entity catalogs can be used in identifying entity mentions in documents. Particularly, a Wikipedia article can be viewed as the description of an entity. Hence Wikipedia itself is a high-quality and comprehensive entity catalog. In addition to treating entities mentioned in articles as facet attributes, we may also use other approaches. For example, if two entities co-occur

frequently in a corpus, one can serve as an attribute of the other. This is particularly useful in Facetedpedia, where Wikipedia is both the text corpus and the named entity catalog. Another example is to use categories of Wikipedia articles as their attributes. Furthermore, metadata of articles can also be utilized in forming facet attributes. Once the facet attributes are identified, category hierarchies of facets can be constructed by utilizing various information sources such as thesaurus (e.g., WordNet [15]), taxonomy (e.g., YAGO [2]), and folksonomy (e.g., Wikipedia category system).

Challenge 2: We need metrics for measuring the "goodness" of facets both individually and collectively.

Given a set of documents, there may be a large number of candidate facets. Hence a goodness metric for ranking the facets is necessary. Since the ultimate objective of a faceted interface over Web documents is to help users explore the documents, it is natural to optimize for finding facets effective for user navigation. A good facet should help users find desirable documents conveniently. Hence we propose to rank facets by their navigational costs, i.e., the amount of effort undertaken by users during navigation, based on a user navigation model.

The problem gets more complex when finding multiple facets to form a faceted interface, because the utilities of multiple facets do not necessarily build up linearly. Since the facets in an interface should ideally describe diverse aspects of the Web documents, a set of individually "good" facets may not be "good" collectively. Our idea is to avoid overlap between multiple facets and more specifically to optimize for small average pair-wise similarity between facets.

Challenge 3: We must design efficient faceted interface discovery algorithms based on the ranking criteria.

A straightforward approach for faceted interface discovery is to enumerate all possible faceted interfaces and apply our ranking metrics directly to find the best interface.

Such a brute-force method results in exhaustive examination of all possible combinations of facets. This can easily be a prohibitively large search space. Furthermore, the interactions between the facets in a faceted interface make the computation of its exact ranking score intractable.

Our faceted interface discovery algorithm hinges on two ideas: (1) reducing the search space; and (2) searching the space efficiently. There are two search spaces in finding a good faceted interface– the space of facets and the space of faceted interfaces. To reduce the space of candidate facets, we focus on a subset of "safe reaching facets". To further reduce the space of faceted interfaces, we rank facets individually by their navigational costs and only consider the top ranked facets that do not subsume each other. Instead of exhaustively examining all possible interfaces, we design a hill-climbing based heuristic algorithm that optimizes for both the average navigational cost and the pair-wise similarity of multiple facets.

The rest of the chapter is organized as follows. In Section 2.1, we propose the generic model of faceted interface and formally define the various relevant concepts. Section 2.2 discusses the metrics for ranking facets. We present our facet discovery algorithm in Section 2.3. Section 2.4 discusses important implementation details and Section 2.5 presents the results of user study and experimental evaluation.

2.1 A Generic Model for Faceted Interfaces and its Instantiation for Faceted Search over Web Documents

In this section we first present a generic model of faceted interfaces, to explain the basic concepts in faceted search systems. We then discuss how to instantiate the model to Facetedpedia and Facetednews, which are our faceted search systems for Wikipedia and news articles, respectively. Finally we formally define the concepts in this model.

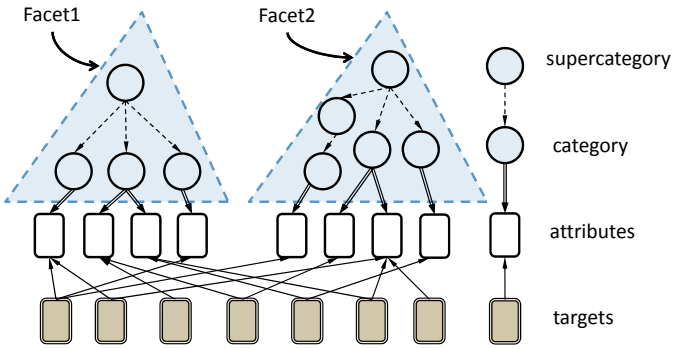


Figure 2.5: The generic model for faceted interfaces.

2.1.1 A Generic Model of Faceted Interfaces

Figure 2.5 is a generic model that shows the basic components in faceted interfaces and their relationships. Most faceted search interfaces consist of three levels: targets, attributes, and category hierarchies. The *targets* are the objects that users browse and search for. Examples of targets include database records, merchandise, photos, videos, web bookmarks, library collections, news articles, and Wikipedia entities (e.g., people, places, organizations, etc.), depending on application scenarios. The *attributes* are the features of the targets. Examples include database schema attributes, product features (e.g., *price*, *manufacturer*, *size*, etc.), tags on social media objects such as photos, videos, and bookmarks, metadata of library collections, terms in articles, and named entities related to target entities. The attributes of objects are partitioned into multiple *facets*, each of which corresponds to a dimension of attributes for exploring the objects. In some systems, each facet is simply a flat group of attribute values. In other systems, the attribute values in each facet can be further organized into a *category hierarchy*, which ideally presents the IS-A relationships between category-subcategory and category-attribute. Various ways can be exploited in constructing category hierarchy. For instance, a category hierarchy can be derived from a folksonomy such as the Wikipedia category system, a thesaurus such as WordNet, or a domain-specific taxonomy (e.g., the city-state-country hierarchy for places).

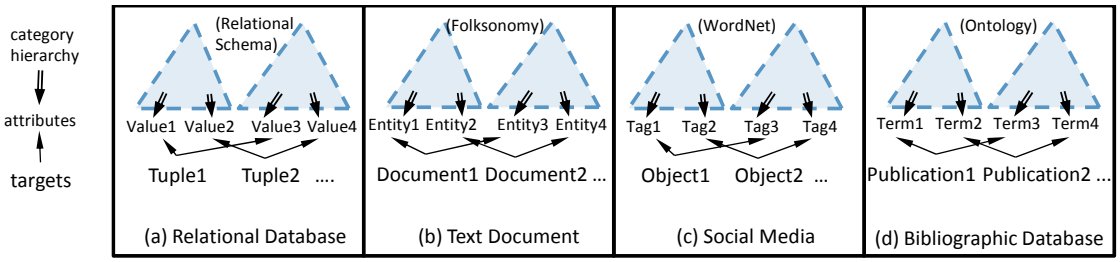


Figure 2.6: Instantiations of the generic faceted interface model for different scenarios.

Figure 2.6 shows several possible instantiations of the above generic model for building faceted interfaces in various applications. Note that the figure is only for illustration purpose. It is not meant to cover all possible scenarios of instantiation. For each scenario in the figure, we show what are the targets, the attributes, and the categories, to instantiate the model. Figure 2.6(a) shows that each facet over a relational database table corresponds to an attribute in its schema. Attribute values in a facet can be hierarchically organized by a domain-specific taxonomy or simply alphabetically ordered. The same instantiation can be used for faceted search over products in online stores and museum and library collections. Figure 2.6 (b) shows how to build a faceted interface for Web documents based on the generic model. The attributes are named entities appearing in target documents and/or related to the documents. The categories of these named entities and the super-categories of the categories form the category hierarchies of facets. These hierarchies are based on a folksonomy such as the Wikipedia category system. The details of this instantiation are given in Section 2.1.2. Figure 2.6(c) is an instantiation of the model for faceted search over social media objects (e.g., videos and photos) and social bookmarks. Each tag is an attribute. Related tags are put together as a facet and hierarchically organized by concepts from a thesaurus taxonomy such as WordNet [15]. By choices made on multiple facets, a user can find those objects that have the specified tags or have tags belonging to the chosen concepts. Figure 2.6(d) shows how to instantiate the model for a faceted interface over

a bibliographic database (e.g., PubMed [5]). The attributes are the scientific terms appearing in target objects, i.e., publications. The terms are organized into multiple facets with hierarchical categories, based on specialized taxonomies such as the Gene Ontology [6].

2.1.2 Instantiation of the Generic Model for Faceted Interfaces over Web Documents

In this section we provide detailed discussion of Figure 2.6(b), i.e., how to instantiate the generic model for faceted search over Web documents. Specifically, we explain its instantiation into two prototype faceted search systems, Facetedpedia and Facetednews, for Wikipedia articles and news articles, respectively.

The basic components in the generic model are targets, attributes, and category hierarchies. Hence the key to realization of the model is to instantiate these basic components. In applications where faceted interfaces are deployed for relational tuples or schema-available objects, the tuples/objects are captured by prescribed schemata with clearly defined attributes (cf. Figure 2.6(a)). On the contrary, Web documents lack such predetermined schemata. To address this challenge, the basis of our instantiation is to exploit user-generated *collaborative vocabulary* in Wikipedia such as its "grassroots" category system and its heavily interlinked articles. The collaborative vocabulary represents the collective intelligence of many users and rich semantic information, and thus constitutes the promising basis for forming faceted interfaces. With regard to the concept of facet attribute, the Wikipedia articles (i.e., entities) hyperlinked from or related to a search result article (i.e. target article) are exploited as its attributes. With regard to the concept of category hierarchy, the Wikipedia category system provides the category-subcategory relationships between categories, allowing users to go from general to specific concepts during exploration.

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

[6] `http://www.geneontology.org/`

In Facetedpedia, the targets are Wikipedia articles, each of which is an encyclopedia entry describing the corresponding entity. For each target entity, its attributes are also Wikipedia entities. They co-occur frequently with the target entity within syntactical units such as sentences or paragraphs in some text corpus. The premise is that if an entity co-occurs frequently with the target entity, there is a good chance that they are highly related. Specifically we use Wikipedia itself as the text corpus for measuring degree of co-occurrence, although other text corpora can also serve the purpose, after Wikipedia entities appearing in documents are annotated. The attribute entities can also be identified by multiple ways together. For example, attribute entities of a target entity can be the hyperlinked entities in the target. In Wikipedia, the fact that the authors of an article (target entity) collaboratively made hyperlinks to other entities is an indication of the significance of the hyperlinked entities in describing the target entity.

In Facetednews, the targets are news articles. Similar to Facetedpedia, Facetednews also uses Wikipedia entities as attributes of target articles. Specifically, given a target article, its attribute entities are mentioned in the article itself. The attribute entities are recognized by entity annotation techniques and particularly entity annotation systems developed for annotating by Wikipedia. We use Wikifier[7] [16] for such purpose. Given a Web document, Wikifier adds hyperlinks to Wikipedia entities in the document. It does so by matching token sequences in the document with titles of Wikipedia entities and disambiguating among candidate matches. After annotation, the news articles in Facetednews are enriched with semantic attributes that are helpful in faceted navigation. This instantiation is applicable to not only news articles but also general Web documents, which extends the application scenarios of our generic model.

Since both Facetedpedia and Facetednews use Wikipedia entities as attributes of target articles, the instantiation of category hierarchies is the same in the two systems, by

[7] http://www.nzdl.org/wikification/docs.html

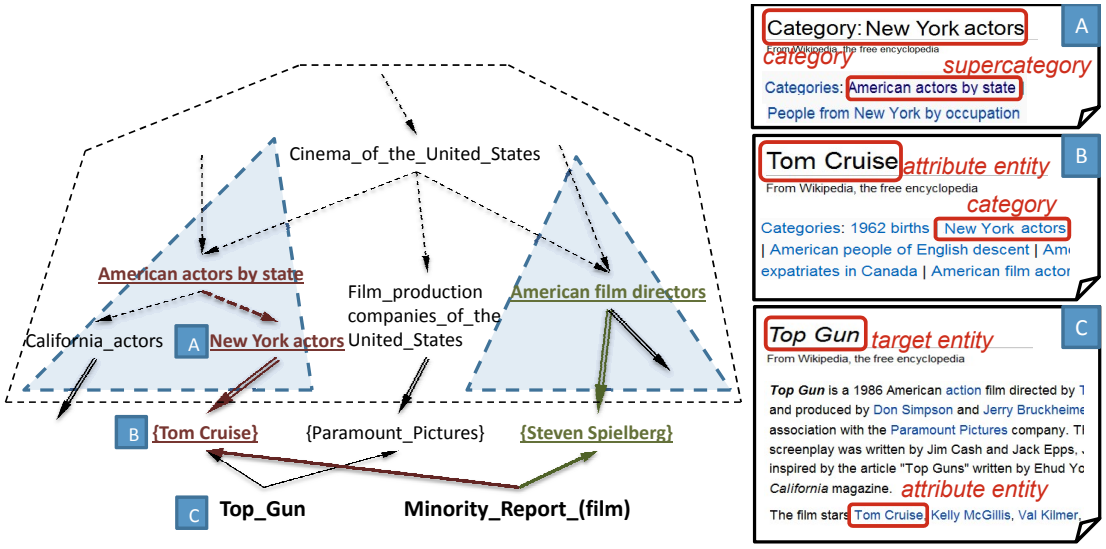


Figure 2.7: Instantiation of the generic model for Facetedpedia. Two highlighted navigational paths corresponding to Figure 2.3c.

exploiting the Wikipedia category system. [8] The Wikipedia category system consists of a large hierarchy of categories. Its root is Category:Fundamental categories. [9] Starting from the root, a category may contain multiple subcategories and multiple instance articles. Subcategories cover Wikipedia articles under more specific concepts, while supercategories cover more general concepts. Articles contained in a category and its subcategories are instances of the concept covered by the category. Hence the Wikipedia category system is a hierarchy formed by supercategory-subcategory relationships and category-instance article relationships.

In both Facetedpedia and Facetednews, each facet is a sub-hierarchy of the Wikipedia category system. A facet groups together highly-related attribute entities (which are Wikipedia articles themselves). Therefore a facet corresponds to a conceptual dimension

[8] Note that the generic model of faceted interfaces is not limited to the specific Wikipedia category system. Ideally, any category hierarchy from a well-structured taxonomy such as WordNet [15] or YAGO [2] can be exploited as the category hierarchy in the model.

[9] `http://en.wikipedia.org/wiki/Category:Fundamental_categories`

and each attribute entity in it corresponds to a value on the dimension. The hierarchy of categories in the facet helps users to explore from general categories to more specific ones, then to instance articles of categories (i.e., attribute entities), and finally to target entities that attain the attribute entities.

Example 2 (Instantiate the Generic Model for Facetedpedia)**.** *Figure 2.7 shows an example of instantiating the generic model into Facetedpedia. The target entities are two Wikipedia articles Top_Gun and Minority_Report_(film). The attribute entities are Tom_Cruise, Paramount_Pictures, and Steven_Spielberg. The two facets are sub-hierarchies under categories American_actors_by_state and American_film_directors. The three boxes on the right side of the figure represent Wikipedia pages A, B, and C. In this particular example, we consider the hyperlinked entities on a target article as the attribute entities of that target. Hence Tom_Cruise is an attribute entity of target entity Top_Gun, according to Wikipedia page C, which is the article for the target entity itself. New_York_actors is a category of attribute entity Tom_Cruise, according to Wikipedia page B, which is the article for the attribute entity itself. American_actors_by_state is a supercategory of New_York_actors, by Wikipedia page A.*

2.1.3 Definitions of Concepts and Faceted Interface Discovery Problem

We now formally define the concepts of faceted interfaces for Web documents, based on the aforementioned instantiation of our generic model. We also provide the specification of faceted interface discovery problem.

Definition 1 (Target Article, Attribute Entity)**.** *Given a keyword query q, the set of top-s ranked result articles from search engine, $\mathcal{T}=\{p_1, ..., p_s\}$, are the* target articles *of q. Each target article can have multiple* attribute entities*, each of which is a Wikipedia article (entity). The relationship between a target article p and its attribute entity p' is represented as $p \rightarrow p'$. The relationship can be established by different measures, such as p and p' co-*

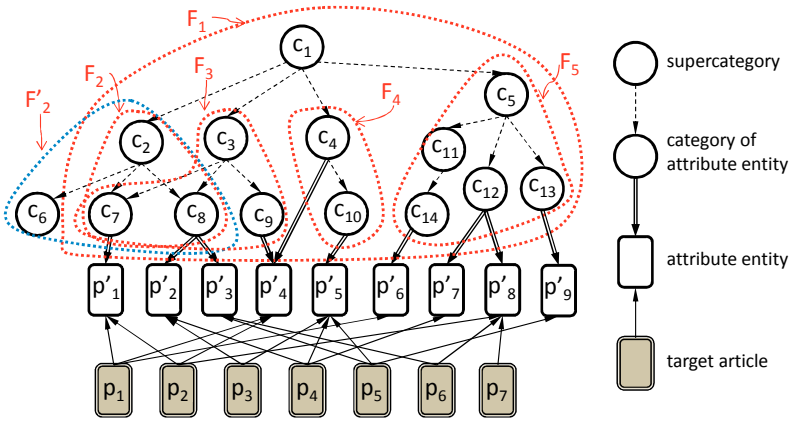


Figure 2.8: The concept of facet for documents.

occurring enough number of times (when p itself is also a Wikipedia entity), p' is mentioned in article p or simply hyperlinked from p, and so on. Given $\mathcal{T}$, the set of attribute entities is $\mathcal{A}$=$\{p'_1, ..., p'_m\}$, where each p'_i is an attribute entity of at least one target article $p_j \in \mathcal{T}$.

Definition 2 (Category Hierarchy)**.** *A* category hierarchy *is a connected, rooted directed acyclic graph $\mathcal{H}(r_\mathcal{H}, \mathcal{C}_\mathcal{H}, \mathcal{E}_\mathcal{H})$, where the node set $\mathcal{C}_\mathcal{H}$=$\{c\}$ is a set of categories, the edge set $\mathcal{E}_\mathcal{H}$= $\{c \dashrightarrow c'\}$ is a set of supercategory(c)-subcategory(c') relationships, and $r_\mathcal{H}$ is the root of $\mathcal{H}$.*

Definition 3 (Facet)**.** *A* facet *$\mathcal{F}(r, \mathcal{C}_\mathcal{F}, \mathcal{E}_\mathcal{F})$ is a rooted and connected subgraph of the category hierarchy $\mathcal{H}(r_\mathcal{H}, \mathcal{C}_\mathcal{H}, \mathcal{E}_\mathcal{H})$, where $\mathcal{C}_\mathcal{F} \subseteq \mathcal{C}_\mathcal{H}$, $\mathcal{E}_\mathcal{F} \subseteq \mathcal{E}_\mathcal{H}$, and $r \in \mathcal{C}_\mathcal{F}$ is the root of $\mathcal{F}$.*

Example 3 (Running Example)**.** *In Figure 2.8 there are 7 target articles (p_1, ..., p_7) and 9 attribute entities (p'_1, ..., p'_9). The category hierarchy has 14 categories (c_1, ..., c_{14}). The figure highlights 6 facets ($\mathcal{F}_1$, ..., $\mathcal{F}_5$, and $\mathcal{F}'_2$). For instance, F_2 is rooted at c_2 and consists of 3 categories (c_2, c_7, c_8) and 2 edges ($c_2 \dashrightarrow c_7$, $c_2 \dashrightarrow c_8$). There are many more facets since every rooted and connected subgraph of the hierarchy is a facet. Note that the figure may give the impression that edges such as $c_{11} \dashrightarrow c_{14}$ and $c_7 \Rightarrow p'_1$ are unnecessary since there is only one choice under c_{11} and c_7, respectively. The example is small due to space limitations. Such single outgoing edge is very rare in a real category hierarchy of a*

folksonomy such as Wikipedia category hierarchy. We will use Figure 2.8 as the running example throughout the section.

The categories in a facet can "*reach*" target articles $\mathcal{T}$ through attribute entities $\mathcal{A}$. That is, by following the category-subcategory hierarchy of the facet, we can find a category, then find an attribute entity belonging to the category, and finally find the target articles that have the attribute. All such target articles are called the *reachable target articles*. A *facet* is a *safe reaching facet* if $\forall c \in \mathcal{C}_{\mathcal{F}}$, there exists a target article $p \in \mathcal{T}$ such that c reaches p, i.e., there exists $c \dashrightarrow ... \Rightarrow p' \leftarrow p$, a navigational path of $\mathcal{F}$, starting from c, that reaches p. In order to capture the notion of "reach", we formally define *navigational path* as follows.

Definition 4 (Navigational Path)**.** *With respect to target articles $\mathcal{T}$, attribute entities $\mathcal{A}$, and a facet $\mathcal{F}(r, \mathcal{C}_{\mathcal{F}}, \mathcal{E}_{\mathcal{F}})$, a* navigational path *in $\mathcal{F}$ is a sequence $c_1 \dashrightarrow ... \dashrightarrow c_t \Rightarrow p' \leftarrow p$, where,*

- *for $1 \leq i \leq t$, $c_i \in \mathcal{C}_{\mathcal{F}}$, i.e., c_i is a category in $\mathcal{F}$;*
- *for $1 \leq i \leq t-1$, $c_i \dashrightarrow c_{i+1} \in \mathcal{E}_{\mathcal{F}}$, i.e., c_{i+1} is a subcategory of c_i (in category hierarchy $\mathcal{H}$) and that category-subcategory relationship is kept in $\mathcal{F}$.*
- *$p' \in \mathcal{A}$, and c_t is a category of p' (represented as $c_t \Rightarrow p'$);*
- *$p \in \mathcal{T}$, and p' is an attribute entity of p (represented as $p \rightarrow p'$).*

Given a navigational path $c_1 \dashrightarrow ... \dashrightarrow c_t \Rightarrow p' \leftarrow p$, we say that the corresponding category path $c_1 \dashrightarrow ... \dashrightarrow c_t$ reaches *target article p through attribute entity p', and we also say that category c_i (for any $1 \leq i \leq t$)* reaches *p through p'. Interchangeably we say p is* reachable *from c_i (for any $1 \leq i \leq t$).*

Definition 5 (Faceted Interface)**.** *Given a keyword query q and the corresponding target articles $\mathcal{T}$, a faceted interface $I = \{\mathcal{F}_i\}$ is a set of safe reaching facets of $\mathcal{T}$. That is, $\forall \mathcal{F}_i \in I$, $\mathcal{F}_i$ safely reaches $\mathcal{T}$.*

Example 4 (Navigational Path and Faceted Interface)**.** *Continue the running example. In Figure 2.8, $I = \{\mathcal{F}_2, \mathcal{F}_5\}$ is a 2-facet interface. Two examples of navigational paths are*

$c_2 \dashrightarrow c_8 \Rightarrow p'_3 \leftarrow p_5$ *and* $c_5 \dashrightarrow c_{13} \Rightarrow p'_9 \leftarrow p_5$. *However,* $\{\mathcal{F}'_2, \mathcal{F}_5\}$ *is not a valid faceted interface because* $\mathcal{F}'_2$ *is not a safe reaching facet, as category* c_6 *cannot reach any target articles.*

Based on the formal definitions, the Faceted Interface Discovery Problem over Web documents is: Given a category hierarchy $\mathcal{H}(r_\mathcal{H}, \mathcal{C}_\mathcal{H}, \mathcal{E}_\mathcal{H})$, for a keyword query q and its resulting target articles $\mathcal{T}$ and corresponding attribute entities $\mathcal{A}$, find the "best" faceted interface with k facets. We shall develop the notion of "best" in Section 2.2.

2.2 Facet Ranking

The search space of the faceted interface discovery problem is prohibitively large. Given the set of s target articles to a keyword query, $\mathcal{T}$, there are a large number of attribute entities which in turn have many categories associated with complex hierarchical relationships. To just give a sense of the scale, in Wikipedia there are about 3 million English articles. The category system $\mathcal{H}$ contains close to half a million categories and several million category-subcategory relationships. By definition, any rooted and connected subgraph of $\mathcal{H}$ that safely reaches $\mathcal{T}$ is a candidate facet, and any combination of k facets would be a candidate faceted interface. Given the large space, we need ranking metrics for measuring the "goodness" of facets, both individually and collectively as interfaces.

Given that faceted interfaces are for users to navigate through the associated category hierarchies and to ultimately reach the target articles, it is natural to rank them by the users' navigational cost, i.e., the amount of effort undertaken by the users during navigation. The "best" k-facet interface is the one with the smallest cost. Therefore as the basis of such ranking metrics, we model users' navigational behaviors as follows.

A user navigates in multiple facets in a k-facet interface. At the beginning, the navigation starts from the roots of all k facets. At each step, the user picks one facet and examines the set of subcategories available at the current category on that facet. She fol-

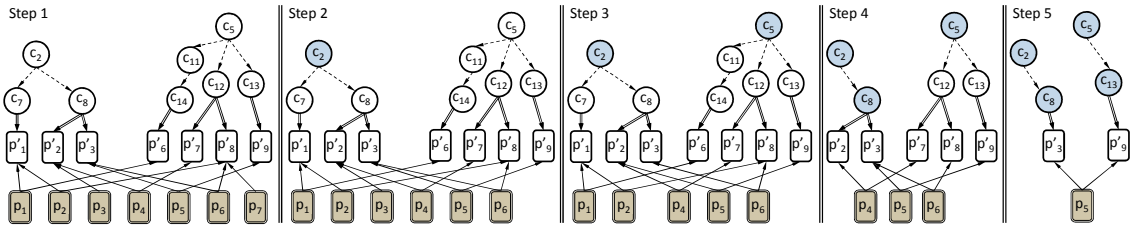


Figure 2.9: The navigation on a 2-facet interface $\mathcal{I} = \{\mathcal{F}_2, \mathcal{F}_5\}$.

lows one subcategory to further go down the category hierarchy. Alternatively the user may select one of the attribute entities reachable from the current category. The selections made on the k facets together form a conjunctive query. After the selection at each step, the list of target articles that satisfy the conjunctive query are brought to the user. The navigation terminates when the user decides that she has seen desirable target articles.

Example 5 (Navigation in Faceted Interface)**.** *Continue the running example in Figure 2.8. Consider a faceted interface I=$\{\mathcal{F}_2, \mathcal{F}_5\}$. A sequence of navigational steps on this interface are in Figure 2.9. At the beginning, the user has not selected any facet to explore, therefore all 7 target articles are available (step 1). Once the user decides to explore $\mathcal{F}_2$ which starts from c_2, p_7 is filtered out since it is unreachable from $\mathcal{F}_2$ (step 2). The user then selects c_5, which further removes p_3 from consideration (step 3). After the user further explores $\mathcal{F}_2$ by choosing c_8 (step 4), c_{11} is not a choice under c_5 anymore because no target articles could be reached by both $c_2 \dashrightarrow c_8$ and $c_5 \dashrightarrow c_{11}$. The user continues to explore $\mathcal{F}_5$ by choosing c_{13} (step 5), which removes p'_2 and also trims down the satisfactory target articles to $\{p_5\}$. The user may decide she has seen desirable articles and the navigation stops.*

2.2.1 Single-Facet Ranking

In this section we focus on how to measure the cost of an individual facet. Based on the navigational model, we compute the navigational cost of a facet as the average cost

of its navigational paths. Intuitively a low-cost path, i.e., a path that demands small user effort, has a small number of steps and at each step only requires the user to browse a small number of choices. Therefore, we formally define the cost of a navigational path as the summation of fan-outs (i.e., number of choices) at every step, in logarithmic form. [10]

Definition 6 (Cost of Navigational Path)**.** *With respect to target articles $\mathcal{T}$, the corresponding attribute entities $\mathcal{A}$, and a facet $\mathcal{F}(r, \mathcal{C}_\mathcal{F}, \mathcal{E}_\mathcal{F})$, the cost of a navigational path in $\mathcal{F}$ is*

$$cost(l) = log_2(fanout(p')) + \sum_{c \in \{c_1, ..., c_t\}} log_2(fanout(c)) \tag{2.1}$$

where l=c_1 - - → ... - - → c_t ⇒ p' ← p.

In Formula 2.1, $fanout(p')$ is the number of (directly) reachable target articles through the attribute entity p',

$$fanout(p') = |\mathcal{T}_{p'}| \tag{2.2}$$

$$\mathcal{T}_{p'} = \{p | p \in \mathcal{T} \wedge p \rightarrow p'\} \tag{2.3}$$

In Formula 2.1, $fanout(c)$ is the fanout of category c in $\mathcal{F}$,

$$fanout(c) = |\mathcal{A}_c| + |\mathcal{C}_c| \tag{2.4}$$

where $\mathcal{A}_c$ is the set of attribute entities belonging to c,

$$\mathcal{A}_c = \{p' | p' \in \mathcal{A} \wedge c \Rightarrow p'\} \tag{2.5}$$

and $\mathcal{C}_c$ is the set of subcategories of c in $\mathcal{F}$,

$$\mathcal{C}_c = \{c' | c' \in \mathcal{C}_\mathcal{F} \wedge c \dashrightarrow c' \in \mathcal{E}_\mathcal{F}\} \tag{2.6}$$

Note that we made several assumptions for simplicity. The cost formula only captures "browsing" cost. A full-fledged formula would need to incorporate other costs, such

[10] The intuition behind the logarithmic form is: When presented with a number of choices, the user does not necessarily scan through the choices linearly but by a procedure similar to binary search.

as the "clicking" cost in selecting a choice and the cost of "backward" navigation when a user decides to change a previous selection. Furthermore, we assume the user always completes a navigational path till reaching target articles. In reality, however, the user may stop in the middle when she already finds desirable articles reachable from the current selection of category. We leave the investigation of more sophisticated models to future study.

Example 6 (Cost of Navigational Path)**.** *We continue the running example. Given* $l = c_5 \dashrightarrow c_{12} \Rightarrow p'_8 \leftarrow p_6$*, a navigational path of* $\mathcal{F}_5$ *in Figure 2.8,* $cost(l) = fanout(c_5) + fanout(c_{12}) + fanout(p'_8) = log_2(3) + log_2(2) + log_2(3) = 4.17$.

Albeit the basis of our facet ranking metrics, the definition of navigational cost is not sufficient in measuring the goodness of a facet. It does not consider such a scenario that a facet cannot fully reach all the target articles in $\mathcal{T}$, which presents an unsatisfactory user experience. In fact, low-cost and high-coverage could be two qualities that compete with each other. On the one hand, a low-cost facet could be one that reaches only a small portion of the target articles. On the other hand, a comprehensive facet with high coverage may tend to be wider and deeper, thus more costly. Therefore we must incorporate into the cost formula the notion of "coverage", i.e., the ability of a facet to reach as many target articles as possible. To combine navigational cost with coverage, we penalize a facet by associating a high-cost *pseudo path* with each unreachable article. We then define the cost of a facet as the average cost in reaching each target article.

Definition 7 (Cost of Facet)**.** *With respect to target articles* $\mathcal{T}$*, the cost of a safe reaching facet* $\mathcal{F}(r, \mathcal{C}_\mathcal{F}, \mathcal{E}_\mathcal{F})$*,* $cost(\mathcal{F}_r)$*, is the average cost in reaching each target article. The cost for a reachable target article is the average cost of the navigational paths that start from* r *and reach the target, and the cost for an unreachable target is a pseudo cost* $penalty$*.*

$$cost(\mathcal{F}_r) = \frac{1}{|\mathcal{T}|} \times \left(\sum_{p \in \mathcal{T}_r} cost(\mathcal{F}_r, p) + penalty \times |\mathcal{T} - \mathcal{T}_r|\right) \tag{2.7}$$

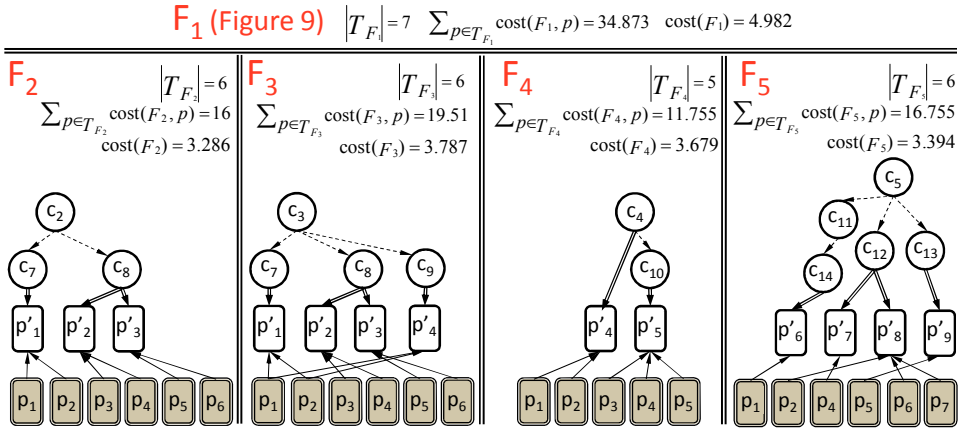


Figure 2.10: Navigational costs of facets.

where $cost(\mathcal{F}_r, p)$ *is the average cost of reaching* p *from* r*,*

$$cost(\mathcal{F}_r, p) = \frac{1}{|l_p|} \times \sum_{l \in l_p} cost(l) \tag{2.8}$$

where l_p *is the set of navigational paths in* $\mathcal{F}$ *that reach* p *from* r*,*

$$l_p = \{l | l = r \dashrightarrow ... \Rightarrow p' \leftarrow p\} \tag{2.9}$$

In Formula 2.7, $penalty$ is the cost of the aforementioned expensive pseudo path that "reaches" the unreachable target articles, i.e., $\mathcal{T} - \mathcal{T}_r$, for penalizing a facet for not reaching them. Its value is empirically selected (Section 2.5) and is at least larger than the highest cost of any path to a reachable target article.

Example 7 (Cost of Facet)**.** *We continue the running example. Figure 2.10 shows the costs of the* 5 *highlighted facets in Figure 2.8, together with their category hierarchies and reachable attribute entities and target articles. It does not show* $\mathcal{F}_1$ *which is Figure 2.8 itself excluding* c_6*. The costs of facets are obtained by Formula 2.7, with* $penalty$=7*. For instance,* $cost(\mathcal{F}_2)$=$\frac{1}{7}\times(\sum_{p\in\{p_1,p_2,p_3,p_4,p_5,p_6\}} cost(\mathcal{F}_2, p)+penalty \times |\mathcal{T}-\mathcal{T}_{\mathcal{F}_2}|)$=$\frac{1}{7}\times(16+7\times1)$=3.286*.* $\mathcal{F}_2$ *and* $\mathcal{F}_5$ *achieve lower costs than other facets. Even though the paths in* $\mathcal{F}_4$ *are cheap,* $\mathcal{F}_4$ *has higher cost due to the penalty for unreachable*

target articles (p_6 and p_7). $\mathcal{F}_1$ is even more costly due to its wider and deeper hierarchy, although it reaches all target articles.

2.2.2 Multi-Facet Ranking

Even with the cost metrics for individual facets, measuring the "goodness" of a faceted interface, i.e., a set of facets, is nontrivial. This is because the best k-facet interface may not be simply the set of cheapest k facets. The reason is that when a user navigates multiple facets, the selection made at one facet has impact on the available choices on other facets, as illustrated by Example 5.

To directly follow the approach of ranking faceted interfaces by navigational cost, in principle we could represent the navigational steps on multiple facets as if the navigation is on one *"integrated" facet*. To illustrate, consider the navigation on a 2-facet interface $\mathcal{I}$=$\{\mathcal{F}_2, \mathcal{F}_5\}$ from Figure 2.8. Two possible sequences of navigational steps are shown in Figure 2.11(a). One is c_2, c_5, c_8, c_{13}, p'_9, p'_3, p_5, which are the steps taken by the user in Figure 2.9, followed by choosing p'_9, p'_3, and finally p_5. (Remember, for simplification of the model, we assumed that the user will always complete navigational paths till reaching target articles.) At each step, the available choices from both facets are put together as the choices in the "integrated" facet. Note that after c_8 is chosen, c_{12} and c_{13} are still valid choices but c_{11} is not available anymore because c_{11} cannot reach the target articles that c_8 reaches. For the same reason, after c_{13} is chosen, p'_3 is still a valid choice but p'_2 is not anymore. The other highlighted sequence is c_5, c_{11}, c_2, c_7, p'_1, c_{14}, p'_6, p_1. There are many more possible sequences not shown in the figure due to space limitations.

With the concept of "integrated" facet, one may immediately apply Definition 7 to define the cost of a faceted interface. That entails computing all possible sequences of interleaving navigational steps across all the facets in a faceted interface. The interaction

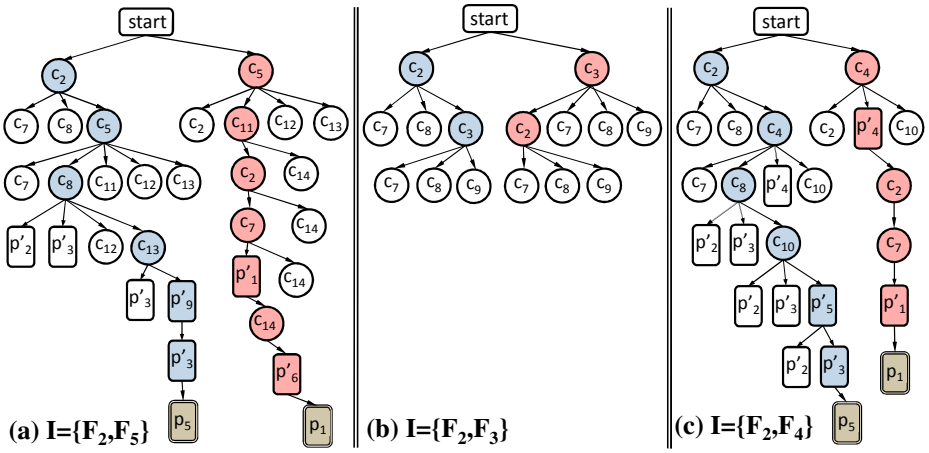


Figure 2.11: The sequences of navigational steps.

between facets is query- and data-dependent, rendering such exhaustive computation practically infeasible.

However, the "integrated" facet does shed light on what are the characteristics of good faceted interfaces. In general an interface should not include two facets that overlap much. Imagine a special case when two facets form a subsumption relationship, i.e., the root of one facet is a supercategory of the other root. Presenting both facets would not be desirable since they overlap significantly, thus cannot capture the expected properties of reaching target articles through different dimensions. As a concrete example, consider the navigational steps of $\mathcal{F}_2$ and $\mathcal{F}_3$ in Figure 2.11(b). After the user selects c_2 from $\mathcal{F}_2$ and then c_3 from $\mathcal{F}_3$, the available choices become $\{c_7, c_8, c_9\}$, which all come from the "dimension" $\mathcal{F}_3$. The same happens if the user selects c_3 and then c_2.

Based on the above observation, we propose to capture the overlap of k facets by their *average pair-wise similarity*. The pair-wise similarity of two facets is the degree of overlap of their category hierarchies and associated attribute entities, defined below.

Definition 8 (Average Similarity of k-Facet Interface)**.** *The average pair-wise similarity of a k-facet interface is*

$$sim(\mathcal{I} = \{\mathcal{F}_1, ..., \mathcal{F}_k\}) = \frac{\sum_{1 \le i < j \le k} sim(\mathcal{F}_i, \mathcal{F}_j)}{k(k-1)/2} \tag{2.10}$$

where the similarity between two facets $sim(\mathcal{F}_i, \mathcal{F}_j)$ is defined by an extension of overlap coefficient *[17],*

$$sim(\mathcal{F}_i, \mathcal{F}_j) = \frac{|\mathcal{C}_{\mathcal{F}_i} \bigcap \mathcal{C}_{\mathcal{F}_j}| + |\mathcal{A}_{\mathcal{F}_i} \bigcap \mathcal{A}_{\mathcal{F}_j}|}{\min(|\mathcal{C}_{\mathcal{F}_i}|, |\mathcal{C}_{\mathcal{F}_j}|) + \min(|\mathcal{A}_{\mathcal{F}_i}|, |\mathcal{A}_{\mathcal{F}_j}|)} \tag{2.11}$$

where $\mathcal{C}_{\mathcal{F}_i}$ is the set of categories in $\mathcal{F}_i$ (Definition 3) and $\mathcal{A}_{\mathcal{F}_i}$ is the set of attribute entities reachable from $\mathcal{F}_i$,

$$\mathcal{A}_{\mathcal{F}_i} = \{p' | p' \in \mathcal{A} \wedge \exists c \in \mathcal{C}_{\mathcal{F}_i} \; s.t. \; c \Rightarrow p'\} \tag{2.12}$$

Example 8 (Similarity of Facets)**.** *Consider facets $\mathcal{F}_1$,. . .,$\mathcal{F}_5$ in Figure 2.8.* $sim(\mathcal{F}_2, \mathcal{F}_3)$= $\frac{|\mathcal{C}_{\mathcal{F}_2} \bigcap \mathcal{C}_{\mathcal{F}_3}| + |\mathcal{A}_{\mathcal{F}_2} \bigcap \mathcal{A}_{\mathcal{F}_3}|}{\min(|\mathcal{C}_{\mathcal{F}_2}|, |\mathcal{C}_{\mathcal{F}_3}|) + \min(|\mathcal{A}_{\mathcal{F}_2}|, |\mathcal{A}_{\mathcal{F}_3}|)}$= $\frac{|\{c_7, c_8\}| + |\{p'_1, p'_2, p'_3\}|}{\min(|\{c_2, c_7, c_8\}|, |\{c_3, c_7, c_8, c_9\}|) + \min(|\{p'_1, p'_2, p'_3\}|, |\{p'_1, p'_2, p'_3, p'_4\}|)}$=5/6. *Other pari-wise facet similarities are computed in the same way. The average pari-wise similarity of faceted interface $\mathcal{I}$=$\{\mathcal{F}_2, \mathcal{F}_3, \mathcal{F}_5\}$ is* $sim(\mathcal{I}) = (sim(\mathcal{F}_2, \mathcal{F}_3) + sim(\mathcal{F}_2, \mathcal{F}_5) + sim(\mathcal{F}_3, \mathcal{F}_5))/3 = 5/18$.

For multi-facet ranking, we do not design a single function to combine the average pair-wise similarity of facets in a faceted interface with its navigational cost, since the measures of similarity and navigational cost are of different natures. Instead, in Section 2.3.3 we discuss how to search the space of candidate interfaces by considering both measures.

2.3 Algorithms

A straightforward approach for faceted interface discovery is to enumerate all possible k-facet interfaces with respect to category hierarchy $\mathcal{H}$ and apply our ranking metrics directly to find the best interface. Such a brute-force method results in the exhaustive ex-

amination of all possible combinations of k instances of all possible facets, i.e., rooted and connected subgraphs of $\mathcal{H}$. Clearly it is a prohibitively large search space, given the sheer size and complexity of the category system in Wikipedia. The brute-force technique would be extremely costly. Therefore finding the best k-facet interface is a challenging optimization problem.

Our k-facet discovery algorithm hinges on (1) reducing the search space; and (2) searching the space effectively and efficiently.

Reducing the Search Space: There are two search spaces in finding a good k-facet interface: the space of facets and the space of k-facet interfaces, which are sets of k facets. To reduce the space of candidate facets, we focus on a subset of the safe reaching facets, $\mathcal{RCH}$-induced facets, which are the facets that contain all the descendant categories of their roots (Section 2.3.1). To further reduce the space of faceted interfaces, we rank the facets individually by their navigational costs (Section 2.3.2) and only consider the top ranked facets that do not subsume each other (Section 2.3.3).

Searching the Space: Instead of exhaustively examining all possible interfaces, we design a hill-climbing based heuristic algorithm to look for a local optimum (Section 2.3.3). To further tackle the challenge of modeling the interactions of multiple facets in measuring the cost of an interface, the hill climbing algorithm optimizes for both the average navigational cost and the pair-wise similarity of the facets.

Based on these ideas, our k-facet discovery algorithm consists of three steps: construction of relevant category hierarchy, ranking single facet, and searching for k-facet interface.

2.3.1 Relevant Category Hierarchy (Algorithm 1)

By Definition 5, the facets in a faceted interface must be safe reaching facets, i.e., they do not contain "dead end" categories that cannot reach any target articles. Therefore

Algorithm 1: Construct RCH and Get Attribute Entities

Input: $\mathcal{T}$: target articles; $\mathcal{H}$: category hierarchy.

Output: $\mathcal{A}$:attribute entities; $\mathcal{RCH}$:relevant category hierarchy.

// get attribute entities.

$\mathcal{A} \leftarrow \emptyset$; $\mathcal{C}_{\mathcal{RCH}} \leftarrow \emptyset$; $\mathcal{E}_{\mathcal{RCH}} \leftarrow \emptyset$

foreach $p \in \mathcal{T}$ **do**

  foreach $p \rightarrow p'$ **do**

    $\mathcal{A} \leftarrow \mathcal{A} \cup \{p'\}$

// start from the categories of attribute entities.

foreach $p' \in \mathcal{A}$ **do**

  foreach $c \Rightarrow p'$, *i.e., a category of* p' **do**

    $\mathcal{C}_{\mathcal{RCH}} \leftarrow \mathcal{C}_{\mathcal{RCH}} \cup \{c\}$

// recursively obtain the supercategories.

$\mathcal{C} \leftarrow \mathcal{C}_{\mathcal{RCH}}$; $\mathcal{C}' \leftarrow \emptyset$

while $\mathcal{C}$ *is not empty* **do**

  foreach $c \in \mathcal{C}$ **do**

    foreach $c' \dashrightarrow c \in \mathcal{E}_{\mathcal{H}}$ **do**

      $\mathcal{E}_{\mathcal{RCH}} \leftarrow \mathcal{E}_{\mathcal{RCH}} \cup \{c' \dashrightarrow c\}$

      if $c' \notin \mathcal{C}_{\mathcal{RCH}}$ **then**

        $\mathcal{C}_{\mathcal{RCH}} \leftarrow \mathcal{C}_{\mathcal{RCH}} \cup \{c'\}$; $\mathcal{C}' \leftarrow \mathcal{C}' \cup \{c'\}$

  $\mathcal{C} \leftarrow \mathcal{C}'$; $\mathcal{C}' \leftarrow \emptyset$

return $\mathcal{A}$ *and* $\mathcal{RCH}(r_{\mathcal{H}}, \mathcal{C}_{\mathcal{RCH}}, \mathcal{E}_{\mathcal{RCH}})$

the categories appearing in any safe reaching facet could only come from the *relevant category hierarchy* ($\mathcal{RCH}$), which is a subgraph of the Wikipedia category hierarchy $\mathcal{H}$, defined below.

Definition 9 (Relevant Category Hierarchy)**.** *Given category hierarchy $\mathcal{H}(r_{\mathcal{H}}, \mathcal{C}_{\mathcal{H}}, \mathcal{E}_{\mathcal{H}})$, target articles $\mathcal{T}$, and attribute entities $\mathcal{A}$, the* relevant category hierarchy *($\mathcal{RCH}$) of $\mathcal{T}$ is a subgraph of $\mathcal{H}$. Given any category in $\mathcal{RCH}$, it is either directly a category of some attribute entity $p' \in \mathcal{A}$ or a supercategory or ancestor of such categories. There exists an edge (category-subcategory relationship) between two categories in $\mathcal{RCH}$ if the same edge exists in $\mathcal{H}$. By this definition the root of $\mathcal{H}$ is also the root of $\mathcal{RCH}$.*

The procedural algorithm for getting $\mathcal{RCH}$ is in Algorithm 1. Based on definition, straightforwardly we can prove every safe reaching facet of the target articles $\mathcal{T}$ is a (rooted and connected) subgraph of $\mathcal{RCH}$. However, not every rooted and connected subgraph of $\mathcal{RCH}$ is a safe reaching facet. Therefore, even though $\mathcal{RCH}$ is much smaller than $\mathcal{H}$, the search space is still very large. Hence we further shrink the space by considering only one type of safe reaching facets, the *$\mathcal{RCH}$-induced facets*.

Definition 10 ($\mathcal{RCH}$-Induced Facet)**.** *Given the relevant category hierarchy $\mathcal{RCH}$ of target articles $\mathcal{T}$, a facet $\mathcal{F}(r, \mathcal{C}_{\mathcal{F}}, \mathcal{E}_{\mathcal{F}})$ is* $\mathcal{RCH}$-induced *if it is a rooted induced subgraph of $\mathcal{RCH}$, i.e., in $\mathcal{F}$ all the descendants of the root r and their category-subcategory relationships are retained from $\mathcal{RCH}$.*

Example 9 ($\mathcal{RCH}$ and $\mathcal{RCH}$-Induced Facet)**.** *Continue the running example. In Figure 2.8, the $\mathcal{RCH}$ contains all the categories in the category hierarchy $\mathcal{H}$ except c_6 (and thus the edge $c_2 \dashrightarrow c_6$), since c_6 cannot reach any target article. $\mathcal{F}_2$ is an $\mathcal{RCH}$-induced facet, but would not be if it does not contain c_7 (or c_8).*

Note that every $\mathcal{RCH}$-induced facet is safe reaching, and the single-facet ranking and the searching for k-facet interfaces are performed on it.

Algorithm 2: Facet Ranking

Input: $\mathcal{T}$:targets;$\mathcal{A}$:attributes; $\mathcal{RCH}$:relevant category hierarchy.

Output: $\mathcal{I}_n$: top n $\mathcal{RCH}$-induced facets with smallest costs.

// get reachable target articles for each attribute entity.

foreach $p' \in \mathcal{A}$ **do**

  $\mathcal{T}_{p'} \leftarrow \{p | p \in \mathcal{T} \wedge \exists\, p \rightarrow p'\}$

  $fanout(p') \leftarrow |\mathcal{T}_{p'}|$

initialize $visited(r)$ to be $False$ for every $r \in \mathcal{C}_{\mathcal{RCH}}$.

$ComputeCost(r_{\mathcal{H}})$ // recursively compute the costs of all $\mathcal{RCH}$-induced facets, starting from the root of $\mathcal{RCH}$.

$\mathcal{I}_n \leftarrow$ the top n $\mathcal{RCH}$-induced facets with the smallest costs.

return $\mathcal{I}_n$

2.3.2 Ranking Single Facet (Algorithm 2 and 3)

Among all $\mathcal{RCH}$-induced facets, only top n facets with the smallest navigational costs are considered in searching for a k-facet interface ($k < n$). In ranking facets by their costs, one straightforward approach is to enumerate all $\mathcal{RCH}$-induced facets and to separately compute the cost of each facet by enumerating all of its navigational paths. This approach is exponentially complex due to repeated traversal of the edges in $\mathcal{RCH}$, because $\mathcal{RCH}$-induced facets would have many common categories and category-subcategory relationships.

To avoid the costly exhaustive method, we design a recursive algorithm that calculates the navigational costs of all $\mathcal{RCH}$-induced facets by only one pass depth-first search of $\mathcal{RCH}$. The details are in Algorithm 2. The essence of the algorithm is to, during the recursive traversal of $\mathcal{RCH}$, record the number of navigational paths in a facet in addition

Algorithm 3: ComputeCost(r)

Input: r: the root of an $\mathcal{RCH}$-induced facet.

Output: $cost(\mathcal{F}_r)$: cost of $\mathcal{F}_r$; $cost(\mathcal{F}_r, p)$: average cost of reaching target article p from $\mathcal{F}_r$; $pathcnt(\mathcal{F}_r, p)$: number of navigational paths reaching p from $\mathcal{F}_r$; $\mathcal{T}_r$: reachable target articles of r.

if $visited(r)$ **then**
  return
$visited(r) \leftarrow True$;
$\mathcal{C}_r \leftarrow \{c | r \dashrightarrow c \in \mathcal{E}_{\mathcal{RCH}}\}$ // subcategories of r.
foreach $c \in \mathcal{C}_r$ **do**
  $ComputeCost(c)$
$\mathcal{A}_r \leftarrow \{p' | p' \in \mathcal{A} \wedge r \Rightarrow p'\}$ // attribute entities belonging to r.
$fanout(r) \leftarrow |\mathcal{A}_r| + |\mathcal{C}_r|$
$\mathcal{T}_r \leftarrow (\cup_{p' \in \mathcal{A}_r} \mathcal{T}_{p'}) \bigcup (\cup_{c \in \mathcal{C}_r} \mathcal{T}_c)$ // reachable target articles of r.
foreach $p \in \mathcal{T}_r$ **do**
  $pathcnt(\mathcal{F}_r, p) \leftarrow |\{p' | p' \in \mathcal{A}_r, p \in \mathcal{T}_{p'}\}| + \sum_{c \in \mathcal{C}_r} pathcnt(\mathcal{F}_c, p)$
  $cost_1 \leftarrow \sum_{p' \in \mathcal{A}_r s.t. p \in \mathcal{T}_{p'}} (log_2(fanout(r)) + log_2(fanout(p')))$
  $cost_2 \leftarrow \sum_{c \in \mathcal{C}_r} (log_2(fanout(r)) + cost(\mathcal{F}_c, p)) \times pathcnt(\mathcal{F}_c, p)$
  $cost(\mathcal{F}_r, p) \leftarrow \frac{cost_1 + cost_2}{pathcnt(\mathcal{F}_r, p)}$
$cost(\mathcal{F}_r) \leftarrow \sum_{p \in \mathcal{T}_r} cost(\mathcal{F}_r, p) + penalty \times |\mathcal{T} - \mathcal{T}_r|$
return

Algorithm 4: k-Facet Interface Selection

Input: $\mathcal{I}_n$: the top n $\mathcal{RCH}$-induced facets with the smallest costs.

Output: $\mathcal{I}_k$: a discovered faceted interface with k facets ($k<n$).

// remove subsumed facets from $\mathcal{I}_n$

$\mathcal{I}_{n^-} \leftarrow \{\mathcal{F}_c | \nexists \mathcal{F}_{c'} \in \mathcal{I}_n$ s.t.$\mathcal{F}_c$ is subsumed by $\mathcal{F}_{c'}$, i.e., c is a descendant category of $c'\}$

// hill climbing

$\mathcal{I}_k \leftarrow$ a random k-facet subset of $\mathcal{I}_{n^-}$; $\mathcal{I}' \leftarrow \mathcal{I}_{n^-} \backslash \mathcal{I}_k$

repeat

  make $\mathcal{I}_k$=<$\mathcal{I}_k[1]$,...,$\mathcal{I}_k[k]$> sorted in increasing order of cost.

  make $\mathcal{I}'$=<$\mathcal{I}'[1]$,...,$\mathcal{I}'[n-k]$> sorted in increasing order of cost

  for $i = k$ **to** 1 **step** -1 **do**

    for $j = 1$ **to** $n-k$ **do**

      $\mathcal{I}_{new} \leftarrow (\mathcal{I}_k \backslash \{\mathcal{I}_k[i]\}) \cup \{\mathcal{I}'[j]\}$

      $S_1 \leftarrow \sum_{\mathcal{F}_c, \mathcal{F}_{c'} \in \mathcal{I}_{new}, \mathcal{F}_c \neq \mathcal{F}_{c'}} sim(\mathcal{F}_c, \mathcal{F}_{c'})$

      $C_1 \leftarrow \sum_{\mathcal{F}_c \in \mathcal{I}_{new}} cost(\mathcal{F}_c)$

      $S_2 \leftarrow \sum_{\mathcal{F}_c, \mathcal{F}_{c'} \in \mathcal{I}_k, \mathcal{F}_c \neq \mathcal{F}_{c'}} sim(\mathcal{F}_c, \mathcal{F}_{c'})$

      $C_2 \leftarrow \sum_{\mathcal{F}_c \in \mathcal{I}_k} cost(\mathcal{F}_c)$

      if *(*$S_1 \leq S_2$ **and** $C_1 < C_2$*)* **or** *(*$S_1 < S_2$ **and** $C_1 \leq C_2$*)* **then**

        $\mathcal{I}_k \leftarrow \mathcal{I}_{new}$; $\mathcal{I}' \leftarrow \mathcal{I}_{n^-} \backslash \mathcal{I}_k$

        go to line 5

until $\mathcal{I}_k$ *does not change*;

return $\mathcal{I}_k$

to its navigational cost. The bookkeeping is performed for each reachable target article because the cost is averaged across all such articles by Definition 7. The cost of a facet rooted at r can be fully computed based on the recorded information of the facets rooted at r's direct subcategories, without accumulating the individual costs of the facets rooted at r's descendants. Therefore it avoids the aforementioned repeated traversal of $\mathcal{RCH}$. More specifically, the lines 11-14 in Algorithm 3 are for computing $cost(\mathcal{F}_r, p)$ in Formula 2.7. However, the algorithm does not compute it by a direct translation of Formula 2.8 and 2.1, i.e., enumerating all the navigational paths that reach p. Instead, line 12 gets $cost_1$, the total cost of all the navigational paths $r \Rightarrow p' \leftarrow p$, i.e., the ones that reach p without going through any other categories; line 13 computes $cost_2$, the total cost of all the navigational paths that go through other categories, by utilizing $cost(\mathcal{F}_c, p)$ and $pathcnt(\mathcal{F}_c, p)$ of the subcategories c, but not other descendants. We omit the formal correctness proof.

2.3.3 Searching for k-Facet Interface (Algorithm 4)

Algorithm 4 searches for k-facet interface. To reduce the search space, our algorithm only considers $\mathcal{I}_n$, the top n facets from Algorithm 2. We further reduce the space by excluding those top ranked facets that are subsumed by other top facets (line 1). In other words, we only keep $\mathcal{I}_{n^-}$, the maximal *antichain* of $\mathcal{I}_n$ based on the graph (category hierarchy) subsumption relationship. This is in line with the idea of avoiding large overlap between facets (Section 2.2.2).

Given $\mathcal{I}_{n^-}$, instead of exhaustively considering all possible k-element subsets of $\mathcal{I}_{n^-}$, we apply a *hill-climbing method* to search for a local optimum, starting from a random k-facet interface $\mathcal{I}_k$. At every step, we try to find a better neighboring solution, where a k-facet interface $\mathcal{I}_{new}$ is a neighbor of $\mathcal{I}_k$ if they only differ by one facet (line 9). Given the $k \times (n-k)$ possible neighbors at every step, we examine them in the order of average navigational costs (line 5, 6, and 9). The algorithm jumps to the first encountered better

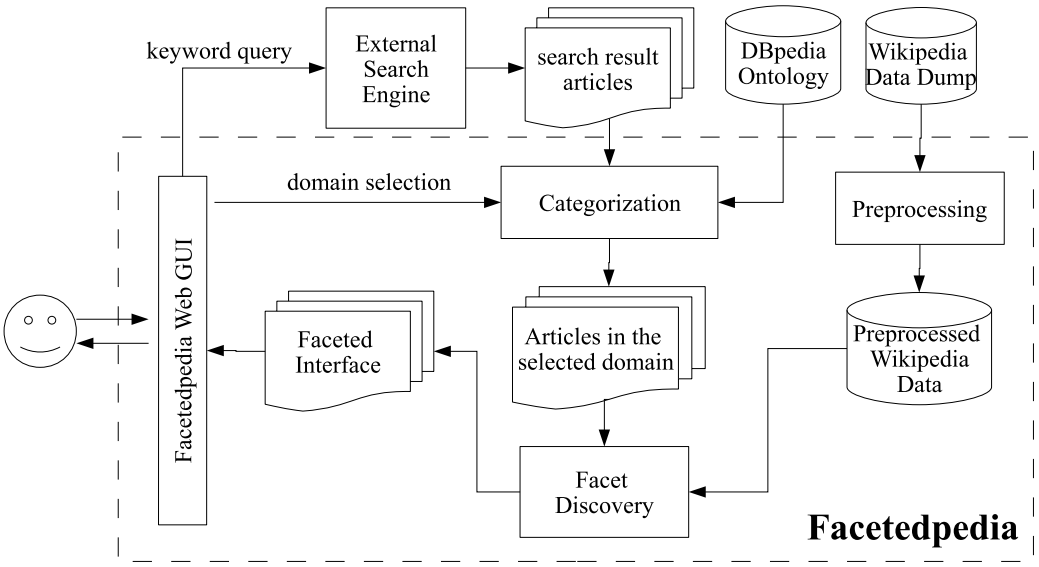


Figure 2.12: The architecture of Facetedpedia.

neighbor. The algorithm stops when no better neighbor can be found. As the goal function to be optimized in hill-climbing, $\mathcal{I}_{new}$ is considered better if the facets of $\mathcal{I}_{new}$ have both smaller pair-wise similarities and smaller navigational costs than that of $\mathcal{I}_k$ (line 14). The idea of considering both similarity and cost is motivated in Section 2.2.2.

2.4 System Implementation

The generic model (Section 2.1), ranking metrics (Section 2.2), and algorithms (Section 2.3) are instantiated into two prototype systems: Facetedpedia and Facetednews. Below we introduce the implementation details of both systems.

2.4.1 Facetedpedia

In Facetedpedia, the targets are Wikipedia entities (i.e., articles). The attributes of a target are Wikipedia entities that co-occur at least twice with the target entity in the text corpus– Wikipedia itself. The Facetedpedia system mainly consists of four components: preprocessing Wikipedia data dump, categorization, facet discovery, and Facetedpedia web

Table 2.1: Characteristics of the Wikipedia dataset.

number of articles	2,445,642
number of hyperlinks between articles	109,165,108
average number of hyperlinks per article	45
number of distinct categories	329,007
average number of categories per article	3
number of category-subcategory relationships	731,097

GUI. The architecture of the system is shown in Figure 2.12. We further elaborate the implementation of the four components as follows.

- Preprocessing Wikipedia Data Dump

  We used the Wikimedia MySQL data dump generated on July 24th 2008[11] and loaded the data into our local database. In particular, we used the tables *page.sql*, *pagelinks.sql*, *categorylinks.sql*, and *redirect.sql*, which provide all the relevant data, including the hyperlinks between articles, categories of articles, and the category system. We performed several preprocessing tasks on these tables. One major preprocessing task is to clean the original category hierarchy to make it cycle-free. Although cycles should usually be avoided as suggested by Wikipedia, the category system in Wikipedia contains a small number of elementary cycles [12] (594 detected in the dataset) due to various reasons. We applied depth-first search algorithm to detect elementary cycles in the original dataset. The category hierarchy is made acyclic by removing the last encountered edge in each elementary cycle during the depth-first search. Other preprocessing steps include: removing tuples irrelevant to

[11] `http://download.wikimedia.org`

[12] A cycle is elementary if no vertices in the cycle (except the start/end vertex) appear more than once.

articles and categories; replacing redirect articles by their original articles; removing special articles such as lists and stubs. We also applied basic performance tuning of the database, including creating additional indexes on *page_id* in various tables. The characteristics of the dataset are summarized in Figure 2.1. The total size of the tables is 1.2GB.

- Categorization

  A faceted interface is more effective on a set of homogeneous target entities. In our implementation, we exploited the DBpedia ontology[13] for assigning Wikipedia entities to about 80 pre-determined domains (e.g., People, Places, etc). This is done offline and the categorization result of all Wikipedia entities is stored in a database table.

  When a user issues a keyword search query, the query is sent to an external search engine which returns a ranked list of Wikipedia entities (i.e., articles). The returned entities are most likely from different domains, thus Facetedpedia asks the user to select one particular domain of her interest. An alternative approach is to let Facetedpedia select the dominant or largest domain of result entities automatically. However, the user may like to select from other domains of her interest. A k-facet interface is then discovered for the top-s search result entities belonging to the chosen domain. In our implementation, we use Google.com as the external search engine.

- Facet Discovery

  The facet discovery component is a multi-thread background daemon program. The main process creates a new thread for each user session. The main process pre-loads all the preprocessed tables (1.2GB in total) into memory. After the user chooses a target domain, a new thread is created to run the facet ranking and search algorithms and generate the resulting faceted interface.

[13] `http://wiki.dbpedia.org/Ontology`

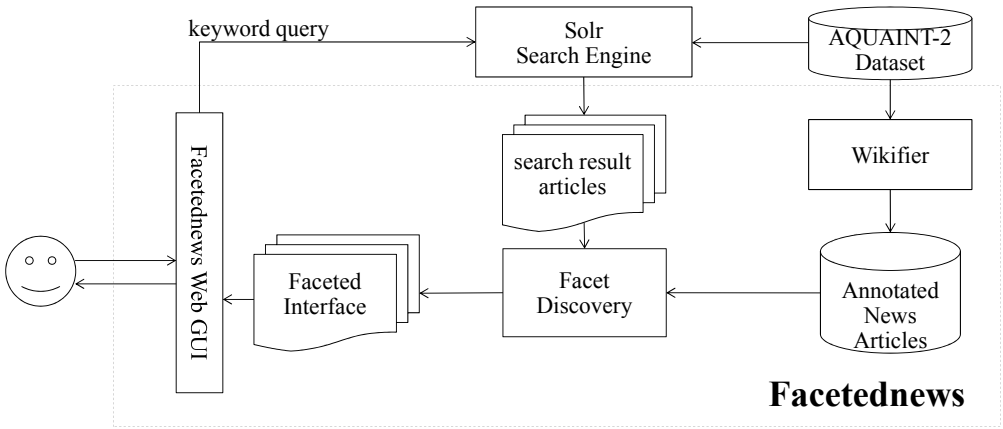


Figure 2.13: The architecture of Facetednews.

- Facetedpedia Web GUI

  The generated faceted interface, including information such as the category hierarchy of each facet and the entities reachable from each category in the hierarchy, is stored in a database. The GUI is a dynamic web page implemented using Ajax. It reads the generated interface data from the database, displays the faceted interface, and updates the interface based on the user's navigation.

2.4.2 Facetednews

In Facetednews, the targets are news articles and the attributes are Wikipedia entities, as mentioned in Section 2.1. Facetednews mainly consists of three components: preprocessing news data, facet discovery, and Facetednews web GUI. The system architecture is shown in Figure 2.13. In Facetednews we do not categorize news articles. We did not find it empirically important to require news articles to be "homogeneous" in order to make a faceted interface over the news articles effective. The implementation of facet discovery and web GUI components in Facetednews is similar to that in Facetedpedia. In order to apply our faceted interface discovery algorithm to news articles, we implemented two ad-

ditional functionalities to preprocess data– indexing and annotating news articles, which we further elaborate below.

- Indexing news articles using Apache Solr

  We used AQUAINT-2 dataset[14] as our news corpus. It consists of 907K news articles from 6 news agencies in the period of October 2004 – March 2006. We index the news articles and provide full-text search over the articles, by using the Apache Solr full-text search engine [15]. For a keyword query to the Facetednews web GUI, the search system returns a list of news articles, which become the target articles for facet discovery. We did not use a commercial news search engine to fetch news articles for queries, to avoid the overhead of query-time news article extraction and annotation. That way we can stay focused on our objective of investigating how to construct faceted interfaces over news articles.

- Annotating news articles using Wikifier

  We applied Wikifier [16] to annotate news articles, i.e., to identify Wikipedia entity names mentioned in the articles. For discovering faceted interfaces over the news articles, the identified entities are their attribute entities. Over the 907K news articles in AQUAINT-2 corpus, Wikifier detected over 13 million attribute entities, i.e., in average 14 attribute entities for each news article.

2.5 Evaluation

Our experiment was conducted on a Dell PowerEdge 2900 III server running Linux kernel 2.6.27, with dual quad-core Xeon 2.0 GHz processors, 2x6MB cache, 8GB RAM, and three 1TB SATA hard drivers in RAID5.

[14] http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2008T25

[15] http://lucene.apache.org/solr/

2.5.1 User Studies

We conducted user studies [16] to evaluate the effectiveness of both Facetedpedia and Facetednews. For faceted interface discovery over Wikipedia articles, we compared the results generated from three systems: Facetedpedia, Castanet [18], and Faceted Wikipedia Search [19]. We obtained the implementation of Castanet from its authors. Note that Castanet is intended for static, short, and domain-specific documents with limited vocabularies. Nevertheless, we applied Castanet on dynamic keyword search results over Wikipedia. We used the same graphical user interface for both systems to make the comparison independent from interface design difference. As for Faceted Wikipedia Search, we do not have the implementation of its internal algorithms. Thus, we utilized its online service for our user studies. This might lead to biases in system preference due to different GUI design. For faceted interface discovery over news articles, we compared the results generated from Facetedpedia and Castanet. As explained in Section 4.1, Faceted Wikipedia Search is inapplicable for faceted interfaces over general Web documents, since it uses Wikipedia infoboxes for generating facets.

In Facetedpedia, each query is sent to Google with site constraint *site:en.wikipedia.org* to get the top 200 (s=200) English Wikipedia articles. In Facetednews, each query is sent to our local Solr search engine to get the top 400 (s=400) news articles. The relevant category hierarchy ($\mathcal{RCH}$) is then generated by applying Algorithm 1 on the aforementioned MySQL database. By default, Algorithm 2 (facet ranking) returns top 200 (n=200) facets and Algorithm 4 (faceted interface selection) generates 20 facets (k=20). The value of $penalty$ in Definition 7 was empirically selected by investigating the relationship between number of unreachable target articles ($|\mathcal{T} - \mathcal{T}_r|$) and the total navigational cost of reachable targets ($\sum_{p \in \mathcal{T}_r} cost(\mathcal{F}_r, p)$).

[16] All the survey pages we used for our user studies are provided at `http://idir.uta.edu/facetsurvey/`.

us action film	computer scientist
American_actors_by_state	Association_of_American_Universities
American_writers	Liberal_democracies
American_film_directors	Software_companies_of_the_United_States
Academy_Awards	Companies_in_the_NASDAQ-100_Index
Liberal_democracies	Host_cities_of_the_Summer_Olympic_Games
Expatriates_in_the_United_States	History_of_human-computer_interaction
Film_directors_by_genre	Xerox
Film_score_composers_by_nationality	!!!_albums
Fictional_secret_agents_and_spies	Computer_hardware_companies
English_film_actors	Companies_established_in_the_1980s
us country singer	**best seller book**
Albums_by_year	American_writers_by_genre
Liberal_democracies	American_writers_by_state
Singles_by_year	English-language_films
Radio_formats	American_military_personnel_of_World_War_II
American_record_labels	Faculty_by_university_or_college_in_the_United_States
Companies_based_in_New_York_City	People_by_high_school_in_the_United_States
Nashville,_Tennessee	People_of_English_descent
2000s_films	American_actors_by_state
Spoken_articles	Agnostics_by_nationality
County_seats_in_Tennessee	Screenwriters_by_nationality

Figure 2.14: Root categories of 10 facets in the faceted interfaces generated by Facetedpedia for 4 queries.

nba	pc game
National_Basketball_Association_draft_picks	Companies_in_the_NASDAQ-100_Index
20th_century_births	Companies_established_in_the_1980s
National_Basketball_Association_players_by_club	Companies_listed_on_the_Hong_Kong_Stock_Exchange
National_Basketball_Association_teams	Dow_Jones_Industrial_Average
United_States_communities_with_African_American	Networking_hardware_companies
Basketball_players_at_the_2004_Summer_Olympics	Companies_established_in_the_1970s
DuPage_County,_Illinois	Companies_based_in_Tokyo
Port_cities_in_the_United_States	Entertainment_Software_Association
Olympic_basketball_players_by_country	20th_century_births
McDonald's_High_School_All-Americans	Companies_listed_on_the_Tokyo_Stock_Exchange
ford	**microsoft**
Car_manufacturers	Companies_in_the_NASDAQ-100_Index
Ford	Liberal_democracies
Liberal_democracies	Living_people
Federal_countries	American_chief_executives
Bus_manufacturers	Circulating_currencies
Truck_manufacturers	Currencies_of_Asia
Companies_based_in_Metro_Detroit	Web_service_providers
G8_nations	Microsoft_employees
Companies_listed_on_the_New_York_Stock_Exchange	Companies_established_in_the_1990s
Motor_vehicle_manufacturers_based_in_Michigan	American_billionaires

Figure 2.15: Root categories of 10 facets in the faceted interfaces generated by Facetednews for 4 queries.

Table 2.2: Querie Keywords for Evaluation.

Query ID	Facetedpedia	Query ID	Facetednews
1	us action film	1	nba
2	us national park	2	ford
3	us country singer	3	mobile phone
4	album	4	pc game
5	us film star	5	layoff
6	pc game	6	bankruptcy
7	computer scientist	7	tsunami
8	football player	8	president election
9	software	9	microsoft
10	best seller book	10	asia market
11	american writer	11	texas university
12	interstate highway	12	terrorism

Both Factedpedia and Facetednews were evaluated for 12 keyword queries, listed in Figure 2.2. For Facetedpedia, we made the query keywords distributed across different domains. For Facetednews, we chose keywords based on news events during the corresponding period of the news corpus. Figure 2.14 shows a sample of resulting facets produced by Facetedpedia for 4 of the queries. For each query, it shows the root categories of 10 facets in the faceted interface discovered by the system. Figure 2.15 shows the same for Facetednews.

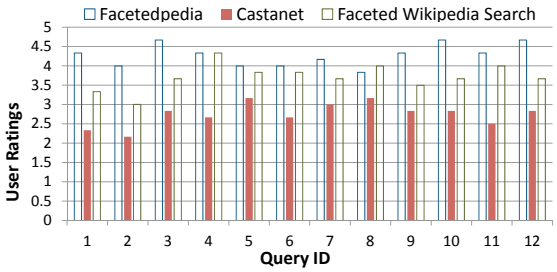
Both Factedpedia and Facetednews were evaluated by the same group of 18 voluntary users. For each system, we partitioned the 12 queries into 3 batches (4 queries in each batch) and asked each user to participate in one batch in evaluating the system's produced

faceted interfaces for the query results. Thus, each batch of queries were evaluated by 6 users.

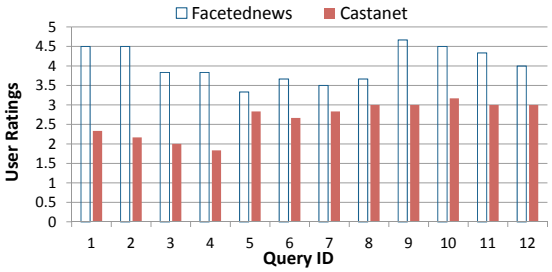
In evaluating the systems for a query, the query keywords and search objective description were shown to the users. The users were asked to explore the query's target articles using the generated faceted interfaces. For Wikipedia articles, the users were presented three different interfaces generated by Facetedpedia, Castanet, and Faceted Wikipedia Search, respectively. For news articles, the interfaces generated by Facetednews and Castanet were shown. After exploring these interfaces, the users were asked to provide responses in the form of ratings. The ratings were in 5-point scale– 1:"useless", 2: "not that useful", 3:"neutral", 4:"useful to some extent", 5:"very useful". The average ratings over the 12 queries are shown in Figure 2.16.

From Figure 2.16, we see that both Facetedpedia and Facetednews got higher ratings than Castanet in all queries. The results are due to the following main advantages of Facetedpedia over Castanet. First, Facetedpedia uses entities related to documents as their attributes, while Castanet only uses general thesaurus concepts. This makes the facet attributes in Facetedpedia more detailed and specific. Second, Facetedpedia uses Wikipedia category system for hierarchical organization of categories in facets, while to organize the general concepts, Castanet uses WordNet which is not as diverse and rich in semantics as the Wikipedia category system.

Figure 2.16 also shows that Facetedpedia outperformed Faceted Wikipedia Search in ratings. As discussed in Section 4.1, Faceted Wikipedia Search uses Wikipedia infoboxes in building facets. The infobox of a Wikipedia article is essentially a collection of attribute-value pairs related to the article. Such structured and table-like metadata makes the generated facets accurate. Given that Facetedpedia directly creates faceted interfaces for Web documents without relying on such metadata, the better ratings obtained by Facetedpedia verify the effectiveness of the proposed methods.



(a) Facetedpedia vs. other systems.



(b) Facetednews vs. other systems.

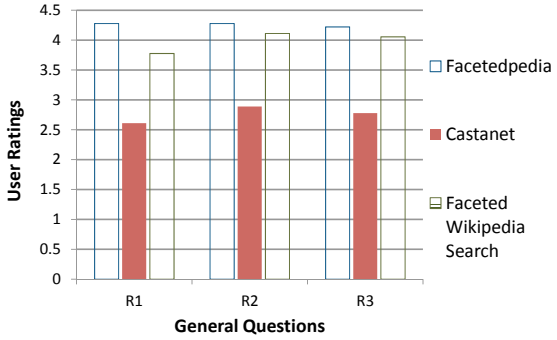
Figure 2.16: Average ratings of compared systems for 12 queries.

In addition to the pre-defined queries, each user was also asked to query the systems with open queries, i.e., arbitrary query keywords that the user came up with during user study. The user was then asked to provide response to three general questions $R1$-$R3$, as follows:

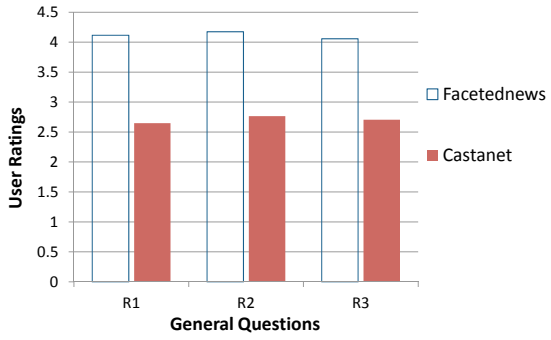
- $R1$: Does the system provide diverse and rich facets? (Being "diverse" means the generated facets cover the target articles from different angles and being "rich" means the generated facets have detailed information about the target articles' attributes.)
- $R2$: Does the system provide precise facets? (Being "precise" means the generated facets cover the target articles in a conceptually correct manner.)
- $R3$: Your overall rating of the system.

The responses to these general questions are also ratings at 5-point scale, ranging from 5 (the best score) to 1 (the worst score).

The rating results for the 3 general questions are shown in Figure 2.17. We see that Facetedpedia and Facetednews received much higher ratings than Castanet on all three questions. With regard to producing diverse and rich interfaces ($R1$), Facetedpedia received stronger ratings than Faceted Wikipedia Search. This is because Faceted Wikipedia Search does not produce query-dependent facets. For target Wikipedia entities in the same domain, it always uses the same set of facets. For instance, if the target entities are people, facets



(a) Facetedpedia vs. other systems.



(b) Facetednews vs. other systems.

Figure 2.17: Average ratings of compared systems for 3 general questions.

	coverage	average pairwise similarity	average category width	average path length
Hill-climbing	**96%**	0.134	**3.4**	**4.5**
Top-k	95%	0.231	3.7	4.8
Random-k	79%	**0.126**	4.5	8.2

Figure 2.18: Characteristics of faceted interfaces produced by various algorithms in Facetedpedia.

related to *age*, *gender*, *citizenship* will be used, no matter if they are politicians or actors. Hence query-dependent facets such as the movies related to the actors and the political events related to the politicians may not appear as facets. On question $R2$ and $R3$, the ratings of Faceted Wikipedia Search are very close to that of Facetedpedia. This can be due to that Faceted Wikipedia Search creates facets over structured infoboxes. Hence the facets are always accurate to some extent, no matter if the facets are useful for a particular query or not.

2.5.2 Characteristics of Generated Faceted Interfaces

In discovering faceted interfaces, our algorithm in Section 2.3 optimizes for mainly 3 objectives– high coverage of target articles, low overlap between multiple facets, and low navigational cost. To evaluate if the algorithms meet these objectives, we measured the characteristics of the faceted interfaces produced by our algorithms by four measures, as

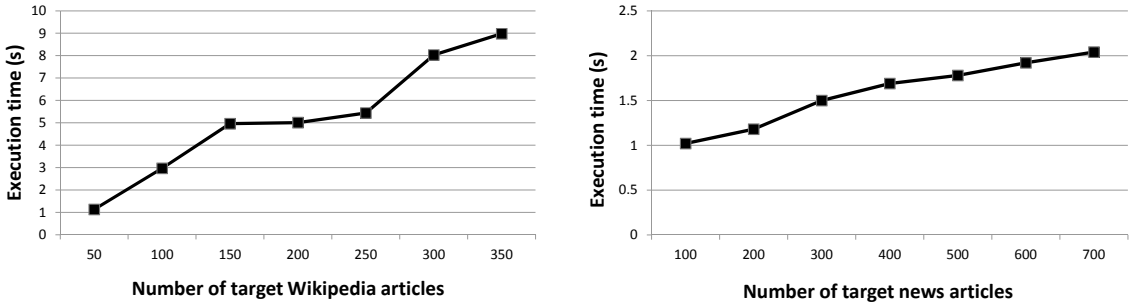
	coverage	average pairwise similarity	average category width	average path length
Hill-climbing	**93%**	**0.111**	**3.4**	**3.3**
Top-k	91%	0.218	3.5	3.6
Random-k	84%	0.190	31.5	11.7

Figure 2.19: Characteristics of faceted interfaces produced by various algorithms in Facetednews.

follows. (1) *Coverage*– the percentage of target articles that can be reached from a faceted interface; (2) *Average pairwise similarity* of the facets in a faceted interface– Equation 2.10; (3) *Average category width*– the average fan-out of all the categories in all k facets of a faceted interface; (4) *Average path length*– the average length of all possible navigational paths in all k facets of a faceted interface.

We measured these values for three algorithms: *hill-climbing* (Algorithm 4), *top-k* which selects the top k facets ranked by Algorithm 2, and *random-k* which chooses k random facets from the top n facets ranked by Algorithm 2. Figure 2.18 and Figure 2.19 show the measured results, for Facetedpedia and Facetednews, respectively. All the values are averaged over the queries listed in Figure 2.2.

From the results, we see that *hill-climbing* and *top-k* had much better coverage than *random-k*. This verifies that our single-facet ranking (Algorithm 2) is effective in choosing high quality individual facets. In terms of average pari-wise facet similarity, *hill-climbing* and *random-k* performed much better than *top-k*. This verifies that highly ranked facets may substantially overlap. Hence simply choosing the top k facets will result in redundant faceted interfaces, which are worse than even randomly chosen interfaces, in terms of pari-wise similarity. *hill-climbing* achieves not only high coverage but also small overlap. Looking into detailed intermediate results, we observed that the *hill-climbing* method started with choosing top k facets and gradually replaced some facets to make them less similar to each other, while still maintaining the high ranks of chosen facets. In terms of average category width and average path length, *hill-climbing* was slightly better than *top-k*



(a) Facetedpedia (b) Facetednews

Figure 2.20: Execution time of Facetedpedia and Facetednews.

and significantly better than *random-k*, which chose very wide or deep facets from time to time. The average category width and path length attained by *hill-climbing* were around 3 and 4. Therefore the fan-outs of categories and the lengths of navigational paths are within a reasonable range for users.

2.5.3 Efficiency Evaluation

We evaluated the scalability of our approach by measuring the average execution time of discovering k=20 facets for varying number of target articles (s from 50 to 350 for Facetedpedia and from 100 to 700 for Facetednews). As can be seen from Figure 2.20, both systems scaled well since the execution time increased linearly with the number of target articles. It also shows that both systems achieved fairly fast response without much performance optimization. (Our facet discovery program was running on a single server without exploiting optimization techniques such as parallel processing for ranking facets and memory caching of data or results.) In average it took 5 seconds for Facetedpedia to discover top 20 facets for 200 Wikipedia articles, and 1.5 seconds for Facetednews to discover top 20 facets for 200 news articles. Facetednews ran faster than Facetedpedia, since the attribute entities in news articles are usually not as diverse as the ones in Wikipedia

articles. Thus the size of relevant category hierarchy ($\mathcal{RCH}$) in Facetednews is usually much smaller than that in Facetedpedia.

CHAPTER 3

GENERATING PREVIEW TABLES FOR ENTITY GRAPHS

In this chapter, we study the problem of generating preview tables for entity graphs. Figure 3.1 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. 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 3.2 is a conceivable preview of the entity graph in Figure 3.1. 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. 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.

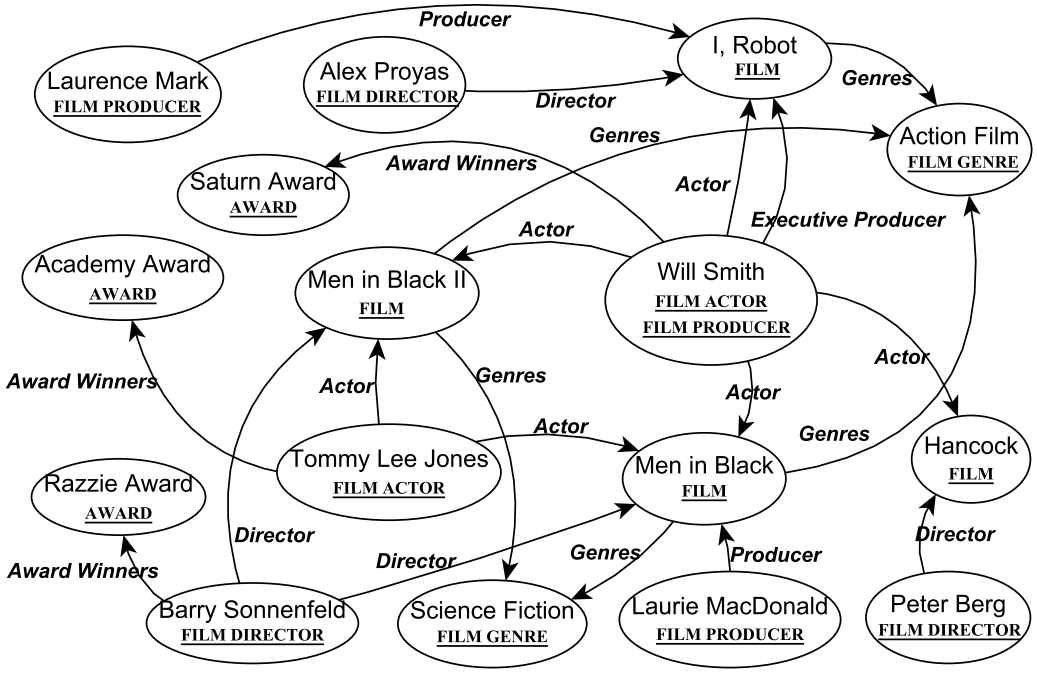


Figure 3.1: An Excerpt of an Entity Graph.

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.

The rest of this chapter is organized as follows. In Section 3.1, we define various concepts and the preview discovery problem. Section 3.2 presents our ideas on scoring measures. In Section 3.3, we formulate the optimal preview discovery problem and prove

	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 3.2: A Two-Table Preview of the Entity Graph in Figure 3.1. (The upper and lower tables are for the subgraphs #1 and #2 in Figure 3.3, respectively.)

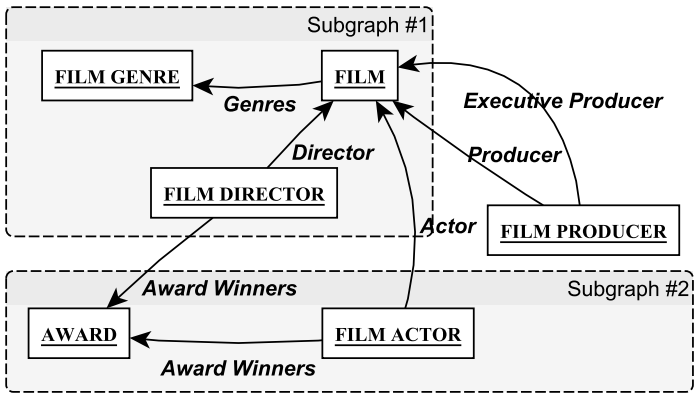


Figure 3.3: The Schema Graph for the Entity Graph in Figure 3.1.

its NP-completeness under distance constraint. Section 3.4 explains the algorithms. Section 3.5 presents the evaluation results.

3.1 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 3.1, there are two edges *Actor* and *Executive Producer* from entity Will Smith to entity I, Robot.)

Table 3.1: Notations

$G_d(V_d, E_d)$	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

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 3.1. (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 3.3 shows the schema graph corresponding to the entity graph in Figure 3.1. 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 11 (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 $T.\tau = \{v | v \in V_d \wedge v$ 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, $t.\gamma(\tau, \tau') = \{u | u \in V_d \wedge e(t.\tau, u) \in E_d \wedge u$ 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, $t.\gamma(\tau', \tau) = \{u | u \in V_d \wedge e(u, t.\tau) \in E_d \wedge u$ 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 11, the upper and lower tables in Figure 3.2 correspond to the star-shape subgraphs #1 and #2 in Figure 3.3, 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 3.2), duplicate (e.g., t_1.*Director*

and t_2.*Director* in Figure 3.2) and multi-valued (e.g., t_1.*Genres* and t_2.*Genres* in Figure 3.2). It also follows that a preview table is not a relational table.

By Definition 11, 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 11 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 3.3.

3.2 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 [20] and decision tree learning [21].

3.2.1 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 3.2.2 and 3.2.3.

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:

$$S(\mathcal{P}) = \sum_{i=1}^{k} S(\mathcal{P}[i]), \tag{3.1}$$

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

$$S(\mathcal{P}[i]) = S(\tau) \times \sum_{\gamma \in \mathcal{P}[i].nonkey} S^{\tau}(\gamma), \tag{3.2}$$

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.

3.2.2 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 3.1 and the corresponding schema graph in Figure 3.3, 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 $M_{\text{FILM,FILM GENRE}}$ = $w_{\text{FILM,FILM GENRE}}/(w_{\text{FILM,FILM GENRE}}$ + $w_{\text{FILM,FILM ACTOR}}$ + $w_{\text{FILM,FILM DIRECTOR}}$ + $w_{\text{FILM,FILM PRODUCER}}) = 5/(5 + 6 + 4 + 3) = 0.28$. The transition probability from FILM to FILM PRODUCER is $M_{\text{FILM,FILM PRODUCER}}$ = $w_{\text{FILM,FILM PRODUCER}}/(w_{\text{FILM,FILM GENRE}}$

+ $w_{\text{FILM,FILM ACTOR}}$ + $w_{\text{FILM,FILM DIRECTOR}}$ + $w_{\text{FILM,FILM PRODUCER}}$) = $3/(5+6+4+3)$ = 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 [7] for ranking relational tables by importance.

$$\pi = \pi M$$

The random-walk based score of a candidate key attribute τ_i is:

$$S_{walk}(\tau_i) = \pi_i, \text{where } \pi_i \text{ is the stationary probability of } \tau_i.$$

3.2.3 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 3.1 and the corresponding schema graph in Figure 3.3, 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., $u \in V_d \wedge u$ has type τ' and $n_j = |\{v | v \in T.\tau \wedge e(v, u) \in E_d$ (or $e(u, v) \in E_d) \wedge e$ 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 3.2), 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 $\gamma(\tau', \tau)$) $\in T.nonkey$, $S^{\tau}_{ent}(\gamma) \not\equiv S^{\tau'}_{ent}(\gamma)$.

3.3 Optimal Previews under Size and Distance Constraints

In this section, based on the scoring measures defined in Section 3.2, 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-complete.

By Equation 3.1 (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 3.2, 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[1] between their key attributes

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

$T_1.key$ 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 3.2 is 1, which is the shortest path length for entity types FILM and FILM ACTOR in schema graph in Figure 3.3. 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 3.5.

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 12 (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.* [2] *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 13 (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.*

$$\mathcal{P}_{opt} \in \arg\max_{\mathcal{P} \in \mathbb{P}} S(\mathcal{P}) \tag{3.3}$$

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-complete.

[2] 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 3.1).

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

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., $S(\mathcal{P}) \geq s$.

First, we show that $TightPreview(G_s, k, n, d, s)$ is in NP. This is because checking whether a preview satisfies the above three conditions can be accomplished in polynomial time. Consider a schema graph G_s, in which the scores of candidate key attributes (i.e., vertices of G_s) and non-key attributes (i.e., edges of G_s) are computed. Such checking includes counting tables and non-key attributes, computing pairwise table distance, and computing the preview's score by aggregating over the scores of its key and non-key attributes.

Next, we construct a reduction, in polynomial-time, from the NP-complete 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. A "yes" answer to $TightPreview(G_s, k, k, 1, 0)$ also confirms the existence of a clique of size k in G, i.e., a "yes" answer to $Clique(G, k)$. □

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

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

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., $S(\mathcal{P}) \geq s$.

First, $DisversePreview(G_s, k, n, d, s)$ is in NP. The proof is essentially the same as in Theorem 1 for $TightPreview(G_s, k, n, d, s)$.

Next, we construct a reduction, in polynomial-time, from the NP-complete $Clique(G, k)$ 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:

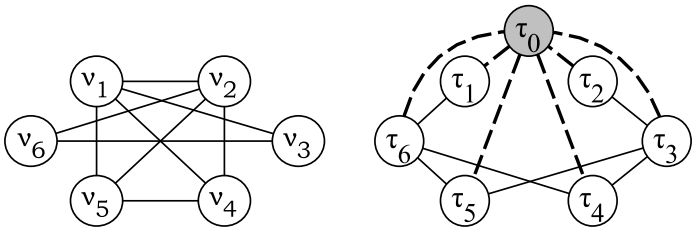


Figure 3.4: Construction of G_s (Right) from G (Left), for Reduction from the Clique Problem to the Optimal Diverse Preview Discovery Problem.

- $\forall \tau, \tau' \in V_s \backslash \{\tau_0\}$, $\gamma(\tau, \tau') \in E_s$ 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. A "yes" answer to $DiversePreview$ $(G_s, k, k, 2, 0)$ also confirms the existence of a clique of size k in G, i.e., a "yes" answer to $Clique(G, k)$. □

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 3.4. 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.

3.4 Algorithms

In this section we discuss algorithms for solving the optimal preview discovery problem. As given in Equation 3.3, 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.

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 3.2 and the problem formulation in Equation 3.3, 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 $S^{\tau}(\gamma) \geq S^{\tau}(\gamma)$.*

3.4.1 A Brute-Force Algorithm

This section presents a brute-force algorithm for the optimal preview discovery problem, shown in Algorithm 5. 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 11. 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.

Algorithm 5: Brute-Force Algorithm for Optimal Preview Discovery

Input : schema graph G_s, size constraint (k, n)

Output: an optimal preview $\mathcal{P}_{opt}$

foreach $\tau \in V_s$ **do**
  $\langle \gamma_1^\tau, \gamma_2^\tau, \ldots \rangle \leftarrow$ sort the candidate non-key attributes $\gamma_j^\tau \in \Gamma^\tau$ by their scores $S^\tau(\gamma_j^\tau)$;
$max\_score \leftarrow 0$; $\mathcal{P}_{opt} \leftarrow \emptyset$;
foreach *k-subset of V_s (denoted V)* **do**
  $score \leftarrow 0$; $\mathcal{P} \leftarrow \emptyset$; $i \leftarrow 1$;
  foreach $\tau \in V$ **do**
    $\mathcal{P}[i].key = \tau$;
    $\mathcal{P}[i].nonkey = \{\gamma_1^\tau\}$;
    $score = score + S(\tau) \times S^\tau(\gamma_1^\tau)$;
    $i \leftarrow i + 1$;
  $\Gamma \leftarrow$ top-$(n-k)$ candidate non-key attributes from all $\tau \in V$ in descending order of $S(\tau) \times S^\tau(\gamma_j^\tau)$;
  foreach $\gamma_j^\tau \in \Gamma$, *where* $\tau = \mathcal{P}[x].key$ **do**
    $score \leftarrow score + S(\tau) \times S^\tau(\gamma_j^\tau)$;
    $\mathcal{P}[x].nonkey \leftarrow \mathcal{P}[x].nonkey \bigcup \{\gamma_j^\tau\}$;
  if $score > max\_score$ **then**
    $max\_score \leftarrow score$;
    $\mathcal{P}_{opt} \leftarrow \mathcal{P}$;
return $\mathcal{P}_{opt}$;

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 5 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 5 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.

3.4.2 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—$\tau_1, \ldots, \tau_K$. 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.

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

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

where $T_x^m.key$ = τ_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 6. Its complexity is $O(KN\log N + Kkn^2)$. Similar to Algorithm 5, Algorithm 6 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 5 and 6 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 3.3, the optimal preview is not required to have exactly n non-key attributes. It is simple to extend Algorithm 5 and 6 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 6: 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 $\quad \langle \gamma_1^{\tau_x}, \gamma_2^{\tau_x}, \ldots \rangle \leftarrow$ 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 $\quad$ **for** $i \leftarrow 1$ **to** $\min(k, x)$ **do**
5 $\quad\quad$ **for** $j \leftarrow i$ **to** n **do**
6 $\quad\quad\quad \mathcal{P}_{opt}(i, j, x) \leftarrow \mathcal{P}_{opt}(i, j, x-1)$;
7 $\quad\quad\quad$ **for** $m \leftarrow 1$ **to** $min(j - i + 1, |\Gamma^{\tau_x}|)$ **do**
8 $\quad\quad\quad\quad T_x^m.key \leftarrow \tau_x$;
9 $\quad\quad\quad\quad T_x^m.nonkey \leftarrow$ top-m candidate non-key attributes in Γ^{τ_x};
10 $\quad\quad\quad\quad \mathcal{P} \leftarrow \mathcal{P}_{opt}(i-1, j-m, x-1) \bigcup \{T_x^m\}$;
11 $\quad\quad\quad\quad$ **if** $S(\mathcal{P}) > S(\mathcal{P}_{opt}(i, j, x))$ **then**
12 $\quad\quad\quad\quad\quad \mathcal{P}_{opt}(i, j, x) \leftarrow \mathcal{P}$;
13 $\mathcal{P}_{opt} \leftarrow \mathcal{P}_{opt}(k, n, K)$;
14 **return** $\mathcal{P}_{opt}$;

3.4.3 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 7. 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

Algorithm 7: 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 $\mathcal{P}_{opt}$;

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 [12] for frequent itemset mining. In [22], 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 [23]. 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 7 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 7). The details of function $ComputePreview$ are omitted. It follows Theorem 3 and is essentially the same as Lines 5– 14 in Algorithm 5, which is described in Section 3.4.1. The score of each preview is computed (details also the same as in Lines 5– 14 of Algorithm 5) and a preview with the highest score is returned.

3.5 Evaluation

We conducted experiments to evaluate the preview scoring measures' accuracy (Section 3.5.1), the preview discovery algorithms' efficiency (Section 3.5.2) as well as the overall quality of discovered previews (Section 3.5.3). 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.[3] 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 precomputed the schema graphs as well as all scoring measures in Section 3.2 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.

3.5.1 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 3.2. 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

[3] `https://developers.google.com/freebase/data`

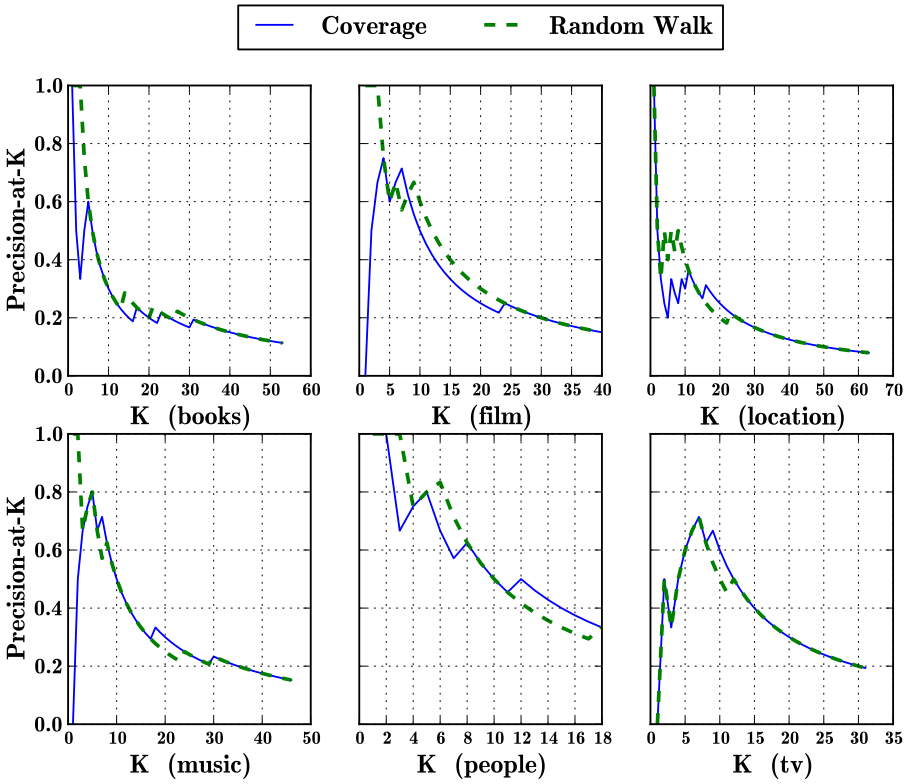


Figure 3.5: Precision-at-K of Key Attribute Scoring.

scoring measures and the pairwise order collected from user study using crowd-sourcing service.

3.5.1.1 Comparison with Gold Standard

Table 3.2: Mean Reciprocal Rank of Non-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

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. [4] 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 3.2.2, 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) [24]. 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 3.5. 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. [5] 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

[4] 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.

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

Freebase allows users to add more attributes 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 3.2.3, we ranked all candidate non-key attributes by their scores. We calculated the accuracy of a scoring measure by Mean Reciprocal Rank (MRR) [24] 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 3.2. 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.

3.5.1.2 User Study

Table 3.3: Pearson Correlation Coefficient for Key Attribute Scoring (Upper) and Non-Key Attribute Scoring (Lower).

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

We conducted an extensive user study in Amazon Mechanical Turk (AMT)[6]—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

[6] `https://www.mturk.com/mturk/`

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) [25] as follows.

$$PCC = \frac{\mathrm{E}(XY) - \mathrm{E}(X)\mathrm{E}(Y)}{\sqrt{\mathrm{E}(X^2) - (\mathrm{E}(X))^2}\sqrt{\mathrm{E}(Y^2) - (\mathrm{E}(Y))^2}} \tag{3.4}$$

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 3.3. 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).

3.5.2 Efficiency of Optimal Preview Discovery Algorithms

This section presents results on the efficiency of the optimal preview discovery algorithms in Section 3.4. On optimal concise preview discovery, we compared the Brute-Force Algorithm 5 and the Dynamic-Programming Algorithm 6. 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

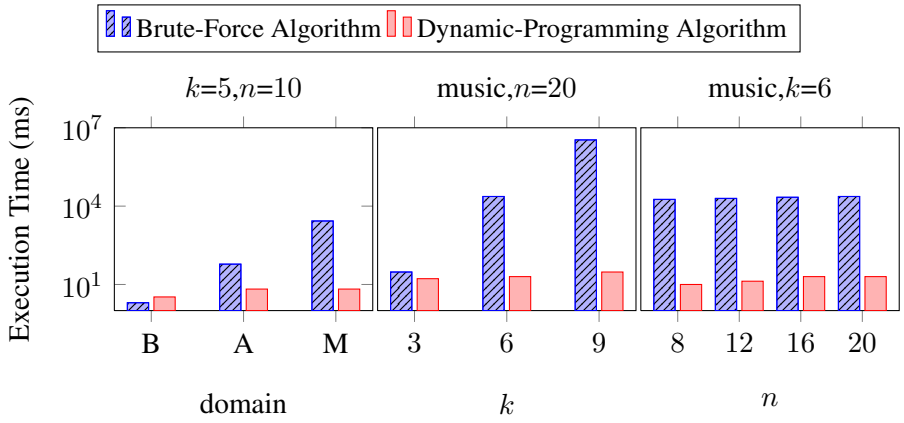


Figure 3.6: Efficiency Evaluation of Optimal Concise Preview Discovery Algorithms.

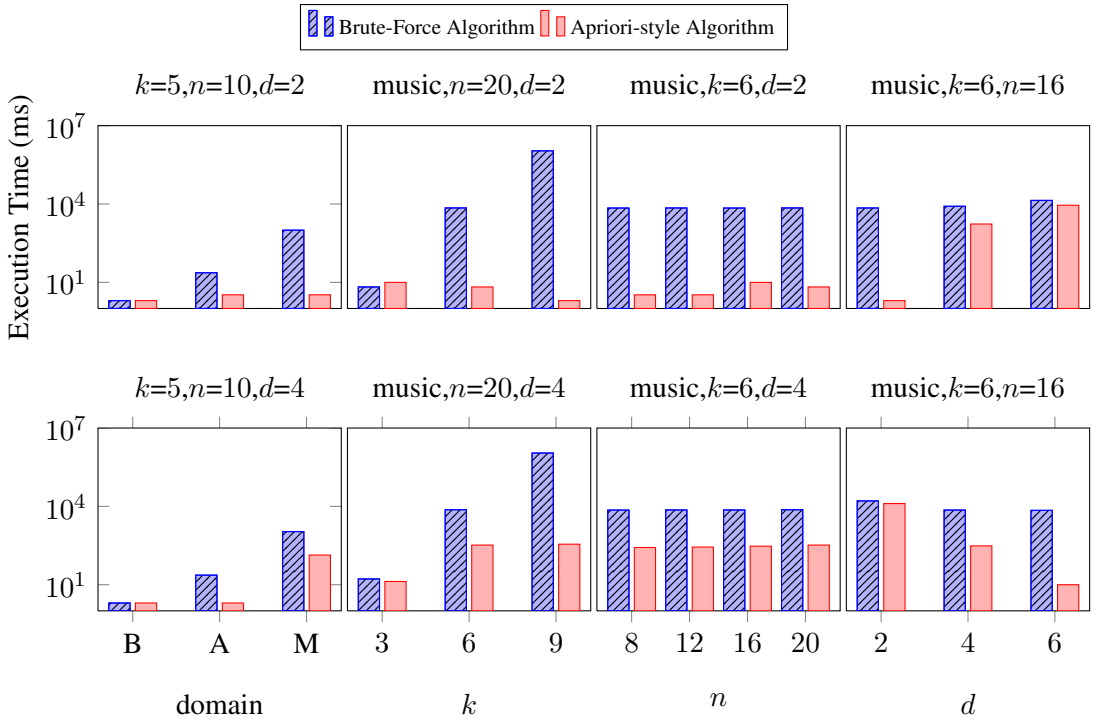


Figure 3.7: Efficiency Evaluation of Optimal Tight (Upper) and Diverse (Lower) Preview Discovery Algorithms.

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 5 and the Apriori-style Algorithm 7, 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 3.6 and Figure 3.7. 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 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 3.7 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.

Table 3.4: Samples of Optimal Concise Previews.

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

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)

3.5.3 Sample Optimal Previews

To demonstrate the combined effectiveness of both scoring measures and preview discovery algorithms, Table 3.4 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 3.5 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 3.4 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.

CHAPTER 4

RELATED WORK

This chapter reviews the related work of this dissertation. In Section 4.1, we conduct a comparative study of various faceted search systems based on two taxonomies that characterize faceted search systems. In Section 4.2, we summarize other work related to faceted search systems based on aspects such as personalization, query log, and user behavior modeling. In Section 4.3, we discuss research work related to generating preview tables for entity graphs.

4.1 Faceted Search Systems: A Comparative Study

Faceted interface has become influential over the last few years and we have seen an explosive growth of interests in its application [26, 27, 5, 27, 5, 18, 28, 29, 30, 31, 32, 33, 34, 19, 35, 36]. Commercial faceted search systems have been adopted by vendors (such as Endeca, IBM, and Mercado), as well as E-commerce websites (e.g., eBay.com, Amazon.com). The utility of faceted interfaces was investigated in various studies [26, 5, 37, 27, 38, 37, 39, 5], where it was shown that users engaged in exploratory tasks often prefer such result grouping over simple ranked result list (commonly provided by search engines), as well as over alternative ways of organizing retrieval results, such as clustering [40, 41, 38].

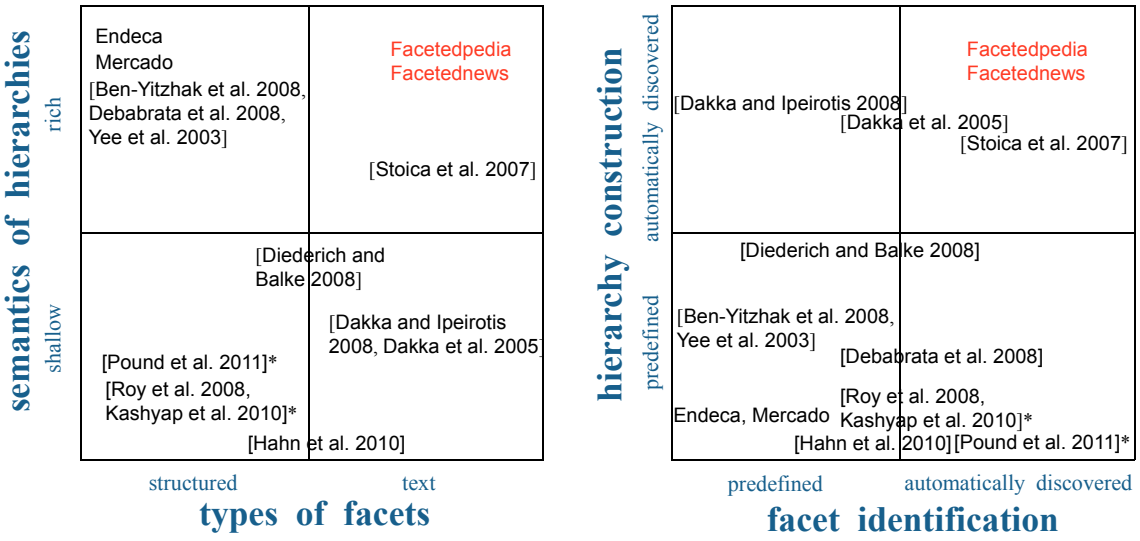
To the best of our knowledge, we are the first to propose a query-dependent faceted search framework that discovers both facet dimensions and category hierarchies dynamically. We also demonstrate our framework through two novel application systems: Facetedpedia and Facetednews. Existing research prototypes and commercial faceted search systems

mostly cannot be applied to meet our goals, because they either are based on manual or static facet construction, or are for structured records or text collections with prescribed metadata. Very few have investigated the problem of dynamic discovery of both facet dimensions and their associated category hierarchies.

There exist some previous works on enabling faceted search over Wikipedia. However, they are different from this work, as explained below. CompleteSearch [42] supports query completions and query refinement in Wikipedia by a special type of "facets" on three dimensions that are very different from our notion of general facets: query completions matching the query terms; category names matching the query terms; and categories of result articles. Recently, another faceted interface for Wikipedia, Faceted Wikipedia Search [19], has been developed as part of the DBpedia project [1]. Their facets are pre-extracted from Wikipedia infobox attributes. Therefore the facets are not dynamically built and hardly query-dependent, as the system often provides the same set of facets for different search result articles. On the contrary, Facetedpedia is fully dynamic and query-dependent. Moreover, it exploits more than 700K Wikipedia categories, in comparison with 1800 pre-extracted attributes from infobox in Faceted Wikipedia Search [19], which makes the facets generated by Facetedpedia more diverse and semantics-rich. Another advantage of our proposed approach is that it can be easily generalized for non-Wikipedia text documents, where infobox does not exist.

- Figure 4.1(a): Taxonomy by Facet Types and Semantics

  Previous systems roughly belong to two groups on this aspect. In some systems the facets are on relational data (e.g., Endeca, Mercado, [31, 35, 36]) or structured attributes in schemata (e.g., [27, 33, 34]) and the hierarchies on attribute values are predefined based on domain-specific taxonomies. The hierarchies could even be manually created, thus could contain rich semantic information. In some other systems a facet is a group of textual terms, over which the hierarchy is built upon thesaurus-



(a) Facet types and semantics.

(b) Automation and dynamism.

Figure 4.1: Taxonomies of faceted search systems. (Works marked by * do not support category hierarchy on facet.)

based IS-A relationships (e.g., [18]) or frequency-based subsumption relationships between general and specific terms (e.g., [28, 29]). These systems cannot leverage as much semantic information. The work [32] is in the middle of Figure 4.1(a) since it has both structured dimensions and a subsumption-based topic taxonomy. In Faceted Wikipedia Search [19] the facets are from metadata (i.e., attributes in Wikipedia infoboxes) of its target data. Hence it is in the middle along the dimension of type of facets.

In contrast, our framework enables semantics-rich facet hierarchies (distilled from Wikipedia category system) over text attributes (Wikipedia article titles). In the absence of predefined schemata, it builds facet hierarchies with abundant semantic information from the collaborative vocabulary in Wikipedia category system, instead of relying on IS-A or subsumption relationships.

- Figure 4.1(b): Taxonomy by Degree of Automation and Dynamism

When building the two pillars in a faceted interface, namely the facet and the hierarchy, our framework is both automatic and dynamic, as motivated in Chapter 2. On this aspect, none of the existing systems could be effectively applied, because none is fully automatic in both facet identification and hierarchy construction.

In some systems (e.g., Endeca, Mercado, [31, 34, 27, 33, 35]) the dimensions and hierarchies are predefined, therefore they do not discover the facets or construct the hierarchy. In [33, 31, 35, 19] a subset of interesting/important facets are automatically selected from the predefined ones. In [28, 29] the set of facets are predefined, but the hierarchies are automatically created based on subsumption. In [32] only one special facet (a topic taxonomy) is automatically generated and the rest are predefined. In [36] facets are automatically discovered, but it does not use hierarchies on facets.

With respect to the automation of faceted interface discovery, the closest work to ours is the Castanet algorithm [18]. The algorithm is intended for short textual descriptions with limited vocabularies in a specific domain. It automatically creates facets from a collection of items (e.g., recipes). The hierarchies for the multiple facets are obtained by first generating a single taxonomy of terms by IS-A relationships from WordNet and then removing the top-level nodes from the taxonomy.

4.2 Other Related Work to Faceted Search

In Section 4.1 we compare our work with prior systems that focus on constructing faceted interfaces. In this section we provide further discussion on other aspects of faceted search systems such as personalization, query log, and user behavior modeling. We also provide a brief discussion of related works on querying and exploring Wikipedia.

Koren et al. [43] studied how to incorporate user preferences into faceted search by a probabilistic user relevance model. This work also studied how to jump start the personalization by a collaborative user relevance model, when there is no cumulated preferences data for an individual user. In [44] the authors study faceted exploration of image search results. For building facets, they use the internal semi-structured data sources in a search engine related to images instead of image metadata. For several pre-defined domains (e.g. *locations*, *movies*, etc), they extract a number of relationships (e.g. *subsumes*, *played in*, *has cast*) to form facet dimensions. The work ranks facets based on statistical analysis of image search query logs and users' tagging behavior. Pound et al. [36] proposed a query-log mining approach that discovers facets for structured data sources from keyword search query logs. Kules et al. [45] applied techniques such as eye tracking, stimulated recall interviews, and direct observation to study user navigation behaviors over faceted interfaces. The results showed that faceted interfaces are useful in searching library catalogs. The study also measured the amount of time that users spend on each component of a faceted interface.

Various approaches have been pursued for enhancing keyword search on Wikipedia. PowerSet [1] uses natural language processing techniques to support simple questions and direct answers. CompleteSearch proactively supports query formulation (by presenting relevant completions) and query refinement through categories (by presenting matching categories) [42]. Several works explicitly support structured queries on Wikipedia. DBPedia [1] allows users to ask expressive queries against structured information extracted from Wikipedia. [46] uses relational tables to support SQL-style queries over the extracted information. [47, 48] studied how to rank resulting entities of keyword queries. Li et al. [49] propose a structured query mechanism, entity-relationship query, for searching entities in Wikipedia corpus by their properties and inter-relationships. An entity-relationship query

[1] `http://www.powerset.com`

consists of multiple predicates on desired entities. The semantics of each predicate is specified with keywords. Entity-relationship query searches entities directly over text instead of pre-extracted structured data stores. YAGO [2] supports semantic queries over a knowledge base on Wikipedia. Semantic Wikipedia [50] extends Wikipedia to allow users to manually specify the types of hyperlinks and data values in articles. [51] automatically creates and enhances various structures in Wikipedia, including infoboxes and link structures. Such manually or automatically generated information could be useful in creating faceted interfaces since they explicitly provide the attributes of articles and the relationships between articles.

4.3 Related Work to Generating Preview Tables for Entity Graphs

DataGuides [52] is among the earliest approach to constructing structural summaries for semi-structured databases. In semi-structured databases, there are usually no fixed schema available. Those databases are hard to use and comprehend for end users. The summary dynamically generated, i.e., DataGuides, servers as the meta data of for traditional databases. They are helpful for database browsing, query formulation as well as means guiding query processor and query optimization for semi-structured databases.

There are also many work related to schema summarization for relational databases [7, 8, 9], XML [9] and general graph data [10, 11]. Below we briefly summarize these work. The notion of summary in [7, 8] 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. Given a pre-defined relational schema, [7] generates a summary consists of several clusters of relational tables, which could potentially improve users understanding of the database schema. The clusters are obtained by a weighted k-center algorithm in which the weights of tables are determined by a random walk process over the

schema graph and the table distance or similarity is based on the number of tuples participating in the join relationship between two tables. [8] further studies, given a set of query tables, how to find the most relevant tables as well as direct join relationships and indirect join paths between the tables. The results preserve most important or informative join path between query tables and form a succinct summary graph for users' digest. [9] 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 summary balances coverage and importance of the elements in the schema tree. The graph summarization in [10, 11] groups graph nodes based on their attribute similarity and allows users to browse the summary from different grouping granularities. [10] enables OLAP-style operations (i.e., SNAP and k-SNAP) over graph data and derive graph summaries by grouping nodes based on the similarity of their categorical attributes. The SNAP operation generates a summary graph by grouping nodes based on user-selected node attributes and relationships, while the k-SNAP operation further allows users to control the resolutions of summaries and provides the drill-down and roll-up abilities to navigate through summaries with different resolutions. These operations helps users view the original graph at different granularities. [11] extends the method by automatically categorizing numerical attributes. It also studies how to find the most insightful summary from summaries of different granularities. In Chapter 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 work on graph clustering [53]. 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.

CHAPTER 5

CONCLUSION

In this dissertation, we studied two aspects of entity-centric information exploration: using entity graphs to browse external data sources and understanding entity graphs. We tackled two specific objectives: faceted interface discovery for Web documents and optimal preview discovery for entity graphs.

The first objective is to develop methods that discover query-dependent faceted interfaces dynamically and automatically to help users navigate Wikipedia articles and general Web documents. Toward this goal, we proposed a generic faceted search model that is instantiated into two faceted search systems– Facetedpedia and Facetednews. Given the sheer size and complexity of the exploited Wikipedia data, there is a prohibitively large space of possible faceted interfaces. We proposed metrics for ranking faceted interfaces as well as efficient algorithms for discovering them. Our experimental evaluation and user study verify the effectiveness of our methods in generating useful faceted interfaces over both Wikipedia and news articles.

The second objective is to develop methods that generate preview tables for entity graphs. The problem is challenging due to the scale and complexity of such graphs. We proposed scoring measures for both key attributes and non-key attributes in each of the individual preview tables, as well as the aggregation method for multiple preview tables. We proved that the optimal preview discovery problem under distance constraint is NP-complete. We designed efficient algorithms for discovering optimal previews. Our experiments on the Freebase dataset verified the accuracy and efficiency of our proposed methods.

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 *Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference (ISWC'07/ASWC'07)*, 2007, pp. 722–735.

[2] F. M. Suchanek, G. Kasneci, and G. Weikum, "YAGO: a core of semantic knowledge," in *Proceedings of the 16th international conference on World Wide Web (WWW)*, 2007, pp. 697–706.

[3] W. Wu, H. Li, H. Wang, and K. Q. Zhu, "Probase: A probabilistic taxonomy for text understanding," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '12. New York, NY, USA: ACM, 2012, pp. 481–492. [Online]. Available: http://doi.acm.org/10.1145/2213836.2213891

[4] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '08. New York, NY, USA: ACM, 2008, pp. 1247–1250. [Online]. Available: http://doi.acm.org/10.1145/1376616.1376746

[5] M. A. Hearst, "Clustering versus faceted categories for information exploration," *Commun. ACM*, vol. 49, pp. 59–61, April 2006.

[6] I. Herman, G. Melançon, and M. S. Marshall, "Graph visualization and navigation in information visualization: A survey," *IEEE Transactions on Visualization and Computer Graphics*, vol. 6, no. 1, pp. 24–43, Jan. 2000. [Online]. Available: http://dx.doi.org/10.1109/2945.841119

[7] X. Yang, C. M. Procopiuc, and D. Srivastava, "Summarizing relational databases," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 634–645, Aug. 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1687627.1687699

[8] ——, "Summary graphs for relational database schemas," *PVLDB*, vol. 4, no. 11, pp. 899–910, 2011.

[9] C. Yu and H. V. Jagadish, "Schema summarization," in *Proceedings of the 32Nd International Conference on Very Large Data Bases*, ser. VLDB '06. VLDB Endowment, 2006, pp. 319–330. [Online]. Available: http://dl.acm.org/citation.cfm?id=1182635.1164156

[10] Y. Tian, R. A. Hankins, and J. M. Patel, "Efficient aggregation for graph summarization," in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '08. New York, NY, USA: ACM, 2008, pp. 567–580. [Online]. Available: http://doi.acm.org/10.1145/1376616.1376675

[11] N. Zhang, Y. Tian, and J. M. Patel, "Discovery-driven graph summarization," in *Proceedings of the IEEE 24th International Conference on Data Engineering (ICDE)*, 2010, pp. 880–891.

[12] R. Agarwal and R. Srikant, "Fast algorithms for mining association rules," in *Proceedings of the 20th International Conference on Very Large Data Bases.*, 1994, pp. 487–499.

[13] C. Li, N. Yan, S. B. Roy, L. Lisham, and G. Das, "Facetedpedia: dynamic generation of query-dependent faceted interfaces for wikipedia," in *Proceedings of the 19th international conference on World wide web (WWW)*, 2010, pp. 651–660.

[14] N. Yan, C. Li, S. B. Roy, R. Ramegowda, and G. Das, "Facetedpedia: enabling query-dependent faceted search for wikipedia," in *Proceedings of the 19th ACM international conference on Information and knowledge management (CIKM)*, 2010, pp. 1927–1928.

[15] C. Fellbaum, "WordNet: An electronic lexical database." MIT Press, 1998.

[16] D. Milne and I. H. Witten, "Learning to link with wikipedia," in *Proceeding of the 17th ACM conference on Information and knowledge management (CIKM)*, 2008, pp. 509–518.

[17] C. J. V. Rijsbergen, *Information Retrieval*, 2nd ed. Newton, MA, USA: Butterworth-Heinemann, 1979.

[18] E. Stoica, M. A. Hearst, and M. Richardson, "Automating creation of hierarchical faceted metadata structures," in *Proceedings of the Human Language Technology Conference (NAACL-HLT)*, 2007, pp. 244–251.

[19] R. Hahn, C. Bizer, C. Sahnwaldt, C. Herta, S. Robinson, M. Bürgle, H. Düwiger, and U. Scheel, "Faceted wikipedia search," in *Business Information Systems*. Springer, 2010, pp. 1–11.

[20] 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*, 1998, pp. 107–117. [Online]. Available: http://dl.acm.org/citation.cfm?id=297805.297827

[21] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, Mar. 1986. [Online]. Available: http://dx.doi.org/10.1023/A:1022643204877

[22] F. Kose, W. Weckwerth, T. Linke, and O. Fiehn, "Visualizing plant metabolomic correlation networks using clique-metabolite matrices." *Bioinformatics*, vol. 17, no. 12, pp. 1198–1208. [Online]. Available: http://dblp.uni-trier.de/db/journals/bioinformatics/bioinformatics17.html#KoseWLF01

[23] C. Bron and J. Kerbosch, "Algorithm 457: finding all cliques of an undirected graph," *Communications of the ACM*, vol. 16, no. 9, pp. 575–577, Sept. 1973. [Online]. Available: http://doi.acm.org/10.1145/362342.362367

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

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

[26] A. S. Pollitt, "The key role of classification and indexing in view-based searching," in *Proceedings of the 63rd International Federation of Library Associations and Institutions General Conference (IFLA)*, 1997.

[27] K.-P. Yee, K. Swearingen, K. Li, and M. Hearst, "Faceted metadata for image search and browsing," in *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI)*, 2003, pp. 401–408.

[28] W. Dakka, P. G. Ipeirotis, and K. R. Wood, "Automatic construction of multifaceted browsing interfaces," in *Proceedings of the 14th ACM international conference on Information and knowledge management (CIKM)*, 2005, pp. 768–775.

[29] W. Dakka and P. Ipeirotis, "Automatic extraction of useful facet hierarchies from text databases," 2008, pp. 466–475.

[30] K. A. Ross and A. Janevski, "Querying faceted databases," in *Semantic Web and Databases*, ser. Lecture Notes in Computer Science, C. Bussler, V. Tannen, and I. Fundulaki, Eds., vol. 3372. Springer Berlin / Heidelberg, 2005, pp. 199–218.

[31] S. B. Roy, H. Wang, G. Das, U. Nambiar, and M. Mohania, "Minimum effort driven dynamic faceted search in structured databases," in *Proceedings of the 17th ACM conference on Information and knowledge management (CIKM)*, 2008, pp. 13–22.

[32] J. Diederich and W.-T. Balke, "FacetedDBLP - navigational access for digital libraries," *Bulletin of IEEE Technical Committee on Digital Libraries*, vol. 4, Spring 2008.

[33] D. Debabrata, R. Jun, N. Megiddo, A. Ailamaki, and G. Lohman, "Dynamic faceted search for discovery-driven analysis," in *Proceeding of the 17th ACM conference on Information and knowledge management (CIKM)*, 2008, pp. 3–12.

[34] O. Ben-Yitzhak, N. Golbandi, N. Har'El, R. Lempel, A. Neumann, S. Ofek-Koifman, D. Sheinwald, E. Shekita, B. Sznajder, and S. Yogev, "Beyond basic faceted search," in *Proceedings of the international conference on Web search and web data mining (WSDM)*, 2008, pp. 33–44.

[35] A. Kashyap, V. Hristidis, and M. Petropoulos, "Facetor: cost-driven exploration of faceted query results," in *Proceedings of the 19th ACM international conference on Information and knowledge management (CIKM)*, 2010, pp. 719–728.

[36] J. Pound, S. Paparizos, and P. Tsaparas, "Facet discovery for structured web search: a query-log mining approach," in *Proceedings of the 2011 international conference on Management of data (SIGMOD)*, 2011, pp. 169–180.

[37] W. Pratt, M. A. Hearst, and L. M. Fagan, "A knowledge-based approach to organizing retrieved documents," in *Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence (AAAI/IAAI)*, 1999, pp. 80–85.

[38] M. Käki, "Findex: search result categories help users when document ranking fails," in *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI)*, 2005, pp. 131–140.

[39] K. Rodden, W. Basalaj, D. Sinclair, and K. Wood, "Does organisation by similarity assist image browsing?" in *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI)*, 2001, pp. 190–197.

[40] D. R. Cutting, D. R. Karger, J. O. Pedersen, and J. W. Tukey, "Scatter/gather: a cluster-based approach to browsing large document collections," in *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR)*, 1992, pp. 318–329.

[41] O. Zamir and O. Etzioni, "Grouper: a dynamic clustering interface to web search results," in *Proceedings of the eighth international conference on World Wide Web (WWW)*, 1999, pp. 1361–1374.

[42] H. Bast and I. Weber, "The CompleteSearch engine: Interactive, efficient, and towards IR & DB integration," in *Conference on Innovative Data Systems Research (CIDR)*, Asilomar, CA, USA, 2007, pp. 88–95.

[43] J. Koren, Y. Zhang, and X. Liu, "Personalized interactive faceted search," in *Proceeding of the 17th international conference on World Wide Web (WWW)*, 2008, pp. 477–486.

[44] R. van Zwol, B. Sigurbjornsson, R. Adapala, L. Garcia Pueyo, A. Katiyar, K. Kurapati, M. Muralidharan, S. Muthu, V. Murdock, P. Ng, A. Ramani, A. Sahai, S. T. Sathish, H. Vasudev, and U. Vuyyuru, "Faceted exploration of image search results," in *Proceedings of the 19th international conference on World wide web (WWW)*, 2010, pp. 961–970.

[45] B. Kules, R. Capra, M. Banta, and T. Sierra, "What do exploratory searchers look at in a faceted search interface?" in *Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries (JCDL)*, 2009, pp. 313–322.

[46] E. Chu, A. Baid, T. Chen, A. Doan, and J. Naughton, "A relational approach to incrementally extracting and querying structure in unstructured data," in *VLDB*, 2007.

[47] H. Zaragoza, H. Rode, P. Mika, J. Atserias, M. Ciaramita, and G. Attardi, "Ranking very many typed entities on Wikipedia," in *CIKM*, 2007, pp. 1015–1018.

[48] A.-M. Vercoustre, J. A. Thom, and J. Pehcevski, "Entity ranking in Wikipedia," in *SAC*, 2008, pp. 1101–1106.

[49] X. Li, C. Li, and C. Yu, "Entity-relationship queries over Wikipedia," *ACM Transactions on Intelligent Systems and Technology (TIST) (In press)*, 2012.

[50] M. Völkel, M. Krötzsch, D. Vrandecic, H. Haller, and R. Studer, "Semantic Wikipedia," in *Proceedings of the 15th international conference on World Wide Web (WWW)*, 2006, pp. 585–594.

[51] F. Wu and D. S. Weld, "Autonomously semantifying Wikipedia," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management (CIKM)*, 2007, pp. 41–50.

[52] 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*, ser. VLDB '97, San Francisco, CA, USA, 1997, pp. 436–445. [Online]. Available: http://dl.acm.org/citation.cfm?id=645923.671008

[53] S. E. Schaeffer, "Survey: Graph clustering," *Computer Science Review*, vol. 1, no. 1, pp. 27–64, Aug. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.cosrev.2007.05.001

BIOGRAPHICAL STATEMENT

Ning Yan was born in Tai'an, a beautiful city of Shandong Province in China. The city is located at the foot of Mount Tai which has been a place of worship for at least 3,000 years and served as one of the most important ceremonial centers of ancient China.

He then spent seven years of his life in Nanjing, China, and graduated with a Bachelor of Engineering degree in Software Engineering and a Master of Engineering degree in Computer Science both from Southeast University in 2005 and 2008, respectively.

As a doctoral student in University of Texas at Arlington, he is doing research mainly focusing on Web data mining, information retrieval, and database management system. Compared with elegant theoretic results, he is more interested in building real world applications that have great impacts.