ABSTRACT

Title of dissertation: QUALITY-AWARE DATA SOURCE MANAGEMENT

Theodoros Rekatsinas,
Doctor of Philosophy, 2015

Dissertation directed by: Professor Amol Deshpande and
Professor Lise Getoor
Department of Computer Science

Data is becoming a commodity of tremendous value in many domains. The ease of collecting and publishing data has led to an upsurge in the number of available data sources — sources that are highly heterogeneous in the domains they cover, the quality of data they provide, and the fees they charge for accessing their data. However, most existing data integration approaches, for combining information from a collection of sources, focus on facilitating integration itself but are agnostic to the actual utility or the quality of the integration result. These approaches do not optimize for the trade-off between the utility and the cost of integration to determine which sources are worth integrating.

In this dissertation, I introduce a framework for quality-aware data source management. I define a collection of formal quality metrics for different types of data sources, including sources that provide both structured and unstructured data. I develop techniques to efficiently detect the content focus of a large number of diverse sources, to reason about their content changes over time and to formally compute the utility obtained when integrating subsets of them. I also design efficient algorithms with constant factor approximation guarantees for finding a set of sources that maximizes the utility of the integration result given a cost budget. Finally, I develop a prototype quality-aware data source management system and demonstrate the effectiveness of the developed techniques on real-world applications.

QUALITY-AWARE DATA SOURCE MANAGEMENT

by

Theodoros Rekatsinas

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2015

Advisory Committee:

Professor Amol Deshpande	Chair/Co-Advisor
Professor Lise Getoor	Co-Advisor
Professor Douglas Oard	Dean's Representative
Professor Mihai Pop	Committee Member
Dr. Divesh Srivastava	Committee Member

© Copyright by
Theodoros Rekatsinas
2015

This thesis is dedicated to my parents Eleni and Yiannis.

Their love will always be the strongest driving force in life.

Acknowledgments

My journey as a student may have come to an end but I am sure I have so much more to learn from my four great mentors, Amol Deshpande, Lise Getoor, Xin Luna Dong and Divesh Srivastava. I will forever be indebted to them for their support in my research career and every other aspect of academic life.

First and foremost, I owe a special thank you to my advisors Amol and Lise. They were the first people to truly believe in my half-baked research ideas and the ones that taught me how to do research and think critically. Amol and Lise taught me how important it is to have faith in your work and be able to crisply communicate it to others. They also gave me the freedom and necessary encouragement to pursue my own research ideas and generously offered me their advice at every step along the way. I will always be grateful to them for shaping me into a researcher.

Next, I would like to express my deepest gratitude to Luna and Divesh. I feel truly privileged for having the opportunity to work with them. Luna and Divesh are not only absolutely brilliant researchers but they are amongst the nicest people I have ever met. They were the first to treat me as a peer rather than a student and the ones that introduced me to the amazing world of data quality. This thesis would have never been accomplished without them. They taught me how to make no compromises, not only, on the quality of research but life in general. Divesh's limitless knowledge and Luna's inexhaustible passion will always be an inspiration for me.

At this point I'd like to thank my fellow students in the database and LINQS group as well as the Computer Science department. They made my graduate studies both intellectually stimulating and incredibly enjoyable. Special thanks go to Ioana Bercea, Rajesh Chitnis, Sameh Khamis, Udayan Khurana, and Ben London for making my life more colorful and helping me go through the daily grinds of graduate life.

I would also like to thank the College Park gang, Alex Tzannes, Evripidis Paraskevas, Jason Filippou, Jason Papanikolaou, Kostas Zampogiannis, Nasos Papanikolaou, Vassilis Lekakis and Vassilis Zikas, for turning the last five years into a magical experience. In particular, Vassilis Z., Jason P. and Alex offered me some of the most enjoyable times during this journey. I am looking forward to those that are yet to come. Also, special thanks go to Kostas Nikolopoulos. Kostas is a great friend and the first person that taught me what it means to be a researcher.

I also want to thank someone very close to my heart. Lida, I am grateful you were there for me for the toughest part of this journey, for your patience, and for all the moments we shared. I will always treasure your love; even if it's only a memory now.

Finally, there are not enough words to express my gratitude, love and respect to my parents. They always offered me their unconditional love and support throughout my life, and were always there for all the ups and downs during the last five years.

Table of Contents

Chapter 1: Introduction

Today we are experiencing an unprecedented deluge of data sources available for integration and analysis. This is mainly because of the ease of publishing data on the Web, the proliferation of services that facilitate the collection and sharing of data (e.g., Google Fusion Tables [65]), and the adoption of open data access policies both in science and government. Integrating data from multiple data sources can significantly enhance the value of data. For example, with more sources, we can increase the coverage of integrated data; In the presence of inconsistencies, we can improve correctness by leveraging the collective wisdom. Reducing the effort to perform data integration, i.e., clean the data provided by sources, resolve entity references, and construct schema mappings across sources, has been one of the most challenging problems in the database community [41].

However, the ease of collecting and publishing data has also led to an increase in the number of data sources providing data of bad or poor quality. This makes reasoning about the data quality of sources of paramount importance. In many real-world scenarios, integrating low-quality sources with noisy data may hurt the overall accuracy of integrated data [47]. Furthermore, integrating every data available may incur a significant monetary cost. To understand the effect of poor data quality, one only needs to consider a report by the SAS Institute Inc.[1], stating that poor data quality is estimated to cost U.S. businesses

[1]http://support.sas.com/resources/papers/proceedings13/079-2013.pdf

$600 billion annually. Despite that, most existing data integration approaches are still agnostic to the quality of the available data sources.

Even in the presence of high-quality data sources, integrating data sources comes with a significant *computational cost* [142]. A substantial effort must be spent in setting up the overall integration pipeline for continuous ingest. Furthermore, acquiring data may involve a *monetary cost*. Although much of the data is freely available, the number of data sources that charge monetary fees for access is rapidly increasing. This trend is expected to continue as data is further commoditized [9, 158]. The cost involved in integrating data sources gives rise to the natural questions of how can one reason about the trade-off between the benefit and cost of integration, and how one can identify sources that are worth integrating. For example, in the presence of redundancy among data sources, integrating new sources may not necessarily increase the coverage of integrated data significantly, if at all, but it increases the total cost.

Given the large number of available data sources and their heterogeneity in terms of quality and integration cost, it is challenging for a user to identify sources that are truly beneficial to her applications. The goal of this dissertation is to develop a formalism for *managing a large number of diverse and dynamic data sources, i.e., reasoning about their content and quality*; and algorithms for *discovering valuable sources for integration, i.e., sources that maximize the user's utility at the minimum cost.*

1.1 Challenges in Determining the Quality and Utility of Data Sources

Assessing data quality has been a longstanding problem in many diverse disciplines ranging from data mining and data curation to operations research and econometrics [175]. Traditionally, the data quality of a data source or a dataset has been measured via reasoning about its completeness and the amount of erroneous information in it. In the last decade, however, there has been a growing interest in defining diverse metrics to assess data quality [125]. In fact more than 200 *quality metrics* have been proposed to characterize data quality [11]. Nevertheless, most of these metrics are hard to quantify and calculate for arbitrary datasets and the most frequently mentioned data quality metrics in the literature are: (i) accuracy (i.e., the degree to which data represents the values of corresponding real-world constructs correctly), (ii) coverage (i.e., the percentage of real-world constructs mentioned in the data) , (iii) timeliness (also called freshness, i.e., the degree to which data represents reality as of a required point in time), and (iv) consistency (i.e., inter-source value conflicts across distinct instances for the same real-world construct).

Although it is possible to talk about the quality of a data source by itself, it is more natural, useful, and accurate to talk about the quality of a source with respect to some specific *context*. For example, “ESPN” has high coverage for “sport in the USA” but has negligible coverage for “politics”. Trying to formalize the notion of context and characterizing the quality of a source with respect to that context, raises several challenges. We will use two real-world scenarios to illustrate these issues: (i) listings aggregation by combining, e.g., business, job, or rental listings from a variety of sources, and (ii) event detection and analysis by combining articles in social and news media [78, 57].

In the first scenario, aggregators offer a search service to end users by integrating listings from multiple sources. Each source provides a set of listings and periodic updates as new listings become available or existing listings get updated or removed. Specifically, we consider the scenario of aggregating business listings (BL) from 43 different data sources providing records for US businesses over a period of two years. This dataset was extracted from Yellow Pages. Source entries correspond to listings for businesses from various categories located in different states across the United States.

In the second scenario, an analyst integrates events mentioned in a diverse set of news media sources and analyzes them collectively to detect patterns characterizing her domain of interest. An example of such an event collection framework is the Global Database of Events, Languages and Tone (GDELT) [96] where news articles from different sources are aggregated into a single repository. Here, we consider a snapshot of GDELT containing news articles from 15,275 sources over a period of one month. The extracted events come from different news portals (e.g., NY Times), correspond to different event types from a predefined dictionary, and are associated with different locations.

Next, we illustrate some of the major challenges when analyzing the content and quality of data sources. We study the quality of sources with respect to their average coverage and average freshness. At a high-level, the coverage of a source is the probability that a randomly chosen data entry from the data domain will be provided by the source. Timeliness is the probability that a randomly chosen entry from the source will be up-to-date, i.e., in agreement with the actual data domain. The averages for each source are computed across all domain points covered by it considering a source snapshot over the available time window.

The first challenge is that data sources may vary significantly in the portions of the data domain they focus on and also exhibit large heterogeneity in their quality for different parts of the domain. User tasks may focus on different parts of the data domain, thus, it is important to consider both types of heterogeneity when assessing the quality of a source.

Example 1. *Consider the business listings scenario described above. Figure 1.1(a) shows the different sources and the number of different locations and business types mentioned in the sources. The radius of each circle indicates the size of the source measured as the total number of entries in it. As shown, there are many sources of varying sizes that provide data for most of the available business types and locations. However, there are also specialized sources that focus either on specific locations or specific business categories and tend to be significantly smaller. Figure 1.1(b) shows the coverage of the largest source in BL for all possible location and business-type combinations. The coverage values in the figure are computed for a single time point within the two-year window. As we can see, the quality of this source (i.e., the coverage) varies significantly depending on the location or business type.*

The utility of a source captures the actual value that a user extracts from using the data provided by it. Certainly the utility of a data source is closely related to its quality, but focusing attention only on the high-quality sources can be a mistake.

Example 2. *Consider the business listings scenario presented above. Here, the utility of a data source can be described using the number of unique entries the source provides, while the quality of a source can be described by its average coverage. The bottom graph in Figure 1.2 shows the percentage of unique entries provided by each source in BL with*

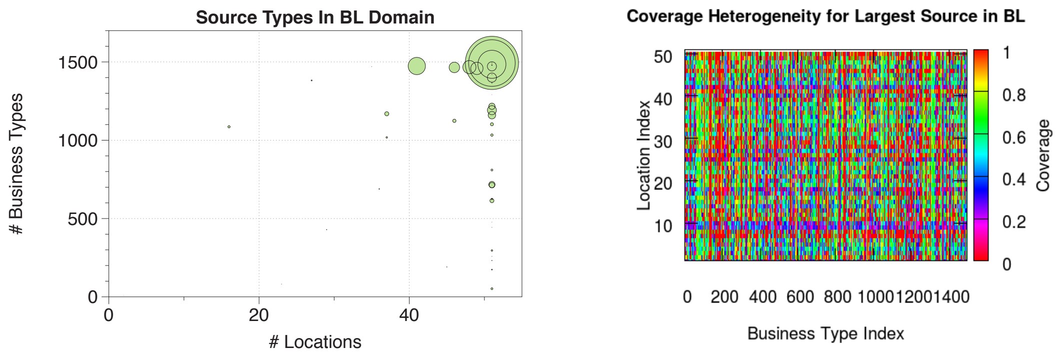


Figure 1.1: (a) The various data sources in the business listings dataset with respect to the location and business types they focus on. The radius of each circle indicates the size of each source. (b) The coverage of the largest source in the business listing dataset for different locations and business types.

respect to its average coverage. The upper one shows the total size of each source. As shown, there is a significant number of sources with low coverage for the overall data domain that provide mostly unique items. While small in size, these specialized sources can be of great utility to several users. Finally, there is a significant number of sources with mid-range coverage (i.e., 0.25 to 0.4) that provide a small number of unique data items. Considering these sources during integration does not promote the utility of a user.

As illustrated in the last example, the dimensions of utility and quality are conceptually different. Thus, focusing integration efforts only on high-quality sources may incur a significant loss of utility.

Finally, in many scenarios, sources are not *static* but rather their content changes *dynamically* over time. This gives rise to additional challenges when reasoning about the quality of sources. One challenge stems from the fact that sources that update their data more frequently are not always more effective at capturing changes in a timely manner.

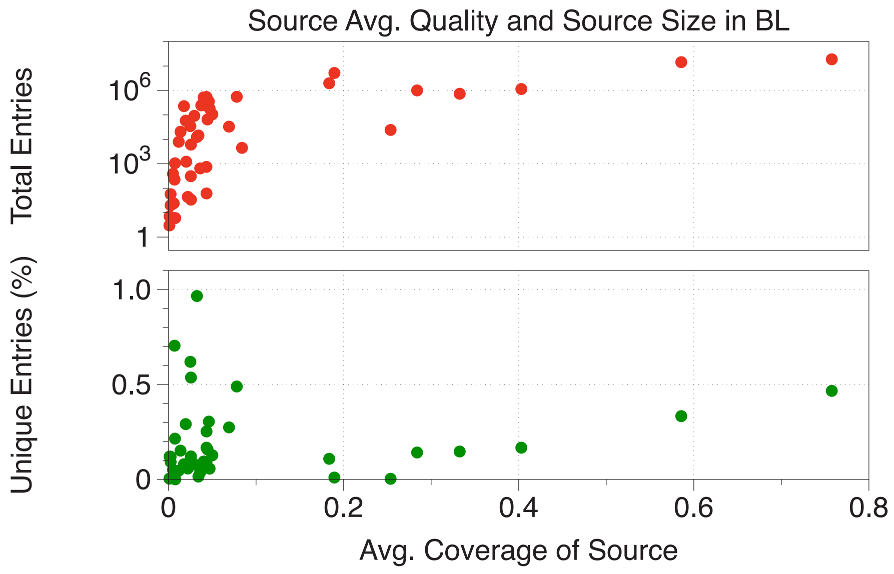
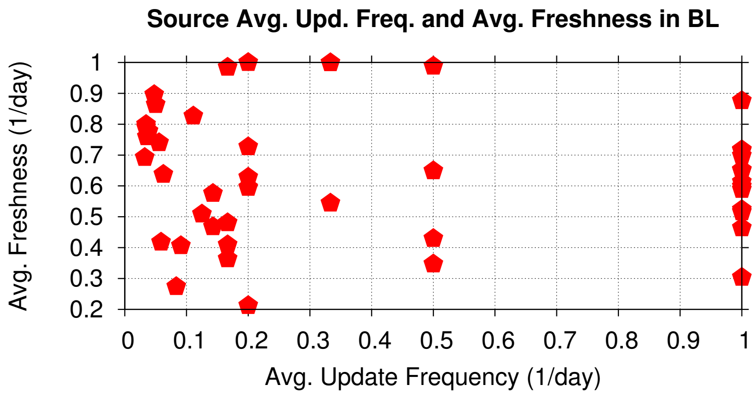


Figure 1.2: The average coverage of data sources in the business listing domain versus the percentage of unique entries they provide and their size.

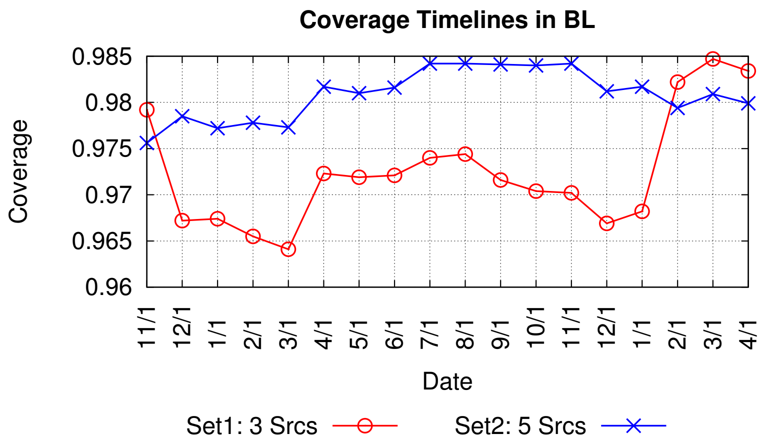
Example 3. *Figure 1.3(a) shows the average update frequency and average freshness for each source in BL over the two year time window. As shown, there is no clear correspondence between the update frequency and freshness of a source; even sources with high update frequencies may have low freshness, indicating that sources may add to their content frequently but are ineffective at deleting stale data or capturing value changes of older data items.*

Even sources with similar update frequencies exhibit different levels of staleness, as exemplified next in the second domain.

Example 4. *This example examines how effective the 20 largest sources in GDELT are at reporting events in a timely manner. Figure 1.4(a) shows the average delay with which events are reported and the corresponding fraction of delayed events over the total content of each source over one month. While all sources get updated daily, one can see that a significant fraction of events are reported with delays.*

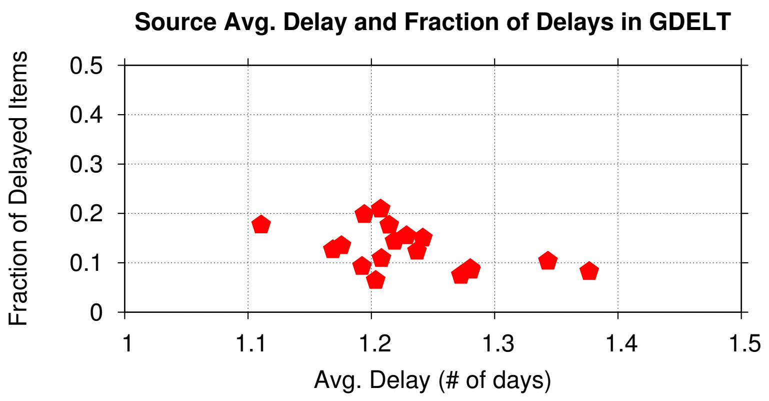


(a) Average update frequencies and average freshness of data sources in BL.

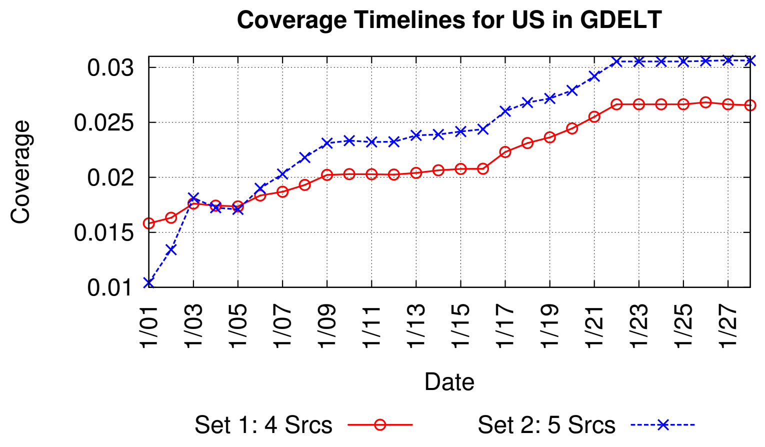


(b) Coverage evolution for two sets of sources in BL.

Figure 1.3: Characteristics of dynamic data sources in BL.



(a) Average delay and fraction of delayed event mentions for the 20 largest sources in GDELT. All sources get updated every day.



(b) Coverage evolution for two sets of sources for GDELT corresponding to events in the US.

Figure 1.4: Characteristics of dynamic data sources in GDELT.

1.2 Dissertation Overview and Contributions

Given the challenges described above, the primary goal of this dissertation is to develop the techniques and algorithms needed to realize a *quality-aware data source management* (QDSM) system that automatically assesses the quality of data sources and not only allows users to discover sources relevant to their applications, but also enables them to discover the *most valuable sources* for integration. That is, sources that maximize the user's utility extracted by the integrated data at the minimum cost. We propose a QDSM system that follows the architecture shown in Figure 1.5[2]. The techniques summarized below serve as building blocks to this architecture. The main components of the proposed architecture are (i) a source analysis engine, and (ii) a source exploration engine. The source analysis engine analyzes the content of data sources and computes their quality through a collective analysis process. It then constructs an index over the quality profiles of the sources while taking into account the domain covered by each source.

The source exploration engine serves user *integration tasks* and enables users to efficiently identify the most valuable sources for integration. User integration tasks correspond to a free-text description of the data domain the user is interested in, accompanied by a desired budget either on the number of sources to be integrated or the amount of money the user can afford for acquiring data. Considering the business listings scenario, an example user integration task would correspond to finding sources that report "Hotels in Maryland" given a constraint that "up to 20 sources" should be used. Given such a task a QDSM system needs to first detect which sources provide listings relevant to "Hotels in

[2]This architecture was first presented in Rekatsinas et al., Finding Quality in Quantity: The Challenge of Discovering Valuable Sources for Integration, CIDR 2015

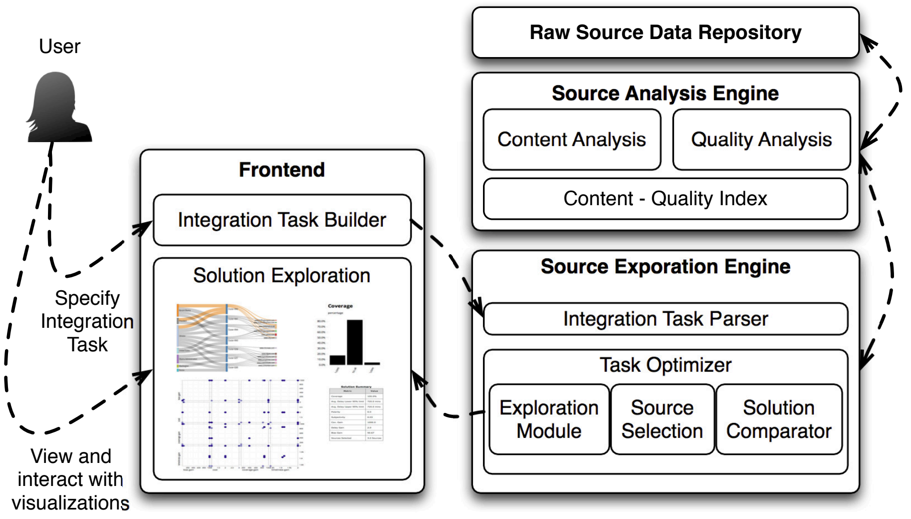


Figure 1.5: Quality-Aware Data Source Management System architecture.

Maryland" and then identify subsets of up to 20 sources maximize either the coverage or the accuracy for the corresponding listings if integrated together.

The main part of the dissertation focuses on the main techniques required to realize the aforementioned QDSM architecture. Subsequently, these techniques are used to design practical systems and develop to real-world applications, which are also described in the dissertation. The main parts of the dissertation, as well as the associated chapters and published papers, are summarized below.

1.2.1 Reasoning About the Content and Quality of Data Sources

In Chapter 3, we describe the techniques and algorithms used by the source analysis engine of the QDSM architecture to analyze the content and to compute the quality of data sources. We present a formalization for characterizing the content of sources. We consider sources that provide both *structured* and *unstructured* data. We focus on dynamic data sources and define a range of quality metrics for characterizing their content.

More precisely, for sources providing structured data, we define the coverage, freshness and accuracy of a dynamic source and also study the structural properties of these quality metrics (e.g., we show that coverage exhibits submodular structure). We also introduce a collection of statistical models that capture the change patterns of each source as well as the overall data domain and allow us to estimate the quality of sources for future time points. Our models for describing the update patters for the overall data domain build upon well-known parametric approaches, such as Poisson processes. On the other hand, our models for describing the change patterns of sources are based on non-parametric empirical models. This provides us with the necessary flexibility to learn update models for highly heterogeneous sources.

To analyze the content of sources that provide unstructured data, we introduce a novel temporal statistical model that allows one to discover the abstract *topics* that occur in the data entries of the sources. This model not only allows us to identify the domain covered by each source but also enables us to find change patterns in the overall data domain. We also show how one can use this model to estimate the coverage of data sources with unstructured data entries for future time points.

We evaluate the effectiveness of the aforementioned models using a collection of diverse real-world datasets. As shown in Section 3.7, our models are capable of detecting and estimating data change patterns and source-quality changes with very small relative errors. This work is presented in a series of papers [137, 138, 136].

The proposed models introduced in the beginning of Chapter 3 assume that all sources provide data from a focused and not fully diverse data domain. This is a fairly strong condition that prevents the proposed approaches from being used in a holistic approach supporting arbitrary domains with highly heterogeneous sources. An example of such a domain is that of news media. If we treat a news portal as a source, it is easy to see that different portals cover significantly different domains. For example, "ESPN" focuses on sports mostly in the United States, while "Eurosport" covers sports in Europe and "Techcrunch" focuses on technology news world-wide. Notice that even for sources with similar topics, e.g., "ESPN" and "Euorsport", the actual sports and real-world entities prevalent in each portal can be different. For example, "ESPN" provides data for NFL, NBA and college sports in the US while "Eurosport" provides entries on Cycling, Wintersports, Formula One, etc.

To address this limitation, we introduce a novel multi-level source quality index that enables us to effectively categorize the content and quality profiles of diverse data sources. We refer to this index as a *correspondence graph*. The correspondence graph extends traditional structured constructs such as ontologies or knowledge bases [68], used to store and organize complex relationships across diverse real-world entities, concepts and entity types. This index is introduced in [136] and is a core part of a prototype QDSM system we describe in Section 7.1 and demonstrate in [133].

1.2.2 Enriching Structured Domain Indexes

While the correspondence graph is effective in organizing diverse data sources, many times the structured indexes it builds upon (i.e., an ontology or a knowledge base) follow a *closed-world* assumption. In other words, their scope is limited to the entries and concepts already present in them. Moreover, such indexes usually focus on "head" data, i.e., popular entities and concepts. However, focusing exclusively on head data leaves behind a considerable volume of "tail" data about less popular entities, non-current (historical) facts and so on. Typically, the closed-world assumption can be relaxed by leveraging the paradigm of *crowdsourcing*, i.e., using a crowd of people, usually online and with a monetary cost, to fulfill a collection of small *human-intelligence tasks* (HITS).

In Chapter 4, we show how one can use the paradigm of crowdsourcing to enrich such domains, thus relaxing the closed world assumption. More precisely, we show how one can use the crowd to enrich structured domains effectively by asking crowd workers to enumerate (i.e., extract) entities that belong to different parts of the domain. Nevertheless, worker answers tend to exhibit significant overlaps when reporting entities from a domain and tend to focus only on the popular entities. Therefore, asking humans to extract entities repeatedly may only incur a large monetary cost without increasing the total number of extracted entities significantly. To address this limitation, we develop a new *adaptive crowd-querying policy* that maximizes the total number of extracted entities by crowd-workers while operating under a monetary budget. At a high-level, our querying policy exploits the structure of the underlying data domain to diversify the entity extraction queries (e.g., asks workers to enumerate different types of entities), and thus maximizes the number of extracted entities by issuing queries for different subparts of the data domain.

To design this adaptive querying policy, we develop new statistical tools that allow us to reason about the gain of issuing *additional queries* and focus on the problem of *budgeted entity extraction* where we seek to maximize the number of extracted entities given a monetary and latency budget. In Section 4.5, we present an empirical evaluation of our proposed algorithms on both real-world and synthetic datasets, demonstrating a yield of up to 4X over competing approaches for the same budget. Finally, data sources themselves can be viewed as entities, hence, the proposed techniques can be used to identify new data sources in domains where only a limited number of sources is available. For example, we can ask the crowd to provide us with data sources on "Mongolian Philosophy" for which only limited information is provided by OpenLibray.org. This work is described in [135].

1.2.3 Selecting Valuable Data Sources for Integration

In Chapter 5, we design algorithms for finding sets of sources that maximize the user utility of the final integration result at the minimum cost. For instance, given the content and quality profiles of the available sources, if a user is interested in retrieving business listings for "Lawyers in Maryland" we want to detect which sources, if integrated together, will maximize the coverage of business listings for "Lawyers in Maryland" at the minimum monetary cost. In Chapter 5, we show how the utility of integrated data can be defined as a function of the overall quality of the integration result, using the quality metrics introduced in Chapter 3. To estimate the quality of integrated data, we assume an integration scheme across sources that follows the union semantics. For example, consider integrating two sources at a time point t and a restaurant listing that is mentioned in the first one but was never mentioned in the second. In this case, the restaurant entry will be present in the integration result of the two sources. On the other hand, if the listing was present in the second source for a time point prior to t but deleted by time t then this entry will not be present in the integration result. This integration scheme is used in many practical applications to form the integration result [76, 151]. The latter corresponds to computing the OR-probability that a desired data entry will be reported by at least one source in the selected sets of sources.

Following these semantics, we study this problem of *source selection* for dynamic sources and introduce rigorous formalizations of the problem for various setups. More specifically, we study the scenarios where users can choose to acquire all data (including all updates) provided by a dynamic source or partial data either by acquiring specific time snapshots of the source or data for a subset of the domain of the source. We show that all these problem variations are NP-complete.

For sources that get updated independently (Chapter 5), we propose an efficient local-search algorithm with rigorous theoretical guarantees on the quality of the retrieved solution. The effectiveness and scalability of our proposed algorithm is evaluated on two large-scale real-world datasets in Section 5.4. This work is presented in [137].

When source updates exhibit dependencies, computing the OR-probability that a desired data entry will be reported by a set of sources is challenging due to the underlying dependency structure. In Chapter 6, we present an framework for efficiently estimating the probability of *generic boolean formulas* under the presence of variable dependencies. During an offline phase, our algorithm compiles the dependencies into an efficient data structure that is used afterwards to evaluate the probabilities of boolean formulas. We evaluate the performance of our techniques in Section 6.6 and show speed-ups of at least one order of magnitude compared to baselines. This work is described in [134].

1.2.4 Data Source Management Applications

In this part of the dissertation (Chapter 7), we design a quality-aware data source management system and introduce applications that use the source management techniques described in previous chapters as building blocks, thereby demonstrating their utility. We study the following applications:

SOURCESIGHT (Section 7.1). Here, our goal is to design a quality-aware data source management system using the source management techniques reviewed above. This system, called SOURCESIGHT, enables efficient and effective source selection over large heterogeneous data domains [133] and allows users to not only explore and identify the most valuable sources for their integration tasks but also understand the quality and cost trade-off between different integration options. SOURCESIGHT introduces a collection of visualizations that allow users to interactively perform source selection and evaluate the integration solutions recommended by the system. We evaluate the system on diverse news data sources reporting real-world events and extend the quality metrics described in Chapter 3 to consider the position bias of news sources.

SOURCESEER (Section 7.2). Here, our goal is to improve the forecasting of rare disease outbreaks, such as Hantavirus outbreaks, when analyzing news reports obtained from diverse news portals. News portals in this correspond to data sources. We apply our techniques described in Chapter 3 to identify the different quality characteristics and exploit that to obtain more accurate forecasts. Our techniques, described in detail in Section 7.2 and introduced in [138], improve the quality of rare-disease forecasting significantly compared to source-agnostic baselines. We evaluate them on forecasting Hantavirus outbreaks in Latin America using real data from thousands of different data sources. We show that our techniques can not only forecast outbreaks in a more timely manner but can also forecast outbreaks at a finer spatial granularity, i.e., at the state level and not country level as baseline techniques do.

Chapter 2: Background

This chapter serves as a primer on data integration and data source management as related to this dissertation. First, we review techniques developed to minimize the effort in integrating data from multiple data sources into a single repository. Then, we focus on approaches for managing the content of large numbers of heterogeneous data sources and techniques for reasoning about their quality and integration costs.

2.1 Data Integration

Combining heterogeneous data sources has been a longstanding problem in the data management literature [97]. Traditionally, *data integration* was studied in the context of data warehouses where multiple heterogeneous data sources (or databases) are combined into a repository under a single schema so that data becomes compatible with each other [74]. Later, data integration research expanded to addressing the problems in providing a unified query interface to access real-time data over a *mediated* schema [97], which allows information to be retrieved directly from the original data sources. This approach relies on mappings between the mediated schema and the schema of the original sources, and transforming a user query into specialized queries to match the schema of the original databases. Such mappings can be divided mainly in three classes [40]: (i) Global-as-

View (GAV) mappings, where the mediated schema is defined as a set of views over the data sources, (ii) Local-as-View (LAV) mappings, where data sources are described as views over the mediated schema, and (iii) Global-and-Local-as-View (GLAV) mappings that correspond to a combination of the two aforementioned approached. In general, data integration systems require *semantic integration* before any services can be provided. Hence, the data integration system needs to know the precise relationships between the terms used in each data source schema and the mediated schema.

The schema mapping classes described above assume that whenever an expression in the mapping requires combining tuples from different sources, the corresponding columns will have comparable values. In practice, sources not only exhibit structure heterogeneity but may also differ considerably in how they represent values and objects in the world. These differences are referred to as *data-level heterogeneity*. Data level heterogeneity can be classified in broadly two classes: (i) differences of scale that occur when there is some mathematical transformation between the values in one source and the other (e.g., product prices in USD versus EUR) and (ii) multiple references to the same entity that occur when there are multiple ways of referring to the same object in the real-world (e.g., USA vs United States).

Scale differences are usually reconciled by adding transformation functions to the schema mapping rules; many tools including Oracle XSL Mapper, MS Excel or the more recent Data Wrangler [87] allow the user to manually specify such functions. On the other hand, the problem of automatic reference reconciliation is significantly more challenging. In fact, it has had a long history and has been studied under many different names including *record linkage*, *entity resolution*, *duplicate matching* and many others. The early works on record linkage originated in the statistics community. In 1959, Newcombe et al. [111] introduced the record linkage problem and suggested many matching ideas: use of Soundex to handle spelling errors, blocking to reduce the number of tuple comparisons, multiple blocking rules to increase the number of matches found, and estimating match probabilities using independence assumptions. In 1969 Fellegi and Sunter [53] proposed an influential theory model for record linkage and later Winkler et al. [169, 170] extended the Fellegi-Sunter model in substantial ways to capture additional domain knowledge. Later, the problem of record linkage received increasing attention in the database, data mining and AI communities where a multitude of different approaches and models were introduced. We refer the reader to Getoor and Machanavajjhala [60] and Doan et al. [39] for details on a variety of record linkage techniques.

2.2 Data Source Management

All data integration approaches described thus far require a human to put a considerable amount of effort in setting up the data integration system. However, in many applications putting this effort may not be practical or even possible. For example, schema mappings may not exist, either because it is hard to create them, or because it is even impossible to generate precise mappings. To overcome this shortcoming, later work on data integration introduced the concept of *dataspaces* [70]. Dataspaces aim to reduce the effort required to set up a data integration system by shifting the emphasis to a data *co-existence* approach providing base functionalities over all data sources, regardless of how integrated they are. For example, a dataspace support platform (DSSP) can provide *keyword search* over all its

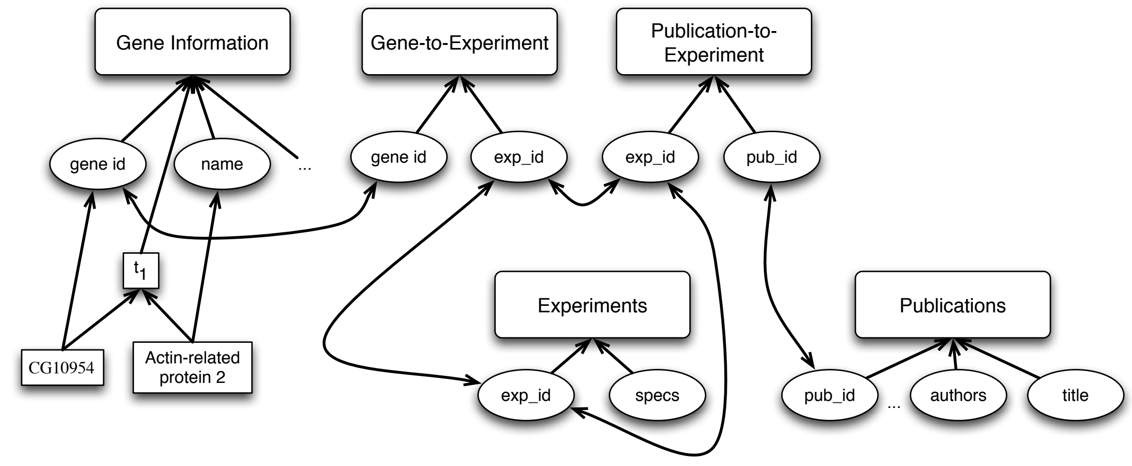


Figure 2.1: A data graph for a set of relations in a biology domain. Nodes can represent relations (rounded rectangles), attribute labels(ellipses), tuples(t_1), and attribute values(rectangles). Edges represent membership links and foreign key relationships.

data sources. In general, given a query, a dataspace support platform generates *best-effort* approximate answers from data sources where perfect mappings may not exist. When a large number of operations (e.g., answering relational queries, data mining, etc.) over certain sources are detected, it guides users to integrate those sources.

Keyword search has been recently adopted by many data integration systems to enable non-expert users to pose ad hoc queries over structured, integrated data. An overview of the work in keyword search over databases was recently published by Yu, Lu and Chang [172]. Keyword search in data integration systems is more complex than in a typical information retrieval system or search engine as it does not merely match against a single document or object. Intuitively, the set of keywords describes a set of concepts in which the user is interested. Given this set, the data integration system is tasked with the job of finding a way of relating the data source content to these concepts, e.g., through performing multiple joins.

The general approach to answering keyword queries over *structured* data sources is to represent a collection of structured databases as a *data graph* relating data and/or metadata items. Nodes in a data graph represent attribute values and potentially metadata items such as attribute labels or relations. Directed edges represent conceptual links between the nodes, where the links in a traditional DBMS include foreign key, containment and "instance-of" relationships. An example of a data graph is shown in Figure 2.1. The graph includes schema components from a biology setting with five tables (focusing on genes, experiments and publications), indicated as rounded rectangles. The attribute labels for each table are shown as ellipses and member tuples as rectangles. Edges include foreign keys and membership relationships. Such a graph typically corresponds to a logical construct used for defining the semantics of query answering and for efficiency reasons it is generally computed lazily. Finally, data graphs are commonly associated with node-level and edge-level weights. Node weights usually represent authoritativeness, reliability, accuracy, or trustworthiness. Node weights are generally assigned using one of the following approaches: (i) link-based analysis similar to PageRank, (ii) voting or expert rating, where an *expert* assigns scores to nodes, and (iii) query answer based feedback, where the system takes feedback on the quality of specific *query results* to learn the authoritativeness of nodes. Edge weights are assigned based on known relationships such as integrity constraints (e.g., foreign keys) or compatibility across data values.

Queries correspond to a set of keyword terms. Given these terms and a data graph, a keyword search system will match each keyword against the nodes in the graph and compute a similarity score or weight. After these matches are formed, the actual query processing computation finds a set of trees from the data graph - with keyword nodes as leaf nodes - and returns the top-scoring trees. Several techniques have been proposed to compute tree-scores and all of them are based on summing the weights of paths (considering both node and edge weights) within candidate trees [13, 86, 89]. Once the tree scores are computed, a variety of top-k query processing algorithms can be used to retrieve the k most relevant answers. These include Fagin's threshold algorithm [51] and other algorithms for performing joins in a ranked model [55, 66, 80].

The approaches described thus far focus mainly on enabling users to discover sources that are of interest to them and facilitate the actual integration. Recent work [47], showed how, given a fixed data domain, the benefit of integration can be quantified using rigorous data quality metrics, and introduced the paradigm of *source selection* that focuses on characterizing the marginal benefit of integrating a new source. Finally, Kruse et al. [142] proposed a collection of techniques for characterizing the cost of integration via reasoning about the effort required to perform schema matching, data cleaning and data transformation when integrating multiple sources; however, these techniques do not reason about the actual benefit of integrating multiple sources.

2.3 Knowledge Bases and Uncertain Data

Reasoning about the content of sources via a data graph is feasible when the structure of sources is relatively simple. However, in most cases, data sources and their contents lend themselves to rather complex modeling. Determining the relationships between data sources, or between a data source and a mediated schema, often requires subtle reasoning. For these reasons researchers have considered applying *knowledge representation* techniques to data integration. Knowledge representation systems, such as ontologies and knowledge bases, act as information repositories that provide a means for complex structured or unstructured information to be collected, organized, shared, searched and utilized. A knowledge base can be viewed as a collection of *facts* that describe information about entities and their properties, and *concepts* that describe information about the entity types and their properties. Moreover, a knowledge base can be represented as a graph where entities, facts and concepts correspond to nodes, while edges determine the relationships among entities, facts and concepts. Knowledge bases are often associated with *description logics*, i.e., formal knowledge representation languages [98] and have been an important component of many artificial intelligence applications, such as planners, robots, natural language processors, and game-playing systems.

Catarci and Lenzerini [22] were the first to articulate how description logics can be used to model data sources, and reason about the relationships between them. They proposed a structured representation language that allows one to express semantic interdependencies between different database schemas and presented a method for reasoning over such interdependencies. More recent work explored how, given an existing knowledge base, one can reason about the content of different data sources by matching the content of source entries and any attributes accompanying them to the content of a knowledge base [101, 160, 37, 153, 20]. Finally, a different line of work has explored the semantic integration of knowledge bases using automated data matching and schema matching techniques [42, 114, 72].

Recently, we have witnessed a proliferation of large-scale knowledge base systems, including Wikipedia, Freebase, YAGO [154], DBpedia [7] and Google's Knowledge Graph. However, most of these systems are curated by humans, and thus, have limited scopes with respect to the entities and facts they cover. To increase the scale of knowledge bases even further, recent work has focused on automated methods for constructing knowledge bases [21, 113, 161, 43, 128] that usually store millions of facts about the world, such as information about people, places and other real-world entities.

Most of the automated knowledge base construction approaches focus on compiling knowledge bases out of text-based extractions. During this compilation process multiple extractions are joined to form the facts stored in the knowledge base. Text-based extractions are usually uncertain and associated with an *accuracy value* characterizing the extractor's confidence on the result [44]. The field of uncertain data management and the paradigm of *probabilistic databases* [155] was introduced to enable the efficient management and querying of *uncertain data*.

In a probabilistic database, each tuple is associated with a probability $\in (0, 1]$, with 0 representing that the tuple is not present in the database, and 1 representing that the tuple is certainly present in the database. Also, each attribute is associated with a probability distribution over the potential values it may obtain. A probabilistic database could exist in multiple states referred to as *possible worlds*. For example, if we are uncertain about the correctness of a tuple then the database could be in two different states with respect to that tuple - in the first state the tuple is correct and thus contained in the database, while the second one does not. Recent work has shown how probabilistic reasoning techniques that were used to enable efficient query processing in probabilistic databases can also be used to construct knowledge bases out of uncertain data [112, 161, 48, 159].

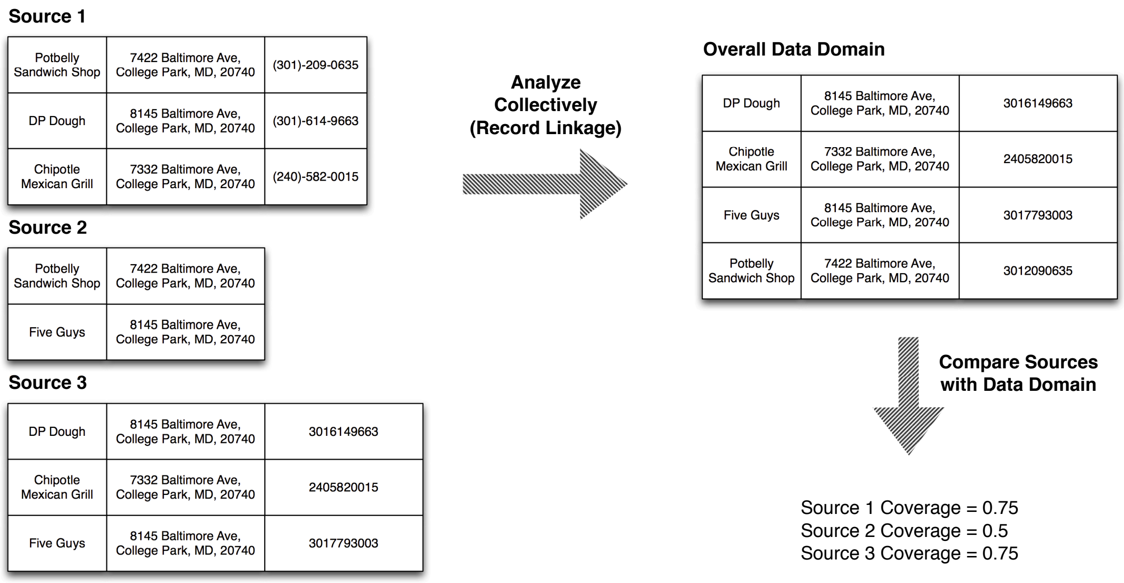
Chapter 3: Analyzing the Content and Quality of Data Sources

In this chapter, we develop algorithms and techniques for analyzing the content of data sources and computing their quality. We focus on dynamic data sources and consider sources that provide either structured or unstructured data entries. This chapter, being our first technical chapter, will also formalize the data source management scenarios we focus on throughout the dissertation. In Section 3.1, we describe the setting we consider, and in Section 3.2, we formalize the notion of dynamic sources, and define the content of a source as well as the notion of a data domain. Subsequently, in Sections 3.3 and 3.4, we introduce a collection of statistical models describing the changes in data sources and the underlying data domain, and in Section 3.5 we use these models to define a range of quality metrics and provide efficient estimators for those. In Section 3.6, we introduce a novel index that allows a quality-aware data source management system to effectively organize the quality profiles of data sources for largely heterogeneous data domains. Finally, in Section 3.7, we present an experimental evaluation of the techniques described in this chapter, and in Section 3.8 we discuss related work.

3.1 Introduction

We consider a setting where we have access to the content of a set of sources providing data entries from a data domain (also referred to as the *world*), but, we assume no ground-truth information is available about the content of that data domain. Under this assumption, the data domain corresponds to a latent universe and data sources provide their own, often partial, view of the universe. For example, if we consider the business listings scenario described in Chapter 1, sources may correspond to listing providers, such as Yelp, Yellow Pages, Foursquare, etc. While we can obtain information about the businesses listed in these sources, we assume that we do not have the means to verify the actual information (e.g., phone number or open hours) provided for each listing. The latter assumption applies in most real-world scenarios as obtaining ground-truth information is a strenuous and labor-intensive task. Furthermore, for the most part, we follow a *closed-world* assumption, considering that the world only contains information provided by at least one of the available sources.

In the aforementioned setup, the only available information is the content of sources. Our goal is to collectively analyze the source content to obtain information about the true state of the underlying data domain. Once we have this information, we can compute the quality of each source by comparing its individual content with the true content of the data domain. For example, consider the three sources shown in Figure 3.1 providing listings of fast food restaurants. If we integrate them, we get that the underlying data domain contains four entries. We can now compute the *coverage* of each source by comparing the integration result with each of the sources.


Source 1
Source 2
Source 3
Analyze Collectively (Record Linkage)
Overall Data Domain
Compare Sources with Data Domain

Source 1

Potbelly Sandwich Shop	7422 Baltimore Ave, College Park, MD, 20740	(301)-209-0635
DP Dough	8145 Baltimore Ave, College Park, MD, 20740	(301)-614-9663
Chipotle Mexican Grill	7332 Baltimore Ave, College Park, MD, 20740	(240)-582-0015

Source 2

Potbelly Sandwich Shop	7422 Baltimore Ave, College Park, MD, 20740
Five Guys	8145 Baltimore Ave, College Park, MD, 20740

Source 3

DP Dough	8145 Baltimore Ave, College Park, MD, 20740	3016149663
Chipotle Mexican Grill	7332 Baltimore Ave, College Park, MD, 20740	2405820015
Five Guys	8145 Baltimore Ave, College Park, MD, 20740	3017793003

Overall Data Domain

DP Dough	8145 Baltimore Ave, College Park, MD, 20740	3016149663
Chipotle Mexican Grill	7332 Baltimore Ave, College Park, MD, 20740	2405820015
Five Guys	8145 Baltimore Ave, College Park, MD, 20740	3017793003
Potbelly Sandwich Shop	7422 Baltimore Ave, College Park, MD, 20740	3012090635

Source 1 Coverage = 0.75
Source 2 Coverage = 0.5
Source 3 Coverage = 0.75

Figure 3.1: An example of collectively analyzing sources to identify their coverage.

To determine the actual state of the underlying data domain, we need to identify matching entries across data sources. When sources provide structured data entries, we need to identify entries that refer to the *same* real-world entities. This is equivalent to performing record linkage across sources, and, as discussed in Section 2.1 there has been a great amount of work on this problem. Solving this problem is not the focus of this dissertation but when record linkage is required by our techniques, we use existing methodologies described in the literature. Thus, we only provide the appropriate references for the techniques used and do not discuss them in detail.

In many cases, sources may provide only unstructured data. Thus, record linkage techniques are not applicable. To determine the actual state of the underlying data domain in the presence of unstructured data, we introduce a novel topic modeling framework that allows us to reason about the content of sources and identify the latent abstract "topics" present in the underlying data domain.

Finally, the main focus of this dissertation is on *dynamic data sources*, i.e., sources whose content changes over time. As described in Section 1.1, the quality of dynamic data sources can change over time. To reason about the quality changes of sources, we need to detect the content change patterns of each data source, and also detect the change patterns of the underlying data domain. For instance, consider a source that provides business listings for a state and updates its catalog only once per year. If new businesses open every month due to economic development, it is easy to see that the coverage of this source will keep deteriorating as time passes due to the changes in the real world.

At this point the reader may ask herself whether obtaining information about the underlying data domain by collectively analyzing the available sources is equivalent to performing the full integration across all sources. While we need to actually integrate data across the available sources, for such collective analysis, it suffices to do so on small samples of the sources as we discuss later in this chapter. Thus, we avoid paying the cost of full integration across all sources.

3.2 Preliminaries

We consider a data domain $\mathcal{D}$ characterized by a set of discrete-valued attributes $\mathcal{A}_D$. We also assume a fixed set of data sources $\bar{\mathrm{S}}$ providing objects from $\mathcal{D}$. In the remainder of the dissertation we use the terms *data domain* and *world* interchangeably for convenience. A *closed domain* [97] assumption is followed, stating that the data domain $\mathcal{D}$ contains only objects stored in the sources in $\bar{\mathrm{S}}$.

The data domain $\mathcal{D}$ changes dynamically over time, i.e., new entries may *appear*, or *disappear* or, in the case of structured data, the values of existing entries may *change* over time. *Dynamic sources* update their content by capturing changes that occur in the underlying data domain.

Definition 1. A data source S is dynamic when it updates its content with a frequency f_S, reporting entry appearances, disappearances and value changes from a data domain $\mathcal{D}$.

Given that we consider dynamic sources, we assume knowledge of a collection of historical snapshots for sources in $\bar{\mathbf{S}}$ over a past time window T ending at time t_0. For structured domains, entries in the snapshots correspond to tuples with attributes obtained from a superset of $\mathcal{A}_D$. For unstructured domains, entries correspond to *elemental units of information*, e.g., words or articles from a newspaper, associated with structured metadata following the schema defined by $\mathcal{A}_D$.

The following assumptions are made in the remainder of this chapter:

- For both structured and unstructured data, non-numeric attributes in $\mathcal{A}_D$ and their values are obtained from a known dictionary $\mathcal{V}_{\mathcal{A}_D}$.
- The majority of inaccuracies in data sources occur due to sources being ineffective at capturing changes from the world and not erroneous insertions. As demonstrated in Section 3.7, this assumption holds in several real-world domains including business listings, news articles and health-related news listings, since stale data and source delays dominate the mistakes.
- At any time point the entries in the world can be fully determined using an integration scheme across sources that follows the *union semantics*. For example, consider inte-

grating two sources at a time point t and a restaurant listing that is mentioned in the first one but was never mentioned in the second. In this case, the restaurant entry will be present in the integration result of the two sources. On the other hand if the listing was present in the second source for a time point prior to t but deleted by time t then this entry will not be present in the integration result. This integration scheme is used in many practical applications to form the integration result [76, 151].

- When sources provide structured entries, the changes in the underlying domain are assumed to follow a Poisson random process; the lifespan of an entry and the time interval between consecutive updates are assumed to follow an exponential distribution. In Section 3.3.1, we show that both assumptions hold for two diverse real-world domains. No such assumptions are made for the domain changes when sources provide unstructured data or the content changes of sources in either the structured or unstructured case. For these, generic statistical models are used, based on empirical distributions in the case of structured domains and auto-regressive techniques in the case of unstructured domains, that are capable of capturing complex change patterns that depend on the update frequency of each source.
- Finally, we require that sufficient historical data are available to learn the statistical models described next. This assumption is well suited for highly dynamic sources as more training points are available and hence more accurate models can be learned.

3.3 Modeling Changes in the Overall Data Domain

In this section we describe a collection of statistical models for learning the change patterns of the underlying data domain from the available historical source snapshots over the time window T. We focus first on domains where data sources provide structured data entries and then discuss domains with unstructured entries.

3.3.1 Structured Data Entries

To extract the change patterns of the underlying data domain from source-based snapshots, we first need to integrate the available snapshots by solving the *history integration* problem [46, 118]. Namely, we need to unify the streams of the sources into a single stream describing the evolution of the world. We rely on the techniques proposed by Dong et al. [46] and Pal et al. [118] for doing this. Once we have a stream containing the changes in the underlying data domain, we can use the models described next to learn its change patterns.

Recall that for a data domain $\mathcal{D}$, the following assumptions are adopted: (a) entry appearances, disappearances and value changes follow a Poisson random process; and (b) the lifespan of an entry and the time interval for which it does not get updated follow an exponential distribution.

Appearances: The number of entry appearances $N_i(\cdot)$ during the time interval $(t, t+\tau]$ follows a Poisson distribution with intensity parameter λ_i:

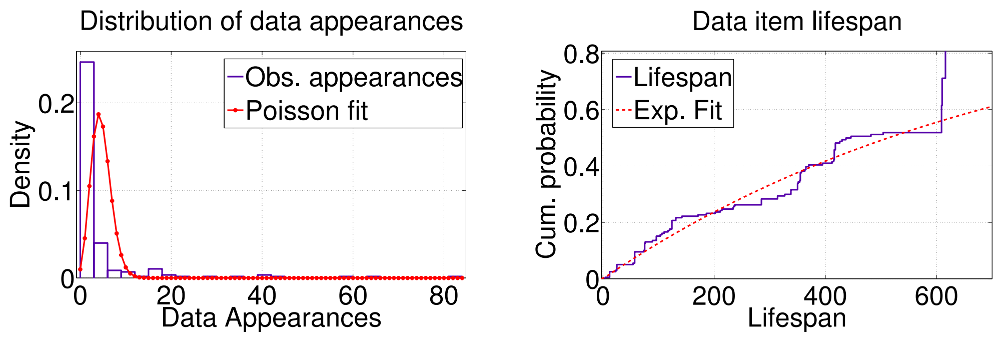
$$\Pr[(N_i(t+\tau) - N_i(t)) = k] = \frac{e^{-\lambda_i \tau}(\lambda_i \tau)^k}{k!} \tag{3.1}$$

The parameter λ_i is approximated by its maximum likelihood estimate (MLE) corresponding to the average rate of data appearances in $\mathcal{D}$. To compute this, the time window T is divided into intervals of fixed length, and the average occurrence rate of entry appearances is calculated over these intervals. The starting point of the Poisson process is extracted by calculating the total number of entries in the world by the end of T.

Disappearances: The lifespan of an entry follows an exponential distribution with rate parameter γ_d, i.e., the probability that the lifespan of an entry is at most τ is $F_d(\tau) = 1 - e^{-\gamma_d \tau}$. The parameter γ_d is approximated by its MLE which is equal to the inverse of the average entry lifespan observed over the time window T. Due to the fixed length of the historical time window the available data contains incomplete observations, that is, there are entries for which only a lower bound of their lifespan is known but not their exact lifespan since they did not disappear until the end of T. These observations are called *right censored* and the MLE of γ_d for right censored data is given by:

$$\gamma_d^{-1} = \frac{\text{total lifespan of entries}}{\text{number of disappeared entries}} \tag{3.2}$$

According to the superposition property of Poisson processes [59], if the appearances of entries occur based on a Poisson process and the lifespan of each entry follows an exponential distribution, the disappearances of entries should also occur based on a Poisson random process with an intensity rate λ_d. Given a time window $(t, t+\tau]$ and with $|\mathcal{D}|_x$ denoting the total number of entries in the world at time x one has that $\lambda_d = \frac{1}{\tau} \sum_{x=t}^{t+\tau} \gamma_d \cdot |\mathcal{D}|_x$. The intensity parameter λ_d can be estimated by its MLE which corresponds to the average rate of disappearances over the time window T.



(a) Fitting a Poisson distribution to the appearances of data items per time point in BL.

(b) Fitting an exponential distribution to the lifespan of data items in BL.

Figure 3.2: Fitting change models for the business listing dataset (BL).

Value Updates: Assume that the interval between consecutive value changes of an entry follows an exponential distribution with parameter γ_u. This parameter can be learned similarly to entry disappearances. Moreover, one can easily show that value updates in the world occur based on a Poisson random process with intensity parameter λ_u, following the same steps presented above.

The models described above can be used to predict the content of the underlying data domain at future time points.

Discussion: The aforementioned modeling is presented considering the entire data domain $\mathcal{D}$ for ease of exposition. However, these techniques are directly generalizable to heterogeneous data domains where different subdomains $\mathcal{D}_{<i>} \subseteq \mathcal{D}$ exhibit different change patterns, such as the business listing and GDELT domains presented in Section 1.1. In the case of heterogeneous data domains, a collection of separate models for different homogeneous data subdomains is required. This enables capturing non-uniform change patterns commonly observed in real-world domains.

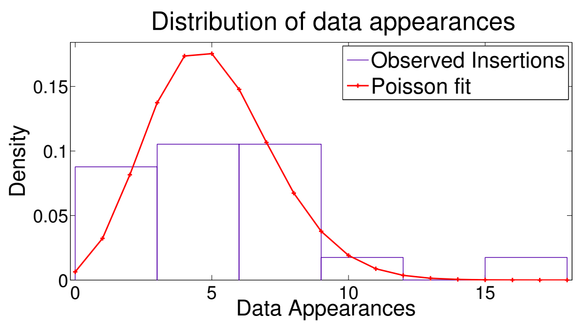


Figure 3.3: Fitting a Poisson distribution to data appearances for GDELT.

Figures 3.2(a) and 3.3 present evidence that both BL and GDELT (Section 1.1) fit these assumptions. Studying the distribution of observed appearances per day for various domain points in BL and GDELT, one observes that indeed the number of updates per day follows a Poisson distribution. The figures show show the fitted and exact distribution for a domain point in BL and GDELT respectively. For BL, Figure 3.2(b) shows the observed and fitted lifespan of data entries for the same domain point as before. Indeed the lifespan of entries follows an exponential distribution. The observed cumulative distribution for the lifespan presents a peak after 600 days which corresponds to censored data. Similar results were observed for all points in both domains.

3.3.2 Unstructured Data Entries

In this section, we focus on domains where sources provide unstructured data entries. Sources are assumed to provide entries with textual information that are associated with a time stamp and a structured metadata tuple following the schema defined by $\mathcal{A}_D$. For example, such a data domain can correspond to news articles associated with a news paper, a time stamp, and potentially additional metadata such as the location corresponding to the article. Let $\mathcal{V}_{\mathcal{A}_D}$ denote the union of all attribute values for attributes in $\mathcal{A}_D$. We now need to find the hidden topics that *best summarize* the entries in the data domain, their temporal patterns and prominence with respect to the metadata tuple values in $\mathcal{V}_{\mathcal{A}_D}$.

Following the closed-world assumption, we obtain information about the evolution of the underlying data domain by collectively analyzing the source snapshots. We view each source $s \in \bar{\mathbf{S}}$ as an *evolving document* containing N_s entries corresponding to $<$word,time stamp, attribute value$>$ tuples. Since we may have multiple attributes for each $<$word, time stamp$>$ combination in a source, we have $|\mathcal{A}_D|$ entries. Let $\{1, 2, \ldots, T\}$ be the time points associated with the available snapshots and assume that the words in the documents come from a vocabulary V. To deal with the topic and pattern discovery problem, we introduce a temporal topic model that explicitly models time and each of the metadata attribute value in $\mathcal{V}_{\mathcal{A}_D}$ jointly with the word co-occurrence patterns over the entries of the data sources. The output of this model can be used to characterize the evolution of the content of the world.

Two topic models that are related to the proposed approach are the basic Latent Dirichlet Allocation (LDA) model [15] and the Author-Topic (AT) model [139]. LDA is a Bayesian network that generates a document using a mixture of topics. In its generative process, for each document d, a multinomial distribution θ_d over topics is randomly sampled from a Dirichlet with parameter α, and then to generate each word, a topic z_{di} is chosen from this topic distribution, and a word, w_{di}, is generated by randomly sampling from a topic-specific multinomial distribution $\phi_{z_{di}}$. Rosen-Zvi et al. [139], extended the basic LDA model by explicitly modeling author's interests as a mixture of topics and used author and topic specific distributions to model the generation of words.

Table 3.1: Notation used in this section.

Symbol	Description
K	Number of topics
$\bar{\text{S}}$	Number of sources
$\mathcal{A}_D$	Metadata attributes
$\mathcal{V}_{\mathcal{A}_D}$	Metadata attribute values
V	Number of words
T	Number of discrete time-points
N_s	Number of entries in each source s
θ_v	Topic multinomial distr. for value $v \in \mathcal{V}_{\mathcal{A}_D}$
ϕ_z	Word multinomial distr. for topic z
ξ_z	Time point multinomial distr. for topic z
z_{si}	Topic of the ith entry from source s
v_{si}	Metadata attribute value of the ith entry from source s
w_{si}	Word of the ith entry from source s
t_{si}	Time point of the ith entry from source s

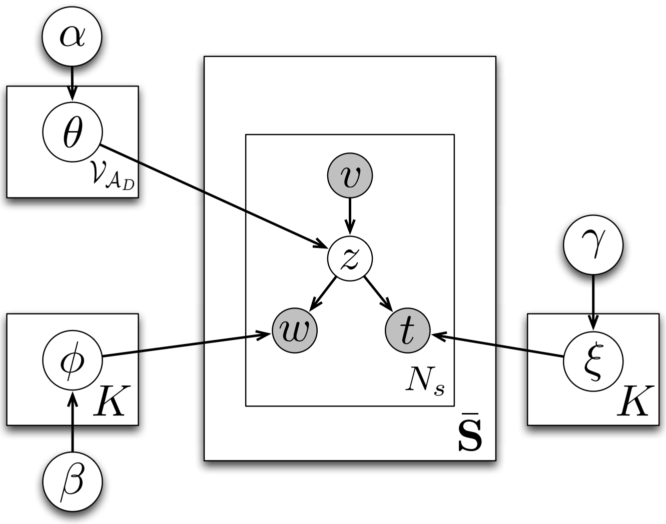


Figure 3.4: Plate notation for the proposed topic model.

In the proposed temporal topic model, values from $\mathcal{V}_{\mathcal{A}_D}$ play a similar role to that of authors in the AT model. Topic discovery is influenced not only by word co-occurrences, but also temporal information and information associated with the values from $\mathcal{V}_{\mathcal{A}_D}$. The notation used in the section is summarized in Table 3.1, and the graphical model representation of the model is shown in Figure 3.4. The model's generative process for the word and time point of each source entry is:

Topic model generative process

- Draw K multinomials $\phi_z \sim \text{Dir}(\beta)$ for each topic z
- Draw K multinomials $\xi_z \sim \text{Dir}(\gamma)$ for each topic z
- Draw $|\mathcal{A}_D|$ multinomials $\theta_v \sim \text{Dir}(\alpha)$ for each metadata value $v \in \mathcal{V}_{\mathcal{A}_D}$
- For each source $s \in \bar{\mathbf{S}}$ and entry $i \in N_s$ with v_{si}:
  - Draw a topic z_{si} from the multinomial $\theta_{v_{si}}$
  - Draw a word w_{si} from multinomial $\phi_{z_{si}}$
  - Draw a time-point t_{si} from multinomial $\xi_{z_{si}}$

Each source entry is associated with an attribute value $v_{si} \in \mathcal{V}_{\mathcal{A}_D}$. Consider a distribution $\theta_{v_{si}}$ over topics that is randomly sampled from a Dirichlet with parameter α. To generate each entry $i \in N_s$ for source s, first, a topic z_{si} is chosen from the topic distribution $\theta_{v_{si}}$, and then, a word w_{si} and time-point t_{si} are generated by randomly sampling from the topic-specific multinomial distributions $\phi_{z_{si}}$ and $\xi_{z_{si}}$. A fixed number of topics

K is assumed. We use a Gibbs sampling algorithm to perform approximate inference. Using a Dirichlet conjugate prior for the multinomial distributions allows to easily integrate out θ, ϕ and ξ. To estimate the model parameters, we calculate the conditional probability distribution $\Pr(z_{si}|\mathbf{w}, \mathbf{t}, \mathbf{v}, \mathbf{z}_{-si}, \alpha, \beta, \gamma)$ where $\mathbf{z}_{-si}$ represents the topic assignments for all entries in s except the i-th entry.

$$\begin{aligned}
&\Pr(z_{si}|\mathbf{w}, \mathbf{t}, \mathbf{v}, \mathbf{z}_{-si}; \alpha, \beta, \gamma) \\
&= \frac{\Pr(z_{si}, w_{si}, t_{si}|\mathbf{w}_{-si}, \mathbf{t}_{-si}, \mathbf{v}, \mathbf{z}_{-si}; \alpha, \beta, \gamma)}{\Pr(w_{si}, t_{si}|\mathbf{w}_{-si}, \mathbf{t}_{-si}, \mathbf{v}, \mathbf{z}_{-si}; \alpha, \beta, \gamma)} \\
&\propto \frac{n_{w_{si}}^{k,-(s,i)} + \beta_{w_{si}}}{\sum_{r=1}^{V} n_r^{k,-(s,i)} + \beta_r} \cdot \frac{m_{t_{si}}^{k,-(s,i)} + \gamma_{t_{si}}}{\sum_{t=1}^{T} m_t^{k,-(s,i)} + \gamma_t} \cdot \frac{o_{v_{si}}^{k,-(s,i)} + \alpha_{v_{si}}}{\sum_{v \in \mathcal{V}_{\mathcal{A}_D}} o_v^{k,-(s,i)} + \alpha_a}
\end{aligned} \tag{3.3}$$

where n_r^z denotes the number of times word r was associated with topic z across all sources and entries, m_t^z denotes the number of times time-point t was associated with topic z across all sources, o_v^z denotes the number of times value v was associated with topic z across all sources and their entries, and $-si$ in the superscript indicates that the current example has been excluded by the count summations. The derivation of the Gibbs sampling algorithm is deferred to Section A.1. Once the sampler has converged, the parameters of the multinomials θ, ϕ, and ξ are estimated as:

$$\begin{aligned}
\theta_{v,z} &= \frac{o_v^z + \alpha_v}{\sum_{z=1}^{K} o_v^z + \alpha_v} \\
\phi_{z,w} &= \frac{n_w^z + \beta_w}{\sum_{r=1}^{V} n_w^z + \beta_w} \\
\xi_{z,t} &= \frac{m_t^z + \gamma_t}{\sum_{t=1}^{T} m_t^z + \gamma_t}
\end{aligned} \tag{3.4}$$

Each entry in $\mathcal{D}$ is assigned a hidden topic z according to Equation 3.3, and update the appropriate counts. After the sampling, the distributions $\boldsymbol{\theta}$, $\boldsymbol{\xi}$ and $\boldsymbol{\phi}$ can be computed using Equation 3.4.

The discovered topics together with the distributions $\xi_{t,z}$ provide a concise description of the evolution of the underlying data domain $\mathcal{D}$ for the past time points $\{1, 2, \ldots, T\}$. The output of the topic model presented above can be used to predict the content of the world at future time points. Consider a future time point T_f. Estimating the content of the world at that time point requires estimating (a) the prominence of the different topics at time T_f and (b) the expected word content of the world at time T_f for any value $v \in \mathcal{V}_{\mathcal{A}_D}$. Estimating the future prominence of a topic $z \in \{1, 2, \ldots, K\}$ is equivalent to estimating the probability ξ_{z,T_f}. This can be done by using the values of distribution ξ_z corresponding to past time points. In particular, we use an autoregressive model with lag p over the values of topic z for p time points in the past giving:

$$\xi_{z,T_f} = c + \sum_{i=1}^{p} a_i \cdot \xi_{z,T_f-i} + \epsilon \tag{3.5}$$

where $a_1, a_2, \ldots.., a_p$ are the regression coefficients, c is a constant and ϵ corresponds to white noise. The word content of the world at future time points can be estimated as follows. Let $\hat{x}_{w,v,T_f}$ denote the future word count in the world at time T_f for a value $v \in \mathcal{V}_{\mathcal{A}_D}$. This is equal to:

$$\hat{x}_{w,a,T_f} = \bar{x}_w \sum_{z=1}^{K} \theta_{a,z} \cdot \phi_{z,w} \cdot \xi_{z,T_f} \tag{3.6}$$

where $\bar{x}_w$ denotes the average rate of occurrences of word w in $\mathcal{D}$ [107] over the available historical time points, and $\phi_{z,w}$, $\theta_{v,z}$, and $\xi_{z,t}$, can be retrieved by the output of the topic model and ξ_{z,T_f} is computed as in Equation 3.5.

3.4 Modeling Changes in Data Sources

In this section we provide a collection of statistical models for learning the change patterns of data sources. To do so, we compare the changes in each source over time with the changes in the unified stream corresponding to the underlying data domain. As before, we distinguish between sources providing structured and unstructured data entries.

3.4.1 Structured Data Entries

The change patterns of a data source depend on its effectiveness in capturing changes from the world. The effectiveness of a source S in capturing an entry appearance is defined as the probability $G_i(\tau)$ that S will incorporate this entry appearance in its content in a maximum of τ time units. Probabilities G_d and G_u are defined in a similar fashion for entry disappearances and value changes.

These distributions can be learned using the available historical data. For ease of exposition, we only present the learning procedure for G_i. The derivations of G_d and G_u are similar. The distribution G_i is approximated by a Kaplan-Meier empirical distribution [88] corresponding to the delay between the appearance of an entry in the world and its insertion in a source S. Given the evolution of source S and the world over the time-window T, one can extract two delay histograms characterizing the insertions in S:

(a) one corresponding to exact observations, that is, insertions of items that appeared in the world and were also inserted in S before the end of the observed time window T, and (b) one corresponding to right-censored observation, that is, insertions of items that appeared in the world during T but were not inserted in S until the end of T. These two histograms are then combined to extract the empirical distribution G_i. For example, consider the business listing domain. Figure 3.5 shows the two delay histograms corresponding to exact and right-censored observations for a source in BL, together with the learned effectiveness distribution G_i of the source.

The effectiveness distribution G_i assumes as input the duration of the time interval $t - t_c$ between a time point t and the actual occurrence of an entry appearance t_c. However, data sources get updated with a fixed frequency, and hence, the time point t may not be aligned with the latest update point of the source. The effectiveness distributions G_i, G_u and G_d can be extended to account for the common case of fixed update frequencies of the sources. Again, the discussion below focuses on G_i for ease of exposition. Given a source S that gets updated with a frequency f_S, define $T_S(t)$ to be a function that returns the latest update time point of S until time t inclusive. In particular, define $T_S(t)$ as $T_S(t) = \frac{\lfloor (t - t_0^S) f_S \rfloor}{f_S} + t_0^S$, where t_0^S denotes the last time S was updated during the historical time window T. Using this, the definition of G_i can be updated to:

$$G_i(t, t_c) = \begin{cases} G_i(T_S(t) - t_c) & \text{if } t \geq T_S(t) \geq t_c \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

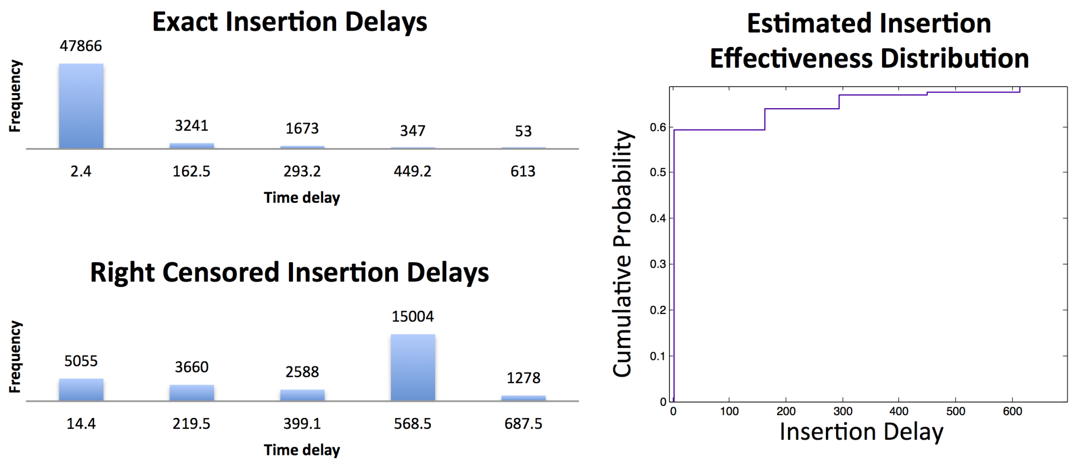


Figure 3.5: Exact and right censored insertion delay histograms with the effectiveness distribution G_i for a source in BL.

Finally, the update frequency of a source S is computed: Consider that $f_S = \frac{1}{u_S}$, where u_S denotes the average update interval of S. Let $M_S = \{t_1, t_2, \ldots, t_m\}$ be the timestamps of different content updates in S ordered by time, and let $I_S = \{t_2 - t_1, t_3 - t_2, \ldots, t_m - t_{m-1}\}$ be the set of observed time intervals for S. The parameter u_S is computed by taking the average over the elements of I_S.

Given the effectiveness distributions G_i, G_u and G_d, we can now characterize the content changes of a source as a function of the change patterns of the underlying data domain. Recall that changes in the domain $\mathcal{D}$ correspond to appearances, disappearances and value updates of entries. As a data source observes these changes it can perform one of the three following operations: (i) *insert* newly appeared data entries to its content, (ii) *delete* existing entries that disappeared from the world, and (iii) *update* the values of existing entries that got updated in the world. To characterize the actual changes in the content of a source we associate the three aforementioned operations with a probability of success. We start with insertions. Let $\Pr(\mathsf{Ins}(S,t,\tau))$ be the probability that a data appearance at time τ in $\mathcal{D}$ is reflected in S by time t. This probability is defined as:

$$\Pr(\mathsf{Ins}(S,t,\tau)) = G_i^S(T_S(t),\tau) \tag{3.8}$$

Next, we consider the deletions in S corresponding to disappeared items from $\mathcal{D}$. Let $\Pr(\mathsf{Del}(S,t,\tau))$ be the probability that an entry disappearance in $\mathcal{D}$ at time τ is captured by $F(S_I)$ until time t. In order for source S to delete this entry, the entry must already be present in S. As we discuss in Section 3.5, the probability of an entry being mentioned in a source at a particular time point is equal to its coverage. Let $\mathsf{Cov}(\mathsf{S},\tau)$ denote the coverage of source S at time τ. We have that

$$\Pr(\mathsf{Del}(S,t,\tau)) = \mathsf{Cov}(\mathsf{S},\tau)G_d^S(T_S(t),\tau) \tag{3.9}$$

Following a similar process, the probability of a value update being captured by S is:

$$\Pr(\mathsf{Upd}(S,t,\tau)) = \mathsf{Cov}(\mathsf{S},\tau)G_u^S(T_S(t),\tau) \tag{3.10}$$

These source-specific probabilities can be combined with the evolution models for the world to estimate the actual content of a source. However, we are not directly interested in the source's content but rather its quality. We discuss how to use these models to estimate the quality of arbitrary sets of sources in Section 3.5.

Discussion: Similarly to world changes, the above techniques are generalizable to sources that exhibit varying effectiveness at capturing updates for different data subdomains. In this case, a collection of separate models is required for the different homogeneous subdomains to capture the complex change patterns commonly exhibited by real-world sources.

3.4.2 Unstructured Data Entries

Now we focus on sources providing unstructured data. To model the content changes for such sources we reason about the relevance of a source's content to the topics discovered by the topic model introduced in Section 3.3.2. This problem is an instantiation of *document classification* [152]. The relevance between the content of a source and a topic can be instantiated using *cosine similarity*.

For each topic $z \in K$, the topic model outputs a distribution ϕ_z over all words V. The occurrence rate $\bar{x}_w$ for each word $w \in V$ across all entries can be used to construct an *average representative document* for each topic $z \in Z$, characterized by a vector F_z that contains the expected occurrence frequency of each word $w \in V$ given the topic. Let the w-th entry of F_z corresponding to word w be defined as $F_z[w] = \bar{x}_w \cdot \phi_{z,w}$. Similarly, given a source s, a metadata attribute value v and a time point t, the content of a source is described with a word frequency vector $F_{s,v,t}$. Given the vectors F_z and $F_{s,v,t}$ the relevance of the content of source s for value v at time t to topic z can be defined as:

$$\mathsf{Relevance}(s, z; v, t) = \mathsf{CosineSimilarity}(F_{s,v,t}, F_z) \tag{3.11}$$

where the cosine similarity of two vectors A and B is:

$$\mathsf{CosineSimilarity}(A, B) = (A \cdot B)/(\,|\,A\,|\;|\,B\,|\,) \tag{3.12}$$

However, since sources are dynamic, we need to estimate the content of each source at future time points when it is not available. Therefore, given a source s, a value v and a future time point T_f, one needs to estimate the entries of F_{s,v,T_f} considering the expected frequency of each word in source s. Let $\hat{F}_{s,v,T_f}[w]$ denote the expected frequency for word $w \in V$ when associated with the metadata value v in source s. To compute the expected frequency $\hat{F}_{s,v,T_f}[w]$, consider the conditional probability of source s mentioning word w at a future time point T_f, denoted by $\Pr(T_f|s,w)$, the conditional probability of source s publishing word w in an article related to the metadata value v, denoted by $\Pr(w|s,v)$, and the probability of word w being generated by any topic $z \in K$, given the value v and time point T_f. More precisely:

$$\hat{F}_{s,v,T_f}[w] = \bar{x}_w \cdot \Pr(T_f|s,w) \cdot \Pr(w|s,v) \cdot \sum_{z \in K} \phi_{z,w} \cdot \theta_{v,z} \cdot \xi_{z,T_f} \tag{3.13}$$

where $\bar{x}_w$ denotes the average occurrence rate of word w over the past time points, and $\phi_{z,w}$, $\theta_{v,z}$, and $\xi_{z,t}$, can be retrieved by the output of the topic model. Given the historical data, the probability $\Pr(w|v,s)$ can be estimated by its maximum likelihood as:

$$\Pr(w|s,v) = \frac{n_{w,s,v}}{\sum_{w \in V} n_{w,s,v}} \tag{3.14}$$

where $n_{w,s,v}$ denotes the number of mentions of word w from source s associated with the metadata value v. The probability ξ_{z,T_f} can be computed using the auto-regressive techniques described in the previous section (Equation 3.5). Also the probability $\Pr(T_f|s,w)$ corresponds to a future time point and needs to be estimated. According to the setup description presented above, the available historical data spans up to time point T. Thus, the probability of the source mentioning a particular word w at a future time t can be estimated considering the weighted average occurrence rate of word w in the source:

$$\Pr(T_f|s,w) = \frac{\sum_{\tau=1}^{T} \frac{1}{T_f-\tau} I(\tau,s,w)}{\sum_{\tau=1}^{T} \frac{1}{T_f-\tau}} \tag{3.15}$$

where $I(s,\tau,w)$ is an indicator variable equal to one if source s mentioned word w at least once at time τ, and zero otherwise. Eventually, the source-topic relevance for each source, metadata attribute value and topic combination at future time points is computed using the aforementioned techniques.

3.5 Quality of Integrated Data

We now provide time-dependent definitions of data quality metrics such as coverage, freshness, and accuracy to characterize the data provided by a single source or by integrating a set of sources. We mainly focus on sources providing structured data, but we discuss how some of these metrics can be used for sources providing unstructured data. Finally, we show how one can estimate these quality metrics for future time points using the statistical models introduced in Sections 3.3 and 3.4.

3.5.1 Defining Quality

The entries in a source or the integration result at a time point t can be characterized using three categories: (a) *up-to-date*, denoted by Up, corresponding to entries mentioned in the source that also exist in the world and whose attribute values in the source are in agreement with the world, (b) *out-of-date*, denoted by Out, corresponding to entries mentioned in the source that are present in the world but whose latest value changes are not captured by the source, and (c) *non-deleted*, denoted by NDel, corresponding to entries mentioned by the source that have disappeared from the world. The quality of the integration result can be defined using these categories.

Let S_I be the selected set of sources to be integrated at time t, F an integration model (e.g., majority voting), and $F(S_I)$ the integration result using model F. Coverage of $F(S_I)$ at time t, denoted by $\mathsf{Cov(F(S_I),t)}$, is defined as the probability that a random entry from the world $\mathcal{D}$ at time t belongs to $F(S_I)$. This probability is expressed as:

$$\mathsf{Cov(F(S_I),t)} = \frac{\mathsf{Up(F(S_I),t) + Out(F(S_I),t)}}{|\mathcal{D}|_\mathsf{t}} \tag{3.16}$$

where $|\mathcal{D}|_t$ denotes the total entries in the world at time t.

A localized freshness measure for the integrated data at time t corresponds to the probability that a randomly selected entry of $F(S_I)$ is up-to-date. This metric, referred to as *local freshness* and denoted by LF, can be expressed as:

$$\mathsf{LF(F(S_I),t)} = \frac{\mathsf{Up(F(S_I),t)}}{|\mathsf{F(S_I)}|_\mathsf{t}} \tag{3.17}$$

where $|F(S_I)|_t$ denotes the total number of entries in the integration result at time t. The coverage and local freshness are orthogonal, that is, a source with high-freshness does not necessarily exhibit high coverage. Moreover, while the coverage is expected to increase monotonically as more sources are integrated the same does not hold for freshness.

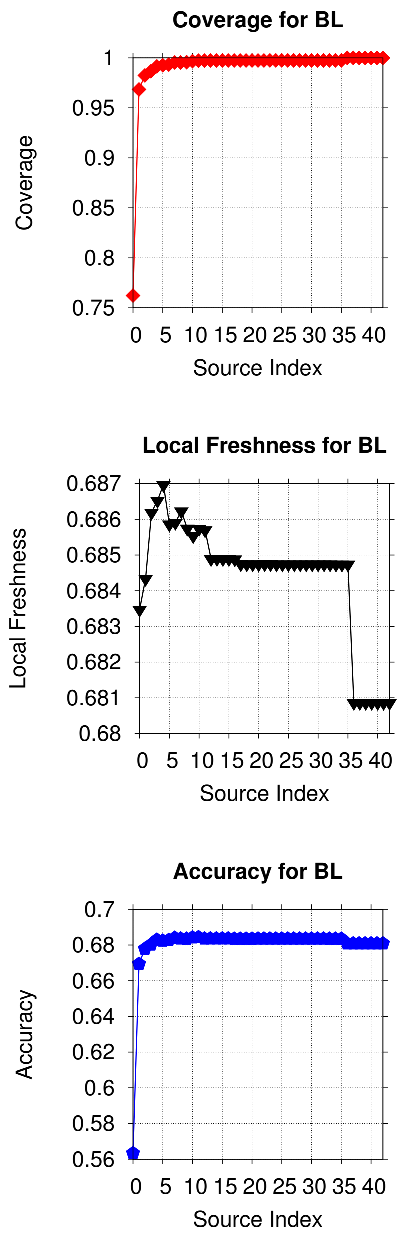


Figure 3.6: (a) Coverage, (b) freshness, and (c) accuracy of integrated data for the BL scenario introduced in Section 1.1; Sources processed in decreasing order of coverage.

Example 5. *Consider the business listing scenario introduced in Section 1.1. We integrate the available sources in decreasing order of coverage. Figure 3.6(a) shows the coverage of the integration result. Local freshness is shown in Figure 3.6(b). While coverage increases monotonically, local freshness decreases as more sources are integrated.*

In many cases, one wishes to reason about coverage and local freshness collectively. Coverage and local freshness are similar to *recall* and *precision* in information retrieval, and hence, can be combined using an F-type measure. Thus, we define a measure of *accuracy* of a source or the integration result as the *match rate accuracy* [110]. Defining accuracy requires defining a global measure of freshness, namely *global freshness*, as the probability that a randomly selected entry from Ω at time t belongs to $F(S_I)$ and its reference is up-to-date:

$$\mathsf{GF}(\mathsf{F}(\mathsf{S}_\mathsf{I}), \mathsf{t}) = \frac{\mathsf{Up}(\mathsf{F}(\mathsf{S}_\mathsf{I}), \mathsf{t})}{|\mathcal{D}|_\mathsf{t}} \tag{3.18}$$

Eventually, accuracy is defined as the percentage of correctly matched entries, corresponding to up-to-date entries in $F(S_I)$, to all entities in $F(S_I)$ together with entries that are present in Ω and not mentioned in $F(S_I)$:

$$\mathsf{Acc}(\mathsf{F}(\mathsf{S}_\mathsf{I}), \mathsf{t}) = \frac{\mathsf{Up}(\mathsf{F}(\mathsf{S}_\mathsf{I}), \mathsf{t})}{|\ \mathsf{F}(\mathsf{S}_\mathsf{I}) \cup \mathcal{D}\ |_\mathsf{t}} \tag{3.19}$$

Using Equations 3.16, 3.17 and 3.18, accuracy can be computed by:

$$\mathsf{Acc}(\mathsf{F}(\mathsf{S}_\mathsf{I}), \mathsf{t}) = \frac{\mathsf{GF}(\mathsf{F}(\mathsf{S}_\mathsf{I}), \mathsf{t})}{1 - \mathsf{Cov}(\mathsf{F}(\mathsf{S}_\mathsf{I}), \mathsf{t}) + \frac{\mathsf{GF}(\mathsf{F}(\mathsf{S}_\mathsf{I}),\mathsf{t})}{\mathsf{LF}(\mathsf{F}(\mathsf{S}_\mathsf{I}),\mathsf{t})}} \tag{3.20}$$

Figure 3.6(c) shows the accuracy corresponding to Example 5.

Unstructured Data Entries. When sources provide unstructured data entries, e.g., when the sources are news portals and the entries correspond to news articles, it is not appropriate to characterize entries as non-deleted or out-of-date. Therefore, our previous quality definitions are not directly applicable. However, the topic relevance computation presented in Section 3.4.2 is equivalent to the metric of coverage discussed above. In the case of structured entries, coverage is defined as the probability that a random entry from a source will be present in the world. Similarly, the source-topic relevance in Equation 3.11 corresponds to the probability that a source is *covering* a certain topic that is present in the world. Therefore, the techniques introduced in Section 3.4.2, not only characterize the change patterns of a source providing unstructured data but also characterize its quality (i.e., coverage). Later, in Section 7.2, we discuss how these techniques can be used to derive source quality metrics that are application specific.

3.5.2 Estimating Quality

Now, we discuss how coverage, freshness, and accuracy of the integration result for a set of data sources S_I can be estimated at a future time point t. The quality of the integration result for S_I is affected by the content changes of sources in S_I. First, we extend the techniques discussed in Section 3.4.1 to estimate the content changes in $F(S_I)$ and then present how the different quality metrics can be estimated. We point out that this section focuses on *independent* sources, i.e., sources that update their content independently. We discuss source dependencies, e.g., sources copying from each other, in Chapter 6.

3.5.2.1 Content Changes Under Union Semantics

Given a set of sources S_I one wants to estimate the content of integrated data $F(S_I)$ at a future time point t. For this, one needs to characterize the content of $F(S_I)$, in terms of up-to-date, out-of-date and non-deleted entries, at the end t_0 of the available historical time window T and then examine how the content of $F(S_I)$ changes by estimating how effectively the entry appearances, disappearances and value changes occurring in the world up to time $t \succ t_0$ are captured in $F(S_I)$.

To determine the content of $F(S_I)$ for a set S_I at time t_0, we consider the up-to-date, out-of-date and non-deleted entries in each source $S \in S_I$ extracted by comparing the content of S with the actual entries in the world. The set of up-to-date entries in $F(S_I)$ is computed by taking the union of up-to-date entries across all sources in S_I. The set of out-of-date and non-deleted entries are extracted in a similar fashion. Conflicts between entries that are up-to-date in one source and out-of-date in another are resolved by considering only the reference with the most recent time-stamp.

Procedurally, we store three different signatures (bit arrays) for each source $S \in \bar{\mathbf{S}}$: (a) a signature B_S^{up} for the up-to-date items, (b) a signature B_S^{cov} for the up-to-date and out-of-date (i.e., the covered) items, and (c) a signature B_S for all the items in the source. All similar bit arrays have the same size across different sources. Using these signatures the number of entries mentioned in $F(S_I)$ is $|\bigvee_{S \in S_I} B_S|$, the number of up-to-date entries is $|\bigvee_{S \in S_I} B_S^{up}|$, and the number of covered entries is $|\bigvee_{S \in S_I} B_S^{cov}|$.

To estimate the content changes in $F(S_I)$ at time t, the effectiveness of S_I in capturing changes in the world needs to be estimated. We follow a similar approach to that described in Section 3.4.1 for a single source. We focus on insertions of new entries. Let $\Pr(\mathsf{Ins}(F(S_I), t, \tau))$ be the probability that a data appearance at time τ was captured in $F(S_I)$ by time t. Following the union semantics, this probability corresponds to the probability that at least one source in S_I capturing the appearance of the new data item. Let $\mathsf{Ins}(S, t, \tau)$ be a boolean indicator variable taking the value true when a data appearance at time τ was captured by source S until time t. The probability $\Pr(\mathsf{Ins}(F(S_I), t, \tau))$ is:

$$\Pr(\mathsf{Ins}(F(S_I), t, \tau)) = \Pr(\bigvee_{S \in S_I} \mathsf{Ins}(S, t, \tau) = True) \tag{3.21}$$

In the presence of arbitrary source correlations computing this equation is not efficient as it corresponds to the inclusion-exclusion formula, thus, it may contain exponentially many terms. However, when sources in S_I are *independent* we have:

$$\begin{aligned} \Pr(\mathsf{Ins}(F(S_I), t, \tau)) &= \left(1 - \prod_{S \in S_I} (1 - \Pr(\mathsf{Ins}(S, t, \tau)))\right) \\ &= \left(1 - \prod_{S \in S_I} \left(1 - G_i^S(T_S(t), \tau)\right)\right) \end{aligned} \tag{3.22}$$

We discuss how one can compute the probability of such boolean formulas evaluating to true under the presence of source correlations in Chapter 6. For the remainder of this chapter we focus on independent sources.

Next, consider deletions in $F(S_I)$ corresponding to disappeared items from the world. Again, we adopt the union semantics of integration. Let $\Pr(\mathsf{Del}(F(S_I), t, \tau))$ be the probability that an entry disappearance at time τ was captured by $F(S_I)$ until time t and $\mathsf{Del}(S, t, \tau))$ be the corresponding indicator variable for each source $S \in S_I$.

This probability corresponds to the probability that at least one of the sources in S_I that mentioned this entry at time τ captured the disappearance event until time t. We consider that sources are independent. According to Equation 3.16, the probability of an entry being mentioned in a source at a particular time point is equal to its coverage.

$$\begin{aligned}\Pr(\mathsf{Del}(F(S_I), t, \tau)) &= \Pr(\bigvee_{S \in S_I} \mathsf{Del}(s, t, \tau) = True) \\ &= 1 - \prod_{S \in S_I} (1 - \mathsf{Cov}(\mathsf{S}, \tau)\mathsf{G}^{\mathsf{S}}_{\mathsf{d}}(\mathsf{T}_{\mathsf{S}}(\mathsf{t}), \tau))\end{aligned} \tag{3.23}$$

Following a similar process, the probability of a value update getting captured is:

$$\begin{aligned}\Pr(\mathsf{Upd}(F(S_I), t, \tau)) &= \Pr(\bigvee_{S \in S_I} \mathsf{Upd}(S, t, \tau) = True) \\ &= 1 - \prod_{S \in S_I} (1 - \mathsf{Cov}(\mathsf{S}, \tau)\mathsf{G}^{\mathsf{S}}_{\mathsf{u}}(\mathsf{T}_{\mathsf{S}}(\mathsf{t}), \tau))\end{aligned} \tag{3.24}$$

where $\mathsf{Upd}(S, t, \tau))$ is an indicator variable for a source $S \in S_I$ that is true if source S captured an updated from the world at time τ by time t.

3.5.2.2 Quality Estimation at Future Time Points

We now describe how to estimate the coverage and freshness of the integrated data for a future time point $t \succ t_0$. Accuracy can be derived using Equation 3.20.

Coverage: Let $\mathbb{E}[|\mathcal{D}|_t]$ be the expected number of entries in the world at time t, $\mathbb{E}[\mathsf{Ins}(F(S_I), t)]$ be the expected number of entries of newly appeared entries in the world that have not been deleted until time t and were also insured in $F(S_I)$ up to time t, and $\mathbb{E}[\mathsf{OldCov}(F(S_I), t)]$

the expected number of entries that were already covered by $F(S_I)$ at time t_0 and have not disappeared from the world until time t. The coverage is computed as follows:

$$\mathsf{Cov}^*(\mathsf{F}(\mathsf{S_I}), \mathsf{t}) = \frac{\mathbb{E}[\mathsf{OldCov}(\mathsf{F}(\mathsf{S_I}), \mathsf{t})] + \mathbb{E}[\mathsf{Ins}(\mathsf{F}(\mathsf{S_I}), \mathsf{t})]}{\mathbb{E}[|\mathcal{D}|_{\mathsf{t}}]} \tag{3.25}$$

To compute $\mathbb{E}[\mathsf{OldCov}(F(S_I), t)]$, consider the number of covered entries in $F(S_I)$ at t_0, i.e., the sum of up-to-date and out-of-date entries, and multiply that with the probability of an entry not disappearing until time t. Using the memoryless property of the Poisson process for data disappearances one obtains:

$$\mathbb{E}[\mathsf{OldCov}(F(S_I), t)] = \mathsf{Cov}(\mathsf{F}(\mathsf{S_I}), \mathsf{t_0}) \cdot |\mathcal{D}|_{\mathsf{t_0}} \cdot \mathsf{e}^{-\gamma_{\mathsf{d}}(\mathsf{t}-\mathsf{t_0})} \tag{3.26}$$

where $\mathsf{Cov}(\mathsf{F}(\mathsf{S_I}), \mathsf{t_0})$ and $|\mathcal{D}|_{t_0}$ can be computed by the signatures and extracted statistics described above. Since data appearances and disappearances occur based on a Poisson process the quantity $\mathbb{E}[|\mathcal{D}|_t]$ is:

$$\mathbb{E}[|\mathcal{D}|_t] = |\mathcal{D}|_{t_0} + \sum_{\tau=t_0}^{t} [\lambda_i - \lambda_d] \tag{3.27}$$

One can compute $\mathbb{E}[\mathsf{Ins}(F(S_I), t)]$ using Equation 3.22 and the fact that data appearances follow a Poisson random process and the entry lifespan is exponentially distributed:

$$\mathbb{E}[\mathsf{Ins}(F(S_I), t)] = \sum_{\tau=t_0}^{t} \lambda_i \cdot e^{-\gamma_d(t-\tau)} \cdot \Pr(\mathsf{Ins}(F(S_I), t, \tau)) \tag{3.28}$$

The coverage estimator corresponds to a non-decreasing submodular function. A set function $G : 2^V \rightarrow \mathbb{R}$ mapping subsets $A \subseteq V$ into the real numbers is *submodular* [52] if for all $A \subseteq B \subseteq V$, and $v' \in V \setminus B$, it holds that $G(A \cup \{v'\}) - G(A) \geq G(B \cup \{v'\}) - G(B)$ (i.e., adding v' to a set A increases G no less than adding v' to a superset B of A). Function G is *nondecreasing*, if for every $A \subseteq B \subseteq V$, it holds that $G(A) \leq G(B)$.

Theorem 1 (Submodular Coverage)**.** *The coverage estimate* $\mathsf{Cov}^*(\cdot)$ *for any set of independent sources S_I and time t is a non-decreasing submodular function.*

Proof [Sketch] The coverage estimator (Equation 3.25) is a non-decreasing submodular function as it is a non-negative linear combination of two monotonic submodular functions. The first function referring to the coverage of $F(S_I)$ at time t_0, can be shown to be non-decreasing submodular as it is derived by the set union function. The second function, corresponding to future time points, is also non-decreasing submodular as it is derived from the probability inclusion exclusion formula for independent events. A detailed proof is provided in Section A.2. ■

Freshness: Let $\mathbb{E}[\mathsf{Up}(\mathsf{F}(\mathsf{S_I}), \mathsf{t})]$ be the expected number of up-to-date entries in the integration $F(S_I)$ of S_I at time t, $\mathbb{E}[|F(S_I)|_t]$ be the expected number of all entries in $F(S_I)$, and $\mathbb{E}[|\mathcal{D}|_t]$ the expected number of entries in the world at time t. The local and global freshness for S_I are estimated as:

$$\mathsf{LF}^*(\mathsf{F}(\mathsf{S_I}),\mathsf{t}) = \frac{\mathbb{E}[\mathsf{Up}(\mathsf{F}(\mathsf{S_I}),\mathsf{t})]}{\mathbb{E}[|F(S_I)|_t]} \tag{3.29}$$

$$\mathsf{GF}^*(\mathsf{F}(\mathsf{S_I}),\mathsf{t}) = \frac{\mathbb{E}[\mathsf{Up}(\mathsf{F}(\mathsf{S_I}),\mathsf{t})]}{\mathbb{E}[|\mathcal{D}|_t]} \tag{3.30}$$

First, consider $\mathbb{E}[|F(S_I)|_t]$. To compute this quantity, the number of newly inserted and newly deleted entries in $F(S_I)$ until time t needs to be estimated. Let $|F(S_I)|_{t_0}$ be the number of entries in $F(S_I)$ at t_0 computed by the signatures in Section 3.5.2.1:

$$\mathbb{E}[|F(S_I)|_t] = |F(S_I)|_{t_0} + \mathbb{E}[\mathsf{Ins}(F(S_I),t)] - \mathbb{E}[\mathsf{Del}(F(S_I),t)] \tag{3.31}$$

where $\mathbb{E}[\mathsf{Ins}(F(S_I),t)]$ is as in Equation 3.28 and $\mathbb{E}[\mathsf{Del}(F(S_I),t)]$ denotes the expected number of deleted items from $F(S_I)$. To compute the expected number of deleted items multiply the average number of data disappearances per time unit λ_d given by the Poisson occurrence of data disappearances with the probability that an entry disappearance was captured by the sources in S_I. The probability $\Pr(\mathsf{Del}(F(S_I),t,\tau))$ is as in Equation 3.23:

$$\mathbb{E}[\mathsf{Del}(F(S_I),t)] = \sum_{\tau=t_0}^{t} \lambda_d \cdot \Pr(\mathsf{Del}, F(S_I),t,\tau)) \tag{3.32}$$

The expected up-to-date items $\mathbb{E}[\mathsf{Up}(\mathsf{F}(\mathsf{S_I}),\mathsf{t})]$ can be expressed as the summation of three quantities:

- $\mathbb{E}[\text{OldUp}]$: the expected up-to-date entries already present in $F(S_I)$ that did not change in the world until time t.

- $\mathbb{E}$[InsUp]: the expected newly inserted entries in $F(S_I)$ that appeared in the world during $[t_0, t]$ and their values were not updated until t.
- $\mathbb{E}$[ExUp]: the expected entries that were present in both $F(S_I)$ and the world, their latest update was captured in $F(S_I)$, and have not disappeared from the world by t.

Eventually, $\mathbb{E}[\mathsf{Up}(\mathsf{F}(\mathsf{S_I}), \mathsf{t})] = \mathbb{E}[\text{OldUp}] + \mathbb{E}[\text{InsUp}] + \mathbb{E}[\text{ExUp}]$.

To compute the three aforementioned quantities one first needs to compute the probability of an entry not disappearing until t, denoted by $\Pr(\text{In } \mathcal{D} \text{ at } t)$ and the probability of none of its values getting updated during $[t_0, t]$, denoted by $\Pr(\text{Not Upd.}, t)$. Since the lifespan and update intervals follow exponential distributions, $\Pr(\text{In } \mathcal{D} \text{ at } t) = e^{-\gamma_d(t-t_0)}$ and $\Pr(\text{Not Upd.}, t) = e^{-\gamma_u(t-t_0)}$. Finally, recall that the up-to-date entries in $F(S_I)$ are $\mathsf{Up}(\mathsf{F}(\mathsf{S_I}), \mathsf{t_0}) = |\bigvee_{\mathsf{S} \in \mathsf{S_I}} \mathsf{B}^{\mathsf{up}}_{\mathsf{S}}|$. According to the Poisson arrival of changes:

$$\begin{aligned}
\mathbb{E}[\text{OldUp}] &= \mathsf{Up}(\mathsf{F}(\mathsf{S_I}), \mathsf{t_0}) \Pr(\text{In } \mathit{world} \text{ at } \mathsf{t}) \Pr(\text{Not upd.}, \mathsf{t}) \\
\mathbb{E}[\mathsf{InsUp}] &= \sum_{\tau=t_0}^{t} \lambda_i \Pr(\text{In } \mathcal{D} \text{ at } t) \Pr(\text{Not upd.}, t) \Pr(\mathsf{Ins}(F(S_I), t, \tau)) \\
\mathbb{E}[\mathsf{ExUp}] &= \sum_{\tau=t_0}^{t} \lambda_u \Pr(\text{In } \mathcal{D} \text{ at } t) \Pr(\text{Not upd.}, t) \Pr(\mathsf{Upd}(F(S_I), t, \tau))
\end{aligned}$$

where $\Pr(\mathsf{Ins}(F(S_I), t, \tau))$ and $\Pr(\mathsf{Upd}(F(S_I), t, \tau))$ are as in Equations 3.22 and 3.24.

The global freshness estimate is also a non-decreasing submodular function. However, the same does not hold for local freshness.

Theorem 2 (Submodular Global Freshness)**.** *The global freshness estimate* $\mathsf{GF}^*(\cdot)$ *for any set of independent sources* S_I *and time t is a non-decreasing submodular function.*

The proof of this theorem follows similar steps to the previous proof.

Estimator Complexity: Given a set of time points of interest T_f, one needs to estimate the quality for each $t \in T_f$. The run time complexity is $\mathcal{O}(\sum_{t \in T_f}(t - t_0) \cdot |S_I|)$, since evaluating the estimators presented above requires $\mathcal{O}((t - t_0) \cdot |S_I|)$ operations for each $t \in T_f$.

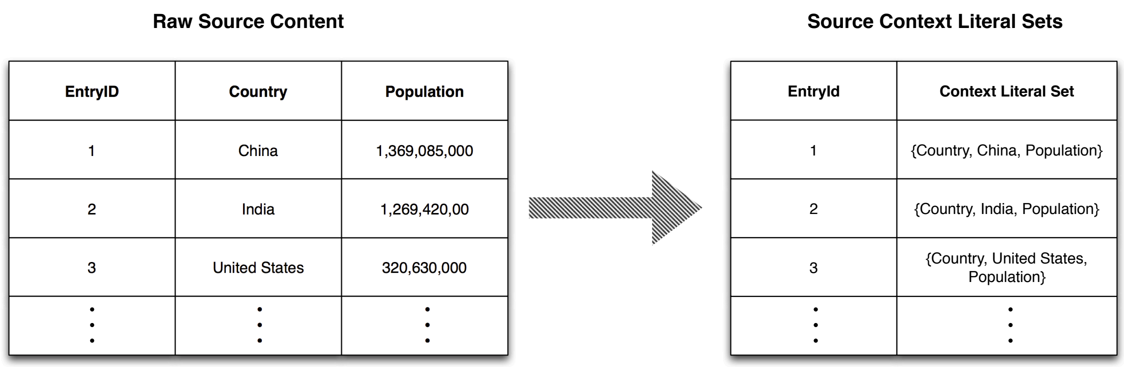
Unstructured Data Entries. When sources provide unstructured data, the quality of the integration result for a future time point can be estimated by adapting Equation 3.13 to consider the union of selected sources for integration. The expected frequency of a word w for a future time point T_f in the integration result of a set of sources S_I is given by:

$$\hat{F}_{S_I,v,T_f}[w] = \bar{x}_w \cdot \prod_{S \in S_I} (1 - \Pr(T_f|S,w) \cdot \Pr(w|S,v)) \cdot \sum_{z \in K} \phi_{z,w} \cdot \theta_{v,z} \cdot \xi_{z,T_f} \qquad (3.33)$$

3.6 Reasoning about Diverse Data Domains

The techniques introduced thus far, considered that all sources provide data from a data domain described by a pre-specified set of attributes $\mathcal{A}_D$. This is a fairly strong condition that prevents the proposed techniques from being used in a holistic approach supporting arbitrary domains with highly heterogeneous sources. We now describe a technique for relaxing this assumption.

As discussed in Section 3.2, we assume that both the non-numeric attributes in $\mathcal{A}_D$ and their values characterizing the entries of sources come from a dictionary $\mathcal{V}_{\mathcal{A}_D}$. For each entry we collapse all non-numeric attribute names and their corresponding values characterizing the entry to a set of *context literals*. An example is shown in Figure 3.7. Here, we have a source providing entries about the population of different country across



Raw Source Content

EntryID	Country	Population
1	China	1,369,085,000
2	India	1,269,420,00
3	United States	320,630,000
⋮	⋮	⋮

Source Context Literal Sets

EntryId	Context Literal Set
1	{Country, China, Population}
2	{Country, India, Population}
3	{Country, United States, Population}
⋮	⋮

Figure 3.7: An example of a source and the context literal sets for its entries.

the world. We associate each entry of the source with a context-literal set. In particular entry <China, 1,369,085,000> with schema <Country, Population> is associated with the set {Country, China, Population}. We now discuss how one can use these context literal sets to identify sets of sources that cover heterogeneous domains and build a content and quality index for diverse sources.

The context literals described above focus on non-numeric attributes. In fact, these literals often correspond to *real-world entities* and *abstract concepts*. Therefore, we can use an knowledge base, such as YAGO, Freebase, DBPedia, etc. to represent $\mathcal{V}_{\mathcal{A}_D}$. Essentially, this serves as a backbone global relaxed schema for describing arbitrary data domain. Figure 3.8 shows an example knowledge base with concept literals being hierarchically structured (e.g., “Country” is subsumed by “Location”) and entity literals being semantically associated with concept literals (e.g., “USA” has a specific “Population”). The dictionary $\mathcal{V}_{\mathcal{A}_D}$ allows one to identify the domains covered by each source by analyzing the union of context-literal sets for the entries of the source. To reason about the content and quality of different sources we augment $\mathcal{V}_{\mathcal{A}_D}$, i.e., an existing knowledge

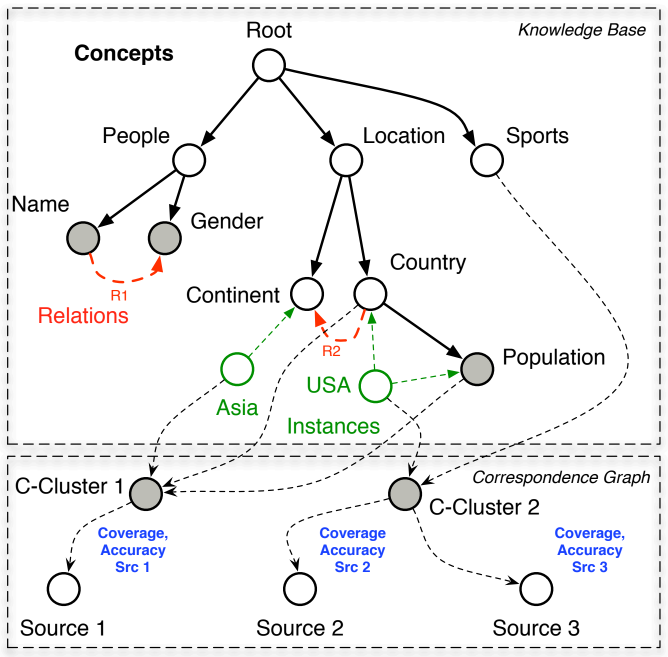


Figure 3.8: An example knowledge base extended with a correspondence graph.

base, with a *correspondence graph*. An example of a correspondence graph is shown in Figure 3.8. The nodes in the correspondence graph are either data sources (*source nodes*) or clusters of literals as dictated by the available sources (*c-cluster nodes*). The edges in the correspondence graph connect each source node with c-cluster nodes and c-cluster nodes with the corresponding literals in the knowledge base. In the example above, there are two c-cluster nodes, one corresponding to the population of countries in Asia and one to sports in the USA (i.e., "USA and Sports"). The edges connecting c-cluster nodes to literals follow conjunctive semantics. Each edge from a source to a c-cluster node is annotated with a quality profile of that source for that specific c-cluster, and each c-cluster node is associated with local information about the dependencies of the data sources that are connected to it. Intuitively, each c-cluster corresponds to a single homogeneous part of the data domain and corresponds to the world as discussed in Section 3.3.

The correspondence graph serves as a content and quality index for the available sources. To construct it we first learn the latent c-cluster nodes and then compute the quality profiles and data source dependencies for each c-cluster node. A canonical representation of the literals associated with each source entry is obtained by mapping them to entities and concepts from the available knowledge base. This is done by using semantic matching techniques introduced in the literature [101, 72]. Given these canonical representations, we construct the c-cluster nodes using a frequent pattern mining approach based on the FP-growth algorithm [71]. This allows us to discover domains that are prevalent in multiple sources. After discovering the c-cluster nodes, we compute the quality of each source, for each c-cluster node it is connected with, by using the techniques described previously in this chapter.

3.7 Experimental Evaluation

In this section we experimentally evaluate the effectiveness of the statistical models described in Section 3.3 and Section 3.4 at capturing the changes in the overall data domain and the individual sources. Our evaluation is separated in two parts one focusing on sources providing structured entries and one on sources providing unstructured entries.

3.7.1 Structured Data Entries

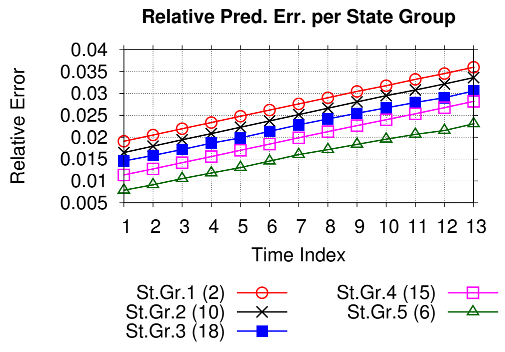
We describe the experimental setup used for evaluation and then discuss our findings.

Data. We use two real-world datasets. The first corresponds to the business listings (BL) dataset from Section 1.1 containing daily snapshots from 43 data sources providing business listings over a period of 23 months. Each data entry includes the source-id, a description of the business (i.e., phone, address, category) and the timestamp of the last insertion or update operation performed on it. A deletion timestamp is assigned to an entry using the timestamp of the latest snapshot mentioning it. If that timestamp corresponds to the end of the observed time window, the entry is assumed not to be deleted.

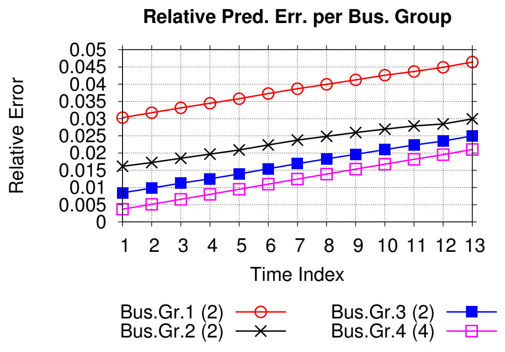
The evolution of the world is extracted by first detecting duplicates across the source snapshots using standard canonicalization and format standardization techniques together with an exact matching algorithm, and then applying an integration scheme following the union semantics described earlier. The output was verified against a gold standard provided with BL containing a subset of businesses. The sources provide 84,791,789 listings for 28,094,382 distinct businesses over 51 locations (i.e., states including Washington, DC) for 1496 business types.

The second dataset is GDELT (Section 1.1). GDELT contains daily snapshots of events extracted from articles published in 15,275 news sources over a period of 22 days. All entries contain information about the source reporting the event, and characteristics such as the actors associated with the event, the location and the type of the event. The evolution of the world is extracted using similar techniques as for BL. In total the sources provide 2,833,755 entries for 2,219,704 distinct events corresponding to 242 different locations and 236 different event types.

Preprocessing. The statistical models that describe the changes in BL are trained using the data corresponding to the first 10 months. The next 13 months are used for evaluation. For GDELT, we use the first 15 days for training and the next seven days for evaluation.



(a) State predictions.



(b) Business-type predictions.

Figure 3.9: The relative error for predicting the total listings in BL for (a) five state groups and (b) four business category groups over 13 future time points.

Experimental Results. We study the effectiveness of the proposed change models at predicting the world and source changes both in BL and GDELT.

For BL, the predicted number of businesses for the 51 locations, and the ten largest business categories is considered. The relative error between the actual and predicted values is reported in Figure 3.9(a). The states are divided in five groups based on the absolute value of their prediction error. The relative error for the representative state of each group is shown. The size of each group is mentioned in the legend. The ten largest business categories are divided in four error groups. Figure 3.9(b) shows the relative error for the representative business category of each group. Similar behavior was observed for the rest of the business categories. Our models can accurately predict the number of listings as the average relative error is around 2%. The increase rate of the error is 0.001.

For GDELT, the predicted number of events for four event-location pairs over 7 days in the future is shown in Figure 3.11(a). The prediction error is relatively small, considering that the amount of training data used in GDELT spanned over a time period of only 15 days. The variation observed is due to the dynamic nature of this domain.

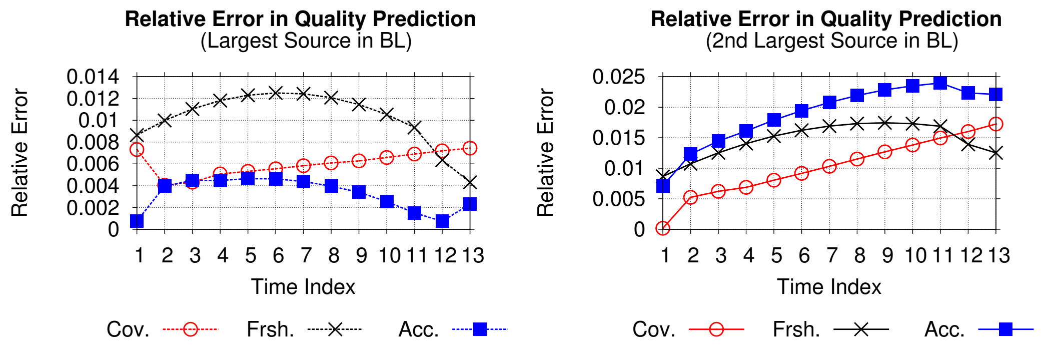
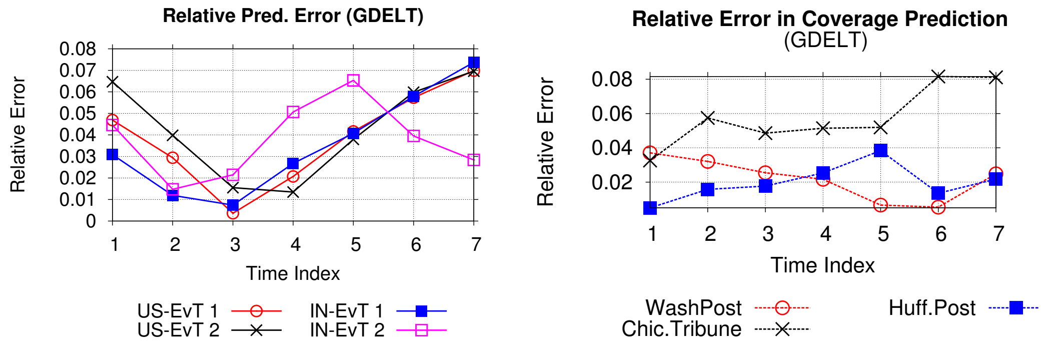


Figure 3.10: The relative error for predicting the quality of the two largest sources in BL for 13 consecutive future time points.



(a) Total-events prediction error.

(b) Quality prediction error.

Figure 3.11: The relative error for predicting (a) the total events for four different event-location pairs in GDELT and (b) the quality of three large US news sources from GDELT for 7 consecutive future time points.

Next, we evaluate the proposed models on predicting the source quality over time. Figure 3.10 shows the relative error for predicting the coverage, accuracy and local freshness of the two largest sources in BL for 13 months in the future. The maximum relative error is less than 1.5% for the largest source and less than 2.5% for the other source. Figure 3.11(b) shows the relative error corresponding to the coverage of the four largest US data sources in GDELT for 7 days in the future. Observe that the relative error is small.

3.7.2 Unstructured Data Entries

We now evaluate the model introduced inSection 3.3.2 detecting the actual topics in the underlying domain, their temporal patterns and their patterns with respect to the metadata attributes associated with the source entries.

Data. We use a dataset corresponding to a corpus of public health-related news articles and tweets extracted from HealthMap [57], a prominent online aggregator of news articles and tweets for disease outbreak monitoring and real-time surveillance of emerging public health threats. The collected articles span from January 2013 to March 2014. Articles in this dataset have only one metadata attribute corresponding to a location associated with the article. The locations correspond to states in four Latin American countries. Therefore, the vocabulary used here consists of Spanish and Portuguese words and does not contain only disease related words. Traditional IR pre-processing such as stop-word removal and term frequency modeling is performed over a fixed vocabulary of words. The dictionary contains words that are either commonly associated with diseases (e.g.,“contagious”) or words associated with a specific disease (e.g.,“rodents”, “hanta” for Hantavirus). Finally, each article is associated with a data source corresponding to an online news media.

We extract data source snapshots on a weekly basis. Because of this, the size of the input data varies over time, as new articles are added every week. The number of words ranges from 20,908 to 48,700, the number of locations from 74 to 144 and the number of data sources from 381 to 798.

Parameter Setup. The parameters of the Dirichlet priors are set to $\alpha = 2/K$, $\beta = 0.01$ and $\gamma = 0.01$ where K is the number of topics. The topic model was evaluated with $K = \{8, 12, 15\}$ and setting $K = 12$ was found to provide the most meaningful topics.

How effective is the proposed topic model in identifying disease topics in the world and their temporal patterns? The HealthMap corpus contains mentions to multiple diseases, both common and rare, over multiple countries in Latin America. The most prevalent diseases mentioned in the dataset are avian flu (i.e., type h5n1), dengue fever, swine flu (i.e., h1n1 flu), the Hantavirus pulmonary syndrome (HPS) and the Hantavirus hemorrhagic fever with renal syndrome (HFRS) [85].

While the first three diseases are widespread with a large number of incidences throughout a calendar year, Hantavirus syndromes are rather rare with a small number of incidences. To study the actual incidences for the six aforementioned diseases we used a gold standard report that gives ground truth determinations of whether a disease incidence (Hantavirus) happened in a given location. This report is determined by analysts considering multiple news sources and studying bulletins issued by health reporting organizations such as ProMED [1]. Figure 3.12 shows the Hantavirus outbreaks over time for each of the countries where outbreaks were reported. Most outbreaks occurred in Chile.

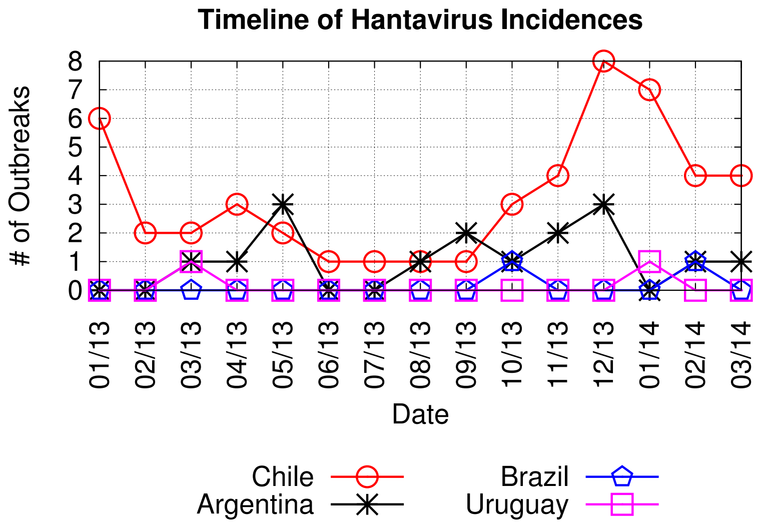
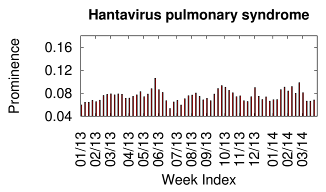
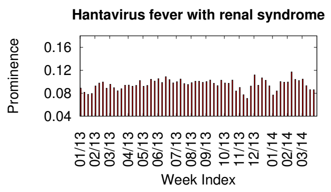
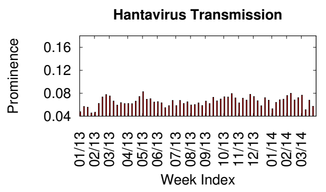


Figure 3.12: Timeline of Hantavirus outbreaks from January 2013 to March 2014 for Chile, Argentina, Brazil and Uruguay. No hantavirus outbreaks were reported for other countries in Latin America.

Now we evaluate the the topics discovered by our topic model. Six out of the twelve topics are related to the diseases mentioned above, while the rest are generic topics related to non-disease aspects of the news articles including administrative information about the reported incidents. We focus only on disease-related topics. To evaluate the disease topics, a vocabulary of 184 health-related words is considered. For each topic, the most likely words based on the health-related vocabulary and their prominence over time are reported. Given a time point, the prominence of a topic is defined as the fraction of articles of that topic over the total number of articles published at that time point.

Table 3.2 shows three topics related to Hantavirus, their most likely words based on the health-related vocabulary and their prominence histograms over time. The first topic refers to the HPS syndrome with words such as “pneumonia”, “sangre” (blood), and “cardiopulmonar” being ranked higher. One can see that the proposed topic model

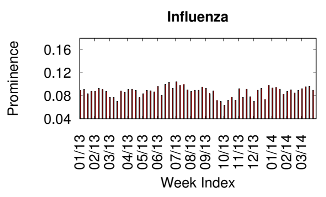
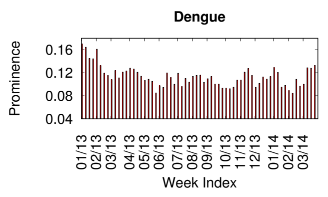
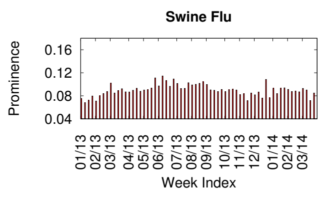
Table 3.2: Three discovered topics that are related to Hantavirus. The first two topics are related to the two different Hantavirus syndromes (i.e., HPS and HFRS) and the third topic is related to generic information about the transmission of the virus. Histograms show the topic prominence over time; The top words with their probability in each topic are shown.

					
virus	0.0468	vacuna	0.0057	paciente	0.0220
epidemia	0.0443	campos	0.0031	transmissor	0.0133
enfermos	0.0066	provincial	0.0028	lixo	0.0099
hanta	0.0068	hantavirus	0.0024	criaderos	0.0088
viral	0.0038	tosse	0.0022	respiratorias	0.0061
territorio	0.0027	nariz	0.0019	manos	0.0056
pneumonia	0.0014	estornudar	0.0011	boca	0.0047
sangre	0.0014	abdominal	0.0008	rural	0.0038
ratones	0.0006	lluvia	0.0008	musculares	0.0028
cardiopulmonar	0.0002	renal	0.0005	roedores	0.0022

is able to retrieve the correlation between words “hanta” and “ratones” (mice) successfully. The second topic focuses on the HFRS syndrome with words as “nariz” (nose), “estornudar” (sneeze), “renal” being more prevalent. Finally, the third topic focuses on the hantavirus transmission routes with words as “lixo” (garbage), “criaderos” (breeding places), “manos” (hands) and “roedores” (rodents) being ranked higher than others. According to Jonsson et al. [85] HPS is the main syndrome observed in the Americas while HFRS cases are mainly observed in Eurasia. Thus, observing a topic focusing on HFRS for Latin America seems unexpected. However, after analyzing the actual articles in the corpus, we found that articles reporting Hantavirus incidents usually mention both forms of Hantavirus syndromes for informational purposes. Focusing on the prominence histograms, one can see that the HFRS and Hantavirus transmission topics show small fluctuations across the different time points. However, one can observe that the HPS topic follows a trend similar to that of the Hantavirus incidence time line. Observe that the prominence of these topics peaks towards the end of May‘13 and from December‘13 to March‘14 exactly during the months when the number of Hantavirus incidences increases.

Next, we focus on the remaining three topics focusing on diseases other than Hantavirus. Table 3.3 shows three topics related to avian flu, dengue and swine flu. Again, their most likely words and their prominence histograms over time are reported. For all three topics one can see that the corresponding disease keywords, i.e., “influenza”, “dengue” and “gripe” (flu) are ranked first. For the avian influenza topic, the proposed topic model is able to discover the correlation among words referring to both the *causes*, i.e., “mosquito”, “larvas”, “zancudos” (mosquitos), and the *symptoms*, i.e., “fiebre” (fever), of the disease. Regarding the dengue topic, the proposed approach is able to identify the

Table 3.3: Topics related to Avian Flu, Dengue and Swine Flu. Histograms show the topic prominence over time; The top words with their probability in each topic are shown.

					
influenza	0.0567	dengue	0.2095	gripe	0.0522
mosquito	0.0495	aegypti	0.0166	h1n1	0.0351
pacientes	0.0258	agua	0.0137	infectadas	0.0043
aviar	0.0144	mosquitos	0.0058	flu	0.0024
larvas	0.0096	agricultura	0.0019	bacteria	0.0021
fiebre	0.0088	respiratoria	0.0018	enfermo	0.0008
surto	0.0061	rurales	0.0006	vacinas	0.0008
zancudos	0.0008	agropecuario	0.0006	nasal	0.0008
avian	0.0006	hemorragias	0.0005	paracetamol	0.0007
h5n1	0.0003	suero	0.0004	swine	0.0005

main transmission root of dengue which is via the aides aegypti mosquito, as well as, the fact that dengue is more prominent in rural and agricultural areas. Finally, a similar performance is observed for the swine flu topic. The proposed approach can identify the correlation between the word "bacteria" and swine flu - bacteria co-infections play a key role in swine flu deaths - and the correlation between "paracetamol" and swine flu, one of indicated medication substances for the disease.

How effective is the proposed topic model in identifying spatial patterns? We examined the correlations between the prominence of each topic and the countries under consideration (Figure 3.13). Our model was effective at determining that HPS and HFRS are more prominent in Chile compared to other countries.

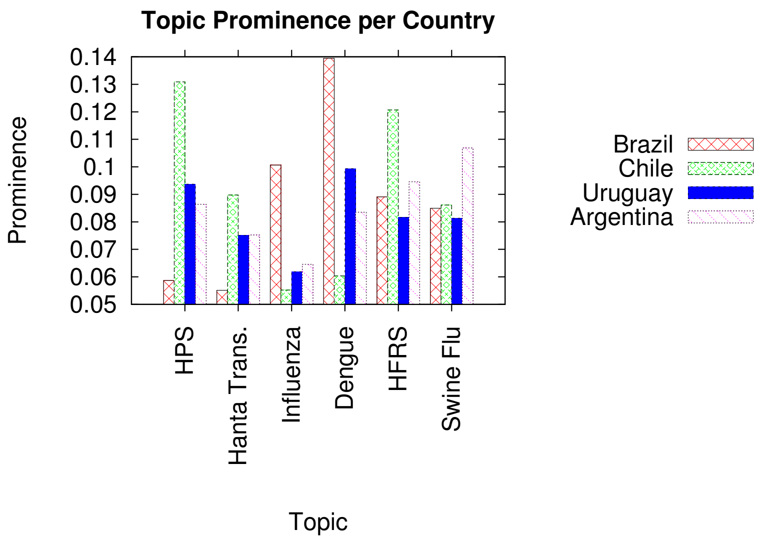


Figure 3.13: The topic prominence corresponding to the disease topics for Brazil, Chile, Uruguay and Argentina. The numbers reported per country are averaged over all states.

3.8 Related Work

Reasoning About Data Sources. A large amount of work has focused on identifying sources relevant to a given query or domain [108]. However, this work does not consider the quality of sources. A different line of work has considered the problem of online data integration [46, 118], however the proposed techniques are agnostic to the quality of sources. Moreover, a fair amount of work has considered the problem of determining the quality of multiple data sources and leveraging this information during data integration to improve the quality of the outcome [100, 109, 129, 174]. However, this work does not consider dynamic sources. Finally, Cho et al. [28], considered the problem of finding the optimal data extraction frequency from web-pages, but the authors do not reason about their quality as it is not of high importance in the web-page crawling scenario.

Topic Modeling. A number of methods have been proposed for analyzing the time evolution of topics in document collections, such as the topics over time (TOT) model [165], the dynamic topic model (DTM) [14], and TriMine model [107]. More precisely, TOT handles time-windows of fixed size and uses a Beta distribution to model the evolution of a topic over time. DTM also focuses on a time-window of fixed size but uses Kalman filters to align topics with different time points. Finally, TriMine is able to analyze windows of variable size and unlike TOT and DTM is able to find cyclic time patterns with different timescales, which enables predicting future events. While TOT and DTM focus on the dimension of time alone, TriMine can associate the generation of different modalities with topics, however is agnostic to correlations across the different modalities.

A different line of work [164] focuses on discovering spatial patterns jointly with the word co-occurrences. In particular, the authors introduced the Spatial Latent Dirichlet Allocation (SLDA), which better encodes spatial structure among words. While the model focuses on computer vision applications where documents are comprised by visual words the proposed techniques can be trivially extended to regular text documents. A similar approach was introduced by Ramage et al. [131] for labeled documents where the labels can correspond to multiple modalities, i.e., locations as well.

3.9 Summary

Most of the prior work on data source management has studied the problem of organizing heterogeneous data sources by introducing various schema-level or instance-level indexes. However, none of the previous approaches put sufficient emphasis on the quality of sources. In fact, reasoning about the content and the quality of data sources have been studied as orthogonal problems by previous approaches. This chapter fills in this gap by proposing a collection of statistical models, as well as a novel indexing technique, that enable one to simultaneously reason about the content and quality of sources. Our experimental results show the effectiveness of our approaches for diverse real-world datasets containing sources providing both structured and unstructured data.

Chapter 4: Enriching Structured Domain Indexes

In the previous chapter, we described a series of techniques for analyzing the content of heterogeneous sources and computing their quality. We also showed how these techniques can be applied to large heterogeneous data domains by exploiting the presence of a generalized structured domain index, such as an ontology or a knowledge base, and extending it with a correspondence graph that groups together related sources and indexes their quality. However, many knowledge bases follow a *closed-world* assumption, i.e., their scope is limited to real-world entries and concepts already present in them. Moreover, such indexes usually focus on the "head" of data, i.e., popular entities and concepts, leaving behind a considerable volume of "tail" data about less popular entities, non-current (historical) facts and so on.

In this chapter, we design an algorithmic framework for enriching structured domain indexes with new entities and thus relaxing the aforementioned closed world assumption. Recently, researchers have used the paradigm of *crowdsourcing* to relax the closed world assumption and operate in an *open world* setting [56]. In fact, crowdsouring has been recently proven beneficial in extracting knowledge and acquiring data for many application domains, including recommendation systems [3], knowledge base completion [93], entity extraction and structured data collection [122, 157]. Inspired by this work we build our framework on top of crowdsourced information extraction techniques. In fact, our goal is to design practical crowdsourced entity extraction techniques that exploit the structure of the existing index to minimize the monetary cost and latency incurred by issuing queries to human workers. Therefore, in this chapter we study the problem of *budgeted crowdsourced entity extraction* over structured domains.

The remainder of the chapter is organized as follows. First, in Section 4.1, we provide the reader with a primer on the problem of crowdsourced entity extraction, the challenges involved in it, and how in the presence of structured domains, one can design algorithms to address these challenges. In Section 4.2, we formally define the problem of budgeted crowdsourced entity extraction, and present the underlying query response model. Then, in Section 4.3, we introduce a methodology for estimating the number of new entities extracted from the crowd by issuing further extraction queries. In Section 4.4, we introduce an algorithm for designing querying policies inspired by the multi-armed bandits literature. Subsequently in Section 4.5 we evaluate the effectiveness of our algorithms on extracting entities and discuss related work in Section 4.6. Finally, in Section 4.7, we summarize the contributions of this chapter.

4.1 Introduction

A fundamental challenge in crowdsourced entity extraction is reasoning about the completeness of the extracted information. Given a task, e.g., “extract people from newspapers”, that seeks to extract entities from a specific domain by asking human workers, it is not easy to judge if we have extracted all entities due to the “open world” assumption [56].

Recent work [157] has considered the problem of crowdsourced entity extraction using a single type of *query* that is asked to humans; for our people case, the query will be "give me another person from New York Times". That work determines how many times this query must be asked to different human workers before we are sure we have extracted most of the people mentioned in a newspaper. However, given the monetary cost inherent in leveraging crowdsourcing, it is easy to see that just using this query repeatedly will not be practical for real-world applications, for two coupled reasons: (a) *wasted cost:* we will keep receiving the most popular entities (i.e., the "head" of data) and will have to issue many additional queries before receiving new or unseen entities, thus, increasing the cost; (b) *lack of coverage:* beyond a point all the entities we get will already be present in our set of extracted entities — thus, we may never end up receiving less popular entities (i.e., the "tail" of data) at all. To illustrate the effect of the aforementioned reasons on crowd-sourced entity extraction we conducted a real-world experiment using Amazon Mechanical Turk. We asked workers to provide us with people from five popular newspapers. Figure 4.1 shows the number of times each entity was provided by different workers. As shown, there is a small number of very popular entities reported by multiple workers. However, there is a very long tail of unpopular entities that only a single worker reported. It is easy to see that due to the skew in the underlying popularity distribution of People, we spent a significant number of queries in extracting the same popular entities over and over again.

Given the above, our main goal is to *make crowdsourced entity extraction practical*, i.e., maximize the number of unique entities extracted and focus on the tail of data. To do so, we focus on entity extraction over structured domains, i.e., a domain that can be fully

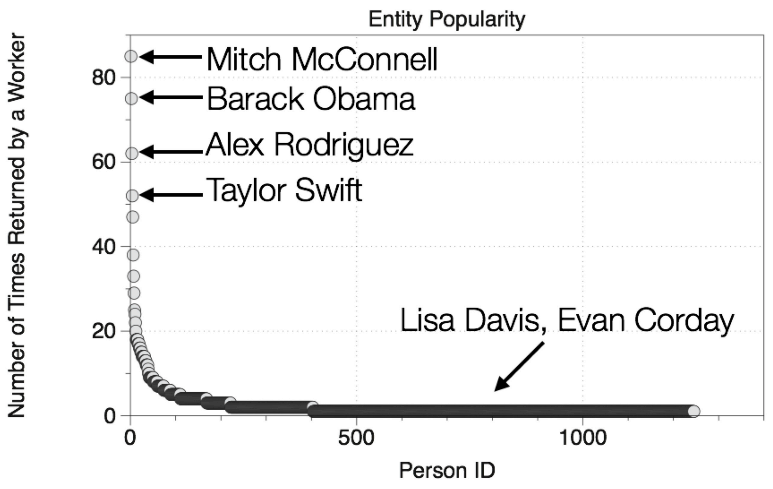


Figure 4.1: Number of times a Person was extracted by crowd workers from a collection of newspapers.

described by a collection of attributes, each potentially being hierarchically structured. For example, in our people extraction case, we could have one attribute about location, one about occupation, and one about nationality. Often the structure of domains in practical applications is already known by design. We can then leverage this structure to use a much richer space of queries asked to human workers, considering all combinations of values for each of these attributes, e.g., "give me another Basketball player from the United States that is of Greek origin". In this manner, we can leverage these *specific, targeted queries* to diversify entity extraction and obtain not-so-popular entities as well.

If we view the structured data domain as a *partially ordered set* (poset), then each query can be mapped to a node in the graph describing its topology. Thus, our goal is to traverse the graph corresponding to the input poset by issuing queries corresponding to various nodes, often multiple times at each node. However, the poset describing the domain can be often large, leading to many additional challenges in deciding which queries to issue at any node: (a) *Sparsity:* Many of the nodes in the poset are likely to be empty, i.e., the queries corresponding to those nodes are likely to not have any answers; avoiding asking queries corresponding to these nodes is essential to keep monetary cost low. (b) *Interrelationships:* Many of the nodes in the poset are "coupled" with one another; for example, the results from a few queries corresponding to "give me another Basketball player" can inform whether issuing queries corresponding to "give me another Basketball player in the United States" is useful or not. We elaborate more on these challenges in Section 4.1.1 using examples from a real-world scenario.

Previously proposed techniques [157] do not directly apply to the scenario where we are traversing a poset corresponding to this structured data domain, and new techniques are needed. The main limitation of the aforementioned techniques is that they focus on estimating the completeness of a specific query and are agnostic to cost. As a consequence they do not address the problem of deciding which additional queries are *worth* issuing. To mitigate these shortcomings, one needs to tune the queries that are asked. However, deciding which queries to ask among a large number of possible queries (exponential in the number of attributes describing the input domain) and when and how many times to ask each query, are both critical challenges that need to be addressed. Furthermore, unlike previous work, we focus on the budgeted case, where we are given a budget and we want to maximize the number of retrieved entities; we believe this is a more practical goal, instead of the goal of retrieving all entities.

4.1.1 A Real-World Scenario

To exemplify the aforementioned challenges we review a large-scale real-world scenario where crowdsourcing is used to extract entities. We consider Eventbrite[1], an online event aggregator, that relies on crowdsourcing to compile a directory of events with detailed information about the location, type, date and category of each event. Typically, event aggregators are interested in collecting information about diverse events spanning from conferences and music festivals to political rallies across different location, i.e., countries or cities. In particular, Eventbrite collects information about events across different countries in the world. Each country is split into cities and areas across the country. Moreover, events are organized according to their type and topic. The attributes and their corresponding structure are known in advance and are given by the design of the application. We collected a dataset from Eventbrite spanning over 63 countries that are divided into 1,709 subareas (e.g., states) and 10,739 cities, containing events of 19 different types, such as rallies, tournaments, conferences, conventions, etc. and a time period of 31 days spanning over the months of October and November.

Two of the three dimensions, i.e., location and time, describing the domain of collected events are hierarchically structured. The poset characterizing the domain can be fully specified if we consider the cross product across the possible values for location, event type and time. For each of the location, time, type dimensions we also consider a special *wildcard* value. Taking the cross-product across the possible values of these dimensions results in poset with a total of 8,508,160 nodes containing 57,805 distinct events

[1]`www.eventbrite.com`

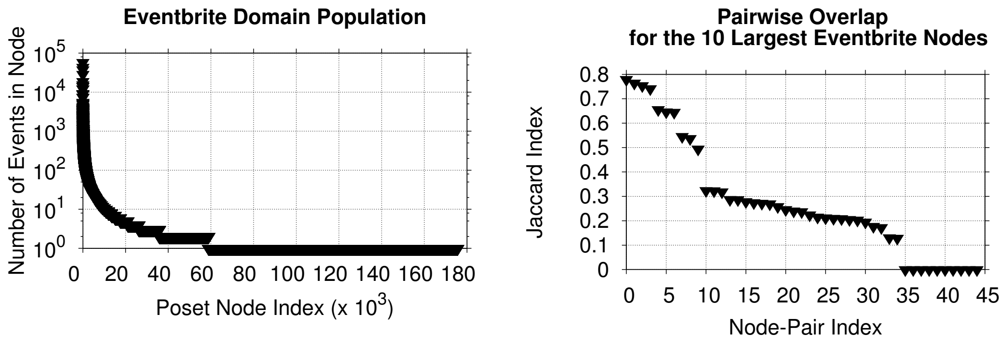


Figure 4.2: (a) The population of different nodes and (b) pairwise overlaps for the 10 most populous nodes in the Eventbrite domain.

overall. We point out that the events associated with a node in the poset overlap with the events corresponding to its descendants. First, we demonstrate how the sparsity challenge applies to Eventbrite.

Example 6. *We plot the number of events for each node in the poset describing Eventbrite's domain. Out of 8,508,160 nodes only 175,068 nodes are associated with events and the remaining are empty. Figure 4.2(a) shows the number of events per node (y-axis is in log-scale). Most of the populated nodes have less than 100 events. Additionally, the most populated nodes of the domain correspond to nodes at the higher levels of the poset. When extracting events from such a sparse domain one needs to carefully decide on the crowdsourced queries to be issued especially if operating under a monetary budget.*

As mentioned before, a critical challenge in such large domains is deciding on the queries to ask. However, the hierarchical structure of the data domain presents us with an opportunity. One approach would be to perform a top-down traversal of the poset and issue queries at the different nodes. Nevertheless, this gives rise to a series of challenges: (i) how can one decide on the number of queries to be asked at each node, (ii) when should one progress to deeper levels of the poset and (iii) which subareas should be explored. We elaborate on these in Section 4.2. Next, we focus on the second challenge, i.e., the interdependencies across poset nodes.

Example 7. *We consider again the Eventbrite dataset and plot the pairwise overlaps of the ten most populous nodes in the domain. Figure 4.2(b) shows the Jaccard index for the corresponding node pairs. As shown the event populations corresponding to these nodes overlap significantly. It is easy to see that when issuing queries at a certain node, we not only obtain events corresponding to this node but to other nodes in the domain as well.*

A critical issue that stems from the overlaps across nodes is being able to decide how many answers to expect when issuing an additional query at a node whose underlying population overlaps with nodes associated with previous queries. In Section 4.2, we elaborate more on the dependencies across nodes of the poset.

4.2 Preliminaries

In this section we first define structured domains, then describe entities and entity extraction queries or interfaces, along with the response and cost model for these queries. Then, we define the problem of *crowd entity extraction* over *structured domains* that seeks to maximize the number of extracted entities under budget constraints and present an overview of our proposed framework.

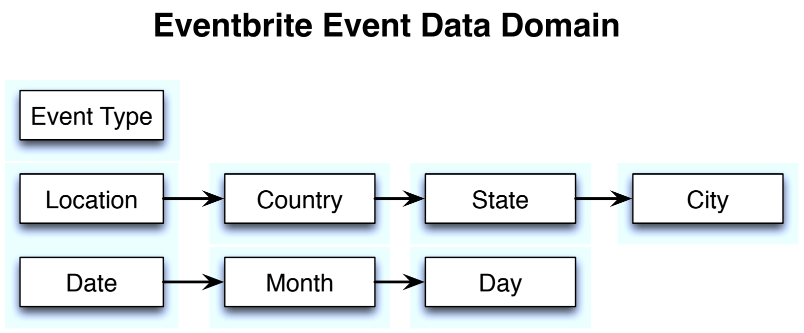


Figure 4.3: The attributes describing the Eventbrite domain and the hierarchical structure of each attribute.

4.2.1 Structured Data Domain

Let $\mathcal{D}$ be a data domain described by a set of discrete attributes $\mathcal{A}_D = \{A_1, A_2, \ldots, A_d\}$. Let $dom(A_i)$ denote the domain of each attribute $A_i \in \mathcal{A}_D$. We focus on domains where each attribute A_i is hierarchically organized. For example, consider the Eventbrite domain introduced in Section 4.1.1. The data domain $\mathcal{D}$ corresponds to all events and the attributes describing the entities in $\mathcal{D}$ are $\mathcal{A}_D = \{$"Event Type", "Location", "Date"$\}$. Figure 4.3 shows the hierarchical organization of each attribute. Notice that this definition of a structured data domain matches the knowledge bases considered in Section 3.6.

The domain $\mathcal{D}$ can be viewed as a *poset*, i.e., a partially ordered set, corresponding to the cross-product of all available hierarchies[2]. Part of the poset corresponding to the previous example is shown in Figure 4.4. We denote this cross-product as $\mathcal{H}_D$. As can be seen in Figure 4.4, there are nodes, such as $\{\}$, where no attributes are specified, and nodes, such as $\{X1\}$ and $\{C1\}$ where just one of the attribute values is specified, as well as nodes, such as $\{X2, ST2\}$, where multiple attribute values are specified.

[2]Note that $\mathcal{D}$ is not a lattice since there is no unique infimum.

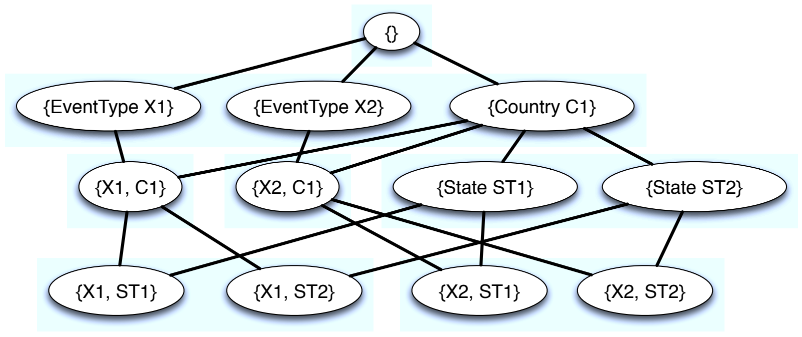


Figure 4.4: Part of the poset defining the entity domain for Eventbrite.

4.2.2 Entities and Entity Extraction Queries

Entities. Our goal is to extract entities that belong to the domain $\mathcal{D}$. We assume that each entity e can be uniquely associated with one of the leaf nodes in the hierarchy $\mathcal{H}_D$; that is, there is a unique set of "most-specific" values of $A_1, \ldots, A_d$ for every entity. For example, in Eventbrite, each entity (here, a local event) takes place in a specific city, and on a specific day. Our techniques also work for the case when entities can be associated only with "higher level" nodes, but we focus on the former case for simplicity.

Queries. Next, we describe queries for extracting entities from the crowd. First, a query q is issued at a node $v \in \mathcal{H}_D$; that is, a query specifies zero or more attribute values from $A_1, \ldots, A_d$ that are derived from the corresponding values of v, implicitly requiring the worker to find entities that match the specified attribute values.

Given a query issued at a node, there are three different configurations one can use to extract entities from the crowd: The first configuration corresponds to *single entity queries* where workers are required to provide "one more" entity that matches the specified attribute values mentioned in the query. Considering the Eventbrite example introduced in the previous section, an example of a single entity query would be asking a worker to provide "a concert in Manhattan, New York". The second configuration corresponds to *queries of size k* where workers are asked to provide up to k distinct entities. Finally, the last configuration corresponds to *exclude list queries*. Here, workers are additionally provided with a list E of l entities that have already been extracted and are required to provide up to k distinct entities that are not present in the exclude list. It is easy to see that the last configuration generalizes the previous two. Therefore, in the remainder of the chapter, we will only consider queries using the third configuration. To describe a query, we will use the notation $q(k, E)$ denoting a query of size k accompanied with an exclude list E of length l. We denote query configurations as (k, l).

Query Response. Given a query $q(k, E)$ issued at a node $v \in \mathcal{H}_D$, a human worker gives us k distinct entities that belong to the domain $\mathcal{D}$, match the specified attribute values mentioned in the query (derived from v), and are not present in E. Furthermore, the human worker provides us the information for the attributes that are not specified in q for each of the k entities. For example, if our query is "a concert in Manhattan, New York", with $k = 1, E = \emptyset$, the human worker gives us one concert in Manhattan, New York, but also gives us the day on which the concert will take place (here, the missing, unspecified attribute). If the query is "a concert in the US", with $k = 1, E = \emptyset$, the human worker gives us one concert in the US, but also gives the day on which the concert will take place, as well as the specific city. If less than k entities are present in the underlying population, workers have the flexibility to report either an empty answer or a smaller number of entities (Section 4.3.2).

While the reader may wonder if getting additional attributes for entities is necessary, note that this information allows us to reason about which all nodes in $\mathcal{H}_D$ the entity belongs to; without this, it is difficult to effectively traverse the poset. Furthermore, we find that in most practical applications, it is useful to get the values of the missing attributes to organize and categorize the extracted entities better. Similar interfaces that ask users to fully specify the attributes of entities have been proposed in recently [130].

Finally, answers are expected to be duplicated across workers, who may also specify or extract an entity incorrectly. Resolving duplicate entities during extraction is crucial as this information is later used to estimate characterize the completeness of extracted entities, and thus, reason about the gain of additional queries. Extraction errors can be resolved by leveraging the presence of duplicate information and by applying de-duplication and entity resolution techniques. At a high-level one can use an entity resolution or string similarity (e.g., Jaccard coefficient) algorithm to identify duplicate entities. Furthermore, the additional attributes for each entity, can be used to further ascertain similarity of entities. We refer the user to Getoor and Machanavajjhala [61] for an overview of entity resolution techniques. Finally, standard truth discovery techniques can be used to identify the correct attribute values for entities. Nevertheless entity resolution and truth discovery are orthogonal problems and not the focus of this chapter. In our experiments on real datasets, we found that there were no cases where humans introduced errors to the attribute values of extracted entities. Only minor errors (e.g., misspelled entity names) were detected and fixed manually.

Query Cost. In a typical crowdsourcing marketplace, tasks have different costs based on their difficulty. Thus, crowdsourced queries of different difficulties should also exhibit different costs. We assume we are provided with a cost function $c(\cdot)$ that obeys the following properties: (a) given a query with fixed size its cost should increase as the size of its exclude list is increasing, and (b) given a query with a fixed exclude list size its cost should increase as the number of requested answer increases. These are fixed upfront by the interface-designer based on the amount of work involved.

4.2.3 Crowdsourced Entity Extraction

The basic version of *crowdsourced entity extraction* [157] seeks to extract entities that belong to $\mathcal{D}$, by simply using repeated queries at the root node, with $k = 1, E = \emptyset$. When considering large entity domains, one may need to issue a series of entity extraction queries at multiple nodes in $\mathcal{H}_D$ — often overlapping with each other — so that the entire domain is covered. Issuing queries at different nodes ensures that the coverage across the domain will be maximized.

We let π denote a *querying policy*, i.e., a chain of queries at different nodes in $\mathcal{H}_D$. Notice that multiple queries $q(k, E)$ can be issued at the same node. Let $C(\pi)$ denote the overall cost, in terms of monetary cost of a querying policy π. We define the gain of a querying policy π to be the total number of unique entities, denoted by $\mathcal{E}(\pi)$ extracted when following policy π. Thus, there is a natural trade-off between the gain (i.e., the number of extracted entities) and the cost of policies.

Here, we require that the user will *only* provide a monetary budget τ_c imposing a constraint on the total cost of a selected querying policy, and optimize over all possible querying policies across different nodes of $\mathcal{H}_D$. Our goal is to identify the policy that maximizes the number of retrieved entities under the given budget constraint. We define the problem of budgeted crowd entity extraction as follows:

Problem 1 (Budgeted Crowd Entity Extraction)**.**
Let $\mathcal{D}$ be a given entity domain and τ_c a monetary budget on the total cost of issued queries. The Budgeted Crowd Entity Extraction problem seeks to find a querying policy π^* using queries $q(k, E)$ over nodes in $\mathcal{H}_D$ that maximizes the number of unique entities extracted $\mathcal{E}(\pi^*)$ under the constraint $C(\pi^*) \leq \tau_c$.

The optimal policy not only specifies the nodes at which queries will be executed but also the size and exclude list of each query.

The cost of a querying policy π is defined as the total cost of all queries issued by following π. We have that $C(\pi) = \sum_{q \in \pi} c(q)$ where the cost of each query q is defined according to a cost model specified by the user. Computing the total cost of a policy π is easy. However, the gain $\mathcal{E}(\pi)$ of a policy π is unknown as we do not know in advance the entities corresponding to each node in $\mathcal{H}_D$, and hence, needs to be estimated.

4.2.4 Underlying Query Response Model

To reason about the occurrence of entities as response to specific queries, we need an underlying query response model. Our model is based on the notion of *popularity*.

Popularities. We assume that each underlying entity has a *fixed, unknown popularity value* with respect to crowd workers. Given a query $q(1, \emptyset)$, asking for one entity without using an exclude list, the probability that we will get entity e that satisfies the constraints specified by q is nothing but the popularity value of e divided by the popularity value of all entities e' that also satisfy the constraints in q. As an example, if there are only two entities e_1, e_2 that satisfy the constraints specified by a given query q_1, with popularity values 3 and 2, then the probability that we get e_1 on issuing a query $q_1(1, \emptyset)$ is 3/5. If an exclude list E is specified, then the probability that we will get an entity $e \notin E$ is the popularity value of e divided by the popularity values of all entities $e' \notin E$ also satisfying the constraints specified by q. **We do not assume that all workers follow the same popularity distribution**. Rather the overall popularity distribution can be seen as an average of the popularity distributions across all workers.

Thus, since workers are asked to provide a limited number of entities as response to a query, each entity extraction query can be viewed as taking a random sample from an unknown population of entities. In the rest of the chapter, we will refer to the distribution characterizing the popularities of entities in a population of entities as the *popularity distribution* of the population. This is equivalent to the underlying assumption in the species estimation literature [23] (Section 4.3).

Then, estimating the gain of a query $q(k, E)$ at a node $v \in \mathcal{H}_D$ is equivalent to estimating the number of new entities extracted by taking additional samples from the population of v given all the retrieved entities by past samples associated with node v [157].

Samples for a Node. When extracting entities, the retrieved entities for a node v (i.e., the *running sample*) may correspond to two different kinds of samples: (i) those that were extracted by considering the **entire population** corresponding to node v (ii) and those that we obtained by sampling **only a part of the population** corresponding to v. Samples for a node v can be obtained either by querying node v or by indirect information flowing to v by queries at other nodes. We refer to the latter case as *dependencies across queries*.

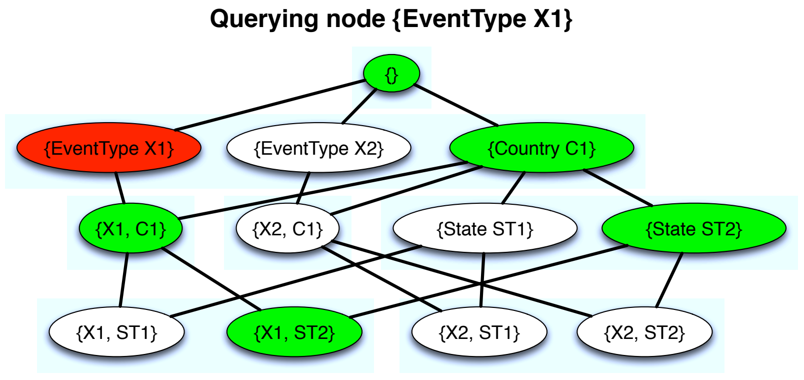


Figure 4.5: An example query that extract an entity sample from the red node. The nodes marked with green correspond to the nodes for which indirect entity samples are retrieved.

We use an example considering the poset in Figure 4.4, to illustrate these two cases (see Figure 4.5). Assume a query $q(k, \emptyset)$ issued against node $\{$EventType X1$\}$. Assume that the query result contains entities that correspond only to node $\{$X1,ST2$\}$. The green nodes in Figure 4.5 are nodes for which samples are obtained indirectly without querying them. All these nodes are ancestors of $\{$X1,ST2$\}$. We have:

- The samples corresponding to nodes $\{$X1, C1$\}$ and $\{$X1,ST2$\}$ were obtained by considering their *entire population*. The reason is that node $\{$EventType X1$\}$ is an ancestor of both and the entity population corresponding to it fully contains the populations of both $\{$X1,C1$\}$ and $\{$X1,ST1$\}$.

- The samples corresponding to nodes $\{\ \}$, $\{$Country C1$\}$ and $\{$State ST2$\}$ were obtained by considering only part of their population. The reason is that the population of node $\{$EventType X1$\}$ does not fully contain the populations of these nodes.

Samples belonging to both types need to be considered when estimating the gain of a query at a node in $v \in \mathcal{H}_D$. To address this issue we merge the extracted entities for each node in $\mathcal{H}_D$ into a single sample and treat the unified sample as being extracted from the entire underlying population of the node. As we discuss later in Section 4.4 we develop querying strategies that traverse the poset $\mathcal{H}_D$ in a top-down approach, hence, the number of samples belonging in the first category, i.e., samples retrieved considering the entire population of a node, dominates the number of samples retrieved by considering only part of a node's population. Moreover, it has been shown by Hortal et al. [77] that several of the techniques that can be used to estimate the gain of a query (see Section 4.3) are insensitive to differences in the way the samples are aggregated.

4.2.5 Framework Overview

The optimization problem in Section 4.2.3 can be viewed as a multi-round adaptive optimization problem where at each round we solve the subproblems below:

- **Estimating the Gain for a Query.** For each node in $v \in \mathcal{H}_D$, consider the retrieved entities associated with v and estimate the number of new unique entities that will be retrieved if a new query $q(k, E)$ is issued at v. This needs to be repeated for different query configurations.

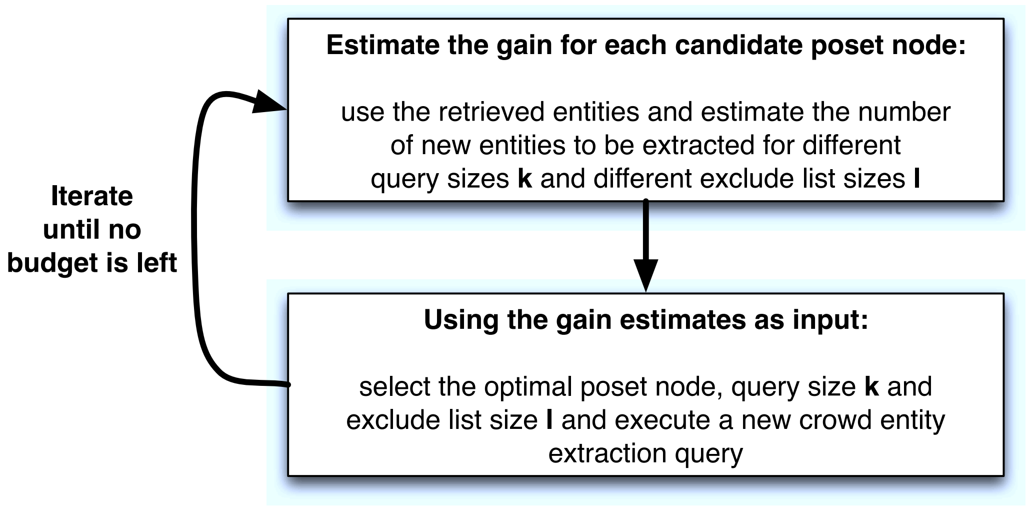

Estimate the gain for each candidate poset node:
use the retrieved entities and estimate the number of new entities to be extracted for different query sizes **k** and different exclude list sizes **l**
Iterate until no budget is left
Using the gain estimates as input:
select the optimal poset node, query size **k** and exclude list size **l** and execute a new crowd entity extraction query

Figure 4.6: Framework overview for budgeted entity extraction.

- **Detecting the Optimal Querying Policy.** Using the gain estimates from the previous problem as input, identify the next (query configuration, node) combination so that the total gain across all rounds is maximized with respect to the given budget constraint. When identifying the next query we do not explicitly optimize for the exclude list to be used. We rather optimize for the exclude list size l. Once the size is selected, the exclude list is constructed in a randomized fashion. We elaborate more on this design choice in Section 4.4.2.

Our proposed framework iteratively solves the aforementioned problems until the entire budget is used. Figure 4.6 shows a high-level diagram of the framework.

4.3 Estimating the Gain of Extraction Queries

Previous work [157] has drawn connections between this problem and the species estimation literature [23]. However, the proposed techniques therein do not work for queries that specify an exclude list. Moreover, they rely on the presence of a relatively large sample and tend to exhibit negative biases [79, 146], i.e., they underestimate the expected gain. Negative biases can severely impact entity extraction over large domains since nodes that contain entities that belong in the long tail of the popularity distribution may never be queried as they may be deemed to have zero population. In this section, we first review the existing methodology for estimating the gain of a query. Then we discuss how these estimators can be extended to consider an exclude list. Finally, we propose a new gain estimator for queries $q(k, E)$ that exhibits lower biases, and thus, improved performance, in the presence of little information than previous techniques (see Section 4.5).

4.3.1 Previous Estimators

Consider a specific node $v \in \mathcal{H}_D$. Prior work only considers samples retrieved from the entire population associated with v and does not consider an exclude list. Let Q be the set of all existing samples retrieved by issuing queries against v without an exclude list. These samples can be combined into a single sample corresponding to multi-set of size $n = \sum_{q \in Q} \mathsf{size}(q)$. Let f_i denote the number of entities that appear i times in this unified sample, and let f_0 denote the number of unseen entities from the population under consideration. Finally, let C be the population coverage of the unified sample. i.e., the fraction of the population covered by the sample $C = \frac{f_1+f_2+..}{f_0+f_1+...}$.

A new query $q(k, \emptyset)$ at node v can be viewed as increasing the size of the unified sample by k. Prior work used techniques from species estimation to estimate the expected number of new entities returned in $q(k, \emptyset)$. Shen et al. [146], derive an estimator for the number of new species $\hat{N}_{Shen}$ that would be found in an increased sample of size k. The approach assumes that unobserved entities have equal relative popularity. An estimate of the unique elements found in an increased sample of size k is given by:

$$\hat{N}_{Shen} = f_0 \left(1 - \left(1 - \frac{1-C}{f_0}\right)^k\right) \tag{4.1}$$

The second term of Shen's formula corresponds to the probability that at least one unseen entity will be present in a query asking for k more entities. Thus, multiplying this quantity with the number of unseen entities f_0 corresponds to the expected number of unseen entities present in the result of a new query $q(k, \emptyset)$.

The quantities f_0 and C are unknown and thus need to be estimated considering the entities in the running unified sample. The coverage can be estimated by considering the Good-Turing estimator $\hat{C} = 1 - \frac{f_1}{n}$ for the existing retrieved sample. On the other hand, multiple estimators have been proposed for estimating the number of unseen entities f_0. Trushkowsky et al. [157] proposed a variation of an estimator introduced by Chao et al. [23] to estimate f_0. Nevertheless, the authors argue that the original estimator proposed by Chao performs similarly with their approach when estimating the gain of an additional query $q(k, \emptyset)$. Next, we discuss how one can estimate the return of a query $q(k, E)$ in the presence of an exclude list E of size l and potential negative answers.

4.3.2 Exclude Lists and Negative Answers

A query $q(k, E)$ with $E \neq \emptyset$ issued at node $v \in H_D$ effectively limits the sampling to a restricted subset of the entity population corresponding to node v. To estimate the expected return of such a query, we need to update the estimates $\hat{f}_0$ and $\hat{C}$ before applying

Equation 4.1, by removing the entities in E from the running sample for node v and updating the frequency counts f_i and sample size n. This approach requires that the exclude list is known in advance. We discuss how we construct an exclude list in Section 4.4.2.

Next, we study the effect of *negative answers* on estimating the gain of future queries. It is possible to issue a query at a specific node $v \in \mathcal{H}_D$ and receive no entities, i.e., we receive a negative answer. This is an indication that the underlying entity population of v is empty. In such a scenario, we assign the expected gain of future queries at v and all its descendants to zero. Another type of negative answer corresponds to issuing a query at an ancestor node u of v and receiving no entities for v. In this case, we do not update our estimates for node u as entities from other descendants of u may be more popular than entities associated with u.

4.3.3 Direct Gain Estimation

The techniques reviewed in Section 4.3.1 result in negative bias when the number of observed entities from a population represents only a *small fraction* of the entire population [79, 146]. This holds for the large and sparse domains we consider in this chapter. To address this problem, Hwang and Shen [79] proposed a regression based technique to estimate f_0 and show that it results in smaller biases. However, estimating the total gain of a query requires coupling this new estimator with Equation 4.1, thus, it may still exhibit negative bias. To eliminate negative bias, we propose a direct estimator for the gain of generalized queries $q(k, E)$ without using Equation 4.1. We build upon the techniques in [79] and use a regression based technique that captures the structural properties of the expected gain function. The proofs for the results below are included in the Appendix of this dissertation. Only, the appropriate section references are provided below.

Let S denote the total number of entities in the population under consideration and p_i the abundance probability (i.e., popularity) of entity i. Given a sample of size n from the population, define $K(n)$ to be $K(n) = \frac{\sum_{i=1}^{S}(1-p_i)^n}{\sum_{i=1}^{S} p_i(1-p_i)^{n-1}}$. First, we focus on queries without an exclude list. Later we relax this and discuss queries with exclude lists. We have the following theorem on query gain:

Theorem 3. *Given a node $v \in \mathcal{H}_D$ and a corresponding entity sample of size n, let f_1 and f_2 denote the number of entities that appear exactly once (i.e., singletons) and exactly twice respectively. Let G denote the number of new items retrieved by a query $q(m, \emptyset)$. We have that:*

$$G = \frac{1}{(1 + \frac{K'}{n+m})}(K\frac{f_1}{n} - K'\frac{f_1(1 - \frac{1}{n}2\frac{f_2}{f_1})^m}{n+m}) \tag{4.2}$$

where $K = K(n)$ and $K' = K(n+m)$.

The proof of this theorem is deferred to Section A.3.

All quantities apart from K and K' in Equation 4.2 are known. The value of K can be estimated using the regression approach introduced by Hwang and Shen [79]. From the Cauchy-Schwarz inequality we have that:

$$K = \frac{\sum_{i=1}^{S}(1-p_i)^n}{\sum_{i=1}^{S} p_i(1-p_i)^{n-1}} \geq \frac{(n-1)f_1}{2f_2} \tag{4.3}$$

This can be generalized to:

$$K = \frac{nf_0}{f_1} \geq \frac{(n-1)f_1}{2f_2} \geq \frac{(n-2)f_2}{3f_3} \geq \dots \tag{4.4}$$

Let $g(i) = \frac{(n-i)f_i}{(i+1)f_{i+1}}$. From the above we have that the function $g(x)$ is a smooth monotone function for all $x \geq 0$. Moreover, let y_i denote a realization of $g(i)$ mixed with a random error. Hwang and Shen show how one can use an exponential regression model to estimate K. The proposed model corresponds to:

$$y_i = \beta_0 \exp(\beta_1 i^{\beta_2}) + \epsilon_i \tag{4.5}$$

where $i = 1, \dots, n-1$, $\beta_0 > 0$, $\beta_1 < 0$, $\beta_2 > 0$ and ϵ_i denotes random errors. It follows that $K = \beta_0$. To estimate the value of K' for an increased sample of size $n + m$, we first show that K increases monotonically as the size of the running sample increases.

Lemma 1. *The function* $K(n) = \frac{\sum_{i=1}^{S}(1-p_i)^n}{\sum_{i=1}^{S} p_i(1-p_i)^{n-1}}$ *increases monotonically, i.e.,* $K(n + m) \geq K(n), \forall n, m > 0$.

The proof of this lemma is presented in Section A.4.

Given the monotonicity of function K, we model K as a generalized logistic function of the form $K(x) = \frac{A}{1+exp(-G(x-D))}$. As we observe samples of different sizes for different queries we estimate K as described above and therefore we observe different realizations of $f(\cdot)$. Thus, we can learn the parameters of f and use it to estimate K'. In the presence of an exclude list of size l we follow the approach described in Section 4.3.2 to update the quantities f_i and n used in the analysis above.

4.4 Discovering Querying Policies

Next, we focus on the second component of our proposed algorithmic framework and introduce a multi-round adaptive optimization algorithm for identifying querying strategies that maximize the total gain across all rounds under the given budget constraints. We build upon ideas from the multi-armed bandit literature [6, 50]. At each round, the proposed algorithm uses as input the estimated gain or return for different generalized queries $q(k, E)$ at the different nodes in $\mathcal{H}_D$. Before presenting our framework we list the main two challenges associated with this adaptive optimization problem.

- The first challenge is that the number of nodes in $\mathcal{H}_D$ is exponential in the number of attributes $\mathcal{A}_D$ describing the domain of interest. Querying every possible node to estimate its expected return for different queries $q(k, E)$ is prohibitively expensive. That said, typical budgets do not allow algorithms to query all nodes in the hierarchy, so this intractability may not hurt us all that much. For example, we keep estimates for each of the nodes for which at least one entity has been retrieved.
- The second challenge is balancing the trade-off between *exploitation* and *exploration* [6]. The first refers to querying nodes for which sufficient entities have been retrieved and hence we have an accurate estimate for their expected return; the latter refers to exploring new nodes in $\mathcal{H}_D$ to avoid locally optimal policies.

4.4.1 Balancing Exploration and Exploitation

While issuing queries $q(k, E)$ at different nodes of $\mathcal{H}_D$ we obtain a collection of entities that can be assigned to different nodes in $\mathcal{H}_D$. For each node we can estimate the return of a query $q(k, E)$ using the estimators presented in Section 4.3. However, this estimate is based on a rather small sample of the underlying population. Thus, exploiting this information at every round may lead to suboptimal decisions. This is why we need to balance the trade-off between exploiting nodes for which the estimated return is high and nodes that have not been queried many times. This corresponds to upper-bounding the expected return of each potential action with a confidence interval that depends on both the variance of the expected return and the number of times an action has been evaluated.

Let $r(\alpha)$ denote the expected return of action α that is an estimate of the true return $r^*(\alpha)$. Moreover, let $\sigma(\alpha)$ be an error component on the return of action α chosen such that $r(\alpha) - \sigma(\alpha) \leq r^*(\alpha) \leq r(\alpha) + \sigma(\alpha)$ with high probability. The parameter $\sigma(\alpha)$ should take into account both the empirical variance of the expected return as well as our uncertainty if an action or similar actions (e.g., queries with different k, E but at the same node) has been chosen few times. Let $n_{\alpha,t}$ be the number of times we have chosen action α by round t, and let $v_{\alpha,t}$ denote the maximum value between some constant c (e.g., $c = 0.01$) and the empirical variance for action α at round t. The latter can be computed using bootstrapping over the retrieved sample and applying the estimators presented in Section 4.3.3 over these bootstrapped samples. Several techniques have been proposed in the multi-armed bandits literature to compute the parameter $\sigma(\alpha)$ [156]. Teytaud et al. [156] showed that techniques considering both the variance and the number of times an action has been chosen tend to outperform other proposed methods. Based on this observation, we choose to use the following formula for sigma:

$$\sigma(\alpha) = \sqrt{(v_{\alpha,t} \cdot \log(t))/(n_{\alpha,t})} \tag{4.6}$$

4.4.2 A Multi-Round Querying Policy Algorithm

We now introduce a multi-round algorithm for solving the budgeted entity enumeration problem. At a high-level, the algorithm proceeds as follows: Instead of considering all potential queries $q(k, E)$ that can be issued at the different nodes of $\mathcal{H}_D$, we consider all potential query configurations (k, l). In particular, we do not optimize directly for the exclude list to be used in a further query but rather for the size l of it. Once we decide on l the exclude list E can be constructed following a randomized approach, where l of the retrieved entities are included in the list uniformly at random. The generated list can be used to update the frequency counts f_i and sample size n and estimate the gain of the query. Bootstrapping can also be used to obtain improved estimates.

We follow a randomized approach as a deterministic construction of E that picks the *l-most* popular items in the running sample is very sensitive to the *observed popularity distribution*. When the number of observed entities corresponds to a small portion of the entire population - as in the scenarios we consider in this chapter - the individual entity popularity estimates tend to be very noisy. We empirically observed that a deterministic construction of a limited size exclude list, especially during early queries, leads to poor popularity estimates. Thus, we choose to follow a randomized approach.

Let $\mathcal{S}$ denote the set of all potential query configurations (k, l) that can be issued at the different nodes of $\mathcal{H}_D$ during a round r. Moreover, let $r(\alpha) + \sigma(\alpha)$ and $c(\alpha)$ be the upper-bounded return (i.e., gain) and cost for an action $\alpha \in \mathcal{S}$. At each round the algorithm identifies an action in $\mathcal{S}$ that maximizes the quantity $\frac{r(\alpha)+\sigma(\alpha)}{c(\alpha)}$ under the constraint that the cost of action α is less or equal to the remaining budget. Since we are operating under a specified budget one can view the problem in hand as a variation of the typical knapsack problem. If no such action exists then the algorithm terminates. Otherwise the algorithm issues the query corresponding to action α, updates the set of unique entities obtained from the queries, the remaining budget and updates the set of potential queries that can be executed in the next round. An overview of this algorithm is shown in Algorithm 1.

As discussed before, the size of $\mathcal{H}_D$ is exponential to the values of attributes describing it, and thus, considering all the possible queries for the different nodes of $\mathcal{H}_D$ can be prohibitively expensive. Next, we discuss how one can initialize and update the set of potential actions as the algorithm progresses based the structure of the poset $\mathcal{H}_D$ and the retrieved entities from previous rounds.

4.4.3 Updating the Set of Actions

Due to the exponential size of the poset $\mathcal{H}_D$, we need to limit the set of possible actions Algorithm 1 considers by exploiting the structure the given domain $\mathcal{H}_D$. We propose an algorithm that updates the set of actions by traversing the input poset in a top-down manner and adds new actions that correspond to queries for nodes that are *direct descendants*

Algorithm 1 Overall Algorithm

```
1: Input: $\mathcal{H}_D$: the hierarchy describing the entity domain; $r, \sigma$: value oracle access to
   gain upper bound; $c$: value oracle access to the query costs; $\beta_c$: query budget;
2: Output: $\mathcal{E}$: a set of extracted distinct entities;
3: $\mathcal{E} \leftarrow \{\}$
4: $RB \leftarrow \beta_c$ /* Initialize remaining budget */
5: $\mathcal{S} \leftarrow$ UpdateActionSet($\mathcal{H}_D$,NULL,$\emptyset$)
6: while $RB > 0$ and $S \neq \{\}$ do
7:    $\alpha \leftarrow \arg\max_{\alpha \in \mathcal{S}} \frac{r(\alpha)+\sigma(\alpha)}{c(\alpha)}$ such that $RB - c(\alpha) > 0$
8:    if $\alpha$ is NULL then
9:       break;
10:   $RB \leftarrow RB - c(\alpha)$ /* Update budget */
11:   Issue query corresponding to $\alpha$
12:   $E \leftarrow$ entities from query
13:   $\mathcal{E} \leftarrow \mathcal{E} \cup E$ /* Update unique entities */
14:   $\mathcal{S} \leftarrow$ UpdateActionSet($\mathcal{H}_D, a, S$)
15: return $\mathcal{E}$
```

of already queried nodes. Due to the hierarchical structure of the poset nodes at higher levels of the poset correspond to larger populations of entities. Therefore, issuing queries at these nodes can potentially result in a larger number of extracted entities. Traversing the poset in a top-down manner allows us to detect sparsely populated areas of the poset.

Our approach for updating the set of available actions (Alg. 2) proceeds as follows: If the set of available actions is empty start by considering all possible queries that can be issued at the root of $\mathcal{H}_D$ (Ln. 4-5). The set of possible queries corresponds to queries $q(k, E)$ for all combinations of the values of parameters k and l. Recall that E is constructed in a randomized fashion once l is determined. Recall that these are pre-specified by the designer of the querying interface. If the set of available actions is not empty, we consider the node associated with the action selected in the last round and populate the set of available actions with all the queries corresponding to its direct descendants (Ln. 7-9), i.e., by traversing the input poset in a top-down fashion. As mentioned

Algorithm 2 UpdateActionSet

```
1: Input: $\mathcal{H}_D$: the hierarchy describing the entity domain; $u$: a node in $\mathcal{H}_D$ associated with the last selected action; $\mathcal{S}_{old}$: the running set of actions; $V_k$: set of values for query parameter $k$; $V_l$: set of values for query parameter $l$;
2: Output: $\mathcal{S}_{new}$: the updated set of actions;
3: /* Extend Set of Actions*/
4: if $\mathcal{S}_{old}$ is empty then
5:     return $\{$Root of $\mathcal{H}_D\}$
6: $\mathcal{S}_{new} \leftarrow \mathcal{S}_{old}$
7: for all $d \in$ Set of Direct Descendant Nodes of $u$ do
8:     $A_d \leftarrow$ Set of queries at $u$ for all configurations in $V_k \times V_l$
9:     $\mathcal{S}_{new} \leftarrow \mathcal{S}_{new} \cup A_d$
10: /* Remove Bad Actions*/
11: /* Find maximum lower bound on gain over all actions in $\mathcal{S}_{new}$*/
12: $thres \leftarrow \max_{\alpha' \in \mathcal{S}_{new}}(r(\alpha') - \sigma(\alpha'))$
13: $\mathcal{B} \leftarrow$ All actions $a$ in $\mathcal{S}_{new}$ with $r(\alpha) + \sigma(\alpha) < thres$
14: $\mathcal{S}_{new} \leftarrow \mathcal{S}_{new} \setminus \mathcal{B}$
15: return $\mathcal{S}_{new}$
```

above the number of nodes in $\mathcal{H}_D$ can be prohibitively large, therefore we also *remove* any *bad actions* from the running set of actions (Ln. 10-14). An action α is bad when $r(\alpha) + \sigma(\alpha) < \max_{\alpha' \in \mathcal{S}}(r(\alpha') - \sigma(\alpha'))$. Intuitively, this states that we do not need to consider an action as long as there exists another action such that the upper-bounded return of the former is lower than the lower bounded return of the latter. This is a standard technique adopted in multi-armed bandits to limit the number of actions considered by the algorithm [50].

4.5 Experimental Evaluation

We present an empirical evaluation of our proposed algorithmic framework using both real and synthetic datasets. First, we discuss the experimental methodology, then we describe the data and results that demonstrate the effectiveness of our framework on crowdsourced entity extraction. The evaluation is performed on an Intel(R) Core(TM) i7 3.7 GHz 32GB machine; all algorithms are implemented in Python 2.7.

4.5.1 Experimental Setup

Gain Estimators. We evaluate the following gain estimators:

- Chao92Shen: This estimator combines the methodology proposed by Chao [23] for estimating the number of unseen species with Shen's formula, i.e., Equation 4.1.
- HwangShen: This estimator combines the regression-based approach by Hwang and Shen [79] for estimating the number of unseen species with Shen's formula.
- NewRegr: This estimator corresponds to our new technique proposed in Section 4.3.3.

All estimators were coupled with bootstrapping to estimate their variance to retrieve an upper bound on the return of a query as shown in Section 4.4.1.

Entity Extraction Algorithms. We evaluate the following algorithms for crowdsourced entity extraction:

- Rand: This algorithm executes random queries until all the available budget is used. It selects a random node from the input poset $\mathcal{H}_D$ and a random query configuration (k, l) from a list of pre-specified k, l value combinations. We expect Rand to be effective for extracting entities in small and dense data domains that do not have many sparsely populated nodes.
- RandL: Same as Rand but only executes queries *only at the lowest level nodes* (i.e., leaf nodes) of the input poset $\mathcal{H}_D$ until all the available budget is used. We expect RandL to be effective for *shallow* data domains when the majority of nodes corre-

sponds to leaf nodes. Like Rand, the performance of RandL is expected to be reasonable for small and dense data domains without sparsely populated nodes.

- BFS: This algorithm performs a breadth-first traversal of the input poset $\mathcal{H}_D$, executing one query at each node. The query configuration is randomly selected from a list of pre-specified k, l value combinations. This algorithm promotes exploration of the action space when extracting entities. It also takes into account the structure of the input domain but is agnostic to sparsely populated nodes of the input $\mathcal{H}_D$.
- RootChao: This algorithm corresponds to the entity extraction scheme of Trushkowsky et al. [157] that utilizes the Chao92Shen estimator to measure the gain of an additional query. The proposed scheme is agnostic to the structure of the input entity domain, and thus, equivalent to issuing queries only at the root node of the poset $\mathcal{H}_D$. Since the authors only propose a pay-as-you-go scheme, we coupled this algorithm with Alg. 1 to optimize for the input budget constraint. The algorithm considers different query configurations (k, l) but restricts its queries to the root node.
- GSChao, GSHWang, GSNewR: Our proposed querying policy algorithm (Section 4.4.2) using Chao92Shen, HwangShen and NewRegr respectively.
- GSExact: This algorithm is used as a near-optimal, omniscient baseline that allows us to see how far off our algorithms are from an algorithm with perfect information. In particular, we combine the algorithm proposed in Section 4.4.2 with an exact computation of the return or gains from queries. More precisely, the algorithm proceeds as follows: At each round we speculatively execute each of the available actions (i.e., all query configurations across all nodes) and select the one that results in the largest

number of return to cost ratio. Since the return of each query is known, the algorithm is not coupled with any of the aforementioned estimators.

Rand, RandL and BFS promote exploration when extracting entities. The other algorithms balance exploration with exploitation. For the results reported below, we run each algorithm ten times and report the average gain achieved under the given budget.

Querying Interface. For all datasets we consider generalized queries of the type "Give me k more entities that satisfy certain conditions and are not present in an exclude list of size l". The conditions correspond to matching the attribute values associated with a node from the input poset. The configurations considered for (k, l) are $\{(5,0),(10,0),(20,0),(5,2),(10,5),(20,5),(20,10)\}$. Larger values of k or l were deemed unreasonable for crowdsourced queries. The gain of a query is computed as the number of new entities extracted. The cost of each query is computed using an additive model comprised by three partial cost terms that depend on the characteristics of the query.

The three partial cost terms are: (i) CostK that depends on the number of responses k requested from a user, (ii) CostL that depends on the size of the exclude list l used in the query, and (iii) CostSpec that depends on the *specificity* of the query q_s, e.g., we assume that queries that require users to provide more specialized entities (e.g., "Give me one concert for New York on the 17th of Nov") cost more than more generic queries (e.g., "Give me one concert in New York"). More formally, we define the specificity of a query to be equal to the number of attributes assigned non-wildcard values for the node $u \in \mathcal{H}_D$ the query corresponds to.

The overall cost for a query with configuration (k, l) with specificity s is computed as: $Cost(q) = \alpha \cdot \frac{k}{\text{max. query size}} + \beta \cdot \frac{l}{\text{max. ex. list size}} + \gamma \cdot \frac{s}{\text{max. specificity}}$. The cost of a query should be significantly increased when an exclude list is used, thus we require that β is set to a larger value than α and γ. For the results reported below, we set $\alpha = \gamma = 1$ and $\beta = 5$. Similar results were observed for other settings.

Data. First, we evaluate the proposed framework on extracting entities from a large sparse domain. We consider the event dataset collected from Eventbrite. As described in Section 4.1, the poset corresponding to the Eventbrite domain contains 8,508,160 nodes with 57,805 distinct events overall. However, only 175,068 nodes are populated leading to a rather sparsely populated domain. Due to lack of popularity proxies for the extracted events, we assigned a random *popularity value* in $(0, 10]$ to each event. These weights are used during sampling to form the actual popularity distribution characterizing the population of each node in the poset.

We further evaluate the performance of the extraction algorithms for a more dense domain, that we constructed ourselves. We used Amazon's Mechanical Turk [2] to collect a real-world dataset, targeted at extracting "people in the news". While different from the event extraction domain studied before this new domain is still structured. We asked workers to extract the names of people belonging to four different types from five different news portals. The people types we considered are "Politicians", "Athletes", "Actors/Singers" and "Industry People". The news portals we considered are "New York Times", "Huffington Post", "Washington Post", "USA Today" and "The Wall Street Journal". This data domain, referred to as the People's domain, is essentially characterized

Table 4.1: The population characteristics for the People's domain.

Person Type	People
Industry People	743
Athletes	743
Politicians	748
Actors/Singers	744

News Portal	People
WSJ	594
WashPost	597
NY Times	595
HuffPost	599
USA Today	593

by the type of the individual and the news portal. Workers were paid \$0.20 per HIT. We issued 20 HITS for each leaf node of the domain's poset, resulting in 600 HITS in total. After manually curating name misspelling's, we extracted 1,245 unique people in total. Table 4.1 shows the number of distinct entities for the different values of the people-type and news portal attributes. Finally, the popularity value of each extracted entity was assigned to be equal to the number of times it appeared in the extraction result. The values are normalized during sampling time to form a proper popularity distribution. Collecting a large amount of data in advance from Mechanical Turk and then simulating the responses of human workers by revealing portions of this dataset allows us to compare different algorithms on an equal footing; this approach is often adopted in the evaluation of crowdsourcing algorithms [120, 106, 157].

4.5.2 Experimental Results

Next, we evaluate different aspects of the aforementioned extraction techniques.

How does our querying policy algorithm compare against baselines? We evaluate the performance of the different extraction algorithms in terms of number of entities extracted for different budgets. The results for Eventbrite and the People's domain are shown in Figure 4.7(a) and Figure 4.7(b) respectively. As shown, our proposed algorithms, i.e.,

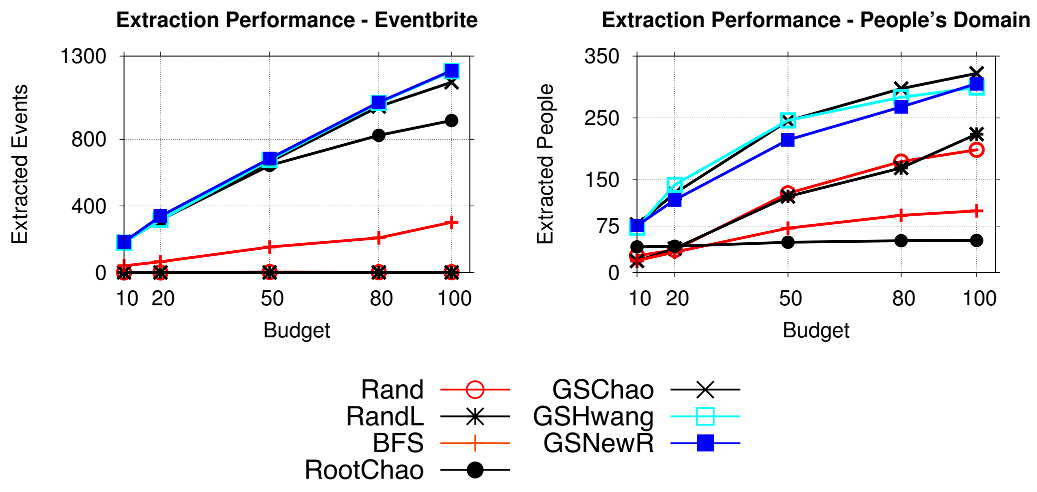


Figure 4.7: A comparison of the proposed entity extraction techniques against several baselines for (a) Eventbrite and (b) the People's domain.

GSChao, GSHwang, GSNewR *outperform all baselines for at least 30% across both datasets*. This behavior is expected as our techniques not only exploit the structure of the domain to diversify entity extraction by targeting entities that belong to the tail of the popularity distribution but also optimize the queries for the given budget.

When comparing again the naive baselines Rand, RandL, and BFS, we see that GSChao, GSHwang and GSNewR extract at least 2X more entities for the sparse Eventbrite domain and around 100% more entities for small budgets and 54% for larger ones when considering the dense People's domain. For example for Eventibrite and a budget of $50 all schemes coupled with our querying policy discovery algorithm (Section 4.4) extracted more than 600 events while Rand and RandL extracted 1.1 and 0.2 events and BFS extracted 207.7 events, an improvement of over 180%.

Comparing against RootChao, we see that GSChao, GSHwang and GSNewR, are able to retrieve up to 30% more entities for Eventbrite and 5X for the People's domain.

This performance difference is due to the fact that the gain achieved by RootChao saturates at a faster rate compared to GSChao, GSHwang and GSNewR as the cost increases. This is because, RootChao focuses on issuing queries at the root of the input poset, and hence, it is not able to extract entities belonging to the long tail of the popularity distribution. Moreover, for the People's domain we see that RootChao performs poorly even compared to the naive baselines Rand, RandL and BFS. This is due to the popularity skew.

How do our techniques compare against a near-optimal policy discovery algorithm? Next, we evaluate GSChao, GSHwang and GSNewR against the near-optimal querying policy discovery algorithm GSExact. The results for Eventbrite and the People's domain are shown in Figure 4.8(a) and Figure 4.8(b) respectively. Regarding the dense domain Eventbrite, we observe that for smaller budgets our proposed techniques perform comparably to GSExact that has "perfect information" about the gain of each query, typically demonstrating a performance gap of less than 10%. For larger budgets this gap increases to 25%. Note that our estimators have access to few samples and sparse information; the fact that we are able to get this close to GSExact is notable. Finally, for the People's domain, our techniques present an increased performance gap compared to GSExact. Nevertheless the performance drop is at most 50%.

How do the different techniques compare with respect to the total number of queries issued during extraction? We compare the performance of RootChao (i.e., the extraction scheme proposed by Trushkowsky et al. [157]) against our algorithms GSChao, GSHwang and GSNewR with respect to the *total number of queries issued during extraction*. Notice that this new evaluation metric characterizes directly the overall latency of the crowd-extraction process. Figure 4.9 shows the corresponding results for a run

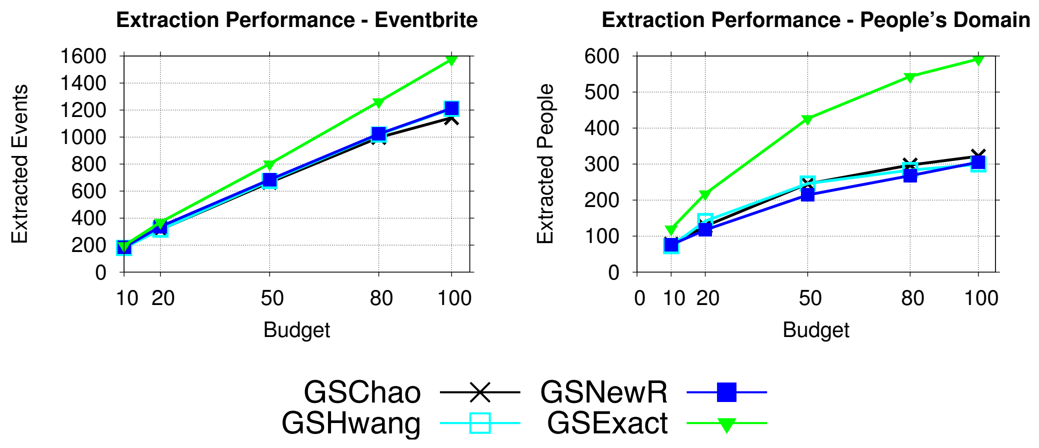


Figure 4.8: A comparison of the proposed entity extraction techniques against a near-optimal algorithm for (a) Eventbrite and (b) the People's domain.

for Eventbrite and a budget of $80. As shown *RootChao requires almost up to 3x more queries* to extract the same number of entities as our proposed techniques, thus, exhibiting significantly larger latency compared to GSChao, GSHwang and GSNewR.

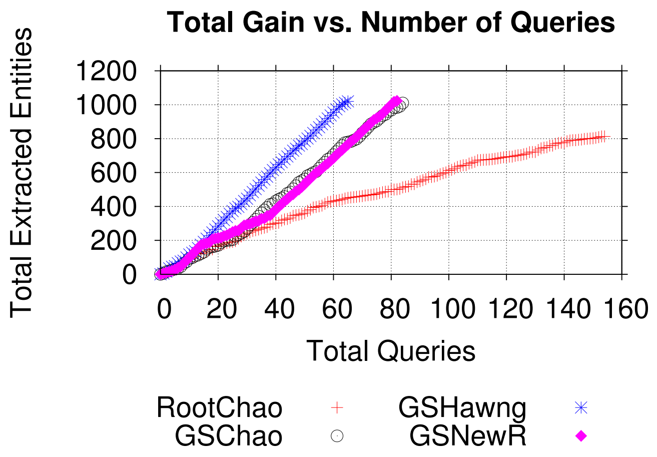


Figure 4.9: The number of events extracted by different algorithms for the Eventbrite data domain and the corresponding total number of queries.

How our different algorithms traverse the poset and use different query configurations? We next explore how our different algorithms traverse the poset, and how they use different query configurations. The results reported are averaged over ten runs and correspond to the People's domain. We begin by considering how many queries these algorithms issue at various levels of the poset. In Figure 4.10, we plot the different number of queries issued at various levels by our algorithms when the budget is set to 10 and 100 respectively. Given a small budget, we observe that all algorithms prefer issuing queries at higher levels of the poset. Notice that inner nodes of the poset are preferred and only a small number of queries is issued at the root (i.e., level one) of the poset. This behavior is justified if we consider that due to their popularity, certain entities are repeatedly extracted, thus leading to a lower gain. As the budget increases, we see that all algorithms tend to consider more specialized queries at deeper levels of the poset. It is interesting to observe that all of our algorithms issue the majority of their queries at the level two nodes, while GSExact, which has perfect information, focuses mostly on the leaf nodes. Thus, in this case, our techniques could benefit from being more aggressive at traversing the poset and reaching deeper levels; overall, our techniques may end up being more conservative in order to cater to a larger space of posets and popularity distributions. In Figure 4.11, we plot the different query configurations chosen by our algorithms when the budget is set to 10 and 100 respectively. We observe that GSExact always prefers queries with $k = 20$ and $l = 0$ for both small and large budgets. On the other hand, our algorithms issue more queries of smaller size when operating under a limited budget and prefer queries of larger size for larger budgets. Out of all algorithms we see that GSNewR was the only one issuing queries with exclude lists of different sizes, thus exploiting the rich diversity of query interfaces. However, the number of such queries is limited.

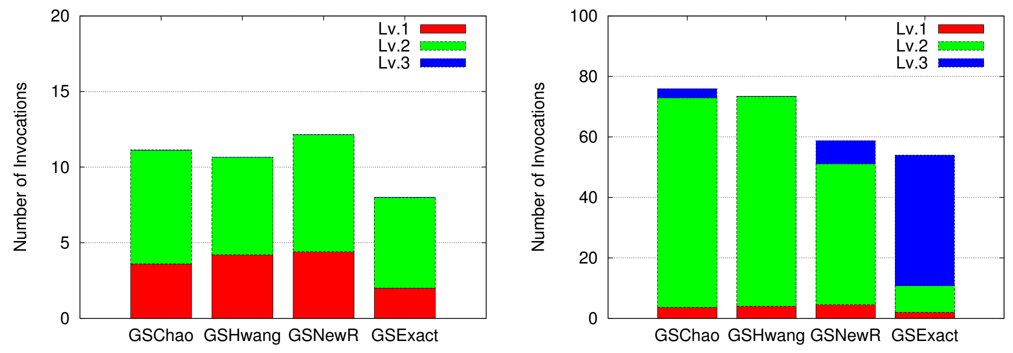


Figure 4.10: The number of queries issued at different levels used when budget is set at 10 or 100.

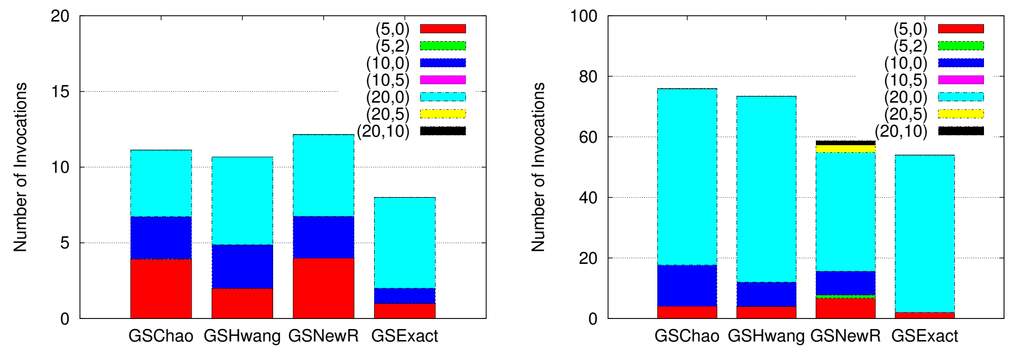


Figure 4.11: The query configurations used when budget is set at 10 or 100.

How effective are the different estimators at predicting the gain of additional queries?

Finally, we point out that GSNewR was able to outperform GSChao and GSHwang for Eventbrite but the opposite behavior was observed for the People's domain. To further understand the relative performance of GSChao, GSHwang and GSNewR, we evaluate the performance of the gain estimators Chao92Shen, HwangShen and NewRegr at predicting the number of new retrieved events for different query configurations. For Eventbrite, we choose ten random nodes containing more than 5,000 events and for each of them and each of the available query parameter configurations (k, l), we execute ten queries of the form "Give me k items from node $u \in \mathcal{H}_D$ that are not included in an exclude list of size l". As mentioned in Section 4.3.2 the exclude list for each query is constructed following

Table 4.2: Average absolute relative error for estimating the gain of different queries for Eventbrite.

Q. Size k	**EL. Size l**	**Chao92Shen**	**HwangShen**	**NewRegr**
5	0	0.470	0.500	0.390
5	2	0.554	0.612	0.467
10	0	0.569	0.592	0.544
10	5	0.580	0.696	0.29
20	0	0.642	0.756	0.471
20	5	0.510	0.60	0.436
20	10	0.653	0.756	0.631

a randomized approach. For the People's domain, we issue ten queries over all nodes of the input poset for all available query configurations. We measure the performance of each estimator by considering the absolute relative error between the predicted return and the actual return of the query.

Table 4.2 reports the relative error for each of the three estimators averaged over all points under consideration for Eventbrite. As shown, all three estimators perform equivalently with the new regression-based technique slightly outperforming Chao92Shen and HwangShen for certain types of queries. For example, for $k = 10, l = 5$, Chao92Shen has a relative error of 0.58, HwangShen had a relative error of 0.7, and NewRegr had a relative error of 0.29. We attribute the improved extraction performance of GSNewR to these improved estimates. The relatively large values for relative errors are justified as the retrieved samples correspond to a very small portion of the underlying population for each of the points. This is a well-known behavior for non-parametric estimators and studied extensively in the species estimation literature [79].

Table 4.3 shows the results for the People's domain. We observe that for smaller query sizes the regression technique proposed in this chapter offers better gain estimates. However, as the query size increases, and hence, a larger portion of the underlying population is observed Chao92Shen outperforms both regression-based techniques. Thus, we are able to explain the performance difference between GSChao and the other two algorithms. Eventually, we have that for sparse domains regression-based techniques result in better performance. However, for dense domains the Chao92Shen estimator results in better performance as a larger portion of the underlying population can be sampled.

Table 4.3: Average absolute percentage error for estimating the gain of different queries for the People's data domain.

Q. Size k	**EL. Size l**	**Chao92Shen**	**HwangShen**	**NewRegr**
5	0	0.295	0.299	0.228
5	2	0.163	0.156	0.144
10	0	0.306	0.305	0.277
10	5	0.341	0.349	0.293
20	0	0.359	0.371	0.467
20	5	0.2615	0.264	0.249
20	10	0.1721	0.162	0.127

4.6 Related Work

The prior work related to the techniques proposed in this chapter can be placed in a few categories; we describe each of them in turn:

Crowd Algorithms. There has been a significant amount of work on designing algorithms where the unit operations (e.g., comparisons, predicate evaluations, and so on) are performed by human workers, including common database primitives such as filter [121], join [105] and max [69], machine learning primitives such as entity resolution [10, 163] and clustering [64], as well as data mining primitives [4, 148].

Previous work on the task of crowdsourced extraction or enumeration, i.e., populating a database with entities using the crowd [122, 157] is the most related to ours. In both cases, the focus is on a single entity extraction query; extracting entities from large and diverse data domains is not considered. Moreover, the proposed techniques do not dynamically adapt crowd queries to optimize for a specified monetary budget.

Knowledge Acquisition Systems. Recent work has also considered the problem of using crowdsourcing within knowledge acquisition systems [83, 93, 166]. This line of work suggests using the crowd for curating knowledge bases (e.g., assessing the validity of the extracted facts) and for gathering additional information to be added to the knowledge base (e.g., missing attributes of an entity or relationships between entities), instead of augmenting the set of entities themselves. As a result, these papers are solving an orthogonal problem. The techniques described in this chapter for estimating the amount of information from a query and devising querying strategies to maximize the amount of extracted information will surely be beneficial for knowledge extraction systems as well.

Deep Web Crawling. A different line of work has focused on data extraction from the deep web [84, 147]. In such scenarios, data is obtained by querying a form-based interface over a hidden database and extracting results from the resulting dynamically-generated answer (often a list of entities). Typically, such interfaces provide partial list of matching entities to issued queries; the list is usually limited to the top-k tuples based on an unknown ranking function. Sheng et al. [147] provide near-optimal algorithms that exploit the exposed structure of the underlying domain to extract all the tuples present in the hidden database under consideration. Our work is similar to this work in that our goal is to also extract entities via a collection of interfaces (in our case the interfaces correspond to queries asked to the crowd).

The main difference between this line of work and ours is that answers from a hidden database are deterministic, i.e., a query in their setting will always retrieve the same top-k tuples. This assumption does not hold in the crowdsourcing scenario considered in this chapter and thus the proposed techniques are not applicable. In their setting, it suffices to ask each query precisely once. In our setting, since crowdsourced entity extraction queries can be viewed as random samples from an unknown distribution, one needs to make use of the query result estimation techniques introduced in Section 4.3.

4.7 Summary

In this chapter, we studied the problem of crowdsourced entity extraction over large and diverse data domains. We introduced a novel crowdsourced entity extraction framework that combines statistical techniques with an adaptive optimization algorithm to maximize the total number of unique entities extracted. We proposed a new regression-based technique for estimating the gain of further querying when the number of retrieved entities is small with respect to the total size of the underlying population. We also introduced a new algorithm that exploits the often known structure of the underlying data domain to devise adaptive querying strategies. Our experimental results show that our techniques extract up to 4X more entities compared to a collection of baselines, and for large sparse entity domains are at most 25% away from an omniscient adaptive querying strategy with perfect information.

Chapter 5: Selecting Valuable Sources for Integration

So far in the dissertation we focused on the techniques used by the source analysis engine of the quality-aware data source management architecture (Section 1.2) to index the content and discover the quality of available sources. In this chapter, we focus on the second major part of the QDSM architecture that uses the quality profiles of the sources to discover the most valuable sources for integration. We introduce a collection of algorithms that, given a set of available sources, discover the subset of sources that, if integrated together, will maximize the utility of integrated data at the minimum cost. We will refer to the utility of integrated data as the *gain of integration*. The gain of integration can be quantified using the quality metrics for integrated data described in Section 3.5. The cost of integration can incorporate the monetary cost of acquiring source data and the computation cost of performing integration. Recently, Dong et al. [47] formalized this problem of maximizing the gain of integration while minimizing its cost by introducing the paradigm of *source selection*. While the definition of the problem is generic, the authors considered only static sources with no content updates and univariate gain functions.

In the following sections, we first review the basic definition of source selection and provide the necessary background for the reader (Section 5.1). Then, we introduce the problem of *time-aware source selection* extending the formulation of Dong et al. Further, we formalize variations of the problem that, in addition to selecting a subset of sources, allow us to decide the optimal frequency to acquire data from each source, as well as, the optimal subset of data to acquire from each source (Section 5.2). All this variations of the source selection problem are shown to be NP-complete. However, in Section 5.3, we show that many of the time-aware source selection instances (e.g., where the gain is a function of time-dependent coverage, and the cost is an additive function) correspond to well-studied submodular optimization problems for which efficient local-search algorithms with constant factor approximations are known. We also discuss how one can solve time-aware source selection for arbitrary objective functions and multiobjective objective functions. In Section 5.4, we present an experimental evaluation of the performance and scalability of the source selection algorithms introduced in Section 5.3 for the business listings and GDELT datasets introduced in Section 1.1. Finally in Section 5.5 we discuss related work and summarize the main results of this chapter in Section 5.6.

5.1 Preliminaries

We consider a set of sources $\bar{\mathrm{S}}$ that provide data from a data domain $\mathcal{D}$. As before, we assume that $\mathcal{D}$ follows a closed-world assumption, i.e., $\mathcal{D}$ contains only objects stored in the sources in $\bar{\mathrm{S}}$. The integrated data for any subset of sources from $\bar{\mathrm{S}}$ is characterized by its *cost* and *gain*. The cost of integration is a function of the monetary cost to acquire data, and the total resources needed for integration. The gain quantifies the benefit of integration and is a function of the integration quality using the same unit as for cost.

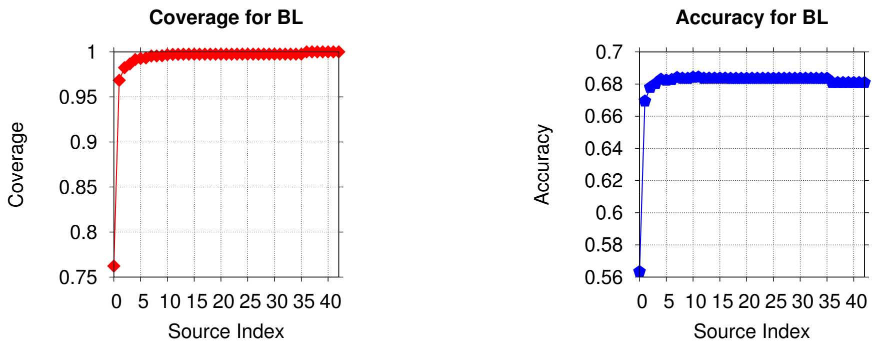


Figure 5.1: (a) Coverage and (b) accuracy of integrated data for the BL scenario introduced in Section 1.1; Sources processed in decreasing order of coverage.

In many real-world applications it is not always worthwhile to integrate all available sources in a domain. For example, in the presence of redundancy among data sources, integrating new sources may not increase the coverage significantly, if at all, while it increases the total cost. This can be seen in Figure 5.1(a), where we plot the coverage of integrated data for the sources in the business listings domain described in Section 1.1. Even worse, low-quality sources can even hurt the accuracy of integrated data while still increasing the integration cost. This can be observed in Figure 5.1(b) for the same business listings domain.

To address this problem, Dong et al. [47] introduced the paradigm of *source selection*, which is performed before real integration to balance the cost and the gain of integration. Source selection is defined as follows:

Definition 2. (SOURCE SELECTION) Let $\bar{\mathrm{S}}$ be a set of sources, F be an integration model, $G_F(\cdot)$ be a gain function and $C_F(\cdot)$ a cost function using model F, and β_c be a budget on cost. The Source Selection problem finds a subset $S_I \subseteq \bar{\mathrm{S}}$ that maximizes $G_F(S_I) - C_F(S_I)$ under constraint $C_F(S_I) \leq \beta_c$.

The gain of integration can be quantified using the quality of integrated data. Let $Q(F(S_I))$ denote the overall quality of the integration result $F(S_I)$ for a set of sources S_I. The quality of integrated data can be measured using a combination of different quality metrics, such as the coverage, freshness or accuracy of the integration result. Given $Q(F(S_I))$ the gain of integration can be defined as $G_F(S_I) = f(Q(F(S_I)))$ where $f(\cdot)$ is a function converting the quality of integration to monetary units. The cost of integration can vary from simple additive functions over the individual source prices [47] (i.e., the amount of money each source requires to acquire its data) to more elaborate models that compute the cost of integration by estimating the effort needed to integrated the data of the selected sources [142]. To reason about the trade-off between the gain and cost of integration, it is necessary that the cost and gain functions to have the same range expressed in monetary units. Given the aforementioned gain and cost function families, Dong et al. show that the problem of source selection described above is NP-complete.

5.2 Selecting Dynamic Data Sources

We now consider that sources in $\bar{\mathrm{S}}$ are dynamic, i.e., they change their content over time. Recall, that source selection is a pre-processing step to data integration, therefore, when one selects the optimal set of sources to be integrated, she makes a decision for *future time points*. Consider a fixed set of future time points, denoted by T_f. The goal is to maximize the profit of integration, i.e., the difference between the gain and cost, for $\mathcal{D}$ and T_f. Let $G_F(S_I, T_f)$ be the overall gain of integrating S_I using the fusion model F for T_f, and $C_F(S_I, T_f)$ be the corresponding integration cost. With $G_F(S_I, t)$ denoting the gain of integrating S_I for a single time point $t \in T_f$, and $A_{t \in T_f}$ denoting an aggregate function (e.g., average or max) over the time points in T_f, define the overall gain as $G_F(S_I, T_f) = \mathcal{A}_{t \in T_f} G_F(S_I, t)$. Similarly to Dong et al. [47], consider an additive cost model with $C_F(S_I, T_f) = \sum_{S \in S_I} C(S, T_f)$, where $C(S, T_f)$ denotes the cost of source $S \in S_I$ for T_f. The problem of time-aware source selection is defined as follows:

Definition 3. (TIME AWARE SOURCE SELECTION) Let $\bar{\mathbf{S}}$ be a set of sources, F be an integration model, and β_c be a budget on cost. Let T_f be a set of time points of interest. The Time-Aware Source Selection problem finds a subset $S_I \subseteq \bar{\mathbf{S}}$ that maximizes $G_F(S_I, T_f) - C_F(S_I, T_f)$ under the constraint $C_F(S_I, T_f) \leq \beta_c$.

It is easy to see that the problem of time-aware source selection is a strict generalization of the source selection problem introduced in the previous section. Therefore, time-aware source selection is also NP-complete. Next, we introduce two variations of the basic time-aware source selection.

Varying update frequencies: Dynamic sources offer significant opportunities to lower the integration cost while maintaining the quality of integrated data. In particular, choosing to integrate data from a source at a lower frequency than the source update frequency can lead to similar integration quality but reduced cost.

Example 8. *We focus on the BL and GDELT domains introduced in Section 1.1. For BL, we consider the evolution of coverage for the largest source, when its updates are acquired at half the update frequency. As shown in Figure 5.2(a), the quality loss is not significant while the cost is reduced significantly since only half of the updates are acquired. A similar behavior can be observed in GDELT as illustrated in Figure 5.2(b).*

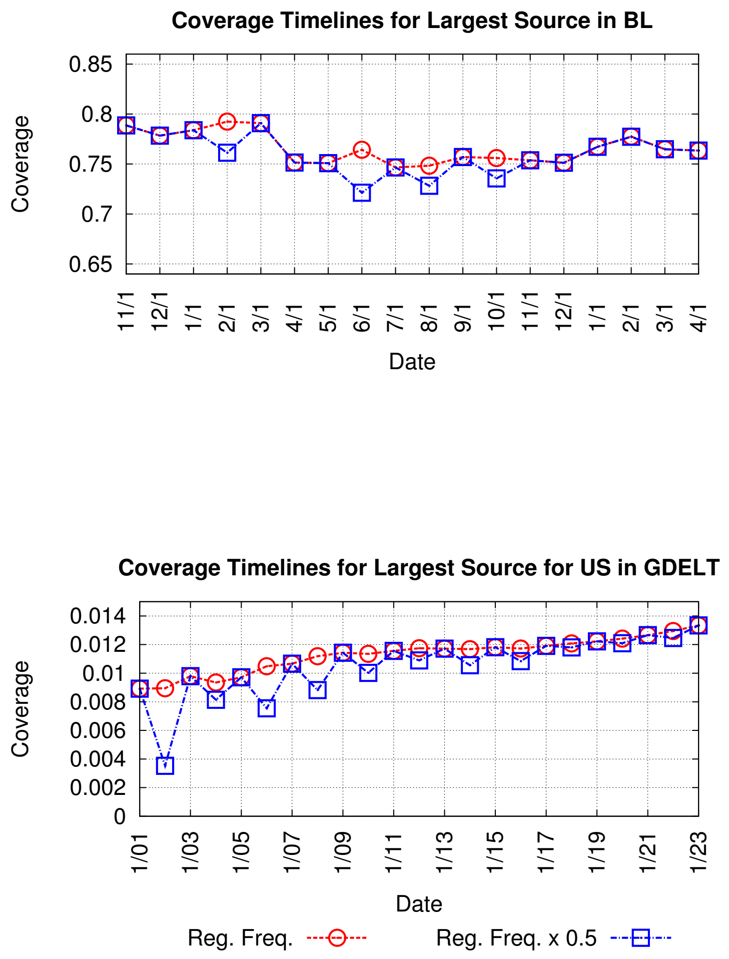


Figure 5.2: (a) Evolution of coverage for the largest source, when incorporating updates with different frequencies for BL. (b) Evolution of coverage for the largest source, when incorporating updates with different frequencies for GDELT.

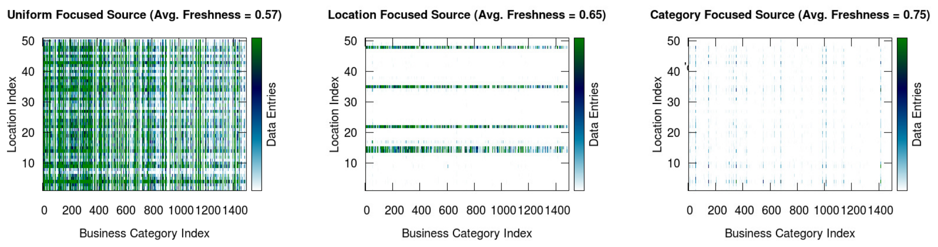


Figure 5.3: Sources covering different parts of the data domain.

We now extend the basic definition of time-aware source selection to exploit this opportunity for lowering the integration cost. Here, we select both the subset of sources that maximizes the integration profit and their *optimal frequencies* with which updates should be acquired. Given a set of selected sources S_I and their selected frequencies f_{S_I} let $G_F(S_I, f_{S_I}, T_f)$ denote the integration gain of S_I, under model F, with the frequencies specified in f_{S_I} for T_F, and $C_F(S_I, f_{S_I}, T_f)$ denote the corresponding integration cost. The problem of *varying update frequencies* is defined as follows:

Definition 4. (VARYING FREQUENCY SOURCE SELECTION) Let $\bar{\mathbf{S}}$ be a set of sources with variable update frequencies, F be an integration model, and β_c be a budget on cost. Let T_f be a set of time points of interest. The Varying Frequency Source Selection problem finds a subset $S_I \subseteq \bar{\mathbf{S}}$ and their corresponding update frequencies f_{S_I} that maximize $G_F(S_I, f_{S_I}, T_f) - C_F(S_I, f_{S_I}, T_f)$ under the constraint $C_F(S_I, f_{S_I}, T_f) \leq \beta_c$.

Integrating slices of data: Often sources may exhibit significant differences in the types of data they cover, and, instead of acquiring all the entries from a source, only a subset can be acquired (i.e., *a slice*), thus, reducing the integration cost.

For example, consider the three sources from BL shown in Figure 5.3. The listings domain is characterized by two dimensions: (a) the location of the listing and (b) the category of business (e.g., restaurants in New York). From the three data sources, (1) the first one provides entries for most location-category pairs, (2) the second one entries for a specific set of locations but across all categories, and (3) the last one entries for a specific set of categories but across all locations. A user focusing on certain locations may consider acquiring the second source and small parts of the first source to increase the overall coverage at a reduced cost.

In such cases, sources can be viewed as aggregates of multiple *micro-sources*, i.e., elemental sources focusing on certain slices of the data domain. The basic definition of time-aware source selection can be extended to account for this case as follows:

Definition 5. (SLICE TIME AWARE SOURCE SELECTION) Let $\bar{\mathbf{S}}_m$ be a set of micro-sources corresponding to slices obtained from a set $\bar{\mathbf{S}}$ of data sources, F be an integration model, and β_c be a budget on cost. Let T_f be a set of time points of interest. The Slice Time-Aware Source Selection problem finds a subset $S_I \subseteq \bar{\mathbf{S}}_m$ that maximizes $G_F(S_I, T_f) - C_F(S_I, T_f)$ under the constraint $C_F(S_I, T_f) \leq \beta_c$.

The SLICE TIME AWARE SOURCE SELECTION problem can be easily extended to identify optimal update frequencies as well.

5.3 Source Selection Algorithms

This section presents how the problem of time-aware source selection and its variations can be solved efficiently. First, we study a specific family of profit functions, i.e., *sub-*

modular profit function, and introduce a collection of efficient algorithms that come with theoretical guarantees. This class of profit functions is specific to independent sources. Then, we discuss how one can solve the problem of time-aware source selection for arbitrary profit functions and multi-variate functions.

5.3.1 Submodular Objective Functions

Dong et al. [47] proved that not only source selection in the context of data fusion is NP-complete but also estimating the integration quality is #P-hard. In contrast to static source selection, the quality estimators for time-dependent metrics can be approximated efficiently when sources are independent and under an integration model using the union semantics. Moreover, as discussed earlier the coverage and global freshness estimates are non-decreasing submodular functions. Exploiting submodularity, a set of local-search algorithms can be used for solving the different versions of time-aware source selection that come with theoretical guarantees on the quality of the solution.

Given a set of time points of interest T_f and a set of independent sources, the necessary conditions for a profit function to be submodular are:

- The integration gain $G_F(S_I, t)$ for each time point $t \in T_f$ has to be a non-negative linear function of either estimated coverage or global freshness of $F(S_I)$.
- The aggregate function A to compute $G_F(S_I, T_f)$ should be an average (or non-negative weighted average) since the class of submodular functions is closed under non-negative linear combinations.

- The cost function $C_F(S_I, T_f)$ has to be an additive function, so that the profit function $G_F(S_I, T_f) - C_F(S_I, T_f)$ is also submodular since the difference of a submodular and an additive function is still submodular.

Time-Aware Source Selection. Consider the basic version of time-aware source selection (Definition 3). For simplicity, assume that no constraint is set on the budget β_c. This version corresponds to the problem of maximizing a monotone submodular function, and can be solved by a local-search algorithm introduced by Feige et al. [52] (Algorithm 3).

Algorithm 3 Submodular Maximization

```
1: Input: $\bar{\mathbf{S}}$: set of sources available; $f$: value oracle access to submodular function; $n$:
   cardinality of $\bar{\mathbf{S}}$;
2: Output: $S_I$: a set of selected sources;
3: Set $v \leftarrow \arg\max\{f(u) | u \in \bar{\mathbf{S}}\}$ and $S_I \leftarrow \{u\}$
4: while one of the following local operations applies do
5:    /* Addition operation on $S_I$. */
6:    if $e \in \bar{\mathbf{S}} \setminus S_I$ such that $f(S_I \cup \{e\}) > (1 + \frac{\epsilon}{n^2} f(S_I))$ then
7:        $S_I \leftarrow S_I \cup \{e\}$
8:    /* Deletion operation on $S_I$. */
9:    if $e \in S_I$ such that $f(S_I \setminus \{e\}) > (1 + \frac{\epsilon}{n^2} f(S_I))$ then
10:       $S_I \leftarrow S_I \setminus \{e\}$
11: return $\arg\max_{\bar{S} \in \{S_I, \bar{\mathbf{S}} \setminus S_I\}}(f(\bar{S}))$
```

This algorithm starts by selecting a single source that maximizes the profit (Ln.3) and then tries to increase the value of the running solution S_I either by *including* a new element in S_I or by *discarding* one of the elements of S_I until a local optimum is reached (Ln. 4 - 10). Once a local optimum is reached, the algorithm checks if the complement of the running selection improves the solution and returns the selected sources (Ln.11). The algorithm is proven to yield a constant-factor approximation of $(1 + \frac{\epsilon}{n^2})$ and is shown to use $\mathcal{O}(\frac{1}{\epsilon} n^3 logn)$ oracle calls [52].

Varying Frequency Source Selection. Selecting the optimal set of sources and their corresponding frequencies can be expressed as an optimization problem with a unified objective function. Let $\bar{\mathbf{S}}$ be the set of available sources. For each source $S_i \in \bar{\mathbf{S}}$ one can select a variable update frequency $f'_{S_i} = \frac{f_{S_i}}{l_i}$, $l_i \in \{1, 2, \ldots, m_i\}, m_i \in \mathbb{Z}^+$ lower than the original frequency f_{S_i} of the source. Define the *augmented set of available sources* as $\mathbf{S_{aug}} = \{S_1^1, S_1^2, \ldots, S_1^{m_1}, \ldots, S_i^1, \ldots, S_i^{m_i}, \ldots S_k^{m_k}\}$ where S_i^j denotes a version of source S_i with an update frequency of $\frac{f_{S_i}}{j}$. One can now select sources from $\mathbf{S_{aug}}$ instead of $\bar{\mathbf{S}}$ - each entry of $\mathbf{S_{aug}}$ can be considered as a different source - under the constraint that only one of the $[l_i]$ versions of an actual source S_i will be selected for integration.

The submodular objective is now defined over the ground set $\mathbf{S_{aug}}$ and the frequency constraints can be expressed as a *uniform matroid constraint*. A uniform matroid U_n^r is defined over a set of n elements, and a subset of the elements is independent if and only if it contains at most r elements. Thus, each of the k constraints corresponds to a uniform matroid constraint of rank 1. Every uniform matroid is also a *partition matroid*. The varying frequency time-aware source selection corresponds to the problem of maximizing a monotone submodular function under a fixed number of partition matroid constraints, and can be solved by an algorithm that yields a constant-factor approximation of $\frac{1}{k+\epsilon}$ [95].

The algorithm is shown in Algorithm 4. The independent sets defined by the matroid constraints divide the ground set of sources in multiple partitions, each corresponding to the intersection of a combination of independent sets from all constraints. The algorithm identifies $k + 1$ disjoint partitions for which the optimization objective is locally maximized and returns the partition with the highest objective value. The algorithm

Algorithm 4 Submodular Maximization with Matroid Constraints

```
1: Input: $\mathbf{S_{aug}}$: ground set of sources; $k$: number of matroid constraints;
2: Output: $S_{opt}$: a set of selected sources;
3: Set $V_1 = \mathbf{S_{aug}}$
4: for $i = 1, \cdots, k+1$ do
5:     Apply the approximate local search procedure $A$ on a ground set $V_i$ to obtain a
       solution $S_i \subseteq V_i$ corresponding to the problem:
           $max\{f(S) : S \in \cap_{j=1}^{k} \mathcal{I}_j, S \subseteq V_i\}$
6:     Set $V_{i+1} = V_i \setminus S_i$
7: return $S_{opt} \leftarrow max\{f(S_1), \cdots, f(S_{k+1})\}$
```

performs $k+1$ iterations (Ln. 4) and at each iteration i uses a local-search procedure (Ln. 5) similar to the one used in the basic version (Algorithm 5) to select an approximately optimal set of sources over a subset V_i of the available sources. Each of the sets V_i corresponds to the union of a subset of the aforementioned partitions. After each iteration the set of available sources for the next iteration is restricted to sources that were not previously selected (Ln. 6). Finally, the algorithm returns the partition, i.e., a subset of the data sources, with the highest objective value (Ln. 7).

The local search procedure is given a set of available data sources and greedily selects a set of available sources that maximizes the optimization objective under the given constraints. The algorithm detects a single source that yields the highest objective value (Ln. 4) and proceeds by searching the neighborhood of the running solution for solutions that improve the objective. The local neighborhood of the running solution is constructed either by removing a source from the solution (Ln. 5-7) or by exchanging a set of selected sources with a new source such that all the constraints are satisfied (Ln. 8-10). The local search procedure iterates until a local optimum is retrieved. The running time of Algorithm 5 is $\frac{1}{\epsilon}n^{\mathcal{O}(k)}$ with $n = |\mathbf{S_{aug}}|$ and k is the number of matroid constraints, and thus the running time of Algorithm 4 is $\mathcal{O}((k+1)\frac{1}{\epsilon}n^{\mathcal{O}(k)})$ [95].

Algorithm 5 Local Search Procedure

```
1: Input: $X$: ground set of sources; $f$: value oracle access to submodular function; $n$: cardinality of $\mathbf{S_{aug}}$;
2: Output: $S_I$: a set of selected sources;
3: Set $v \leftarrow \arg\max\{f(u)|u \in X\}$ and $S_I \leftarrow \{u\}$
4: while one of the following local operations applies do
5:     /* Delete operation on $S_I$. */
6:     if $e \in S_I$ such that $f(S_I \setminus \{e\}) > (1 + \frac{\epsilon}{n^4} f(S_I))$ then
7:         $S_I \leftarrow S_I \setminus \{e\}$
8:     /* Exchange operation on $S_I$. */
9:     if $d \in X \setminus S_I$ and $e_i \in S_I \cup \{\emptyset\}$ (for $1 \leq i \leq k$) are such that $(S_I \setminus \{e_i\}) \cup \{d\} \in \mathcal{I}_i$ for all $i \in [k]$ and $f((S_I \setminus \{e_1, \cdots e_k\}) \cup \{d\}) \geq (1 + \frac{\epsilon}{n^4}) f(S_I)$ then
10:        $S_I \leftarrow (S_I \setminus \{e_1, \cdots, e_k\}) \cup \{d\}$
11: return $S_I$
```

Slice Time-Aware Source Selection. The basic submodular optimization problem of time-aware source selection can be trivially extended to account for this case by including all the micro-sources in $\bar{\mathbf{S}}$. The set of available sources can also be replaced by its augmented set to account for variable update frequencies of the micro-sources.

5.3.2 Arbitrary Objective Functions

When the integration profit is not submodular (e.g., when the gain is quantified using the accuracy or local freshness of $F(S_I)$ or when the sources are dependent), one can apply the *greedy randomized adaptive search procedure* (GRASP) meta-heuristic [54]. GRASP was also used by Dong et al. [47] to solve source selection for static sources. The GRASP algorithm for solving the basic time-aware source selection problem follows similar steps as the algorithm shown in Dong et al. However, GRASP needs to be extended in the case of varying frequencies to account for the matroid constraints introduced above.

GRASP is conceptually similar to the submodular optimization algorithm, in that it starts by selecting the best source and at each step explores the local neighborhood of the running solution in a hill-climbing fashion, by adding and deleting sources. However, in each step GRASP identifies the top-k candidate decisions in terms of resulting profit and chooses one at random. Since the process is randomized, GRASP repeats the overall selection process r times and chooses the best selection out of these repetitions. Depending on the number of repetitions, the complexity of GRASP increases significantly. Finally, GRASP *does not come* with any theoretical guarantees.

5.3.3 Multiobjective Source Selection

As discussed in Section 3.5, multiple metrics can be used to characterize the quality of data. Thus far, the techniques presented for solving source selection, considered a unified objective function that combines the different metrics into a single gain value. However, many times it is hard for users to know the right trade-off between the individual quality metrics, while it is natural for them to specify a cost constraint. Considering this, the source selection problem can be case as a multiobjective optimization problem that finds the set of *Pareto optimal* solutions corresponding to the source selection problem at hand.

Pareto optimality states that for the returned solutions, i.e., *the Pareto front*, it is impossible to improve one of the individual quality metrics without hurting at least one other metric [58]. Discovering all the solutions on the Pareto front is expensive as one needs to reason about all the potential trade-offs amongst the available quality metrics. To address this issue we use a sampling strategy to recover solutions that correspond to different quality trade-offs. Let Q be the set of quality metrics under consideration and $G_F^q(\cdot)$ be a function computing the gain of integration with respect to quality metric $q \in Q$ for any set of sources. We compute the total gain of integration as a weighted linear combination of the individual gain for each quality metric, i.e., $G_F(\cdot) = \sum_{q \in Q} w_q \cdot G_F^q(\cdot)$. Given this definition of the total gain of integration, we sample different combinations for the weights w_q and solve source selection for each of those using the algorithms above.

The result of this sampling process is a collection of vector points where the dimensions of each vector correspond to the different quality metrics in Q. Now, our goal becomes finding the Pareto optimal vectors in the collection. For this, we use the Simple Cull (SC) minimization algorithm [127]. At a high-level this algorithm maintains a set $\mathcal{C}_{min}$ of Pareto points among the points observed so far. Whenever a new point is inspected, either the point is *dominated* (i.e., all of its quality values are lower than an existing point in $\mathcal{C}_{min}$) and the point is discarded or if the point is not dominated it is added in $\mathcal{C}_{min}$ and any points from $\mathcal{C}_{min}$ dominated by this new point are removed. The worst case complexity of the algorithm is $\mathcal{O}(n^2)$ where n is the number of input points. Here, these points correspond to the sampled source selection solutions.

5.4 Experimental Evaluation

We present an experimental evaluation of the source selection algorithms presented above. The main questions we seek to address are: (1) how different source selection algorithms perform under different families of gain and cost functions, and (2) how well do these algorithms scale. We study these questions on both real-world and synthetic datasets.

5.4.1 Experimental Setup

Data. For our real-world experiments we use the business listing and GDELT datasets described in Section 3.7.1. Furthermore, we use a collection of synthetically generated datasets BL+ , using BL as a seed, to evaluate the scalability of the proposed algorithms. We decompose the sources in BL into multiple overlapping micro-sources, where each micro-source covers a randomly selected subset of the initial source. If $|L|$ denotes the locations in a source S, we construct each micro-source to contain all the entities from S belonging to a randomly selected subset of locations from the original source. The number of locations in each micro-source is chosen uniformly at random from a uniform distribution $\mathcal{U}(0.2 \cdot |L|, 0.5 \cdot |L|)$. We vary the total number of micro-sources to be in $\{0, 1, 2, 5, 10, 20, 50, 100, 200\}$ obtaining 9 different datasets with 43 to 8643 sources.

Algorithms. The following algorithms are considered:

- Greedy: The same as the greedy algorithm used by Dong et al. [47]. Starting from an empty selection set the algorithm iteratively selects the source that maximizes the integration profit until it reaches a local optimum.
- MaxSub: Depending on the version of time-aware source selection MaxSub corresponds to the submodular optimization algorithms in Section 4.4.
- GRASP: The GRASP algorithm proposed by Dong et al. [47] for different configurations of (κ, r).

All algorithms are implemented in Java and the evaluation is performed on an Intel(R) Core(TM) i5 2.3 GHz/64bit/8GB machine.

Gain-Cost Models. The following two families of gain models are considered: (1) quality driven, and (2) data driven models. For the first, let Q be the quality (i.e., coverage, freshness or accuracy) of the integrated data. The specific gain functions are: (1) LINEARGAIN assuming that the gain grows linearly with a certain data quality metric Q and sets $G(Q) = 100Q$, (2) QUADGAIN assuming that the gain grows quadratically with Q and sets $G(Q) = 100Q^2$, and (3) STEPGAIN assuming that reaching a milestone of quality increases the gain significantly and sets

$$G(Q) = \begin{cases} 100Q & \text{if } 0 \leq Q < 0.2 \\ 100 + 100(Q - 0.2) & \text{if } 0.2 \leq Q < 0.5 \\ 150 + 100(Q - 0.5) & \text{if } 0.5 \leq Q < 0.7 \\ 200 + 100(Q - 0.7) & \text{if } 0.7 \leq Q < 0.95 \\ 300 + 100(Q - 0.95) & \text{if } 0.95 \leq Q \leq 1.0 \end{cases}$$

For the second category, denoted by DATAGAIN, a gain of \$10 is considered for each covered item in $F(S_I)$ and for a particular time point t the integration gain is $G(F(S_I), t) = 10 \cdot \mathsf{Cov}^*(\mathsf{F}(\mathsf{S_I}), \mathsf{t})|\Omega|_\mathsf{t}$.

Finally, an additive cost function is considered. Similarly to DATAGAIN, each entity has a basic cost of \$10 and an actual cost of $c = \frac{\$10}{(\text{ \#sources mentioning the item})}$. The cost c of a source is the total cost of items contained in it. When considering varying frequencies for sources the source cost is set to $c' = c/(1 + m/10)$, where m is the frequency divisor. Finally, both the gain and cost are rescaled to take values in $[0, 1]$.

5.4.2 Performance of Source Selection Algorithms

We first evaluate the performance of the aforementioned source selection algorithms. We focus on two scenarios.

Fixed update frequencies. Consider a fixed update frequency for each data source and the basic time-aware source selection problem with a user being interested in ten future time points for six data domain points. The overall gain is computed by taking the average gain across time points. The selection tasks for BL correspond to the six largest domain points corresponding to four business types in the states of California and New York. For GDELT, the selection tasks correspond to six domain points for events in the United States. The different algorithms are compared considering DATAGAIN, and LINEARGAIN, QUADGAIN, STEPGAIN with the gain being quantified using coverage and accuracy for BL and coverage for GDELT.

Greedy, MaxSub and GRASP with $\kappa \in \{1, 2, 5, 10\}$ and $r \in \{1, 10, 20, 100\}$ are used for solving the source selection problem for both datasets. GRASP with $(\kappa = 1, r = 1)$ corresponds to a hill-climbing algorithm. For each gain function, I compare the selections by the various algorithms and choose the one with the highest profit as the best. The results presented below report the percentage of times the best selection is returned by each algorithm and for sub-optimal selections the average and maximum, reported in parenthesis, profit difference from the best selection is shown. For GRASP the (κ, r) configuration that obtained the best selection is also reported.

The results for BL are shown in Table 5.1. MaxSub and GRASP outperform Greedy returning solutions that result in up to 9.5% higher objective values on average and up to

Table 5.1: Various algorithms for source selection in BL on the percentage of outputting the best selection and average and worst (reported in parenthesis) profit difference from the best selection. Notation (κ, r) denotes the best performing GRASP algorithm. Sources with a fixed update frequency are considered.

Avg. Selection Quality					
Gain	Metric	Msr.	Greedy	Maxsub	Grasp
Linear	cov.	best	16.7%	50%	100% (5, 20)
		diff.	.005 (.01)%	.001 (.007)%	-
	acc.	best	0%	33.3%	83.3% (2, 100)
		diff.	9.5 (53.7)%	.39 (2.31)%	8.9 (53.7)%
Quad.	cov.	best	33.3%	66.7%	100% (10, 100)
		diff.	.017 (.06)%	.012 (.06)%	-
	acc.	best	100%	100%	100% (1,1)
		diff.	-	-	-
Step	cov.	best	50.0%	66.7%	83.3% (10, 100)
		diff.	7.45 (27.8)%	1.76 (10.6)%	.7 (4.2)%
	acc.	best	50%	66.7%	83.3% (5,100)
		diff.	6 (23.98)%	.8 (4.7)%	3.99 (23.98)%
Data	-	best	16.7%	50%	83.3% (5, 20)
		diff.	.004 (.01)%	.001 (.003)%	.002 (.007)%

53.7% in the worst case. While GRASP returns the best solution most of the times, the solutions returned by MaxSub are on average comparable to the ones obtained by GRASP with a low average profit difference. This behavior is expected since MaxSub unlike Greedy comes with rigorous theoretical guarantees, and GRASP applies a similar local search procedure to the one used by MaxSub. Nevertheless, observe that randomization and multiple iterations help GRASP to obtain marginally better solutions. However, if one considers the run time of the algorithms (shown in Table 5.2), she sees that MaxSub is one to two orders of magnitude faster than GRASP. Eventually, depending on the gain function, MaxSub can be a viable alternative compared to GRASP. Although, if the profit requirements are strict one should use GRASP.

Table 5.2: Average run times of the source selection algorithms for BL. Notation (κ, r) denotes the parameters of GRASP.

Avg. Run Time (sec)							
Gain	Metric	Greedy	Maxsub	(1,1)	(2,10)	(5,20)	(10,100)
Linear	cov.	0.05	0.16	0.16	2.23	4.35	20.13
	acc.	0.42	1.6	1.5	14.9	39.9	144.2
Quad	cov.	0.03	0.11	0.14	1.6	3.3	17.6
	acc.	0.14	0.35	0.43	5.5	11.9	57.8
Step	cov.	0.03	0.11	0.13	1.6	2.8	15.25
	acc.	0.14	0.46	0.5	6.4	14.16	74.02
Data	-	0.04	0.18	0.17	2.4	4.6	25.7

Table 5.3: Performance and runtime comparison of the source selection algorithms for GDELT, showing the percentage of outputting the best selection and average and worst (reported in parenthesis) profit difference from the best selection. Notation (κ, r) denotes the best performing GRASP algorithm.

Avg. Selection Quality				
Gain	Msr.	Greedy	Maxsub	Grasp
Linear Cov.	best	16.7%	50%	100% (10, 100)
	diff.	4.01 (13.7)%	0.5 (2)%	-
	runtime (sec)	8.58 (37)	74.12 (326)	1231.05 (4363)
Data	best	3.3%	0%	100% (10, 100)
	diff.	5.64 (14.9)%	.91 (3)%	-
	runtime (sec)	1.01 (5.03)	8.96 (44)	868.87 (4322)

Similar results are observed for GDELT. Table 5.3 reports the performance and runtime of the various algorithms. One can see that for LINEARGAIN and DATAGAIN both MaxSub and GRASP outperform Greedy. While GRASP never fails to detect the best solution, the profit difference between MaxSub and GRASP is very small and more importantly MaxSub is again one to two orders of magnitude faster. The results for QUADGAIN and STEPGAIN are omitted since all algorithms retrieved the same solution.

The following results focus on the average quality of the retrieved solution and the average number of sources selected for BL and GDELT. The results for BL are shown in Table 5.4. As shown, all algorithms tend to choose fewer sources when the gain is mea-

Table 5.4: Characteristics of the selected sources for various algorithms on BL for fixed source update frequencies.

Alg	Coverage		Accuracy	
	Avg. Qual.	Avg. #Srcs	Avg. Qual.	Avg. #Srcs.
Greedy	0.52	10	0.49	8
MaxSub	0.56	11	0.57	7.6
GRASP	0.56	11	0.57	8.6

Table 5.5: Characteristics of the selected sources for various algorithms on GDELT for fixed source update frequencies.

Alg	Greedy	MaxSub	GRASP
Avg. Coverage	0.57	0.62	0.65
Avg. # Srcs	154	163	167

sured with respect to accuracy. All algorithms tend to select fewer large uniform sources and prefer more specialized smaller sources. Figure 5.4 shows the various source types selected from GRASP when the LINEARGAIN function with coverage and accuracy is used to specify the gain. A similar behavior was observed for all algorithms. Finally, Table 5.5 shows the results for GDELT. GRASP and MaxSub were able to select significantly more sources and increase the coverage of the retrieved solution by 5% and 8%.

Variable update frequencies. The following experiment considers different versions for each source corresponding to different update frequencies. For BL seven different

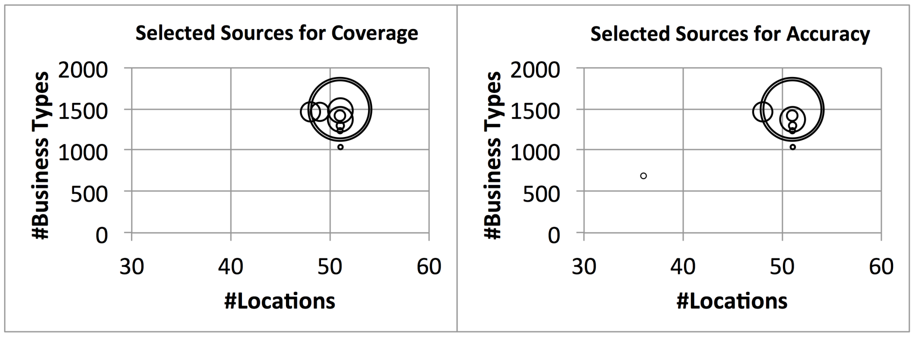


Figure 5.4: Selected sources when the gain is defined using coverage and accuracy. For accuracy smaller and more specialized sources are preferred.

Table 5.6: Characteristics of the selected sources for various algorithms on BL for sources with variable update frequencies.

Alg	Coverage		Accuracy	
	Avg. Qual.	Avg. #Srcs	Avg. Qual.	Avg. #Srcs.
Greedy	0.96	15.6	0.948	14.6
MaxSub	0.976	15.6	0.958	15
GRASP	0.976	16	0.958	16

versions $S_i^1 \cdots, S_i^7$ are taken for each original source. As before, the user is assumed to be interested in the same ten future time points and the same six data domains. Similar performance, as the one presented above, is observed for all the algorithms. Namely, GRASP outperforms both Greedy and MaxSub. The difference in profit between the retrieved solutions is significantly smaller (less than 0.5$\%$ in average) compared to the previous case.

Focusing on the quality of the solutions returned by the various algorithms, Allowing sources to have variable frequencies significantly improves the quality of the retrieved solutions. The average coverage and accuracy rise to 0.976 and 0.958 respectively, compared to 0.56 and 0.57 for the case of fixed frequencies. The reason is that by reducing the update frequency of a source the corresponding cost is reduced, and hence, the algorithms choose to integrate more sources. The corresponding results are reported in Table 5.6. Finally, observe that all algorithms preferred selecting large sources with a significantly reduced update frequency. However, for small specialized sources they either select the original update frequency or a small divisor of that. The average frequency divisors for uniform and specialized sources are reported in Table 5.7.

Table 5.7: Average frequency divisor for uniform and specialized sources in the solution of the various algorithms.

Alg	Greedy	MaxSub	GRASP
Uniform Srcs.	4.9	5.2	4.9
Specialized Srcs.	2.6	2.9	3.2

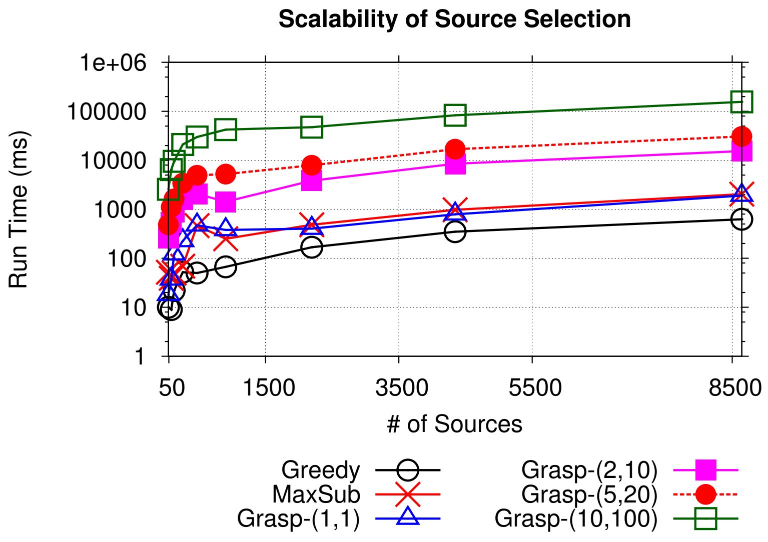


Figure 5.5: Run time of the various algorithms as the number of sources increases.

5.4.3 Scalability

The next set of experiments evaluates the scalability of the various algorithms as the number of available sources increases. The gain function used corresponds to LINEAR-GAIN with coverage, and the synthetically generated datasets BL+ , considering source selection for a single data point for 10 future time points, are used. The corresponding run times are shown in Figure 5.5, where the x-axis corresponds to the number of available data sources and the y-axis (shown in log-scale) to the run time measured in milliseconds. As shown, MaxSub is one to two orders of magnitude faster compared to the best performing alternatives of GRASP, and scales better as the number of sources increases.

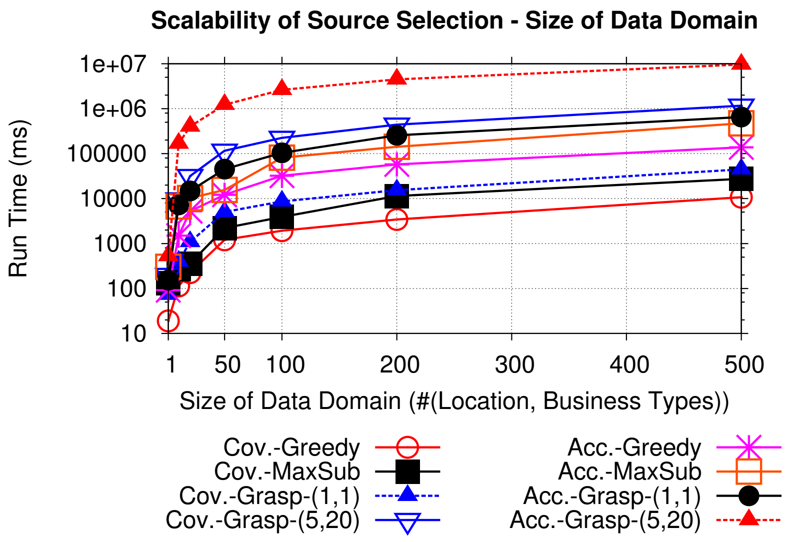


Figure 5.6: Run time of the algorithms as the the size of the input domain increases.

BL is used to examine the scalability of the various algorithms with respect to the size of the input data domain, where the size of the input domain is the number of location-business type pairs specified in a certain user query. The performance of Greedy, MaxSub and GRASP with $(\kappa = 5, r = 20)$ is evaluated for LINEARGAIN with coverage and accuracy. The corresponding run times are shown in Figure 5.6, where the x-axis corresponds to the size of the input domain and the y-axis (shown in log scale) to the run time measured in milliseconds. Again, observe that MaxSub is an order of magnitude faster than GRASP-(5,20).

5.4.4 Main Results

The main results are as follows.

1. Most of the time, GRASP selects the subset of dynamic sources with the highest profit. However, the solutions discovered by MaxSub are mostly comparable to

the best solutions, with an average quality loss of less than 2% and a worst-case quality loss of about 10% compared to the best solution. However, we point out that there are cases where GRASP is significantly worse than MaxSub, with an average quality loss of about 9% and a worst-case quality loss of over 50% compared to the solutions by submodular optimization. Greedy is the worst strategy overall.

2. Finally, MaxSub is one to two orders of magnitude faster than GRASP, and scales better as the number of sources increases. Coupled with the robust quality of its solutions, the significantly faster run-times makes MaxSub a viable alternative to GRASP, especially for large instances of source selection.

5.5 Related Work

The most relevant work to the techniques described in this chapter is that by Dong et al. [47]. The authors introduced the problem of source selection, and showed how one can maximize the profit of integration by optimizing the gain and cost of integrating sources jointly. However, this work focuses on static sources and is not applicable to sources whose content changes dynamically. Moreover, the proposed algorithms, while effective in practice, do not come with rigorous theoretical guarantees on their performance. In this chapter, we considered the problem of source selection for dynamic sources and showed how to select a nearly-optimal set of sources and determine their optimal update frequencies by providing a set of algorithms with rigorous theoretical guarantees. Regarding the cost function families, in this chapter, we limited source selection to simple cost families using a model similar to that of Dong et al. [47]; however, other cost models such as the one introduced by Kruse et al. [142] can be seamlessly incorporated in our techniques. Finally, algorithms for computing the Pareto frontier of a finite set of alternatives have been studied in the skyline query literature [63, 94]. While here we use a simple algorithm for approximating the Pareto optimal frontier, these more sophisticated techniques can be used to extend our methodology.

5.6 Summary

In this chapter, we studied the problem of identifying the most valuable sources for integration in the case of dynamic data sources. We introduced the problem of time-aware source selection and variations of it that not only select a nearly-optimal set of sources for integration but decide the optimal frequency with which to acquire data from each source. We also proposed an algorithmic framework for solving time-aware source selection that comes with rigorous theoretical guarantees on the quality of the selected set of sources. Finally, we experimentally evaluated our source selection algorithms under different families of benefit and cost functions, and showed that our submodular optimization algorithms provide solutions of similar, and in some cases, better quality compared to previous state-of-the-art approaches. Moreover, our algorithms are one to two orders of magnitude faster and scale better as the number of sources increases making them a viable alternative for large instances of source selection.

Chapter 6: Managing Source Dependencies with Probabilistic Databases

Thus far, we considered sources that update their content independently. For instance, in Section 3.5.2.1 we described how the content changes for a set of sources can be estimated efficiently when they are independent. While this assumption is valid for many practical scenarios, sources can exhibit dependencies. In most cases, dependence between sources arises when sources copy information from each other. This is quite prevalent in the news domain where many news agencies collect articles from local correspondents and then larger news papers copy news articles from these agencies. Furthermore, there might be scenarios where sources provide conflicting information due to different perspectives they may have. Recently, there has been an increasing interest in discovering the dependencies between sources [171, 45, 12] and reasoning about the accuracy of the integration result in the presence of source dependencies [126].

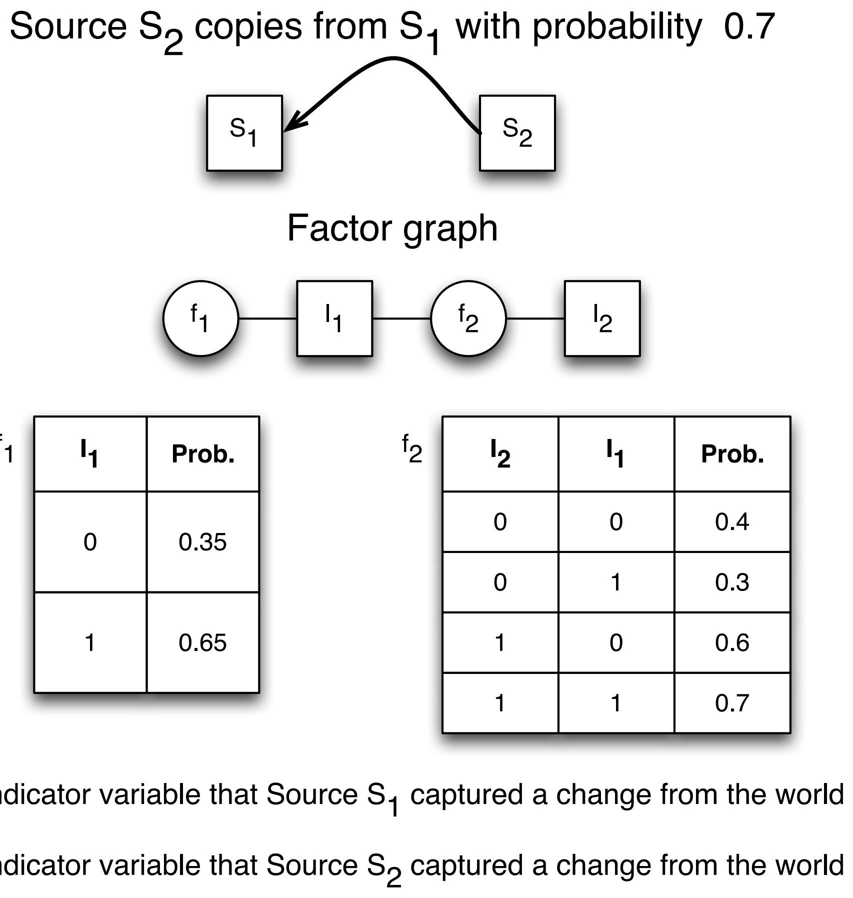
In this chapter, we demonstrate how source dependencies can be represented as a factor graph and how computing the content changes corresponds to performing inference over Boolean formulas with dependent variables (i.e., Boolean queries). When a large number of data sources is available, we discuss how a probabilistic database can be used to store the corresponding factors graphs and evaluate Boolean queries over them.

6.1 Introduction

In Section 3.5.2.1, we showed how, under the union semantics, estimating content changes in the integration result of a set of sources S_I corresponds to computing the probability that at least one of the sources captures a change from the underlying data domain. We also described that this probability is equivalent to computing the probability of a disjunctive Boolean formula evaluating to true. The base variables in that formula correspond to Boolean indicator variables that sources in S_I captured a change or not. When sources are dependent, e.g., they copy from each other following known patterns, one can easily see that the variables in this formula are correlated. Hence, one cannot use Equations 3.22, 3.23 and 3.24 to efficiently estimate the changes in the integration result.

Similarly, Pochampally et al. [126]show that estimating the overall accuracy of the integration result for a set of sources whose precision and recall are known, corresponds to computing the probability of a conjunctive Boolean formula evaluating to true. We refer the reader to Section 4.1 in Pochampally et al. [126] for details on this derivation.

Source dependencies, and hence, the correlations of the aforementioned indicator variables, can be represented using a *factor graph* [92]. For example, Figure 6.1 shows an example of two dependent sources. Source S_2 is copying information from S_1 with probability 0.7. This means that whenever S_1 captures a change in the underlying data domain, then S_2 will reflect it as well with probability 0.7. Variables I_1 and I_2 are the corresponding Boolean indicator variables for capturing changes from the world. The copying dependence between S_1 and S_2 corresponds to I_2 being true with probability 0.7 when I_1 is true. This is captured by the factor graph shown in the figure. Recently, Li



f_1

I_1	Prob.
0	0.35
1	0.65

f_2

I_2	I_1	Prob.
0	0	0.4
0	1	0.3
1	0	0.6
1	1	0.7

I_1: indicator variable that Source S_1 captured a change from the world

I_2: indicator variable that Source S_2 captured a change from the world

Figure 6.1: An example of two sources where S_2 copies from S_1 and the corresponding factor encoding that dependency.

et al. [99] studied the problem of large-scale copy detection and showed how one can retrieve source dependencies and the probabilities as the ones shown in Figure 6.1 by collectively analyzing the data provided by different sources.

Given a factor graph as the one shown above, we can reason about the content changes of sources by computing the probabilities shown in Equations 3.22, 3.23 and 3.24. This corresponds to computing the probability of the corresponding Boolean formulas over the factor graph capturing the dependencies of sources. In the previous example, if we want to compute the probability that the integration result will capture a change in the world, we need to compute the probability $\Pr(I_1 \vee I_2 = true)$. Given the dependencies across sources, this probability is $\Pr(I_1 \vee I_2 = true) = \Pr(I_1 = true) + \Pr(I_1 = false \wedge I_2 = true) = 0.65 + 0.35 * 0.6 = 0.86$. We refer to these Boolean computations as *Boolean queries*. Notice that this probability is lower than considering that sources are independent. More precisely, from the factor graph shown above we have that $\Pr(I_1 = true) = 0.65$ and $\Pr(I_2 = true) = 0.665$. If we assume that sources are independent we have that $\Pr(I_1 \vee I_2 = true) = 0.88275$ which overestimates the corresponding probability by 2%.

Given the large number of data sources and their dependencies, the goal now becomes storing and managing these dependencies efficiently. Moreover, we want to be able to evaluate Boolean queries as the one presented above efficiently. Recently, the paradigm of probabilistic databases has been proposed as a means for storing large factor graphs and evaluating Boolean queries over them efficiently [31, 143].

Query evaluation over probabilistic databases as well as probabilistic inference are known to be $\#P$-hard. To overcome this limitation, a number of different approaches have been explored to make query evaluation over probabilistic databases more efficient, including *knowledge compilation techniques* [115] or approximation techniques [91, 117, 132]. While these approaches perform reasonably well under the scenario that the random variables stored in the database are independent, they exhibit poor scalability in the presence of correlated variables, which is exactly the case for data source dependencies.

In the remainder of this chapter, we introduce an algorithmic framework that increases the efficiency of query evaluation over probabilistic databases. To speed up query evaluation, our techniques exploit *context-specific independence* and *determinism* in the correlations, collectively referred to as *local structure* [36]. Context-specific independence [17, 173], often observed in practice, refers to independences that hold given a specific assignment of values to certain variables. Determinism in the correlations, i.e., assigning zero probability to some joint variable assignments, typically manifests in uncertainties involving logical constraints, e.g., mutual exclusion, implications, etc. Exploiting such local structure enables probabilistic inference to run efficiently in many scenarios, where the standard inference techniques such as variable elimination are not feasible [36]. Our framework builds upon the notion of an *arithmetic circuit* (AC) [36], which is a compiled and compact representation of a factor graph that can effectively exploit local structure to drastically reduce online inference times [24]. The highly-compact compiled form of ACs makes it a non-trivial challenge to compute probabilities of Boolean formulas over them. To address this challenge, we introduce *annotated arithmetic circuits* (AACs), an extension where we add variable annotations on the internal operation nodes of an AC, and develop a novel algorithm for *merging* two AACs to, in essence, combine the uncertainties captured by the AACs. For evaluating queries over an AAC-representation of a probabilistic database, we represent the resulting lineage formulas using *ordered binary decision diagrams* (OBDDs), suggested in prior work [115]. However, the AAC-representation of the database imposes significant constraints on how OBDDs can be generated, requiring us to develop new algorithms for this task. However, we show that our techniques exhibit speed-ups of at least one order of magnitude over competing approaches.

The remainder of the chapter is structured as follows. In Section 6.2, we discuss the necessary background as related to this chapter. In Section Section 6.3 we present an overview of our algorithmic framework. Then in Section 6.4 we define the structure of Annotated Arithmetic Circuits (AACs) and how one can compile a factor graph to an AAC. Then in Section 6.5, we present how one can evaluate boolean queries over AACs. We experimentally evaluate our approach in Section 6.6. Finally, in Section 6.7, we discuss related work and in Section 6.8 we summarize the contributions of this chapter.

6.2 Preliminaries

We present a short review of probabilistic databases and arithmetic circuits.

6.2.1 Probabilistic Databases

A probabilistic database can be defined using the *possible world semantics* [33]. Let $\mathcal{R}$ be a set of relations, $\mathcal{X} = \{X_1, \cdots, X_n\}$ be a set of random variables associated with the tuples or attributes stored in the database (these could either be binary random variables capturing tuple existence uncertainty, or discrete random variables capturing attribute value uncertainty), and Φ be a joint probability distribution over $\mathcal{X}$. A probabilistic database D is defined to be a probability distribution over a set of deterministic databases (or possible worlds) $\mathcal{W}$ each of which is obtained by assigning $\mathcal{X}$ a joint assignment $\mathbf{x} = \{X_1 = x_1, \ldots, X_n = x_n\}$ such that $x_i \in \text{dom}(X_i)$. The probability associated with a possible word obtained from the joint assignment $\mathbf{x}$ is given by Φ.

Given a query q to be evaluated against database D, the result of the query is defined to be the union of results returned by each possible world. Furthermore, the marginal probability of each result t in the union is obtained by summing the probabilities of the possible worlds $W_t \subseteq W$ that return t: $\Pr(t) = \sum\limits_{w \in W_t} \Pr(w)$.

Representation. Typically, we are not able to represent the uncertainty in the dataset using an explicit listing of the joint probability distribution Φ. Instead more compact representations need to be used. The different representations differ in their expressibility and the complexity of query evaluation. The simplest representation associates *tuple existence* probabilities with individual tuples, and assumes that the tuple existences are independent of each other. However, most real-world datasets contain complex correlations, therefore, making an independence assumption can lead to oversimplification and large errors [143, 144].

Instead, we use a general and flexible representation of a probabilistic database, proposed by Sen et al. [143] and also used by Wick et al. [167], that can capture complex correlations among the tuples or the attributes in the database through use of *factor graphs*, a class of graphical models that generalizes both directed Bayesian networks and undirected Markov networks. More formally we have:

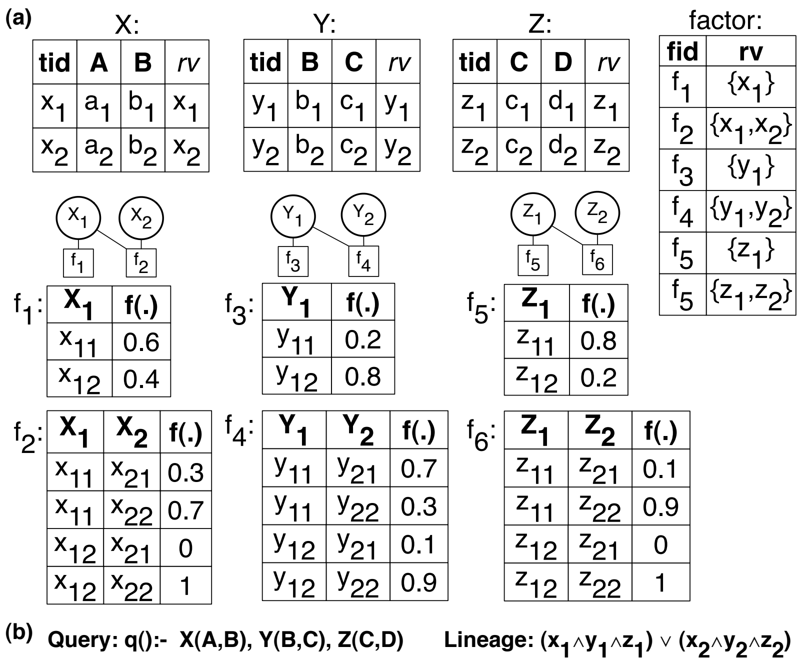
Definition 6. A *factor* $f : \text{dom}(X_1) \times \text{dom}(X_2) \times \cdots \times \text{dom}(X_m) \rightarrow R^+$ is a function over a set of random variables $\mathbf{X} = \{X_1, X_2, \ldots, X_m\}$ such that $f(\mathbf{x}) \geq 0, \forall \mathbf{x} \in \text{dom}(X_1) \times \cdots \times \text{dom}(X_m)$. The set of variables $\mathbf{X}$ is called the scope of the factor and denoted Scope$[f]$.

Definition 7. A factor graph $\mathcal{P} = (\mathcal{F}, \mathcal{X})$ defines a joint distribution Φ over the set of random variables $\mathcal{X}$ via a set of factors $\mathcal{F}$, where $\forall f(\cdot) \in \mathcal{F}$, Scope$[f] \subseteq \mathcal{X}$. Given a complete joint assignment $\mathbf{x} = \{X_1 = x_1, \ldots, X_n = x_n\}$ such that $x_i \in \text{dom}(X_i)$, the joint distribution is defined by $\Phi(\mathbf{x}) = \Pr(\mathbf{x}) = \frac{1}{\mathcal{Z}} \prod_{f \in \mathcal{F}} f(\mathbf{x}_f)$ where $\mathbf{x}_f$ denotes the assignments restricted to Scope$[f]$ and $\mathcal{Z} = \sum_{\mathbf{x}' \in \mathcal{X}} \prod_{f \in \mathcal{F}} f(\mathbf{x}'_f)$.

This leads us to a formal definition of a probabilistic database:

Definition 8. A probabilistic database D is a pair $(\mathcal{R}, \mathcal{P})$ where $\mathcal{R}$ is a set of relations and $\mathcal{P}$ denotes a factor graph defined over the set of random variables associated with the tuples in $\mathcal{R}$.

We require that the joint distribution defined by the factor graph satisfy certain normalization constraints, i.e., the partition function $\mathcal{Z} = 1$. This does not imply a limitation on the applicability of the proposed framework but is only used for ease of representation. Figure 6.2(a) shows an example probabilistic database represented using factors, along with the factor graphs themselves (which contain nodes for each factor and each random variable, and a factor is connected to all variables that it is defined over). We assume we only have tuple existence uncertainties, and the Boolean random variables corresponding to the tuple existences are associated with the tuples $(x_1, x_2, \cdots)$. In the remainder of the chapter, for any Boolean random variable x we will denote its **true** and **false** assignment


(a)

X:

tid	A	B	*rv*
x_1	a_1	b_1	x_1
x_2	a_2	b_2	x_2

Y:

tid	B	C	*rv*
y_1	b_1	c_1	y_1
y_2	b_2	c_2	y_2

Z:

tid	C	D	*rv*
z_1	c_1	d_1	z_1
z_2	c_2	d_2	z_2

factor:

fid	rv
f_1	$\{x_1\}$
f_2	$\{x_1,x_2\}$
f_3	$\{y_1\}$
f_4	$\{y_1,y_2\}$
f_5	$\{z_1\}$
f_5	$\{z_1,z_2\}$

f_1:

X_1	f(.)
x_{11}	0.6
x_{12}	0.4

f_3:

Y_1	f(.)
y_{11}	0.2
y_{12}	0.8

f_5:

Z_1	f(.)
z_{11}	0.8
z_{12}	0.2

f_2:

X_1	X_2	f(.)
x_{11}	x_{21}	0.3
x_{11}	x_{22}	0.7
x_{12}	x_{21}	0
x_{12}	x_{22}	1

f_4:

Y_1	Y_2	f(.)
y_{11}	y_{21}	0.7
y_{11}	y_{22}	0.3
y_{12}	y_{21}	0.1
y_{12}	y_{22}	0.9

f_6:

Z_1	Z_2	f(.)
z_{11}	z_{21}	0.1
z_{11}	z_{22}	0.9
z_{12}	z_{21}	0
z_{12}	z_{22}	1

(b) **Query: q():- X(A,B), Y(B,C), Z(C,D)** **Lineage: $(x_1 \wedge y_1 \wedge z_1) \vee (x_2 \wedge y_2 \wedge z_2)$**

Figure 6.2: (a) A probabilistic database where the uncertainty is represented with factors. (b) The lineage corresponding to a conjunctive query.

with x_1 and x_2 respectively. The factors are stored separately. In this example, we have six factors, $f_1, \cdots, f_6$. The random variables over which they are defined are stored in a separate table (called *factor*). As an example, we have two factors containing variable x_1, namely, f_1 and f_2, the latter of which is a joint factor over x_1 and x_2. The joint probability distribution over all variables is defined as:

$$\Phi = f_1(X_1)f_2(X_1, X_2)f_3(Y_1)f_4(Y_1, Y_2)f_6(Z_1)f_6(Z_1, Z_2).$$

Querying a probabilistic database. Executing an SQL query over a probabilistic database efficiently has been a subject of much research over the last decade in the database community. The approaches can roughly be divided into *extensional approaches* and *intensional approaches*. In an extensional approach, the query evaluation process is guided solely by the query expression, and query operators are extended to directly compute the corresponding probabilities. For example, the probability of a join result tuple $s \bowtie t$ is the product of the probabilities of the tuples s and t. When such extensional evaluation is possible, the query can be evaluated in polynomial time, hence, much research has focused on characterizing datasets, queries, and query plans for which extensional methods can be correctly applied [32, 115, 81, 116]. On the other hand, in an intensional approach, the intermediate tuples generated during query execution and the final result tuples are associated with propositional symbolic formulas (often called *lineage expressions*) over a subset of the random variables corresponding to the base input tuples. One of several general purpose inference algorithms can then be used to compute the result tuple probabilities, either exactly, e.g., using Shannon expansion [115], variable elimination [145], etc., or approximately [91, 117, 132], depending on the complexity of the lineage expression and the uncertainty model.

With our focus on correlated databases, we are restricted to using an intensional approach. In intensional methods the relational operators are extended to build a Boolean formula (called *lineage*) for each intermediate tuple and each result tuple generated during query evaluation (Figure 6.2(b)). The marginal probability of a result tuple can now be obtained by computing the probability of the corresponding Boolean formula evaluating to **true**, which is #P-hard.

Next, we review some of the intensional query evaluation techniques that have been proposed in the literature.

Variable Elimination (VE)-based Approach. In the VE-based approach [143], instead of constructing a lineage formula for each result tuple, we construct an equivalent rep-

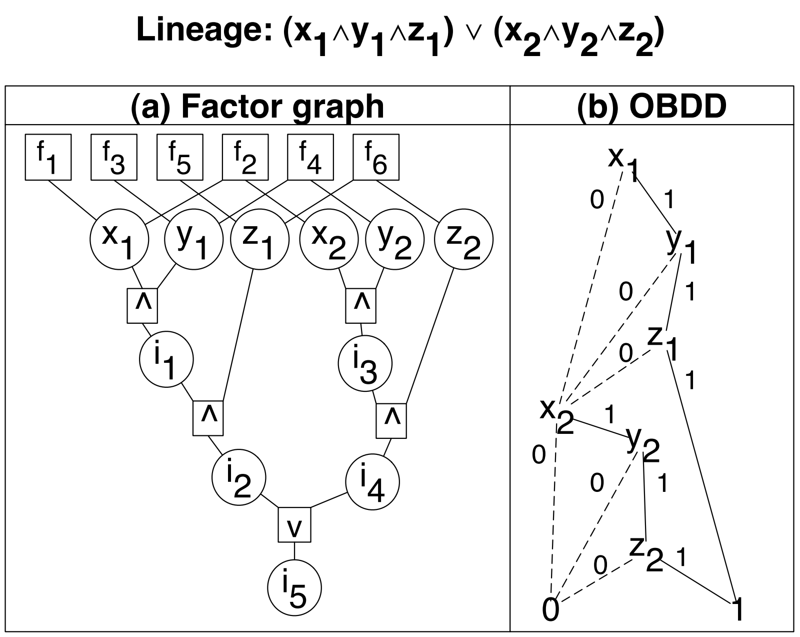


Figure 6.3: A query over the probabilistic database in Figure 6.2. (a) The factor graph for the query with AND and OR factors. (b) The OBDD for the lineage of the result tuple when the random variables are independent.

resentation as a factor graph where each intermediate tuple is explicitly represented (see Figure 6.3(a)). For each intermediate tuple, we add an appropriate factor containing the tuple and the tuples that generated it. For instance, for tuple i_1 generated by joining tuples x_1 and y_1, we introduce an AND factor that captures the logical constraint that i_1 is **true** iff x_1 and y_1 are both **true**. Similarly, for tuple i_5, we add an OR factor capturing the logical constraint that i_5 is **true** if either i_2 or i_4 is **true** (corresponding to a *project* operation). Query evaluation is now equivalent to performing inference on this factor graph to compute the marginal probability distribution of i_5.

Variable elimination [38] is a simple and widely-used technique for performing inference over factor graphs. In essence, VE operates by eliminating one variable at a time from the factor graph until we are only left with the variable of interest (in this case,

i_5). For this purpose, a *variable ordering* needs to be chosen *a priori* to specify the order in which to eliminate the variables. At each iteration two operations are performed: Let X be the variable under consideration. All factors that refer to X are *multiplied* to get a new factor f' and then X is *summed out* of f' to get a factor f'' with no reference to X. As an example, if we choose to eliminate x_1 in the first step, we would multiply the factors f_1, f_2, and the AND factor on x_1, x_2, i_1 to get a factor on x_1, y_1, i_1, x_2, and sum-out x_1 to get a new factor on y_1, i_1, x_2. The complexity of the inference procedure is exponential in the size of the largest factor (measured as the number of variables) created during the process, which is at least the *treewidth* of the factor graph. However, finding the optimal variable ordering is NP-hard and heuristics are typically used.

Sen et al. [144] introduced a *lifted* inference technique that exploits the symmetry in the probabilistic database to reduce the complexity of query evaluation. Our work is orthogonal to their proposal of exploiting symmetry, and it is an interesting future direction to see how these two can be combined.

OBDD-based approach. In a different approach, Olteanu et al. [115] focus on tuple-independent databases and explore the connection between *ordered binary decision diagrams* (OBDDs) [19] and query evaluation for a large class of queries ranging from conjunctive queries with safe plans, to hard queries on restricted databases. OBDDs are rooted, directed acyclic graphs that compactly represent Boolean formulas. They consist of decision nodes and two terminal nodes, called 0-terminal and 1-terminal. Each decision node is labeled with a Boolean variable and has two children, one for each instantiation of the corresponding Boolean variable. Dashed edges represent the assignment of the variable to **false**, while solid edges represent the assignment to **true**. Finally, the two terminal nodes represent the value of the Boolean formula for a particular variable assignment defined by a path from the root node to that terminal node. In the worst case, an OBDD may be a complete binary tree with exponential size, but since it can exploit the structure of the Boolean formula, it is typically much more compact. Figure 6.3(b) shows the OBDD corresponding to the lineage formula in Figure 6.2(b).

Under the tuple-independence assumption, given the OBDD of a lineage formula, each edge can be annotated with the probability of its source decision node taking the corresponding value. The probability of any non-terminal node is computed as the sum over the probabilities of its children, weighted by their corresponding edge probabilities. One can, therefore, compute the probability that the lineage formula evaluates to true by traversing all bottom-up paths from the 1-terminal node to the root, multiplying the probabilities along the way, and then summing the products. This can be done in time linear in the size of the OBDD, hence, when the lineage formula results in an OBDD of polynomial size, the query can be evaluated efficiently.

However not all Boolean formulas admit an OBDD of polynomial size. In fact, OBDD construction is also driven by a variable ordering which dictates the order in which the variables are evaluated and corresponds to the top-down order of decision nodes in the final OBDD. Choosing the optimal variable ordering is NP-hard. A comprehensive review of different construction techniques is presented by Mantadelis et al. [104].

Discussion. All the approaches mentioned above present significant limitations in presence of correlations and local structure. Factor graphs do not exploit the local structure of the factors to reduce the complexity of inference. Moreover, OBDDs are applicable only under the tuple-independence assumption. In the next section we present an approach that combines the representational power of factor graphs with the compactness of decomposition methods leading to more efficient query evaluation over correlated databases.

6.2.2 Arithmetic Circuits

In this section we briefly review how context-specific independence and determinism can be exploited to enable efficient exact inference even in factor graphs with high treewidth, through use of arithmetic circuits [24, 25, 26, 27]. Context-specific independence is prevalent in relational domains, since the underlying structure introduces regularities and conditional independencies that are true only under specific contexts. Furthermore, determinism appears during query evaluation, where every relational operator introduces deterministic constraints over its input tuples, e.g., both input tuples of a join must exist in order for the intermediate tuple to exist. One can also consider the constraints introduced by foreign keys as another source of deterministic correlations. Exploiting such determinism is important for improving the efficiency of probabilistic query processing.

Let $\Phi(\cdot)$ be the joint distribution over a set of random variables $\mathcal{X}$ defined by a factor-graph. We associate Φ with a unique multi-linear function (MLF) [35] over two types of variables:

- *Evidence indicators:* For each random variable $Y \in \mathcal{X}$ with dom$(Y) = \{y_1, \cdots, y_n\}$, we have a set of evidence indicators:
  $\{\lambda_{y_1}, \lambda_{y_2}, \ldots, \lambda_{y_n}\}$, i.e., one evidence indicator for each y_i.

- *Factor parameters:* For each factor f over a set of random variables $\mathbf{X}$, we have a set of parameters $\theta_{\mathbf{X}=\mathbf{x}}$.

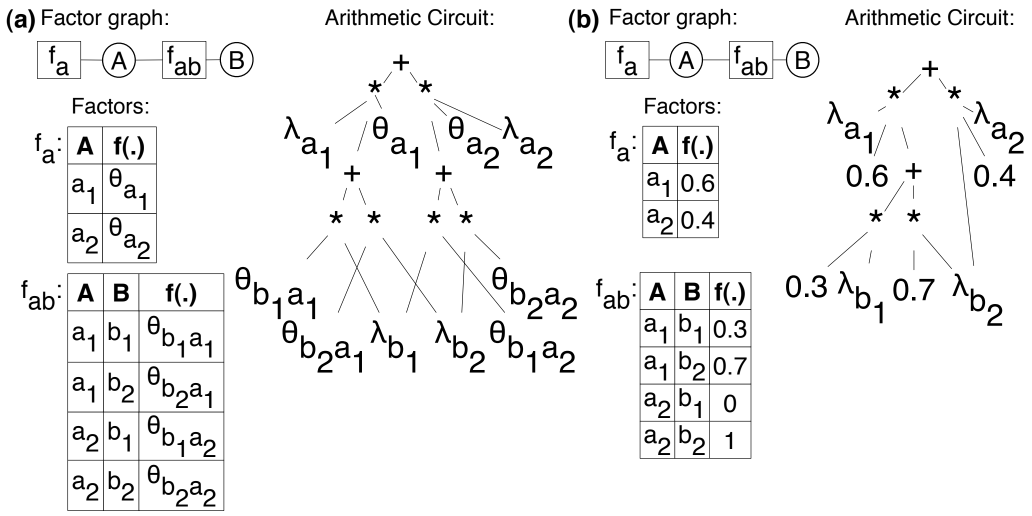
For any unobserved random variable, i.e., a random variable whose value is not fixed, all the evidence indicators are set to 1. When a particular value is assigned to a random variable, the indicator corresponding to that value is set to 1 and all other indicators is set to 0. The factor parameters $\theta_{\mathbf{X}=\mathbf{x}}$ correspond to the actual values in the factors of the factor graph. The MLF for a factor graph has an exponential number of terms, i.e., one term for each joint assignment of the random variables in $\mathcal{X}$. For example, the factor graph in Figure 6.4(a), with only binary random variables, induces the following MLF:

$$\lambda_{a_1}\lambda_{b_1}\theta_{a_1}\theta_{b_1a_1} + \lambda_{a_1}\lambda_{b_2}\theta_{a_1}\theta_{b_2a_1}+$$

$$\lambda_{a_2}\lambda_{b_2}\theta_{a_2}\theta_{b_2a_2} + \lambda_{a_2}\lambda_{b_2}\theta_{a_2}\theta_{b_2a_2}$$

Given the MLF for a factor graph, we can compute the probability of evidence, denoted by $\Pr(e)$, i.e., the probability of a specific joint assignment of all (or of a subset of) the random variables, by setting the appropriate evidence indicators to 0 instead of 1 and evaluating the MLF. While the MLF has exponential size, if we can factor it into something small enough to fit within memory, then we can compute $\Pr(e)$ in time linear in the size of the factorization. The factorization will take the form of an arithmetic circuit [36]. More rigorously we have the following definition.

Definition 9. An *arithmetic circuit* (AC) over variables Σ is a rooted, directed acyclic graph whose leaf nodes are either numeric constants or evidence indicators, internal nodes


(a) Factor graph:
f_a — A — f_{ab} — B
Factors:
Arithmetic Circuit:
λ_{a_1} θ_{a_1} θ_{a_2} λ_{a_2}
$\theta_{b_1a_1}$ $\theta_{b_2a_1}$ λ_{b_1} λ_{b_2} $\theta_{b_1a_2}$ $\theta_{b_2a_2}$
(b) Factor graph:
Factors:
Arithmetic Circuit:
λ_{a_1} λ_{a_2} 0.6 0.4 0.3 λ_{b_1} 0.7 λ_{b_2}

(a) f_a:

A	f(.)
a_1	θ_{a_1}
a_2	θ_{a_2}

(a) f_{ab}:

A	B	f(.)
a_1	b_1	$\theta_{b_1a_1}$
a_1	b_2	$\theta_{b_2a_1}$
a_2	b_1	$\theta_{b_1a_2}$
a_2	b_2	$\theta_{b_2a_2}$

(b) f_a:

A	f(.)
a_1	0.6
a_2	0.4

(b) f_{ab}:

A	B	f(.)
a_1	b_1	0.3
a_1	b_2	0.7
a_2	b_1	0
a_2	b_2	1

Figure 6.4: ACs and their factor graphs. Although ACs are DAGs, the directions on the edges are not explicitly drawn. (a) Assuming no local structure, the size of the AC is exponential in the number of variables. (b) Exploiting determinism (i.e., $Pr(A = a_2, B = b_1) = 0$) leads to an AC of smaller size.

correspond to product and sum operations, and the root node corresponds to the circuit's output. The size of the arithmetic circuit is defined to be the number of its edges.

We elaborate more on the connection between ACs and variable elimination. As mentioned earlier, VE is an algorithm that acts on a set of factors and, driven by a variable ordering, performs two operations at each iteration: First, factors that contain a particular variable are multiplied to create a new factor and, then, that variable is summed out of that factor. An arithmetic circuit can be viewed as the trace of the VE process for a particular factor graph [36].

We refer to the process of producing an AC from a factor graph as *compilation*. One way to do this is to represent the joint distribution by a propositional logical formula in tractable logical form, known as *deterministic, decomposable negation normal form* (d-DNNF), which is then mapped to an AC [34]. Other approaches are based on decision diagrams, and we discuss them in Section 6.4.2. Jha and Suciu [82] show that d-DNNF is a tractable logical form which subsumes decision diagrams. Therefore, arithmetic circuits can be viewed as a generalization of the decision diagrams used in the intensional probabilistic inference methods presented earlier.

A probability of evidence query is computed by assigning appropriate values to the evidence-indicator nodes and evaluating the circuit in a bottom-up fashion to compute the value of the root. For example, using the AC in Figure 6.4(b) we can compute the probability of evidence $\Pr(A = a_1, B = b_1)$ by first setting $\lambda_{a_1} = 1, \lambda_{a_2} = 0, \lambda_{b_1} = 1, \lambda_{b_2} = 0$, and then traversing and evaluating the circuit. This process may be repeated for as many probability of evidence queries as desired and it is only linear in the size of the AC. The size of an AC is in the worst case exponential in the treewidth of the factor graph. However, if local structure is present, the size of an AC is often significantly smaller. Figure 6.4(b) shows one example where the factor value for the joint assignment $A = a_2, B = b_1$ is set to 0, hence, the corresponding sub-circuit is pruned, resulting in a much smaller AC.

6.3 Arithmetic circuits in Probabilistic Databases

In this section we discuss how arithmetic circuits can be used in correlated probabilistic databases. We begin by discussing a naive approach that uses ACs for inference alone, discuss its limitations, and then present an overview of our proposed approach.

6.3.1 Naive Approach

Let D denote a probabilistic database and $\mathcal{P}$ the factor graph representing the correlations among the stored data. Consider a query Q against D. Following the factor graph approach (Figure 6.3(a)), we can construct a new (augmented) factor graph $\mathcal{P}'$ for Q on which inference needs to be performed. Compiling $\mathcal{P}'$ into an arithmetic circuit results in a compact representation of the VE process due to the deterministic intermediate factors introduced. Inference can be performed by parsing the circuit to compute the result probabilities. However, although the inference time in ACs is low, compilation time can be quite expensive. Furthermore, for each different query, the corresponding augmented factor graph needs to be compiled into a new AC. Therefore, such an approach is not a viable means for evaluating queries against probabilistic databases.

Ideally, we would like to avoid repeatedly compiling the base arithmetic circuit, AC_P, from the database. Instead, it would be desirable that we construct AC_P offline once, and save it in the database. Then, for a given query Q, we can construct a new arithmetic circuit, AC_Q, and somehow "merge" the two ACs to get a single AC for computing the query result. However, arithmetic circuits do not support online updates. This significant limitation arises because only the leaf nodes of the circuit provide information about the random variables present in the factor graph through the corresponding evidence indicators. The internal nodes do not have enough information to determine which variables participate in a particular operation. Therefore, it is impossible to merge arithmetic circuits into a unified variable elimination trace, which takes into account all the corresponding correlations.

6.3.2 Overview of the Proposed Framework

We begin with a brief overview of our proposed query evaluation framework. We elaborate on the steps in the next two sections. We also introduce the running example that we will be using.

Phase 1 - Preprocessing: We assume that a probabilistic database D and the factor graph $\mathcal{P}$ representing the correlations among the stored tuples are given as input. The factors in $\mathcal{P}$ are represented using ADDs to capture the local structure. Offline, we compile the given probabilistic database into an *annotated arithmetic circuit* (AAC), an extended version of an AC where sum nodes are annotated with the variable on which the summation is performed. The AACs corresponding to the probabilistic database shown in Figure 6.2 are shown in Figure 6.5. As depicted, we do not have a single AAC for the entire network. Instead, we require that disconnected parts of the factor graph referring to independent sets of variables correspond to separate AACs. An immediate consequence of this is that a random variable can be present only in one AAC. Maintaining a collection of AACs, denoted by A_{Col}, instead of a single AAC, allows for indexing the AACs, thereby, offering more flexibility while merging them.

Phase 2 - Lineage Processing: Given a query Q, we compute a lineage formula for each result tuple using standard techniques.

Phase 3 - Query Evaluation: During this phase we iterate through the result tuples of the given query Q. For each tuple we perform the following steps:

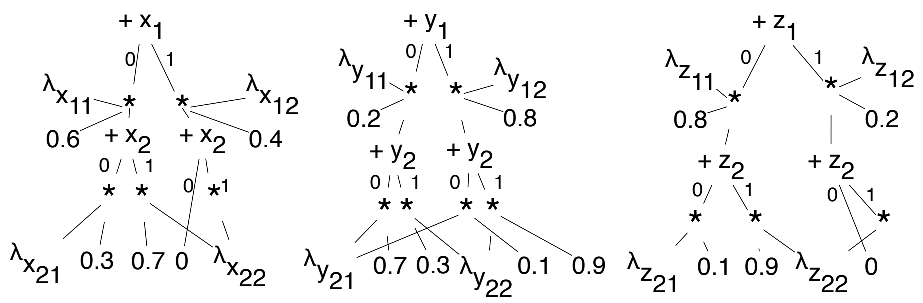


Figure 6.5: The (complete) AACs for the probabilistic database in Figure 6.2.

(a) The lineage formula introduces a set of new deterministic correlations among the random variables present in it. Specifically, the new correlations describe the logical constraints under which the lineage formula evaluates to 1. Then lineage is compiled into a new AAC (called a *lineage-AAC*) that captures the constraints and represents them in a compact way.

(b) The lineage-AAC is merged with the collection of all the AACs that refer to variables present in the lineage-AAC.

(c) The result tuple probability is computed by traversing the resulting merged AAC.

We define and describe AACs in Section 6.4.1, and describe algorithms to compile the database factor graph and the lineage formula into AACs in Sections 6.4.2 and 6.4.3. We then present our merging algorithm for combining multiple AACs in Section 6.5.

6.4 Annotated Arithmetic Circuits

In this section we introduce annotated arithmetic circuits and show how a correlated probabilistic database can be compiled into a collection of AACs. We also introduce a novel algorithm for representing lineage formulas as AACs.

6.4.1 Definitions

An annotated arithmetic circuit is a generalization of an arithmetic circuit that includes full information on the sequence of arithmetic operations performed during inference and the variables that participate in them. We maintain this information by adding variable annotations to the internal operation nodes. We have the following definition:

Definition 10. An *annotated arithmetic circuit* (AAC) over variables Σ is a rooted, directed acyclic graph whose leaf nodes are either numeric constants or evidence indicators, internal nodes correspond to product and sum operations, and the root node corresponds to the circuit's output. Each sum node is *annotated* with the corresponding variable s_i that is being summed out and its outgoing edges are annotated with the values of the different instantiations of variable s_i. The size of the annotated arithmetic circuit is defined to be the number of its edges.

AACs inherit their representational power from regular ACs, and therefore, can capture the local structure and the conditional independences present in a network. Moreover, variable annotations provide the necessary information to detect the exact order of operations performed during variable elimination, as we discuss in the next section. This enables us to detect and directly update the corresponding parts of the circuit when new correlations are introduced and thus plays a key role when merging different arithmetic circuits. To guarantee the correctness of the merging algorithm presented in Section 6.5, we require that the circuit be a *complete trace* of the variable elimination algorithm.

Definition 11. An AAC over variables Σ is a *complete* trace of variable elimination over variables in Σ, if each evidence-indicator $\lambda_{X=x}$ for a random variable X is preceded by a sum node annotated with variable X and none of its sibling nodes is an evidence-indicator.

When an AAC is a complete trace of VE, sum and product nodes appear in an interleaving manner. Every sum node will be followed by a product node and every operation child of a product node will be a sum node. From now on, we will assume that all AACs correspond to complete traces, and we will drop the qualification.

Examples of AACs corresponding to complete traces of VE are shown in Figure 6.5. As we can see the annotations in the sum nodes allow us to trace the exact variable that is being summed out. Furthermore if we compare the first AAC presented in Figure 6.5 with the arithmetic circuit shown in Figure 6.4, we have that both factor graphs present the same pattern of determinism, however the AAC has an extra sum node in the sub-circuit that expresses the deterministic conditional probabilities. This is necessary because the presented AAC keeps full trace of the VE process. Moreover, when a product node has constant 1 as child, we drop it but the product node itself is kept since it is needed during merging. We note that a complete AAC exploits local structure as a regular AC does, and presents only a small increase in size compared to a regular AC.

6.4.2 Compiling Factor Graphs into AACs

We compile a factor graph into the corresponding collection of AACs by extending a compilation algorithm introduced by Chavira [25], which is based on variable elimination and *algebraic decision diagrams* (ADDs) [8]. Analogous to how a BDD is a representation of a Boolean function, an ADD is a graph representation of a function that maps instantiations of Boolean variables to real numbers. Our algorithm to compile factor graphs into AACs extends the algorithm described in [25] by adding variable annotations and merging contiguous product nodes, outputting a complete AAC. For brevity, we omit a detailed description here.

However, one important issue that warrants discussion here is the variable ordering used to generate the AACs. Similar to variable elimination and the OBDD construction algorithm, we need to choose an ordering of the variables to compile the database into AACs using the above procedure. Let Π denote the total ordering over all variables that is used to generate the AACs, and let Π_{Col} denote a collection of partial orderings over the disjoint sets of variables corresponding to the different AACs in A_{Col}.

As we will see in the next section, the AAC corresponding to the lineage of a query result tuple must respect all of these partial orderings. This crucial constraint imposed by the AAC merging algorithm means that we cannot use standard algorithms for constructing an AAC from a lineage expression.

6.4.3 Compiling Lineage Formulas into AACs

Lineage can be represented as a factor graph (Figure 6.3(a)), and we can use the above algorithm to construct an AAC for it. However, the lineage corresponds to a factor graph limited to consist of only two deterministic factors (AND and OR). Hence, we can employ more efficient techniques based on OBDD construction.

Recall that an OBDD corresponding to a lineage formula is a compact decision diagram representing the set of constraints over the instantiations of the random variables under which the lineage formula evaluates to true (Section 6.2), and can be constructed by choosing a ordering of the variables in which the variables are evaluated. As discussed above, the variable ordering we choose must respect all the partial orderings in Π_{Col}. However, none of the standard order selection algorithms can be used for our purpose, as they do not take into account the ordering constraints.

Constructing an AAC from an OBDD. The easiest way to construct an AAC for a given lineage formula is to first construct an OBDD, and then modify it by adding the necessary annotated operation nodes with the appropriate indicator constants. Figure 6.6(b) depicts the OBDD and the AAC (Figure 6.6(c)) corresponding to a conjunctive query, executed against the probabilistic database shown in Figure 6.2. The query generates a single Boolean formula. There is a one-to-one mapping between the OBDD and the corresponding AAC; in particular, each decision node is converted into a sum node, and expanded to add a product node and an appropriate evidence indicator. We see that the AAC represents the deterministic correlations introduced by the query. During the lineage-AAC construction we ignore the correlations among the random variables.

Choosing a variable order for the lineage OBDD. The order in which the variables are evaluated in an OBDD plays a crucial role in determining its size. Given a good variable ordering, the size of the OBDD can be polynomial in the size of the corresponding Boolean expression. In the general case, finding the optimal variable ordering for a given Boolean formula is NP-hard [16]. OBDDs have been extensively used in VLSI design

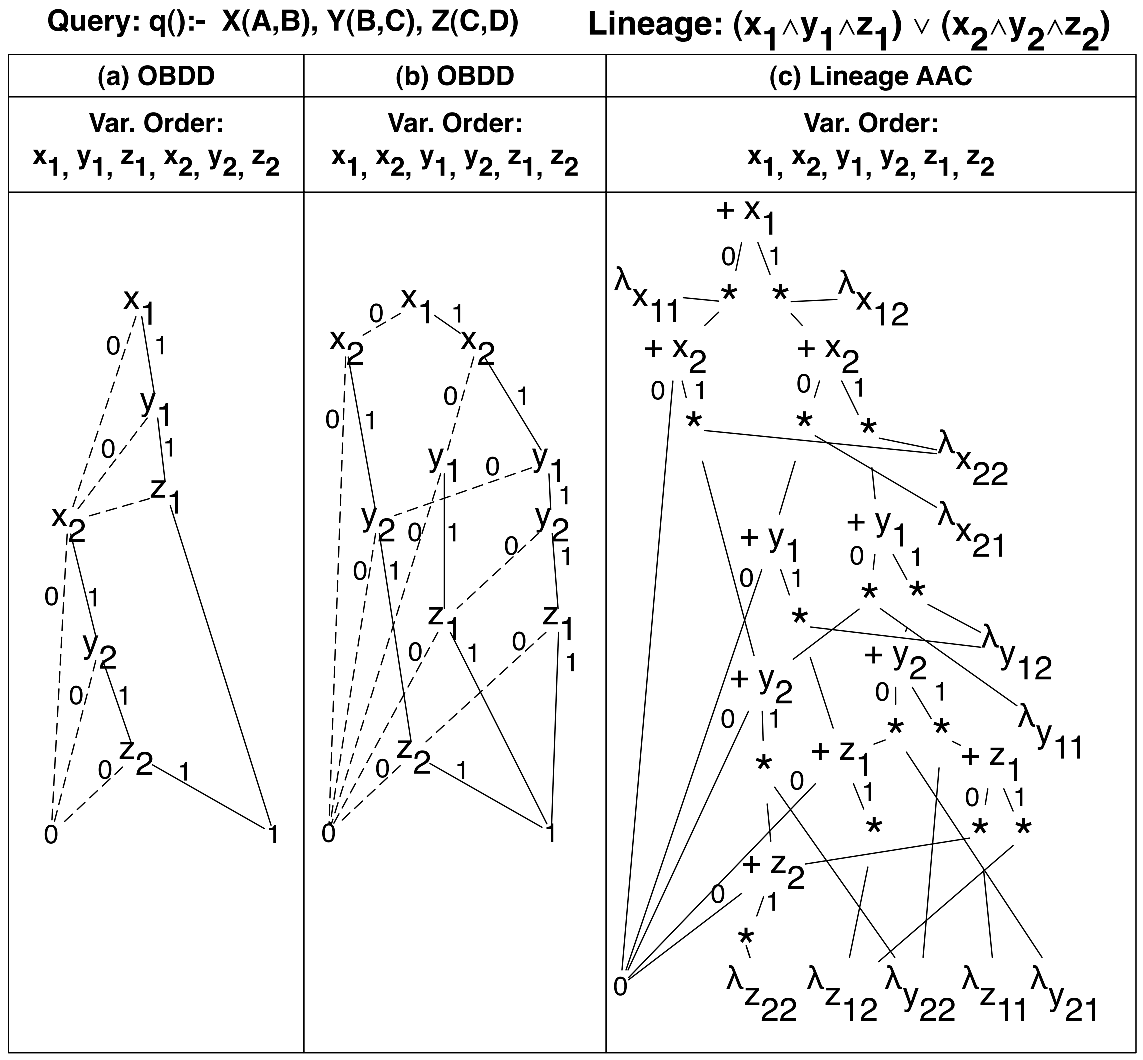

Query: q():- X(A,B), Y(B,C), Z(C,D)
Lineage: $(x_1 \wedge y_1 \wedge z_1) \vee (x_2 \wedge y_2 \wedge z_2)$
(a) OBDD
(b) OBDD
(c) Lineage AAC
Var. Order: $x_1, y_1, z_1, x_2, y_2, z_2$
Var. Order: $x_1, x_2, y_1, y_2, z_1, z_2$
Var. Order: $x_1, x_2, y_1, y_2, z_1, z_2$
$\lambda_{x_{11}}$
$\lambda_{x_{12}}$
$\lambda_{x_{22}}$
$\lambda_{x_{21}}$
$\lambda_{y_{12}}$
$\lambda_{y_{11}}$
$\lambda_{z_{22}}$
$\lambda_{z_{12}}$
$\lambda_{y_{22}}$
$\lambda_{z_{11}}$
$\lambda_{y_{21}}$

Figure 6.6: (a) The OBDD for the given lineage formula when the variable ordering is generated by a constraint oblivious heuristic. (b,c) An example of the lineage AAC corresponding to the given query when executed against the database shown in Figure 6.2. The new constraint aware heuristic was used to generate the variable ordering.

and many heuristics that give good variable orderings have been proposed in the corresponding literature [49]. In particular, there are two main guidelines that the proposed heuristics satisfy:

(a) Input variables of a Boolean formula that are connected should appear together in the ordering. For example, consider a CNF formula where we fix the values of a set of variables that belong to the same clause so that the clause evaluates to zero. The entire formula will then evaluate to zero.

(b) Input variables with higher fan-out should appear sooner in the variable ordering since they have greater influence in the output of the formula. Fixing the values of influential variables first may lead to fixing the values of larger parts of the Boolean formula. By having those variables together in the OBDD, we can significantly reduce the size of the OBDD.

Another advantage of these requirements is that they facilitate *cach-ing.* During OBDD construction the results of intermediate operations are stored in a cache, following a similar rationale as dynamic programming algorithms where the subproblems of the initial problem are cached. This ensures that all intermediate OBDDs are of polynomial size if the final OBDD is of polynomial size.

Partial order constraints. This brings us to the main challenge in constructing an AAC for the lineage formula. Recall that Π_{Col} can be seen as a list of disjoint orderings, each specifying an order over a disjoint subset of the variables. Let Π_{final} denote the variable ordering used to construct the lineage-AAC. We require that Π_{final} satisfy two constraints, one motivated by the merging algorithm (Section 5), and the second motivated by the desire to enable caching in that phase (cf. Section 5). Specifically we require that:

(c) Π_{final} must respect the partial orderings in Π_{Col}. This is necessary since it enables merging multiple AACs that refer to the same set of variables.

(d) Variables that are present in a constraint must be kept together in Π_{final} to enable caching in the merging phase.

Requirement (c) is mandatory for the correctness of the merging algorithm, therefore, we assume that it is always satisfied. We elaborate more on requirement (d). Heuristics that only take into account requirements (a) and (b) may generate a variable ordering that minimizes the size of the lineage-AAC but will not always give a good variable ordering for the final AAC. In particular, disregarding the partial ordering constraints may lead to a variable ordering that does not enable efficient caching during the merging phase, leading to a final AAC of exponential size. As we show in Section 6.5, caching plays an important role in the performance of our proposed AAC merging algorithm.

Keeping the variables that are present in a constraint together in Π_{final}, enables the detection of parts of the final AAC that refer to disjoint sets of variables, allowing caching of those parts. Interleaving variables present in different partial ordering constraints makes it harder to detect isomorphic sub-graphs of the AAC.

Consider the left part of the OBDD shown in Figure6.6(a). As mentioned earlier, we require that the final (after merging) AAC respect a global variable ordering, specifically, the one shown in the figure. Observe that during merging, variables y_1 and z_1 will be inserted before x_2, y_2, and z_2, as the latter depend on the former. Since x_2 is independent of z_1, the parts of the left sub-AAC that refer to x_2 and appear in the two sub-AACs corresponding to $z_1 = 0$ and $z_1 = 1$, denoted by $A_{z_1=0}$ and $A_{z_1=1}$ respectively, will be the same. However, caching can not be used since $A_{z_1=0}$ and $A_{z_1=1}$ are not isomorphic. The reason is that the sub-circuits corresponding to z_2 which appear at the end of $A_{z_1=0}$ and $A_{z_1=1}$ are different. Finally, we note that the problem of minimizing the size of an OBDD given order constraints over the input variables is NP-hard because the OBDD construction without any such constraints is NP-hard. Next, we develop a heuristic for this problem.

Variable ordering heuristic. Ideally, we would like to find a variable order that satisfies all four requirements listed above. However, the requirements will usually conflict with each other. We introduce a new heuristic algorithm to generate a good variable ordering Π_{final} that will be used during the construction of both the lineage AAC and the final AAC. The new heuristic is shown in Algorithm 6. It takes as input the ordering constraints defined in Π_{Col} and the lineage formula L and it returns the variable ordering Π_{final}. In Algorithm 6, we represent orderings as vectors.

Let us elaborate on the algorithm. First, the Boolean formula L is converted to its corresponding Boolean circuit C. Assuming that connections between the input variables are only introduced because of the lineage formula, the algorithm starts by detecting groups of connected input variables. This directly addresses requirement (a). Let S_g denote the set of groups. Variables contained in a single group should appear together.

Variables are assigned to groups in S_g by traversing circuit C recursively: for each variable node v in C we examine its siblings, i.e., other nodes that are connected with v

via some Boolean operation. If any of these siblings is assigned to a group g, we assign v to g. Otherwise, we create a new group and assign v to it. For each internal operation node o we examine its children nodes. If all of them belong to the same group then we assign o to it, otherwise we assign o to a new one.

The algorithm proceeds by generating an ordering O_g^v among the variables contained in a single group. Variables are ordered in descending order of their fan-out in C. This step complies with requirement (b), stating that more influential variables appear sooner. So far the algorithm is oblivious to the ordering constraints in Π_{Col}. The following steps are introduced to account for them.

Ordering variables within groups only generates partial orderings over the variables. To get a total ordering, it is necessary to impose an ordering O_{Cl} on the groups according to their *position score* (PScore). Motivated by requirement (c), we define the following process: Each input variable is associated with a position score, which is defined to be its position in the corresponding ordering in Π_{Col}. The position score of each group in S_g, is defined to be the average position score of the variables contained in it. The intuition is that variables that appear early in an ordering in Π_{Col} should also appear early in Π_{final}.

Until now the proposed algorithm does not explicitly satisfy requirements (c) and (d), presented above. To address them, the algorithm iterates over the groups and the variables in them. Let v denote the variable under consideration at each step of the iteration. The algorithm finds the ordering constraint Constr_v corresponding to v and appends in the final ordering Π_{final} the set of variables S that contains v and all variables $u \in \mathsf{Constr}_v$ that precede v and are not present in Π_{final}. It is easy to see that both requirements (c) and (d) are satisfied.

In Algorithm 6 we show the append operation with the concatenation symbol $|$. Some of the variables may not be present in the lineage formula but are necessary in the merging phase as they appear before v in the corresponding ACs.

6.5 Merging AACs

In this section, we introduce a new algorithm for merging a lineage-AAC with the corresponding complete AACs in the database. We begin with formally defining the problem.

Algorithm 6 variableOrdering(Π_{Col}: a collection of ordering constraints, L: a lineage formula): returns Variable Ordering

```
1:  $C \leftarrow$ transform $L$ into its corresponding Boolean circuit.
2:  $Var_L \leftarrow$ get the set of variables present in $L$.
3:  $S_g \leftarrow$ Get the set of groups of connected variables for circuit $C$.
4:  for $c \in S_g$ do
5:      $O_g^v \leftarrow$ order the variables in $c$ in a decreasing order with respect to their fan-out in $C$.
6:      Assign a position score PScore$_g$ to group $g$.
7:  $O_{\text{Cl}} \leftarrow$ order the groups in an increasing order by their PScore.
8:  $\Pi_{\text{final}} \leftarrow \{\}$
9:  for $c \in O_{\text{Cl}}$ do
10:     for $v \in O_g^v$ do
11:         if $v \notin \Pi_{\text{final}}$ then
12:             $\mathsf{Constr}_v \leftarrow$ get the constraint for $v$ from $\Pi_{\text{Col}}$
13:             $u \leftarrow arg \max_{w \in \mathsf{Constr}_v \cap L}$ (Position of $w$ in $\mathsf{Constr}_v$)
14:             $S \leftarrow \bigcup_{w \in \mathsf{Constr}_v, w \preceq u} w$
15:             $\Pi_{\text{final}} \leftarrow \Pi_{\text{final}} \mid S$
16: return $\Pi_{\text{final}}$
```

The merging problem. Let A_{Col} denote the collection of AACs produced after the compilation of the database, Π_{Col} the collection of partial orderings over the random variables in the database, and Π_{final} the variable ordering generated by the heuristic algorithm presented in the previous section. The merging algorithm takes as input A_{Col} and the lineage-AAC and generates an AAC, which is used to evaluate the probability of the lineage.

The core idea of the algorithm can be simply stated: we traverse the lineage-AAC and all the appropriate database-AACs, i.e., AACs that refer to the variables present in the lineage-AAC, simultaneously by keeping one or more cursors over each of them. At any point, the algorithm considers exactly two AAC nodes, a node from the lineage-AAC and a corresponding node from a database-AAC, and tries to merge them. Since the database-AACs refer to disjoint sets of random variables, a node in the lineage-AAC can only correspond to one database-AAC node. We check whether we can compute the result of the merge operation for the two nodes immediately, otherwise we choose a variable to branch on and recursively perform the merge operation for each instantiation of the variable. The variable is chosen according to Π_{final}. In the remainder of the section, we elaborate on these steps.

Path annotations. When traversing the input AACs, it is important to be able to identify in which path of an AAC a variable appears, since traversing redundant paths will significantly deteriorate performance. In general, a product node can have multiple sum nodes as children. This can happen when two or more variables are conditionally independent given the value of a particular variable. To address this issue, we introduce a new annotation for each variable, which we call the *path annotation* of the variable.

Path annotations are set according to the following process: we start by assigning a path annotation of 0 to the root of each AAC in A_{Col} and then we traverse each AAC in a depth-first manner. If a product node has multiple sum children we extend the path annotation with the count information of each child. Consider for example a product node with two sum nodes as children and a path annotation 0. The annotations of its children will be 0::1 and 0::2. Path annotations are created offline during the compilation phase. The order in which sum nodes appear in the set of children of a product node is determined by the corresponding ordering Π_{Col}. Thus, for different instantiations of the predecessor variables the children of a product node appear in the same order.

Traversing multiple AACs simultaneously. The merging algorithm traverses all the AACs in a breadth-first or a depth-first manner, by keeping multiple cursors at different sum nodes, and recursively traversing down the children of an appropriate product node. If it is traversing down a product node that has two or more sum children, then multiple cursors are generated pointing to those different sum nodes. At any point, we may have at most as many cursors as the number of variables in the AAC. A key requirement here is to be able to identify first which database-AAC contains a particular variable, and then, along which path the variable may be found in that AAC. We use a simple index for the first purpose, whereas the path annotations are used for the second purpose. For example, let the considered variable have a path annotation of 0::1::1, and let the cursors in the corresponding database-AAC point to 0::1 and 0::2. By comparing the prefixes, we can deduce that the variable will be found under the former cursor.

Merging AACs. We now introduce the merging algorithm (shown in Algorithm 7). The multi-merge operation is implemented recursively, reducing the operation of merging the lineage-AAC with the appropriate AACs, into operations over smaller AACs, until we reach boundary conditions: AACs that correspond to constant nodes.

Let a_1 and v_1 denote the root node of the input lineage-AAC and the variable that is associated with it. The algorithm starts by finding the database-AAC A_d that needs to be merged and selects the appropriate cursor of A_d (using the procedure described above, and encapsulated in the function $\mathsf{getAACCursor}()$). Let a_2 denote the sum node that the selected cursor points to.

If the algorithm is given a trivial input, i.e., a lineage-AAC equal to 0 or 1, or if the result of the multi-merge between a_1 and a_2 is present in the cache, the multi-merge operation terminates immediately. Otherwise we must recursively compute the result of the operation for a_1 and a_2. Recall that v_1 is the variable that corresponds to a_1 and let v_2 be the variable that corresponds to a_2. Because all variables present in the lineage-AAC are already present in an AAC in A_{Col} and all AACs respect Π_{final}, there are only two cases that the algorithm needs to consider: (a) $v_2 \prec v_1$ and (b) $v_2 = v_1$. We also note that in each turn the merging algorithm expands two contiguous levels of the AACs under consideration, exploiting that in a complete AAC sum and product nodes appear in turns. The algorithm proceeds as described below.

Case - Same variables. When both sum nodes refer to the same variable v_1, the merge operation outputs a sum node a annotated with v_1. To construct the children of a the algorithm iterates through all values v in the domain of v_1 and performs the following process: let c and c' denote the child node of a_1 and a_2 respectively that correspond to $v_1 = v$. The algorithm checks for the following terminal cases: (1) if either c or c' are 0 it outputs 0 and (2) if c and c' are indicator constants it outputs c'. If none of the terminal cases are met, then it outputs a new product node c_1 with children as the constant children of c. Subsequently the merge operation is propagated by: (1) traversing both input AACs and updating the cursors of the database-AAC appropriately, (2) considering

Algorithm 7 multiMerge(a_1: lineage AAC, a_{col}: AAC Collection): returns AAC

```
1: $a_2 \leftarrow \mathsf{getAACCursor}(\mathsf{Var}(a_1))$
2: if $\mathsf{cache}(a_1, a_2) \neq null$ then
3:     return $\mathsf{cache}(a_1, a_2)$
4: else if $(a_1 == 0)$ or $(a_1 == 1)$ then
5:     return $a_1$
6: if $(\mathsf{Pos}(a_2) < \mathsf{Pos}(a_1))$ then
7:     $a \leftarrow$ new $+$ node; $\mathsf{Var}(a) \leftarrow \mathsf{Var}(a_2)$
8:     for $v \in \mathsf{Values}(\mathsf{Var}(a_2))$ do
9:         $c \leftarrow \mathsf{getChild}(a_2, v)$
10:        //c is either a $*$ node or a constant
11:        if $(c == 0)$ or ($c$ is indicator constant) then
12:            $c_1 \leftarrow c$
13:        else
14:            $c_1 \leftarrow$ new $*$ node
15:            $\mathsf{Const}(c_1) \leftarrow \mathsf{Const}(c)$
16:            $\mathsf{moveCursorToChild}(a_2, v)$
17:            $c_2 \leftarrow \mathsf{multiMerge}(a_1, a_{col})$; $\mathsf{addChild}(c_1, c_2)$
18:            $\mathsf{moveCursorToParent}(a_2, v)$
19:        $\mathsf{addChild}(a, c_1, v)$
20: else if $(\mathsf{Pos}(a_2) == \mathsf{Pos}(a_1))$ then
21:     $a \leftarrow$ new $+$ node; $\mathsf{Var}(a) \leftarrow \mathsf{Var}(a_1)$
22:     for $v \in \mathsf{Values}(\mathsf{Var}(a_1))$ do
23:         $c \leftarrow \mathsf{getChild}(a_1, v)$; $c' \leftarrow \mathsf{getChild}(a_2, v)$
24:         //c and c' are either $*$ nodes or constants
25:         if $(c == 0)$ or $(c' == 0)$ then
26:             $c_1 \leftarrow 0$
27:         else if ($c$ and $c'$ are indicator constants) then
28:             $c_1 \leftarrow c'$
29:         else
30:             $c_1 \leftarrow$ new $*$ node
31:             $\mathsf{Const}(c_1) \leftarrow \mathsf{Const}(c')$
32:             $\mathsf{moveCursorToChild}(a_2, v)$
33:             if ($c$ has a + node in its children) then
34:                 $c \leftarrow \mathsf{getChild}(c)$
35:                 $c_2 \leftarrow \mathsf{multiMerge}(c, a_{col})$; $\mathsf{addChild}(c_1, c_2)$
36:             $\mathsf{moveCursorToParent}(a_2, v)$
37:         $\mathsf{addChild}(a, c_1)$
38: $\mathsf{cacheInsert}(a, a_1, a_2)$
39: return $a$
```

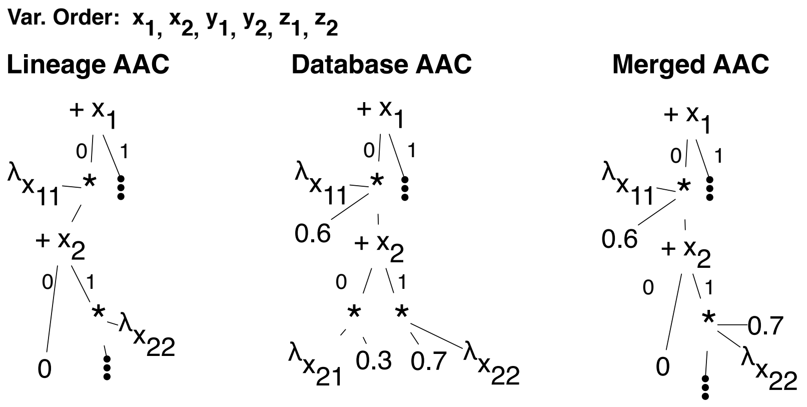


Figure 6.7: Partial AAC produced after merging the lineage-AAC with the corresponding database-AAC.

the descendant sum nodes of a_1 and a_2 and (3) recursively merging them. Finally, the result of the merge operation between a_1 and a_2 is cached.

Figure 6.7 depicts the merging operation as applied to the database-AACs in Figure 6.5 and the lineage-AAC in Figure 6.6(c) using the variable order $X_1, X_2, Y_1, Y_2, Z_1, Z_2$. The first step is to merge the sum nodes that are annotated with X_1 and create a new sum node with the same annotation that contains both 0.6 and $\lambda_{x_{11}}$ in its children nodes. As shown the algorithm continues in a depth-first manner and considers the sum node present in the left sub-AAC.

Case - Different variables. When the merge operation is applied to sum nodes with variables $v_2 \prec v_1$ in Π_{final} the output node is a sum node a annotated with v_2. To construct the children of node a the algorithm iterates through *each* child c of node a_2 corresponding to a particular instantiation of v_2 and performs the following process: if c is a constant it outputs c, otherwise it outputs a new product node c_1 with children the constant children of c. Subsequently, the merge operation is propagated by: (1) traversing the database-AAC and updating its cursors appropriately, (2) considering the descendant sum nodes of

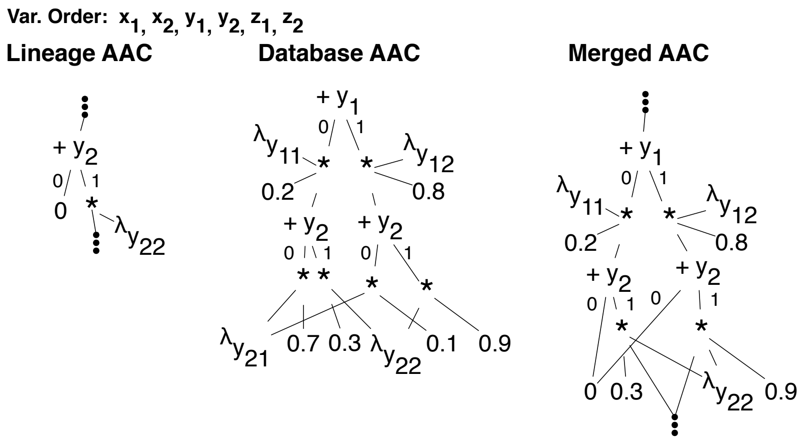


Figure 6.8: A subsequent step of the merging process shown in Figure 6.7. Note that the sum nodes to be merged in this step refer to different variables, namely Y_2 and Y_1.

a_2, and (3) recursively merging them with a_1. Finally, the result of the merge operation between a_1 and a_2 is stored in the cache. Figure 6.8 depicts a subsequent step of the merging process shown in Figure 6.7. The merging operation is performed between two different variables, namely Y_2 and Y_1. Since $Y_1 \prec Y_2$ in the variable order, the output is a copy of the sum node, annotated with Y_1, in the database-AAC. The algorithm proceeds recursively to merge the sum nodes that are annotated with variable Y_2.

In the algorithm, method $\mathsf{getAACCursor}(x)$ returns the appropriate running cursor of the AAC in A_{Col} in which variable x appears. When applied to a sum node, method $\mathsf{Var}()$ returns the variable annotation of that node. Furthermore method $\mathsf{Values}(v)$ returns a set of all possible values in the domain of variable v. For a sum node a, method $\mathsf{addChild}(a, c, v)$ adds a new child c in a for $\mathsf{Var}(a) = v$. For product nodes, method $\mathsf{Const}()$ returns the children of the node corresponding to constant nodes. For a sum node with a variable annotation V, methods $\mathsf{getChild}(v)$ and $\mathsf{Pos}()$ return the child of the node corresponding to $V = v$ and the position of the variable labeling of the node in the global variable order Π_{final} respectively. Method $\mathsf{moveCursorToChild}(a, v)$ sets the AAC cursor to point to the descendant sum node of node a in the path for which $\mathsf{Var}(a) = v$. Finally, $\mathsf{moveCursorToParent}(a, v)$ returns the AAC cursor to the preceding sum node of a.

As mentioned in the previous section, minimizing memory usage and the number of operations performed during merging is important for the performance of the algorithm. A variable in the lineage-AAC may appear in different paths, therefore, after the merge, parts of the database-AACs will be repeated. In order to leverage the detection and caching of those sub-circuits we require that variables which appear together in an AAC from A_{Col} also appear together in the final AAC (Section 6.4.3, Requirement (d)).

Finally, we analyze the complexity of the merging algorithm after caching is introduced. Let m be the size of the query AAC and s_i be the size of the i^{th} AAC from the set $A \subseteq A_{\text{Col}}$ of AACs used during the merge phase. In the worst case the algorithm parses each entire annotated arithmetic circuit at most once. Its complexity is $O(m * \sum\limits_{i \in A} s_i)$. To compute the result probability, we set all indicator variables in the final AAC to 1 and parse the circuit. This operation takes time linear in the size of the final circuit.

6.6 Experiments

In this section we present an experimental evaluation of our framework. The evaluation was performed on an Intel(R) Core(TM) i5 2.3 GHz/64bit/8GB machine running Mac OS X/g++ 4.6.1. Our framework is implemented in C++ for query extraction, lineage processing and probability computation. We used PostgreSQL 9.0 for storing the probabilistic database and the factors. For BDD construction we use the publicly available

CUDD package [149] released by the VLSI group at the University of Colorado. We compare our approach to variable elimination, a generic approach which can support both tuple-independent and correlated tuples [143]. We examine two versions of VE. The first is regular VE using a tabular representation of the factors, and the second is VE where factors are represented using ADDs. VE with ADDs can capture the local structure in the factors of the network. For our results we report wall-clock times of queries averaged over five runs.

6.6.1 Datasets and Queries

We study both tuple-independent and correlated cases. The data used for the experiments was generated by a modified version of the TPC-H data generator. In the first case the generator was modified to create tuple-independent probabilistic databases. We assume that all tables apart from *nation* and *region* are probabilistic and associate each tuple with some existence probability uniformly sampled between $(0, 1]$.

In the second case, we focus on probabilistic databases with arbitrary correlations. We extend the TPC-H data generator to generate correlated probabilistic data according to the following model. We assume that all the tables apart from *nation* and *region* contain uncertain data. Furthermore, tables *customer*, *supplier*, and *partsupp* contain independent tuples. Following the foreign key constraint, each tuple in the *lineitem* table depends on the corresponding tuple from the *orders* table. Many entries of the *orders* table are associated with multiple entries from the *lineitem* table. This introduces many conditionally independent random variables associated with tuples from the *lineitem* table.

For the *part* table, we assume that there is uncertainty over the part price, and that there is a mutual exclusion constraint over price. Finally, for the table *orders* we assume that the orders of a particular customer for a given month are correlated. In particular, this type of correlation can be represented as a chain where the orders are sorted chronologically and then the existence of an order depends on the preceding order. This scenario is realistic as the orders within a particular month may be connected with the same project and they may depend on each other for the fulfillment of that project. The length of the chain varies for databases of different sizes. For a scale factor of 0.001 the maximum length is restricted to two while for a scale factor of 0.1 it increases to ten.

We evaluate our framework for both tractable and hard queries for five different scale factors, namely 0.001, 0.005, 0.01, 0.05 and 0.1. We consider queries Q2, Q3, Q5, Q6, Q8, and Q16. Queries Q6 and Q16 do not contain complicated joins and are easy. However, they are challenging because they generate a large number of result tuples. For every query we remove top-level aggregates and consider its Boolean version.

6.6.2 Experimental Results

We begin by evaluating the scalability of the database compilation technique based on arithmetic circuits. Figure 6.9 illustrates the compilation time as a function of the size of the underlying database and, in particular, the scale factor used to generate it. As illustrated the time required for compiling the database introduces an sizeable overhead to our framework. However, since compilation is performed offline this overhead does not affect the performance of the system during query evaluation.

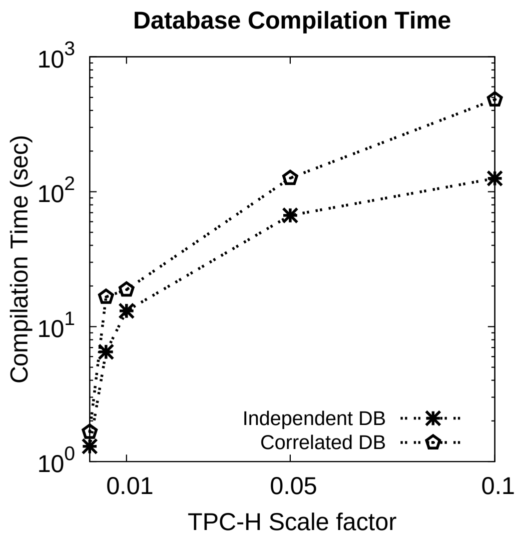


Figure 6.9: The time required to compile the database as a function of its size.

We examine the efficiency of our framework during query evaluation and, in particular, the total execution time for the Boolean versions of the TPC-H queries described above. The results are shown in Figure 6.10. Each graph presents the total evaluation time for a single query for both independent and correlated databases of different sizes. Note that in all graphs the y-axis is in logarithmic scale. Finally, missing values for a particular scale factor correspond to cases where the total evaluation time exceeds the time threshold of 100 seconds.

We focus on hard queries. As shown, for all hard queries, the evaluation based on AACs is at least one order of magnitude faster compared to regular VE but it is also significantly faster than VE with ADDs. As expected, VE with ADDs is faster than tabular VE, since ADDs can capture the local structure in the factors of the network. However, even when using VE with ADDs we still have to pay the cost of multiplying the different ADDs and summing-out variables during online query evaluation. To the contrary, our

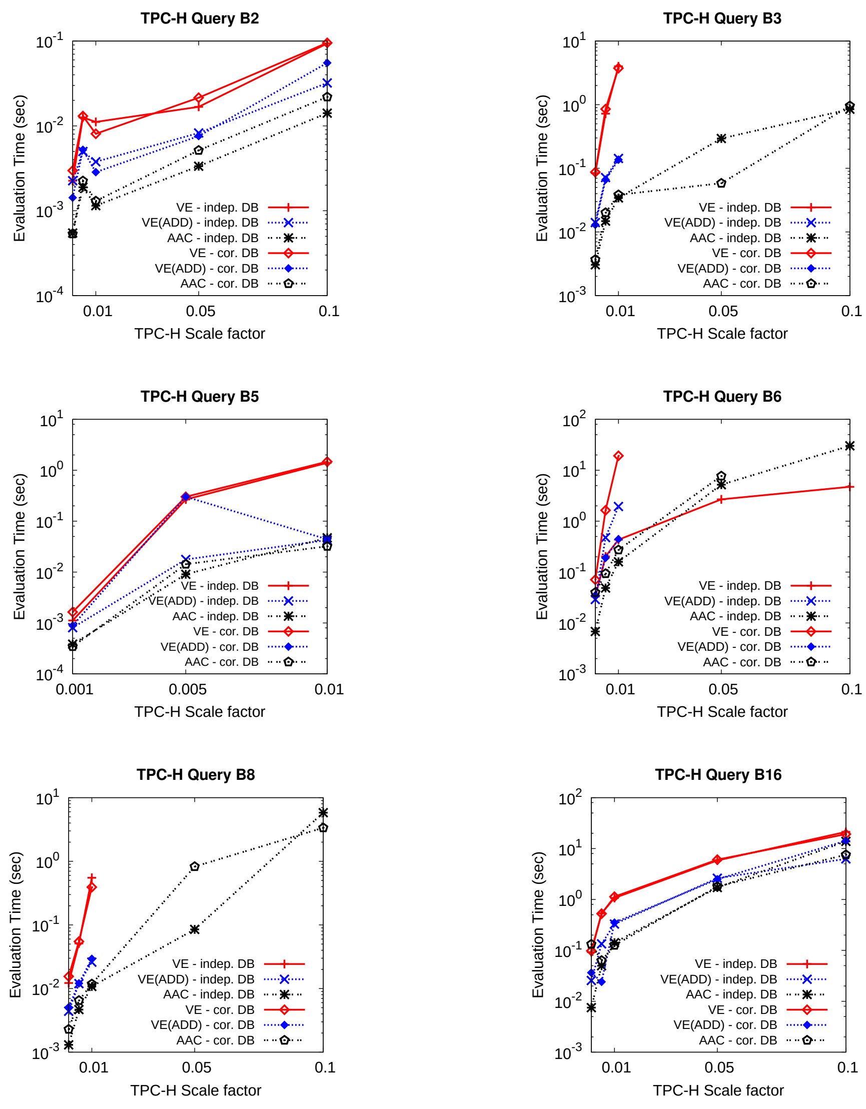


Figure 6.10: Query evaluation times for the Boolean versions of TPC-H queries for both independent and correlated databases of different sizes. Figures (a), (b), (e) refer to hard queries, while the rest to easy queries. Missing values for a scale factor correspond to queries that exceeded the time threshold of 100 seconds.

approach has a significant advantage compared to both baselines as this cost is paid only once, during the offline preprocessing phase. Of particular interest are queries Q3 and Q8 where both versions of VE exceeds the threshold of 100 seconds for both dependency assumptions for scale factors larger than 0.01. This is because of the increasing number of distinct variables in the lineage formulas for larger scale factors. For example the lineage formula for Q3 contains 6276 distinct random variables for a scale factor of 0.1.

Since we are considering the Boolean versions of the queries, the size of the lineage formula is directly associated with the treewidth of the final augmented factor graph. To the contrary, AACs are more scalable since they can fully exploit determinism across the entire network rather than only at a factor level. For example, for Q3 and a scale factor 0.1, the resulting AACs have 42486 and 53803 edges for the independent and the correlated case respectively. Observe that the difference in the size is not that significant despite the presence of correlations as determinism is present in the network. In general, the sizes of the final AACs were sensitive to the queries and the size of the database. For the correlated database experiments the size of the AACs ranged from 19 to 2570 edges for a scale factor of 0.001 and from 1359 to 353214 edges for a scale factor of 0.1.

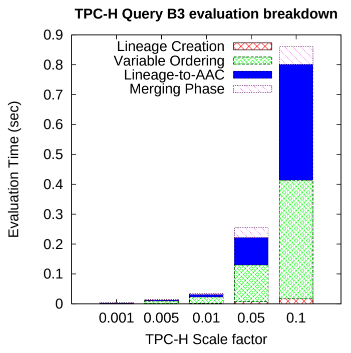
We continue our discussion and focus on queries 6 and 16. Recall that query 6 contains a projection over table *lineitem* and no joins, while query 16 contains a join between tables *partsupp* and *part*. For both, we observe that the performance gain of using AACs is decreasing as the size of the database increases.

To understand this behavior better, we ran micro-benchmarking experiments to investigate the performance of the different components in our framework. We evaluated all queries against correlated databases and we measured the time spent at the different steps of the final AAC creation process. We measure the time for: (a) generating the lineage for the result tuples, (b) generating the final variable ordering using the new algorithm presented in Section 6.4.3, (c) creating the lineage OBDD and converting it to an AAC using the previous variable ordering, and (d) merging all the AACs together. We omit the actual evaluation of the final AAC since it is linear in the size of the final structure, thus, extremely efficient. Due to space constraints we present the results for two representative queries in Figure 6.11. Similar patterns were observed for the rest of the queries.

As shown in the figure, most of the time is spent in creating the lineage OBDD. We demonstrate that this time increases significantly as the size of the database (and consequently the size of the lineage formula) increases. In particular, the size of the linage formula ranges from 59 to 6276 distinct random variables for Q3 and from 142 to 15010 distinct random variables for Q16. Moreover we observed that for all cases where query evaluation with AACs exceeded the threshold of 100 seconds, the actual bottleneck was creating the OBDD for the lineage formula. We would like to point out that an external package was used for creating OBDDs. Improving the performance and optimizing this process is left as future work. Nevertheless, we see that only a small portion of the total running time is spent in the new merging algorithm proposed. Finally, this analysis also explains why for Q6 and Q16 we see a decreasing performance gain when using AACs.

6.7 Related Work

Much of the work in probabilistic database literature has focused on query evaluation under tuple-independence assumption, with some of that work also allowing determin-



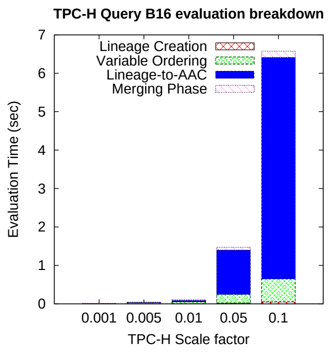


Figure 6.11: Evaluation time breakdown for queries 3 and 16, against correlated databases of multiple sizes.

istic correlations like mutual exclusion. Several recent works have attempted to support more complex correlations, typically represented using graphical models; these include BayesStore [162], PrDB [144], and the work by Wick et al. [167] which uses an MCMC-based technique. However, none of that work exploits local structure for efficient query evaluation. In a followup work to the OBDD-based approach that is limited to tuple-independent databases, Olteanu et al. [117] proposed *decomposition trees* (d-trees), that can support simple correlations expressed via Boolean formulas, but they cannot handle arbitrary correlations in a natural way. While obeying similar structural properties as AACs, d-trees can decompose the lineage formula only partially and can exploit subformulas that can be evaluated efficiently. Moreover d-trees can be used to compute approximate confidence values. It would be an interesting future research direction to combine our approach with d-trees. In a recent work, Jha et al. [81] proposed a framework to combine the intensional and extensional approaches, where they try to use an extensional method as much as possible, falling back to using an intensional approach only when necessary. However, their approach cannot be applied directly to correlated databases represented using factor graphs. Aside from factor graphs, other representations like *pc-tables* [67] can be used to represent correlations. We note that our framework is still applicable in that case, however the preprocessing compilation algorithm (Section 6.4.2) should be replaced with a logical knowledge base compilation algorithm [24] for compiling the database-AACs. Finally, Sanner et al. [140] propose an extension of ADDs, called Affine ADDs, that is capable of compactly representing context-specific, additive, and multiplicative structure. While sharing similarities with AACs, affine ADDs cannot represent conditional independences present in the correlations.

6.8 Summary

In this chapter, we discussed how one can estimate the content changes and the quality of integrated data for a set of dependent sources. We showed how source dependencies can be represented as a factor graph and how computing the content changes corresponds to performing inference over Boolean formulas with dependent variables (i.e., Boolean queries). In the presence of a large number of data sources we described how a probabilistic database can be used to store the corresponding factors graphs and evaluate Boolean queries. We introduced a new algorithmic framework based on knowledge compilation techniques to improve the efficiency of query evaluation in probabilistic databases. Our technique is based on the new structure of annotated arithmetic circuits. In our experimental evaluation we showed that our approach offers speed-ups of at least one order of magnitude over competing approaches.

Chapter 7: Quality-Aware Data Source Management Applications

In this chapter, we demonstrate how the techniques described so far in the dissertation can be used in real-world applications. First, we design a prototype quality-aware data source management system for reasoning about the content and quality of electronic news media including social media, such as Twitter, online news papers and blogs. Then we show how reasoning about the quality of multiple sources can help us forecast outbreaks of rare diseases more accurately than source agnostic forecast models.

7.1 Source Selection for Event Data

Recently, there has been an increasing interest in monitoring news media, blogs, and social media from all over the world and extracting geo-referenced records that correspond to different real-world events and interactions between diverse groups of people, international organizations, countries etc.. These repositories are updated regularly whenever new extractions are obtained from a diverse collection of data sources. Example repositories of such extractions include GDELT[1] and EventRegistry[2] which are updated over fixed time intervals (e.g., daily). The extracted data is stored in a tuple format containing information about its origin (i.e., the source it was extracted from), an event identifier

[1]http://gdeltproject.org/
[2]http://eventregistry.org/

corresponding to a real-world event, a short description of the event and a timestamp. Due to the large number of data sources monitored in such repositories (e.g., EventRegistry monitors around 75,000 news sources daily) it is often hard for analysts to identify sources that are useful for their applications. In this part of the dissertation we introduce SOURCESIGHT, a quality-aware data source management system that implements the techniques proposed in the previous chapters of the dissertation while focusing on the event data domain described above. Next, we discuss SOURCESIGHT's design and present an overview of the system's functionalities.

7.1.1 Design Details

SOURCESIGHT offers a number of unique features:

(1) Users can describe a data domain relevant to their application using a keyword-based interface. They can also explore sources relevant to that domain and discover domains that are highly relevant to their initial search. The latter allows them to refine the specification of their desired task.

(2) Given a domain description, users can perform source selection by selecting their desired quality metrics from a collection of prespecified metrics. SOURCESIGHT casts source selection as a multiobjective optimization problem to help users understand the trade-offs between the different selected quality metrics. It also proposes different sets of sources, with each corresponding to a source selection solution where different weights are assigned to the selected quality metrics.

(3) Finally, the system allows users to interactively explore the recommended solutions. Users can also perform a qualitative comparison between different source selection solutions. This is crucial for evaluating the solutions recommended by SOURCESIGHT and helps users understand why a particular set of sources was proposed by the system.

The core of SOURCESIGHT is built around the techniques introduced in Sections3.3.1, 3.4.1 3.5 and 3.6 and Chapter 5. SOURCESIGHT's design follows the architecture shown in Figure 1.5 and extends it with a frontend that allows users to interact with the system. The frontend is a "thin-client" that allows users to specify a data domain by providing a keyword-based description and displays visualizations that aid users to select the desired sources for integration.

To enable effective source selection over diverse domains, we use a correspondence graph (Section 3.6) to identify the domains covered by the available sources. To discover the literals associated with each source entry we use Thomson Reuter's Open Calais[3], an API for semantic annotations with respect to multiple knowledge bases including DBpedia, Freebase and others. After discovering the context literals associated with each source, we identify the c-cluster nodes in the correspondence graph (see Section 3.6).

To characterize the quality of data sources, SOURCESIGHT considers the metrics of coverage and freshness (referred to as timeliness in the system). Accuracy is not applicable in the domain of event data as no retractions or deletions were observed. Furthermore, SOURCESIGHT extends the quality metrics by reasoning about the *position bias* of each source with respect to the c-clusters in the correspondence graph. The position bias of a source S with respect to a c-cluster $\mathcal{C}$ measures how positive or negative the sentiment of

[3]http://www.opencalais.com/

the entries of S are towards entities contained in the domain $\mathcal{C}.D$ of the c-cluster. The sentiment of S for a single source entry is extracted using standard sentiment analysis techniques [119] that focus on the *subjectivity* and *polarity* of the entry. The position bias of S towards the c-cluster $\mathcal{C}$ is computed by aggregating the subjectivity and polarity values over all source entries relevant to $\mathcal{C}.D$.

7.1.2 SOURCESIGHT Functionalities

Users interact with SOURCESIGHT by providing a keyword-based description of their domain of interest. These keywords are matched against the context literals associated with the different c-clusters to identify domains relevant to their description.

Once a description is provided users can choose among three main functionalities. They can (i) choose to explore which keywords and sources are highly relevant to their description, (ii) choose to perform source selection, and (iii) choose to perform a qualitative comparison between different source selection solutions recommended by the system.

SOURCESIGHT offers a unified interface for users to explore both context-literals and sources related to their desired integration task (see Figure 7.1). Given the description of a user, SOURCESIGHT returns the set of top relevant literals to the search of the user as well as the most relevant sources with respect to coverage or other metrics for the specified keyword search. The user can then select any of the recommended sources to view a summary of the literals that the source covers as well as a quality summary of the source for the corresponding keyword search. Users can also choose to update their description by including new relevant context-literals. Figure 7.1 shows an example use-case where a

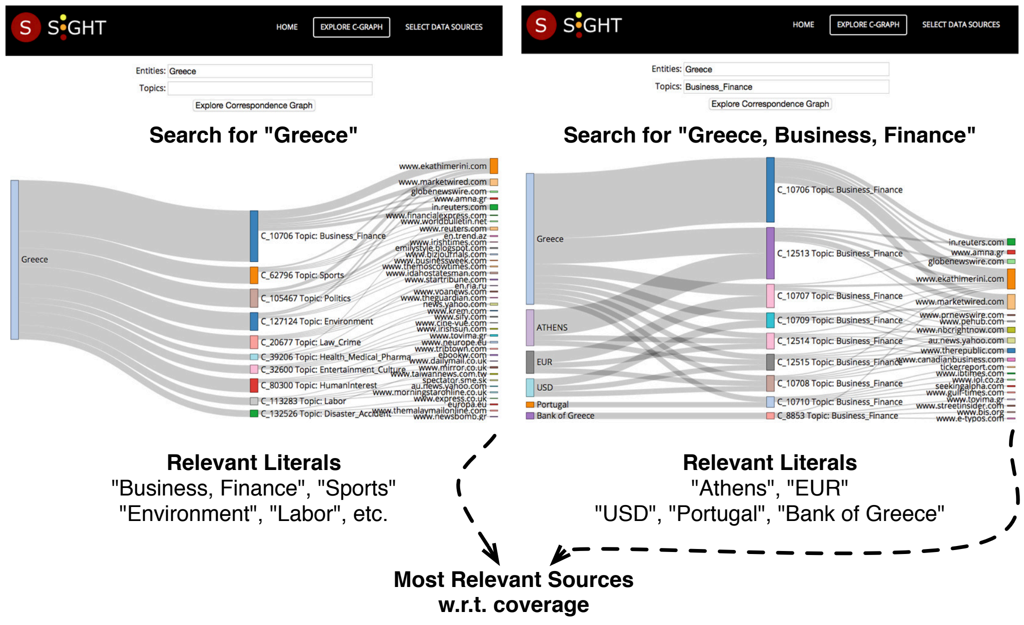
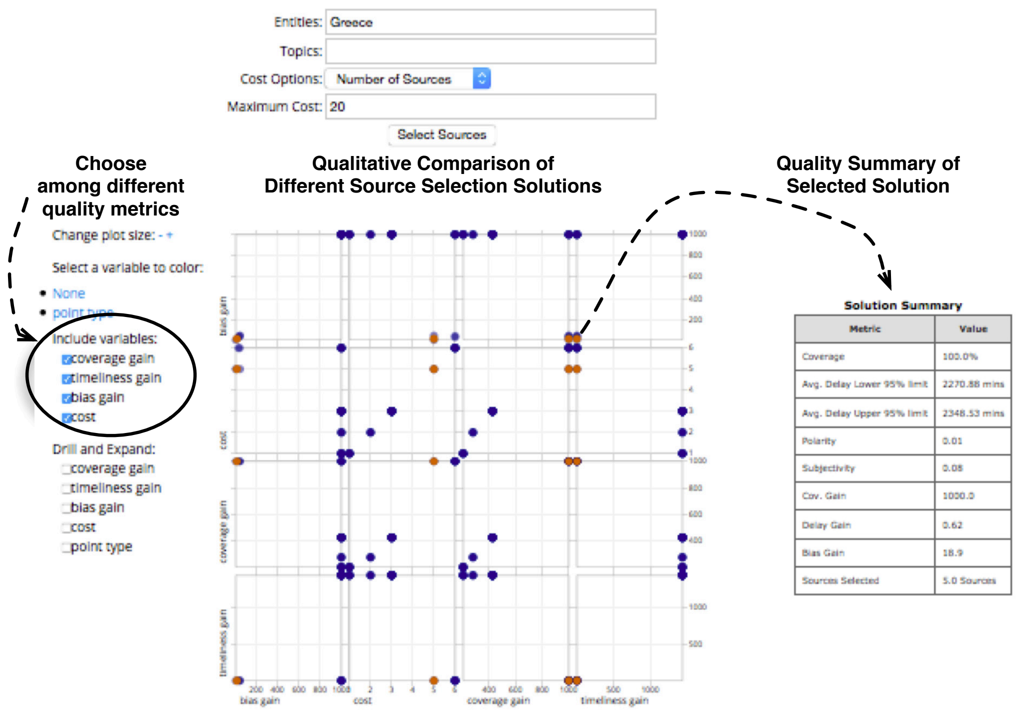

Search for "Greece"
Search for "Greece, Business, Finance"
Relevant Literals
"Business, Finance", "Sports"
"Environment", "Labor", etc.
Relevant Literals
"Athens", "EUR"
"USD", "Portugal", "Bank of Greece"
Most Relevant Sources
w.r.t. coverage

Figure 7.1: Discovering relevant sources and refining an integration task with SOURCE-SIGHT.

journalist wants to write an overview article about the socio-economic situation in Greece. The journalist starts by requesting news sources relevant to the keyword “Greece”. Apart from presenting the relevant sources, SOURCESIGHT additionally recommends that it might be beneficial to explore related and more specialized descriptions, such as “Greece and Business and Finance” or “Greece and Labor”, as the set of relevant sources may change significantly.

Because of the variety of quality metrics SOURCESIGHT supports, it provides the user with multiple source selection solutions that correspond to different weighting configurations for the available quality metrics. The solutions are generated using the technique described in Section 5.3.3. This allows users to explore different trade-offs amongst the available quality metrics and identify the set of sources that best satisfies their quality requirements (seeFigure 7.2).

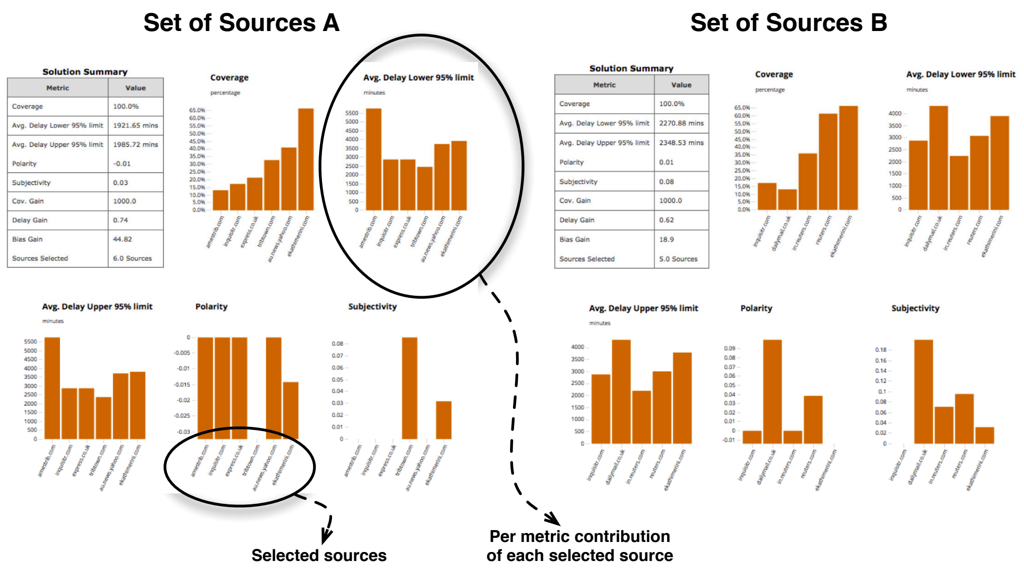


Entities: Greece
Topics:
Cost Options: Number of Sources
Maximum Cost: 20

Solution Summary

Metric	Value
Coverage	100.0%
Avg. Delay Lower 95% limit	2270.88 mins
Avg. Delay Upper 95% limit	2348.53 mins
Polarity	0.01
Subjectivity	0.08
Cov. Gain	1000.0
Delay Gain	0.62
Bias Gain	18.9
Sources Selected	5.0 Sources

Figure 7.2: SOURCESIGHT's interface for exploring source selection solutions.



Solution Summary (Set of Sources A)

Metric	Value
Coverage	100.0%
Avg. Delay Lower 95% limit	1921.65 mins
Avg. Delay Upper 95% limit	1985.72 mins
Polarity	-0.01
Subjectivity	0.03
Cov. Gain	1000.0
Delay Gain	0.74
Bias Gain	44.82
Sources Selected	6.0 Sources

Solution Summary (Set of Sources B)

Metric	Value
Coverage	100.0%
Avg. Delay Lower 95% limit	2270.88 mins
Avg. Delay Upper 95% limit	2348.53 mins
Polarity	0.01
Subjectivity	0.08
Cov. Gain	1000.0
Delay Gain	0.62
Bias Gain	18.9
Sources Selected	5.0 Sources

Figure 7.3: Comparing source selection solutions with SOURCESIGHT.

Finally, users can perform a quality comparison between the solutions recommended by SOURCESIGHT. All the sampled solutions are presented to the user in a way that makes it easy for her to compare the quality of each solution. Users can select a particular solution and view a concise summary of the benefit and cost of integration achieved by it. Users can also drill down and expand only on a subset of the available quality metrics to fully understand specific trade-offs across different solutions. Users can also view a detailed description of a source selection solution with information about the sources included in the result and their individual contributions to the quality of the final integration result (see Figure 7.3).

7.2 Forecasting Rare Disease Outbreaks with Multiple Sources

Recently, there has been a growing interest in developing statistical models to forecast infectious disease outbreaks enabling effective control measures to be taken in a sufficiently timely fashion. In this section, we demonstrate how some of the techniques described in Chapter 3 can be used to forecast the incidences of rare disease outbreaks when the data used for forecasting are collected by multiple sources. We focus on incidences of Hantavirus syndromes over countries in Latin America. Human infections of Hantaviruses are rare and have almost entirely been linked to human contact with rodent excrement.

Many previous approaches rely on integrating publicly available data from the Web, including news articles [18, 102], blogs [29], search engine logs [62] and micro-blogging services, such as Twitter [30, 123, 124]. However, most approaches are agnostic to the quality of each individual source used in the process and usually proposed models that

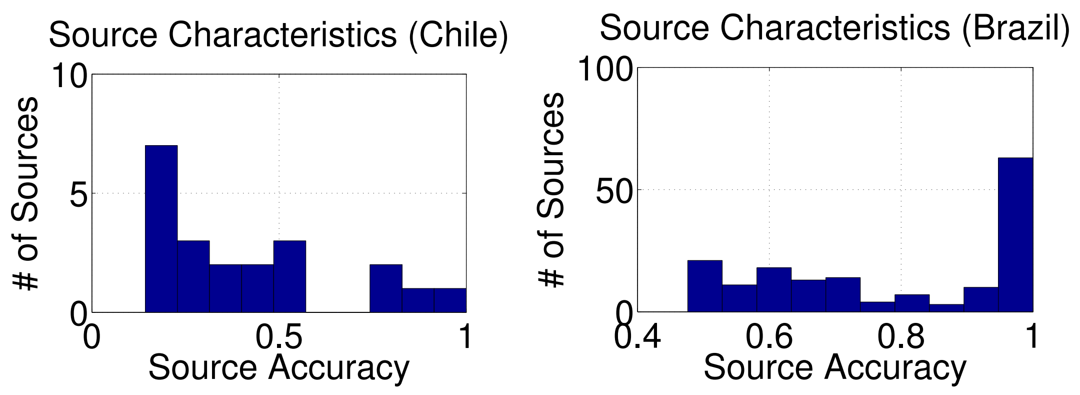


Figure 7.4: Source accuracy histograms for Chile and Brazil.

utilize the data of *all available* sources. Nevertheless, different data sources may exhibit different delays at reporting rare disease incidences, and using their data for predicting outbreaks may lead to predictions of significantly different accuracy. To illustrate this we use the following scenario.

We consider a dataset that corresponds to a corpus of public health-related news articles and tweets from 798 different sources referring to multiple diseases over a timespan of 15 months. All news articles are either in Spanish or Portuguese. The news article feed gets updated on a weekly basis and the goal is to predict disease outbreaks for the next week. We wish to reminder the reader that the same dataset was used in Section 3.7.2.

Example 9. *We consider data sources providing information for Chile and Brazil and examine their accuracy at predicting Hantavirus outbreaks. Figures 7.4(a) and 7.4(b) shows the source accuracy histograms. As shown, the accuracy levels of different data sources vary significantly. The model used for predicting outbreaks is described in Section 7.2.1.*

Motivated by this example, we apply our quality-aware source management techniques to the problem of forecasting Hantavirus outbreaks in countries in Latin America using the data described above. Sources correspond to newspapers publishing news articles and data entries correspond to unstructured data. Therefore, we use the techniques introduced Section 3.3.2 and Section 3.4.2 to identify the quality of available sources.

Using these techniques we can estimate the effectiveness of each data source at covering the topics present in the underlying domain at future time points. The effectiveness of a source in covering certain topics at future time points (i.e., the future source-topic relevance value) can be viewed as a surrogate for its authoritativeness for that topic. Thus, if a source is very effective in covering Hantavirus related topics, then more emphasis should be put on its data compared to other sources.

Each source can be viewed as an *expert* providing a prediction for an outbreak. Given the individual source predictions, one needs to fuse them together into a single prediction. Next, we present how, one can derive source-specific predictions for Hantavirus outbreaks by taking into account the source-topic relevance values, and provide an algorithmic framework for fusing predictions from multiple sources.

7.2.1 Expert Fusion with Sources as Experts

Hantavirus incidences are scarce over time. Therefore, the source-topic relevance values (Equation 3.11 and Equation 3.13) for a rare disease topic will be low for most time points and high only for few time points corresponding to an outbreak. Following this observation, high relevance values for a rare disease topic, can be viewed as anomalous points, and thus, anomaly detection techniques can be used to identify if the source-topic relevance corresponds to an anomalous point.

One-class SVMs [141] (OCSVM) can be used to classify the source-topic relevance values as anomalous or not. OCSVMs have successfully been used in a variety of anomaly detection tasks [103, 150, 73]. OCSVMs present superior performance compared to other anomaly detection techniques, such as Nearest Neighbor classification, in scenarios where a small number of anomalous example is available [90]. Finally, OCSVMs do not make any assumptions on the distribution of the data point values.

Let L be the set of available locations in the data and T the time window for which data from the sources are collected. To predict outbreaks for a future time point t, a separate OCSVM for each source-location pair (s, l), with $l \in L$, is trained using the estimated source-topic relevance values for all time points up to $t-1$ as training data. The training entry for a time point $t' \prec t$ corresponds to a vector $< \mathsf{Relevance}(s, z_1; l, t'), \mathsf{Relevance}(s, z_2; l, t'), \cdots >$ containing the estimated relevance values for all topics $z_1, z_2, \ldots$ that are relevant to the rare disease under consideration. The estimated relevance values for time point t can be computed using the methodology described in Section 3.4.2. Our goal is to forecast disease outbreaks for a specific location, thus, the predictions of all sources should be fused into a single prediction for each location $l \in L$ at time t. For this, we introduce a weighted majority voting algorithm based on the multiplicative weights update framework[5].

Given time t in the future, focus on a location l and view each source $s \in \bar{\mathbf{S}}$ as an expert providing a prediction $d_s \in [-1, 1]$ with the value -1 corresponding to the emergence of an outbreak and 1 otherwise. A weight w_s is assigned to each source, and given the predictions of all sources, predict yes/no for an outbreak at location l by taking the majority vote $\sum_{s \in \bar{\mathbf{S}}} w_s \cdot d_s$. The weights w_s can be learned using the multiplicative weights update algorithm shown in Algorithm 8.

Algorithm 8 Multiplicative Weights Update for Sources

```
1: Input: $S_l$: set of sources for location $l$; $\mathbf{D_l}$: training points; $R_{S_l}$: source-topic relevance dictionary for sources in $S_l$ and points in $\mathbf{D_l}$; $O_{S_l}$: one-class SVMs for $S_l$; $\epsilon$: discount factor
2: Output: $\mathbf{W}$: weights for sources in $S_l$
3: Initialize all weights $W$ to 1
4: for all $d \in \mathbf{D_l}$ do
5:   for all $s \in S_l$ do
6:     /*Extract the expert's vote*/
7:     $v \leftarrow O_{S_l}[s].predict(R_{S_l}[s][d])$
8:     if v is wrong then
9:       $W_k \leftarrow W_k \cdot \exp(-\epsilon)$ /* Decrease the weight */
10:    else
11:      $W_k \leftarrow W_k \cdot \exp(\epsilon)$ /* Increase the weight */
12:  Normalize the weights to sum up to $1.0$
13: return W
```

Consider a location l. To construct the necessary input for the multiplicative weights update algorithm we assume access to a gold-standard report (GSR) is assumed. GSR provides ground truth information for disease outbreaks at locations in L for time points $t \prec T$ and is being updated at a much lower rate than that of the source data and therefore one can observe significant delays at obtaining ground truth information. Given GSR we perform the next steps: (i) identify the set of sources S_l relevant to location l, i.e., sources that have published for location l, and (ii) construct the set of training points $\mathbf{D_l}$ by considering the reported outbreaks in GSR for location l and the disease under consideration. Populate $\mathbf{D_l}$ with tuples of the form $(timepoint, outbreak)$ for all historical time points up to the latest time point present both in Ω and GSR and set the value of $outbreak$ to -1 if an actual outbreak was reported and 1 otherwise. Finally, use the past source-topic relevance values for the sources in S_l and the training points in $\mathbf{D_l}$. The latter step converts the quality metric of coverage extracted by the proposed automated source management techniques to a task specific accuracy metric for each source.

Given the input described above, the algorithm proceeds in an iterative fashion updating the weights of the sources considering the accuracy of their predictions. More precisely, the algorithm iterates over all training points in $\mathbf{D_l}$ (Ln. 4). At each iteration, it examines all available sources (Ln. 5) and extracts their prediction corresponding to a specific training point from the past (Ln. 6-7). If the expert is mistaken, it's corresponding weight is reduced in a multiplicative fashion (Ln. 9), otherwise its weight is increased (Ln. 11). Finally, the algorithm outputs the normalized weights, which are later used to fuse the individual source predictions for future time points. The process is repeated as more ground-truth data are becoming available through GSR.

Finally, each outbreak prediction for location l is associated with a *confidence score*. Let S be the set of relevant sources for location l and S_{-1} be the subset of sources predicting an outbreak. Moreover, let $a_l(s)$ be the overall *accuracy* of a source $s \in S_l$ considering its past predictions for location l. The accuracy of source s is defined over the available past time window as $a_l(s) = \frac{\text{\# correct predictions}}{\text{\#total prediction}}$ and corresponds to the probability of s giving a correct prediction. The confidence score is:

$$ConfScore = \prod_{s \in S_{-1}} a_l(s) \cdot \prod_{s \in S_l \setminus S_{-1}} (1 - a_l(s)) \tag{7.1}$$

Given the confidence score of each outbreak prediction, one can use a threshold mechanism to select the final outbreak predictions, and balance the trade-off between precision and recall as discussed in Section 7.2.2. Fusing the predictions of individual sources, one can predict *if* a disease outbreak will happen during a specific week. To predict the exact day of the incidence, a standard relative date within the week is adopted to be the date at which the rare disease incidence will occur, and is tuned using cross-validation.

7.2.2 Evaluating Outbreak Forecasts

We now evaluate the performance of the proposed source-aware forecasting approach.

Data. We use the news articles dataset introduced in Section 3.7.2.

GSR. The gold standard report gives ground truth determinations of whether a disease incidence (Hantavirus) happened in a given location.. The GSR is used from out multiplicative-weights algorithm to determine the predictive accuracy of different sources. The GSR is determined by analysts considering multiple news sources and studying bulletins issued by health reporting organizations such as ProMED [1].

Models. The following models are evaluated:

- SourceSeer: The source-aware prediction framework that combines the techniques introduced in Section 3.3.2 and Section 3.4.2 with the multiplicative-weights algorithm described above. We also couple this framework with a thresholding mechanism where for a week and country accepts only the predictions with confidence scores in the top-k percentile of all prediction scores for that country.
- LocSeer: A variation of SourceSeer that uses the topic model from Section 3.3.2 to identify disease related topics but integrates this with a *location-only* anomaly detection approach that is similar to the one introduced above. For each location we calculate the location-topic relevance values for future time points and use an OCSVM to detect anomalous points. To calculate the location-topic relevance, we estimate each entry of the location's word frequency vector as:

$$\hat{F}_{l,t}[w] = \bar{x}_w \Pr(t|l,w) \sum_{z \in K} \phi_{z,w} \cdot \theta_{l,z} \cdot \xi_{z,t}$$

where $\Pr(t|s,w)$ is defined similarly to Equation 3.15. Intuitively, LocSeer integrates news articles from multiple data sources ignoring the coverage and accuracy of individual sources. Again a thresholding mechanism similar to that of SourceSeer is used considering the accuracy of each state-based OCSVM.

- KeyWord: A *keyword* based prediction technique that monitors the mentions of Hantavirus related keywords. We considered the set {“hanta”, “hantavirus”, “roedores”, “ratones”, “cardiopulmonar”} and used an OCSVM to predict future outbreaks based on past mentions of words. This word-set reflects the fact that Hantavirus has almost entirely been linked to human contact with rodent excreta and their symptoms affect the heart and lungs.
- BRM: A *base rate model* that assumes a fixed rate for the occurrence of rare disease outbreaks for each location and for each month. To determine this rate, the model extracts the average frequency of outbreak occurrences reported over a past time window of four months. BRM reports disease outbreaks for that location at a frequency equal to the extracted rate. Alerting dates are assigned to the beginning of each month while event dates are assigned uniformly at random to a day within the corresponding month. The average performance over 25 independent runs is taken.

All models are implemented in Python and the evaluation is performed on an Intel(R) Xeon(R) CPU E7-4870 @2.40GHz/64bit/1TB machine.

Parameter Setup. The OCSVM parameters are tuned using leave-one-out cross-validation. The topic model, the parameters of the Dirichlet priors are set to $\alpha = 2/K$, $\beta = 0.01$ and $\gamma = 0.01$ where K is the number of topics. The topic model was evaluated with $K = \{8, 12, 15\}$ and setting $K = 12$ was found to provide the most meaningful topics.

Metrics. Five key measures of performance are adopted. Given the predictions, the precision, recall and F1-score are computed at a country level, grouping together prediction for locations in the same country. An average warning quality for each country is also computed as follows. Each prediction for a location in the country under consideration is assigned a quality score $Q = \frac{4}{3}(1 + a_{loc} + a_{date})$, where a_{loc} and a_{date} denote the location and date accuracy of the prediction. To calculate a_{loc} A two-level topology, considering the country, and state corresponding to the location of a warning is used. A partial score of 0.5 is assigned to a warning if it matches the country of an outbreak correctly and an additional score of 0.5 is assigned if the warning matches the state correctly. The date specific accuracy a_{date} is:

$$a_{date} = 1 - \frac{min(|\text{predicted date} - \text{actual date}|, 7)}{7} \tag{7.2}$$

Finally, the lead-time of the predictions is considered. The lead-time calculated as the time between the date of alerting and the actual date of reporting the outbreak (not the incidence date of the outbreak). Lead-time is different from the date accuracy above.

Mapping Warnings to Events. Since there can be multiple events in a given month, we need a strategy to map events to alerts. A maximum bipartite matching between events and alerts is used where (i) an edge exists if the alert was issued prior to the reporting date of the event, (ii) the weight on the edge denotes the putative quality score.

7.2.2.1 Predicting Disease Outbreaks

How efficient is SourceSeer at forecasting disease outbreaks? We evaluate the performance of the various disease outbreak forecasting algorithms focusing on hantavirus incidences at the country level considering the predicted outbreaks for Argentina, Chile, Uruguay and Brazil. We apply BSR, KeyWord, SourceSeer and LocSeer. We evaluate the performance of SourceSeer and LocSeer with $k \in \{5, 10, 20, 30, 40, 50, 70\}$. We use the three hantavirus topics described above to construct the necessary feature vectors for SourceSeer and LocSeer.

Table 7.1 shows the precision, recall and F1 score of the three approaches from January 2013 to March 2014 aggregated over all countries. For LocSeer and SourceSeer the results for the configuration k that obtained the best performance are reported. As shown, SourceSeer obtains the best F1-score for most of the months. The F1 score of BSR is lower as its recall is significantly lower compared to that of SourceSeer. The latter is expected as BSR can only predict outbreaks for states where a sufficient number of outbreaks has occurred in the past. In fact, due to its design BSR fails completely to forecast outbreaks for states or countries where no outbreaks have been observed in the past (e.g., the outbreak in Brazil for October 2013 and the outbreak in Uruguay for March 2013). However this mechanism limits the number of false positives significantly, and thus, for many months we observe slightly higher or comparable precision scores for BSR with those of SourceSeer. The F1 score of LocSeer is significantly lower compared to SourceSeer due to its significantly lower precision scores. The reason for this behavior is the increased number of false positives returned by LocSeer even after the threshold-

Table 7.1: BSR, KeyWord, LocSeer and SourceSeer on predicting hantavirus outbreaks. Notation $(k\%)$ denotes the best performing configuration for LocSeer and SourceSeer.

	BSR			KeyWord			LocSeer (**5%**)			SourceSeer (**5%**)		
Month	*Prec.*	*Rec.*	*F1*	*Prec.*	*Rec.*	*F1*	*Prec.*	*Rec.*	*F1*	*Prec.*	*Rec.*	*F1*
01/13	0.5	0.17	0.25	**0.67**	0.33	0.44	0.13	**0.67**	0.22	0.44	**0.67**	**0.53**
02/13	0.52	0.78	0.62	**0.67**	**1.0**	**0.80**	0.12	**1.0**	0.21	0.5	**1.0**	0.67
03/13	**0.7**	0.35	0.46	0.6	**0.75**	**0.67**	0.29	0.5	0.37	0.5	0.5	0.5
04/13	**0.78**	0.59	0.67	0.33	0.25	0.28	0.6	0.75	0.67	0.57	**1.0**	**0.73**
05/13	**0.51**	0.48	**0.54**	0.29	0.4	0.34	0.14	0.2	0.16	0.38	**0.6**	0.47
06/13	**0.22**	0.68	**0.33**	0	0	0	0.14	**1.0**	0.25	0.14	**1.0**	0.25
07/13	**0.22**	0.68	**0.33**	0	0	0	0.14	**1.0**	0.25	0.2	**1.0**	**0.33**
08/13	0.4	0.6	0.47	0	0	0	0.2	**1.0**	0.33	**0.67**	**1.0**	**0.80**
09/13	0.5	0.33	0.39	0	0	0	0.23	**1.0**	0.37	**0.67**	**0.67**	**0.67**
10/13	**0.62**	0.24	0.35	0.5	0.4	0.44	0.31	**0.8**	0.45	0.38	0.6	**0.47**
11/13	**0.89**	0.44	0.59	0.75	0.5	**0.6**	0.21	**0.83**	0.34	0.45	**0.83**	0.58
12/13	0.9	0.32	0.47	**0.75**	0.27	0.40	**0.75**	**0.55**	**0.63**	0.67	**0.55**	0.60
01/14	0.65	0.49	0.56	0.43	0.38	0.40	0.19	0.5	0.28	**0.71**	**0.63**	**0.67**
02/14	0.56	**0.74**	0.64	0.43	0.5	0.46	0.27	0.67	0.38	**0.67**	0.67	**0.67**
03/14	0.55	**0.88**	**0.68**	**0.57**	0.8	0.66	0.29	0.8	0.42	0.5	0.8	0.62

ing mechanism was employed. Finally, KeyWord performs reasonably well when there is an increase in the number of outbreaks in previous weeks leading to increased keyword counts. However, the model performs poorly in the presence of low keyword counts. KeyWord failed to forecast the outbreaks in August and September 2013 as only one was reported in July.

Is the performance gain of SourceSeer significant? To obtain a clearer understanding of SourceSeer's performance gain, we perform the Wilcoxon signed-rank [168] test comparing the performance of BSR with SourceSeer, KeyWord with SourceSeer and LocSeer with SourceSeer for precision, recall, and F1-score across all months. In Table 7.2 we report the corresponding test statistic scores W and the z-scores. We consider a baseline confidence level of $\alpha = .05$. As shown, the performance difference between BSR and SourceSeer is statistically significant for recall and F1 (with SourceSeer outper-

Table 7.2: Wilcoxon signed-rank statistical significance test on SourceSeer's performance gain. H_0: The median performance difference between the pairs is zero. Reject H_0: $|z| \geq 1.645$ or $W \geq 15$ when z not applicable. Baseline confidence level of $\alpha = .05$. Bold fonts denotes statistically significant differences.

Metric	**Score**	**SourceSeer v.s. BSR**	**SourceSeer v.s. LocSeer**	**SourceSeer v.s. KeyWord**
Prec.	W	-51	81	36
	z	-1.463	**2.966**	1.349
Rec.	W	114	3	76
	z	**3.223**	-	**2.961**
F1	W	61	101	100
	z	**1.899**	**3.154**	**2.825**

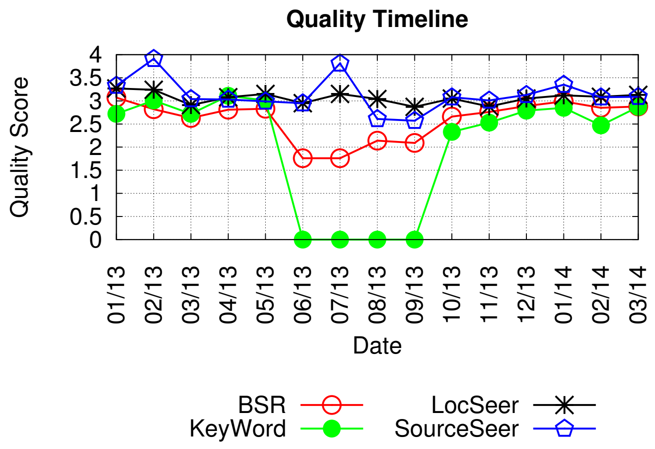


Figure 7.5: Quality score timeline for BSR, KeyWord, LocSeer and SourceSeer on predicting Hantavirus outbreaks.

forming BSR) while the difference for precision is not statistically significant. The same behavior was observed for KeyWord and SourceSeer. For LocSeer and SourceSeer, we see that the performance gain of SourceSeer for precision and F1 is statistically significant while the difference for recall is not. We did not observe significant differences in the performance of LocSeer and SourceSeer for different values of k. We further analyze the performance of the four models by comparing the quality score cross all months under consideration. Figure 7.5 shows the average prediction quality score obtained by

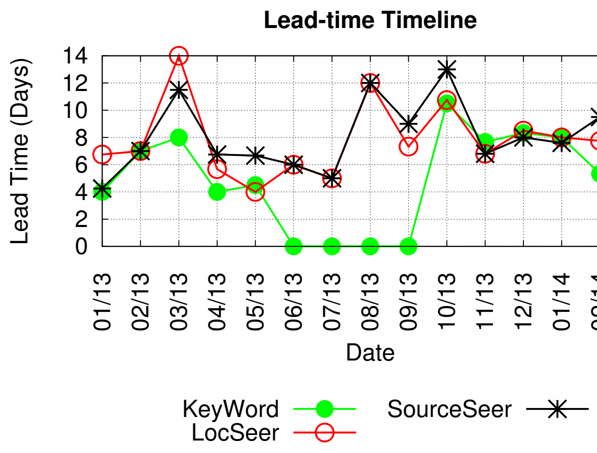


Figure 7.6: Lead-time for KeyWord, LocSeer's and SourceSeer's predictions.

each model from January 2013 to March 2014. A higher quality score is an indicator that a model can predict outbreaks correctly at the state and not only at the country level. As shown, both LocSeer and SourceSeer outperform BSR and KeyWord significantly. This is expected since BSR relies only on past reported events to predict future outbreaks and KeyWord on increased keyword counts, hence, by design both cannot predict outbreaks in states with no reported incidents. We also see that SourceSeer obtains higher quality scores for most of the months compared to LocSeer. This is due to weighting the predictions of difference sources based on their accuracy for each specific state.

What is the lead-time gain of SourceSeer? Finally, we analyze the average lead-time of KeyWord, LocSeer and SourceSeer to examine if the proposed models can forecast outbreaks in a timely manner. Figure 7.6 shows the lead-time timeline of the three models from January 2013 to March 2014. We observe that both models have a significant lead-time advantage when compared against the mention of the outbreak in news sources and also outperform KeyWord.

7.3 Summary

In this chapter, we demonstrated how the techniques proposed in the dissertation can have significant impact on real-world applications. Quality-aware data source management can not only enable users to discover valuable sources for integration but the underlying techniques can also be used to devise significantly more effective forecasting mechanisms.

Chapter 8: Conclusion

In this dissertation, we introduced a framework for quality-aware data source management. We demonstrated that, in the presence of a large number of heterogeneous data sources, it is possible to effectively reason about their quality, and determine the actual utility of the data they provide for diverse user applications. We showed that, without fully integrating all available data from multiple sources, it is possible to find the set of sources that maximize the utility of the integration result given an integration cost budget.

The technical contributions of the dissertation are (i) a collection of formally defined data quality metrics based on probability theory, (ii) statistical models to formally compute the content focus and quality of different types of sources, including sources whose content changes over time and sources that provide both structured and unstructured data, and (iii) efficient algorithms with formal performance guarantees for finding the most valuable sources for integration. We also introduced two systems, SOURCESIGHT and SOURCESEER, that demonstrate how the techniques in this dissertation can not only help users identify the most useful sources for their applications but can also lead to significant accuracy improvements for multi-source learning applications.

8.1 Future Directions

Characterizing and assessing data quality, and therefore data source quality, will be an increasingly important area over the next several years as businesses, governments and analysts realize that data is a commodity similar to standard computational resources (e.g., the number of nodes in a cluster) that everybody is familiar with. Moreover, as the collection and publishing of data is only expected to increase, the problems of understanding *how* useful a data source is and *why* it is useful for a specific application, will only become more important. Next, we identify research directions which we believe to have a high potential of impact.

Automated Quality Assessment with Guarantees. Assessing the quality of a data source or a dataset requires comparing and contrasting it with the real world constructs it refers to. As discussed earlier in this dissertation, one can either do that by considering limited ground truth, if available, or in the case of missing ground truth, one can approximate the real-world by integrating and overlaying samples from different sources. While our models in Chapter 3 were empirically shown to provide accurate estimates for different source quality metrics, they come with no guarantees on the goodness of fit of these estimates. An immediate next step would be to devise new quality estimation procedures that will not only estimate the quality of sources via sampling the content of data sources but will also provide *confidence intervals* for the estimated quality metrics. These intervals need to take into account the size of the sample as well as how representative the sample is with respect to the population of entries provided by the source. For example, consider a source providing business listings from Maryland and New York. We want to estimate the accuracy of this source but we only have access to a sample that contains 1% of the source's data and data entries from Maryland are significantly more abundant. It is obvious that the source's accuracy on this small sample may be significantly different from the source's true accuracy. Therefore, we need to account for the sample specific information and obtain confidence intervals for the source's accuracy.

Expert-Specified Quality Metrics and Explanations. In Chapter 3, we described a collection of generic metrics (i.e., coverage, freshness and accuracy) that can be used to characterize the content of sources and the quality of integrated data. While these metrics are generic and can be instantiated under different semantics, they are not always sufficient to describe the content of a *high-quality source*. For example, consider a scenario where we have multiple data sources corresponding to results of clinical trials and a biologist that wants to evaluate the data published by each source. One measure that characterizes the results of clinical trials is how representative is the population on which the trial was conducted on. Nevertheless, this is a specific metric that is applicable to the scenario of clinical trials and may not be directly applicable to generic data sources.

Given the need of application-specific metrics, a future direction for data quality research is to enable users to specify quality-requirements for their applications via *declarative interfaces*. Such interfaces can be built upon first order logic and data quality can be expressed using a collection of logical rules. Generic metrics such as coverage, freshness, and accuracy, can be used as building blocks, but nonetheless designing the necessary primitives for *expressive* quality specifications is an open research problem. Finally, declarative quality specifications will allow quality-aware data source management systems to provide *explanations* for the quality of integrated data by reasoning about how well the integration result satisfies the provided quality rules. We believe that discovering concise and human-interpretable explanations is a rather challenging problem with a high potential of impact.

Appendix A: Supplemental Derivations and Proofs

A.1 Derivation of Gibbs Sampling Equations

In this section we provide a Gibbs sampling algorithm for learning the parameters of the topic model introduced in Section 3.3.2. Before we proceed with the actual algorithm, we present the joint distribution corresponding to the topic model.

$$\begin{aligned}
&\Pr(\mathbf{w}, \mathbf{t}, \mathbf{v}, \mathbf{z}, \phi, \theta, \xi; \alpha, \beta, \gamma) = \\
&= \prod_{z=1}^{K} \Pr(\phi_z; \beta) \Pr(\xi_z; \gamma) \prod_{v \in \mathcal{V}_{\mathcal{A}_D}} \Pr(\theta_l; \alpha) \\
&\cdot \prod_{s=1}^{\bar{\mathbf{S}}} \prod_{i=1}^{N_s} \Pr(z_{si} | l_{si}, \theta_l) \Pr(w_{si} | \phi_{z_{si}}) \Pr(t_{si} | \xi_{z_{si}})
\end{aligned}$$

Next, marginalize over all ϕ, ξ and θ:

$$
\begin{aligned}
\Pr(\mathbf{w},\mathbf{t},\mathbf{v},\mathbf{z};\alpha,\beta,\gamma) &= \int_{\phi}\int_{\theta}\int_{\xi}\Pr(\mathbf{w},\mathbf{t},\mathbf{v},\mathbf{z},\phi,\theta,\xi;\alpha,\beta,\gamma)d\xi d\theta d\phi \\
&= \int_{\phi}\prod_{z=1}^{K}\Pr(\phi_z;\beta)\prod_{s=1}^{\bar{\mathbf{S}}}\prod_{i=1}^{N_s}\Pr(w_{si}|\phi_{z_{si}})d\phi \\
&\cdot \int_{\xi}\prod_{z=1}^{K}\Pr(\xi_z;\gamma)\prod_{s=1}^{\bar{\mathbf{S}}}\prod_{i=1}^{N_s}\Pr(t_{si}|\xi_{z_{si}})d\xi \\
&\cdot \int_{\theta}\prod_{v\in\mathcal{V}_{\mathcal{A}_D}}\Pr(\theta_v;\alpha)\prod_{s=1}^{\bar{\mathbf{S}}}\prod_{i=1}^{N_s}\Pr(z_{si}|v_{si},\theta_{v_{si}})d\theta \\
&= \int_{\phi}\prod_{z=1}^{K}\Pr(\phi_z;\beta)\prod_{s=1}^{\bar{\mathbf{S}}}\prod_{i=1}^{N_s}\Pr(w_{si}|\phi_{z_{si}})d\phi \\
&\cdot \int_{\xi}\prod_{z=1}^{K}\Pr(\xi_z;\gamma)\prod_{s=1}^{\bar{\mathbf{S}}}\prod_{i=1}^{N_s}\Pr(t_{si}|\xi_{z_{si}})d\xi \\
&\cdot \int_{\theta}\prod_{v\in\mathcal{V}_{\mathcal{A}_D}}\Pr(\theta_v;\alpha)\prod_{s=1}^{\bar{\mathbf{S}}}\prod_{i=1}^{N_s}\Pr(z_{si}|v_{si},\theta_{v_{si}})d\theta
\end{aligned}
$$

Now, one can focus on the different integrals in the expression presented above. Start with the integral over ϕ.

$$\begin{aligned}
&\int_\phi \prod_{z=1}^{K} \Pr(\phi_z;\beta) \prod_{s=1}^{\bar{\mathbf{S}}} \prod_{i=1}^{N_s} \Pr(w_{si}|\phi_{z_{si}}) d\phi \\
&= \prod_{z=1}^{K} \int_{\phi_z} \Pr(\phi_z;\beta) \prod_{s=1}^{\bar{\mathbf{S}}} \prod_{i=1}^{N_s} \Pr(w_{si}|\phi_{z_{si}}) d\phi_z \\
&= \prod_{z=1}^{K} \int_{\phi_z} \frac{\Gamma(\sum_{r=1}^{V}\beta_r)}{\prod_{r=1}^{V}\Gamma(\beta_r)} \prod_{r=1}^{V} \phi_{zr}^{\beta_r-1} \prod_{r=1}^{V} \phi_{zr}^{n^z_{(\cdot),r}} d\phi_z \\
&= \prod_{z=1}^{K} \int_{\phi_z} \frac{\Gamma(\sum_{r=1}^{V}\beta_r)}{\prod_{r=1}^{V}\Gamma(\beta_r)} \prod_{r=1}^{V} \phi_{zr}^{\beta_r+n^z_r-1} d\phi_z \\
&= \prod_{z=1}^{K} \frac{\Gamma(\sum_{r=1}^{V}\beta_r)}{\prod_{r=1}^{V}\Gamma(\beta_r)} \frac{\prod_{r=1}^{V}\Gamma(n^z_r+\beta_r)}{\Gamma(\sum_{r=1}^{V} n^z_r+\beta_r)}
\end{aligned}$$

where n^z_r denotes the number of times word r was associated with topic z across all sources and entries. Similarly for the ξ part:

$$\begin{aligned}
&\int_\xi \prod_{z=1}^{K} \Pr(\xi_z;\gamma) \prod_{s=1}^{\bar{\mathbf{S}}} \prod_{i=1}^{N_s} \Pr(t_{si}|\xi_{z_{si}}) d\xi \\
&= \prod_{z=1}^{K} \int_{\xi_z} \Pr(\xi_z;\gamma) \prod_{s=1}^{\bar{\mathbf{S}}} \prod_{i=1}^{N_s} \Pr(t_{si}|\xi_{z_{si}}) d\xi \\
&= \prod_{z=1}^{K} \frac{\Gamma(\sum_{t=1}^{T}\gamma_t)}{\prod_{t=1}^{T}\Gamma(\gamma_t)} \frac{\prod_{t=1}^{T}\Gamma(m^z_t+\gamma_t)}{\Gamma(\sum_{t=1}^{T} m^z_t+\gamma_t)}
\end{aligned}$$

where m^z_t denotes the number of times time-point t was associated with topic z across all sources. Finally, focus on the θ integral. Following a similar analysis we have:

$$\int_{\theta} \prod_{v \in \mathcal{V}_{\mathcal{A}_D}} \Pr(\theta_v; \alpha) \prod_{s=1}^{\bar{\mathbf{S}}} \prod_{i=1}^{N_s} \Pr(z_{si} | v_{si}, \theta_{v_{si}}) d\theta$$
$$= \prod_{v \in \mathcal{V}_{\mathcal{A}_D}} \int_{\theta_v} \Pr(\theta_a; \alpha) \prod_{s=1}^{\bar{\mathbf{S}}} \prod_{i=1}^{N_s} \Pr(z_{si} | v_{si}, \theta_{v_{si}}) d\theta_l$$
$$= \prod_{v \in \mathcal{V}_{\mathcal{A}_D}} \int_{\theta_v} \frac{\Gamma(\sum_{z=1}^{K} \alpha_z)}{\prod_{z=1}^{K} \Gamma(\alpha_z)} \prod_{z=1}^{K} \theta_{vz}^{\alpha_z - 1} \prod_{z=1}^{K} \theta_{vz}^{o_v^z} d\theta_v$$
$$= \prod_{v \in \mathcal{V}_{\mathcal{A}_D}} \int_{\theta_v} \frac{\Gamma(\sum_{z=1}^{K} \alpha_z)}{\prod_{z=1}^{K} \Gamma(\alpha_z)} \prod_{z=1}^{K} \theta_{vz}^{\alpha_z + o_v^z - 1} d\theta_v$$
$$= \prod_{v \in \mathcal{V}_{\mathcal{A}_D}} \frac{\Gamma(\sum_{z=1}^{K} \alpha_z)}{\prod_{z=1}^{K} \Gamma(\alpha_z)} \frac{\prod_{z=1}^{K} \Gamma(o_v^z + \alpha_z)}{\Gamma(\sum_{z=1}^{K} o_v^z + \alpha_z)}$$

where o_v^z denotes the number of times value v was associated with topic z across all sources and their entries. Eventually the joint distribution is given by:

$$\Pr(\mathbf{w}, \mathbf{t}, \mathbf{v}, \mathbf{z}; \alpha, \beta, \gamma) = \prod_{z=1}^{K} \frac{\Gamma(\sum_{r=1}^{V} \beta_r)}{\prod_{r=1}^{V} \Gamma(\beta_r)} \frac{\prod_{r=1}^{V} \Gamma(n_r^z + \beta_r)}{\Gamma(\sum_{r=1}^{V} n_r^z + \beta_r)}$$
$$\cdot \prod_{z=1}^{K} \frac{\Gamma(\sum_{t=1}^{T} \gamma_t)}{\prod_{t=1}^{T} \Gamma(\gamma_t)} \frac{\prod_{t=1}^{T} \Gamma(m_t^z + \gamma_t)}{\Gamma(\sum_{t=1}^{T} m_t^z + \gamma_t)}$$
$$\cdot \prod_{v \in \mathcal{V}_{\mathcal{A}_D}} \frac{\Gamma(\sum_{z=1}^{K} \alpha_z)}{\prod_{z=1}^{K} \Gamma(\alpha_z)} \frac{\prod_{z=1}^{K} \Gamma(o_v^z + \alpha_z)}{\Gamma(\sum_{z=1}^{K} o_v^z + \alpha_z)}$$

Gibbs sampling is used to approximate the conditional distribution $\Pr(\mathbf{z} | \mathbf{w}, \mathbf{t}, \mathbf{v}; \alpha, \beta, \gamma, \Psi)$. Using the chain rule one gets the following for the conditional probability:

$$\begin{aligned}\Pr(z_{si}|\mathbf{w},\mathbf{t},\mathbf{v},\mathbf{z}_{-si};\alpha,\beta,\gamma) &= \frac{\Pr(z_{si},w_{si},t_{si},v_{si}|\mathbf{w}_{-si},\mathbf{t}_{-si},\mathbf{v}_{-si},\mathbf{z}_{-si};\alpha,\beta,\gamma)}{\Pr(w_{si},t_{si},v_{si}|\mathbf{w}_{-si},\mathbf{t}_{-si},\mathbf{v}_{-si},\mathbf{z}_{-si};\alpha,\beta,\gamma)} \\ &\propto \frac{n_{w_{si}}^{k,-(s,i)}+\beta_{w_{si}}}{\sum_{r=1}^{V} n_r^{k,-(s,i)}+\beta_r} \cdot \frac{m_{t_{si}}^{k,-(s,i)}+\gamma_{t_{si}}}{\sum_{t=1}^{T} m_t^{k,-(s,i)}+\gamma_t} \\ &\cdot \frac{o_{v_{si}}^{k,-(s,i)}+\alpha_{v_{si}}}{\sum_{v\in\mathcal{V}_{\mathcal{A}_D}} o_v^{k,-(s,i)}+\alpha_v}\end{aligned}$$

where $-si$ in the superscript indicates that the current example has been excluded by the count summations.

A.2 Proof of Theorem 1

Consider the coverage estimator introduced in Equation 3.25. It is easy to see that the only terms which depend on the set of selected sources S_I are $\mathsf{Cov}(\mathsf{F}(\mathsf{S_I});\mathsf{t_0})$ and $\Pr[\mathsf{Ins}; F(S_I), t, \tau]$. The estimator $\mathsf{Cov}^*(\mathsf{F}(\mathsf{S_I});\mathsf{t})$ is expressed as a non-negative linear combination of these terms. Thus, it suffices to show that each of these terms corresponds to non-decreasing submodular function. First, focus on $\mathsf{Cov}(\mathsf{F}(\mathsf{S_I});\mathsf{t_0})$. We have that:

$$\mathsf{Cov}(\mathsf{F}(\mathsf{S_I});\mathsf{t_0}) = \frac{\mathsf{Up}(\mathsf{F}(\mathsf{S_I});\mathsf{t_0}) + \mathsf{Out}(\mathsf{F}(\mathsf{S_I});\mathsf{t_0})}{|\mathcal{D}\mathsf{P}|_{\mathsf{t_0}}} \tag{A.1}$$

The denominator does not depend on S_I and is a positive constant. Thus, one only needs to prove that $\mathsf{Up}(\mathsf{F}(\mathsf{S_I});\mathsf{t_0}) + \mathsf{Out}(\mathsf{F}(\mathsf{S_I});\mathsf{t_0})$ is a non-decreasing submodular function. Let $\mathcal{C}(F(S_I);t_0) = \mathsf{Up}(\mathsf{F}(\mathsf{S_I});\mathsf{t_0}) + \mathsf{Out}(\mathsf{F}(\mathsf{S_I});\mathsf{t_0})$ denote the covered data items in $F(S_I)$ at time t_0. From Section 3.5.2.1 one has that:

$$\mathcal{C}(F(S_I); t_0) = |\bigvee_{S \in S_I} B_S^{cov}|$$

where $|\cdot|$ denotes the number of bits set to one in the signature given as input. Let $\mathcal{C}(F(S_I \cup S'); t_0)$ denote the covered items when a new source S' is added in S_I:

$$\begin{aligned}\mathcal{C}(F(S_I \cup S'); t_0) &= |\bigvee_{S \in S_I} B_S^{cov} \vee B_{S'}^{cov}| \\ &= |\bigvee_{S \in S_I} B_S^{cov}| + |B_{S' \setminus S_I}^{cov}| \geq \mathcal{C}(F(S_I); t_0)\end{aligned}$$

where $|B_{S' \setminus S_I}^{cov}|$ denotes the number of bits set to one only in $B_{S'}^{cov}$ and not $\bigvee_{S \in S_I} B_S^{cov}$. Thus the number of covered items is a non-decreasing function. Next, I show that the number of covered items is a submodular function. First, I compute the quantity $\Delta\mathsf{Cov}^*(\mathsf{F}(\mathsf{S_A}); \mathsf{t})$:

$$\begin{aligned}\Delta\mathsf{Cov}^*(\mathsf{F}(\mathsf{S_A}); \mathsf{t}) &= \mathsf{Cov}^*(\mathsf{F}(\mathsf{S_A} \cup \{\mathsf{S'}\}); \mathsf{t}) - \mathsf{Cov}^*(\mathsf{F}(\mathsf{S_A}); \mathsf{t}) \\ &= |\bigvee_{S \in S_I} B_S^{cov} \vee B_{S'}^{cov}| - |\bigvee_{S \in S_I} B_S^{cov}| \\ &= |B_{S' \setminus S_A}^{cov}|\end{aligned}$$

Similarly, $\Delta\mathsf{Cov}^*(\mathsf{F}(\mathsf{S_B}); \mathsf{t}) = |\mathsf{B}_{\mathsf{S'} \setminus \mathsf{S_B}}^{\mathsf{cov}}|$ and:

$$\begin{aligned}|B^{cov}_{S'\setminus S_B}| &= |B^{cov}_{S'\setminus(S_A\cup(S_B\setminus S_A))}| \\ &= |B^{cov}_{(S'\setminus S_A)\cap(S'\setminus(S_B\setminus S_A))}| \\ &= |B^{cov}_{(S'\setminus S_A)} \wedge B^{cov}_{(S'\setminus(S_B\setminus S_A))}| \\ &\leq |B^{cov}_{(S'\setminus S_A)}| \end{aligned} \tag{A.2}$$

Directly from the above equation one has that $|B^{cov}_{S'\setminus S_B}| \leq |B^{cov}_{S'\setminus S_A}|$. Thus, the number of covered items is a submodular function. From the analysis presented above, the coverage at time t_0 is a non-decreasing submodular function.

For each of the terms $\Pr[\mathsf{Ins}; S_I, t, \tau, P]$ one has the following for the set of sources S_A and S_B. From Equation 3.22:

$$\Pr[\mathsf{Ins}; S_A, t, \tau, P] = 1 - \prod_{S\in S_A} \left(1 - G_i^S(T_S(t), \tau; P)\right)$$

Similarly for S_B. The probability corresponding to S_B can be expressed as:

$$\begin{aligned}\Pr[\mathsf{Ins}; S_B, t, \tau, P] &= 1 - \prod_{S\in S_B} \left(1 - G_i^S(T_S(t), \tau; P)\right) \\ = 1 - \prod_{S\in S_A} \left(1 - G_i^S(T_S(t), \tau; P)\right) &\prod_{S\in S_B\setminus S_A} \left(1 - G_i^S(T_S(t), \tau; P)\right)\end{aligned}$$

Since $\prod_{S\in S_B\setminus S_A} \left(1 - G_i^S(T_S(t), \tau; P)\right) \leq 1$ one has that:

$$\Pr[\mathsf{Ins}; S_B, t, \tau, P] \geq \Pr[\mathsf{Ins}; S_A, t, \tau, P]$$

thus proving that each term $\Pr[\mathsf{Ins}; S_I, t, \tau, P]$ corresponds to a non-decreasing function. Next, we prove that this function is submodular:

$$
\begin{aligned}
&\Pr[\mathsf{Ins}; S_B \cup \{S'\}, t, \tau, P] - \Pr[\mathsf{Ins}; S_B, t, \tau, P] \\
&= \prod_{S \in S_B} \big(1 - G_i^S(T_S(t), \tau; P)\big) - \prod_{S \in S_B \cup \{S'\}} \big(1 - G_i^S(T_S(t), \tau; P)\big) \\
&= G_{i,S'}(T_{S'}(t), \tau; P) \prod_{S \in S_B} \big(1 - G_i^S(T_S(t), \tau; P)\big) \\
&= G_{i,S'}(T_{S'}(t), \tau; P) \prod_{S \in S_A} \big(1 - G_i^S(T_S(t), \tau; P)\big) * \prod_{S \in S_B \setminus S_A} \big(1 - G_i^S(T_S(t), \tau; P)\big) \\
&\leq \Pr[\mathsf{Ins}; S_A \cup \{S'\}, t, \tau, P] - \Pr[\mathsf{Ins}; S_A, t, \tau, P]
\end{aligned}
$$

proving that $\Pr[\mathsf{Ins}; S_I, t, \tau, P]$ is submodular.

A.3 Proof of Theorem 3

To derive the new estimator we make used of the generalized jackknife procedure for species richness estimation [75]. Given two (biased) estimators of S, say $\hat{S}_1$ and $\hat{S}_2$, let R be the ratio of their biases:

$$R = \frac{E(\hat{S}_1) - S}{E(\hat{S}_2) - S} \tag{A.3}$$

By the generalized jackknife procedure, we can completely eliminate the bias resulting from either $\hat{S}_1$ or $\hat{S}_2$ via

$$S = G(\hat{S}_1, \hat{S}_2) = \frac{\hat{S}_1 - R\hat{S}_2}{1 - R} \tag{A.4}$$

provided the ratio of biases R is known. Yet, R is unknown and needs to be estimated.

Let D_n denote the number of unique entities in a unified sample of size n. We consider the following two biased estimators of S: $\hat{S}_1 = D_n$ and $\hat{S}_2 = \sum_{j=1}^{n} D_{n-1}(j)/n = D_n - f_1/n$ where $D_{n-1}(j)$ is the number of species discovered with the jth observation removed from the original sample. Replacing these estimators in Equation A.4 gives us:

$$S = D_n + \frac{R}{1-R}\frac{f_1}{n} \tag{A.5}$$

Similarly, for a sample of increased size $n+m$ we have:

$$S = D_{n+m} + \frac{R'}{1-R'}\frac{f_1'}{n+m} \tag{A.6}$$

where R' is the ratio of the biases and f_1' the number of singleton entities for the increased sample. Let $K = \frac{R}{1-R}$ and $K' = \frac{R'}{1-R'}$. Taking the difference of the previous two equations we have:

$$D_{n+m} - D_n = K\frac{f_1}{n} - K'\frac{f_1'}{n+m} \tag{A.7}$$

Therefore, we have:

$$G = K\frac{f_1}{n} - K'\frac{f_1'}{n+m} \tag{A.8}$$

We need to estimate K, K' and f_1'. We start with f_1', which denotes the number of singleton entities in the increased sample of size $n+m$. Notice, that f_1' is not known since we have not obtained the increased sample yet, so we need to express it in terms of f_1, i.e., the number of singletons, in the running sample of size n. We have:

$$f'_1 = G + f_1 - f_1^c \tag{A.9}$$

where f_1^c denotes the number of old singleton entities from the sample of size n that appeared in the additional query of size m. Let E_1 denote the set of singleton entities in the old sample of size n. We approximate f_1^c by its expected value:

$$\hat{f}_1^c = \sum_{e \in E_1} \mathrm{Pr}[\text{e appears in query of size } m] \tag{A.10}$$

We compute the probability of an old singleton entity appearing in an additional query as follows. Let p_e denote the popularity of entity e. As described before, an additional query of size m corresponds to taking a sample of size m from the underlying entity population without replacement. However, m is significantly smaller compared to the size of the underlying population, thus, we can consider a that taking a sample of size m corresponds to taking a sample *with replacement*. Following this we have that:

$$\mathrm{Pr}[\text{e appears in query of size } m] = 1 - (1 - p_e)^m \tag{A.11}$$

Following a standard approach in the species estimation literature we assume that the popularity of retrieving a singleton entity again is the same for all singleton entities. This popularity can be computed using the corresponding Good-Turing estimator considering the running sample. We have:

$$\forall e \in E_1, p_e = p_1 = \hat{\theta}(1) = \frac{1}{n} 2 \frac{f_2}{f_1} \tag{A.12}$$

where f_2 is the number of entities that appear twice in the sample and f_1 is the number of singletons. Eventually we have that:

$$\hat{f}_1^c = f_1(1-(1-p_1)^m) \tag{A.13}$$

and

$$f_1' = G + f_1(1-p_1)^m \tag{A.14}$$

Replacing the last equation in Equation A.8 we have:

$$\begin{aligned}
G &= K\frac{f_1}{n} - K'\frac{G + f_1(1-p_1)^m}{n+m} \\
G &= K\frac{f_1}{n} - K'\frac{G}{n+m} - K'\frac{f_1(1-P)}{n+m} \\
G(1+\frac{K'}{n+m}) &= K\frac{f_1}{n} - K'\frac{f_1(1-P)}{n+m} \\
G &= \frac{1}{(1+\frac{K'}{n+m})}(K\frac{f_1}{n} - K'\frac{f_1(1-p_1)^m}{n+m})
\end{aligned}$$

A.4 Proof of Lemma 1

We will denote $K(n+m)$ as K'. By definition we have that $K = \frac{\sum_{i=1}^{S}(1-p_i)^n}{\sum_{i=1}^{S}p_i(1-p_i)^{n-1}}$ and $K' = \frac{\sum_{i=1}^{S}(1-p_i)^{n+m}}{\sum_{i=1}^{S}p_i(1-p_i)^{n+m-1}}$. We want to show that:

$$\frac{\sum_{i=1}^{S}(1-p_i)^{n+m}}{\sum_{i=1}^{S}p_i(1-p_i)^{n+m-1}} \geq \frac{\sum_{i=1}^{S}(1-p_i)^{n}}{\sum_{i=1}^{S}p_i(1-p_i)^{n-1}}$$

$$\sum_{i=1}^{S}(1-p_i)^{n+m}\sum_{j=1}^{S}p_j(1-p_j)^{n-1} \geq \sum_{i=1}^{S}p_i(1-p_i)^{n+m-1}\sum_{j=1}^{S}(1-p_j)^{n}$$

$$\sum_{i,j:i\prec j}[(1-p_i)^{n+m}p_j(1-p_j)^{n-1} - p_i(1-p_i)^{n+m-1}(1-p_j)^{n}+$$

$$+(1-p_j)^{n+m}p_i(1-p_i)^{n-1} - p_j(1-p_j)^{n+m-1}(1-p_i)^{n}] \geq 0$$

$$\sum_{i,j:i\prec j}[(1-p_i)^{n-1}(1-p_j)^{n-1}(p_j-p_i)((1-p_i)^{m}-(1-p_j)^{m}) \geq 0 \qquad \text{(A.15)}$$

But the last inequality always holds since each term of the summation is positive. In particular, if $p_j \geq p_i$ then also $1-p_i \geq 1-p_j$ and if $p_j \leq p_i$ then $1-p_i \leq 1-p_j$.

Bibliography

[1] International society for infectious diseases. *http://www.promedmail.org/*.

[2] Mechanical Turk. *http://mturk.com*.

[3] Yael Amsterdamer, Susan B. Davidson, Tova Milo, Slava Novgorodov, and Amit Somech. OASSIS: query driven crowd mining. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pages 589–600, 2014.

[4] Yael Amsterdamer, Yael Grossman, Tova Milo, and Pierre Senellart. Crowd mining. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*, pages 241–252, 2013.

[5] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1), 2012.

[6] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2002.

[7] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary 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, pages 722–735, 2007.

[8] R. Iris Bahar, Erica A. Frohm, Charles M. Gaona, Gary D. Hachtel, Enrico Macii, Abelardo Pardo, and Fabio Somenzi. Algebraic decision diagrams and their applications. In *Proceedings of the 1993 IEEE/ACM International Conference on Computer-aided Design*, ICCAD '93, pages 188–191, 1993.

[9] Magdalena Balazinska, Bill Howe, and Dan Suciu. Data markets in the cloud: An opportunity for the database community. *Proc. VLDB Endow.*, 4(12):1482–1485, 2011.

[10] Kedar Bellare, Suresh Iyengar, Aditya G. Parameswaran, and Vibhor Rastogi. Active sampling for entity matching. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 1131–1139, 2012.

[11] Laure Berti-Equille. Measuring and modelling data quality for quality-awareness in data mining. In *Quality Measures in Data Mining*, volume 43 of *Studies in Computational Intelligence*, pages 101–126. 2007.

[12] Laure Berti-Equille, Anish Das Sarma, Xin Dong, Amélie Marian, and Divesh Srivastava. Sailing the information ocean with awareness of currents: Discovery and application of source dependence. In *CIDR*, 2009.

[13] Gaurav Bhalotia, Arvind Hulgeri, Charuta Nakhe, Soumen Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using BANKS. In *Proceedings of the 18th International Conference on Data Engineering, San Jose, CA, USA, February 26 - March 1, 2002*, pages 431–440, 2002.

[14] David M. Blei and John D. Lafferty. Dynamic topic models. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 113–120. ACM, 2006.

[15] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[16] Beate Bollig and Ingo Wegener. Improving the variable ordering of OBDDs is NP-complete. *IEEE Trans. Comput.*, 45(9):993–1002, September 1996.

[17] Craig Boutilier, Nir Friedman, Moises Goldszmidt, and Daphne Koller. Context-specific independence in bayesian networks. In *Proceedings of the Twelfth International Conference on Uncertainty in Artificial Intelligence*, UAI'96, pages 115–123, 1996.

[18] John S. Brownstein, Clark C. Freifeld, Ben Y. Reis, and Kenneth D. Mandl. Surveillance Sans Frontières: Internet-Based Emerging Infectious Disease Intelligence and the HealthMap Project. *PLoS Medicine*, 5(7), 2008.

[19] Randal E. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Comput. Surv.*, 24(3):293–318, September 1992.

[20] Michael J. Cafarella and Alon Y. Halevy. Web data management. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, 2011.

[21] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, 2010.

[22] Tiziana Catarci and Maurizio Lenzerini. Representing and using interschema knowledge in cooperative information systems. *Journal of Intelligent and Cooperative Information Systems*, 2:375–398, 1993.

[23] Anne Chao and Shen-Ming Lee. Estimating the number of classes via sample coverage. *Journal of the American Statistical Association*, 87(417):pp. 210–217, 1992.

[24] Mark Chavira and Adnan Darwiche. Compiling Bayesian networks with local structure. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, pages 1306–1312. Morgan Kaufmann Publishers Inc., 2005.

[25] Mark Chavira and Adnan Darwiche. Compiling Bayesian networks using variable elimination. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2443–2449. Morgan Kaufmann Publishers Inc., 2007.

[26] Mark Chavira and Adnan Darwiche. On probabilistic inference by weighted model counting. *Artif. Intell.*, 172(6-7):772–799, April 2008.

[27] Mark Chavira, Adnan Darwiche, and Manfred Jaeger. Compiling relational Bayesian networks for exact inference. *Int. J. Approx. Reasoning*, 42(1-2):4–20, May 2006.

[28] Junghoo Cho and Hector Garcia-Molina. Effective page refresh policies for web crawlers. *ACM Trans. Database Syst.*, 28(4):390–426, December 2003.

[29] Courtney D. Corley, Diane J. Cook, Armin R. Mikler, and Karan P. Singh. Text and structural data mining of influenza mentions in web and social media. *International Journal of Environmental Research and Public Health*, 7(2), 2010.

[30] Aron Culotta. Towards detecting influenza epidemics by analyzing twitter messages. In *Proceedings of the First Workshop on Social Media Analytics*, SOMA '10, pages 115–122, 2010.

[31] Nilesh Dalvi, Christopher Ré, and Dan Suciu. Probabilistic databases: Diamonds in the dirt. *Commun. ACM*, 52(7):86–94, July 2009.

[32] Nilesh Dalvi, Karl Schnaitter, and Dan Suciu. Computing query probability with incidence algebras. In *Proceedings of the Twenty-ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '10, pages 203–214. ACM, 2010.

[33] Nilesh Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. *The VLDB Journal*, 16(4):523–544, October 2007.

[34] Adnan Darwiche. A logical approach for factoring belief networks. In *Proceedings of the Eights International Conference on Principles and Knowledge Representation and Reasoning*. Morgan Kaufmann, 2001.

[35] Adnan Darwiche. A differential approach to inference in Bayesian networks. *J. ACM*, 50(3):280–305, May 2003.

[36] Adnan Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.

[37] Anish Das Sarma, Lujun Fang, Nitin Gupta, Alon Halevy, Hongrae Lee, Fei Wu, Reynold Xin, and Cong Yu. Finding related tables. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 817–828. ACM, 2012.

[38] Rina Dechter. Bucket elimination: A unifying framework for probabilistic inference. In *UAI '96: Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence, Reed College, Portland, Oregon, USA, August 1-4, 1996*, pages 211–219, 1996.

[39] AnHai Doan, Alon Halevy, and Zachary Ives. *Data Matching*. Morgan Kaufmann Publishers Inc., 1st edition, 2012.

[40] AnHai Doan, Alon Halevy, and Zachary Ives. *Describing Data Sources*. Morgan Kaufmann Publishers Inc., 1st edition, 2012.

[41] AnHai Doan, Alon Halevy, and Zachary Ives. *Principles of Data Integration*. Morgan Kaufmann Publishers Inc., 1st edition, 2012.

[42] AnHai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy. Learning to map between ontologies on the semantic web. In *Proceedings of the 11th International Conference on World Wide Web*, WWW '02, pages 662–673, 2002.

[43] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 601–610, 2014.

[44] Xin Dong, Alon Y. Halevy, and Cong Yu. Data integration with uncertainty. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB '07, pages 687–698. VLDB Endowment, 2007.

[45] Xin Luna Dong, Laure Berti-Equille, and Divesh Srivastava. Integrating conflicting data: The role of source dependence. *Proc. VLDB Endow.*, 2(1):550–561, August 2009.

[46] Xin Luna Dong, Laure Berti-Equille, and Divesh Srivastava. Truth discovery and copying detection in a dynamic world. *Proc. VLDB Endow.*, 2(1):562–573, August 2009.

[47] Xin Luna Dong, Barna Saha, and Divesh Srivastava. Less is more: selecting sources wisely for integration. In *Proceedings of the 39th international conference on Very Large Data Bases*, PVLDB'13, pages 37–48. VLDB Endowment, 2013.

[48] Maximilian Dylla, Iris Miliaraki, and Martin Theobald. A temporal-probabilistic database model for information extraction. *PVLDB*, 6(14):1810–1821, 2013.

[49] Ruediger Ebendt, Goeschwin Fey, and Rolf Drechsler. *Advanced BDD Optimization*. Springer, 2005.

[50] Eyal Even-Dar, Shie Mannor, and Yishay Mansour. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *J. Mach. Learn. Res.*, 7:1079–1105, December 2006.

[51] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. In *Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '01, pages 102–113, 2001.

[52] Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. *SIAM J. Comput.*, 40(4):1133–1153, July 2011.

[53] I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64:1183–1210, 1969.

[54] Thomas A. Feo and Mauricio G. C. Resende. Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, 6, 1995.

[55] Jonathan Finger and Neoklis Polyzotis. Robust and efficient algorithms for rank join evaluation. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, pages 415–428, 2009.

[56] Michael J. Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin. CrowdDB: Answering queries with crowdsourcing. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 61–72, 2011.

[57] Clark C. Freifeld, Kenneth D. Mandl, Ben Y. Reis, and John S. Brownstein. HealthMap: Global Infectious Disease Monitoring through Automated Classification and Visualization of Internet Media Reports. *JAMIA*, 15, 2008.

[58] Drew Fudenberg and Jean Tirole. *Game Theory*. MIT Press, Cambridge, MA, 1991. Translated into Chinesse by Renin University Press, Bejing: China.

[59] R. G. Gallager. *Discrete Stochastic Processes*. Kluwer Academic Publishers, Boston, 1996.

[60] Lise Getoor and Ashwin Machanavajjhala. Entity resolution: Theory, practice & open challenges. In *International Conference on Very Large Data Bases*, 2012.

[61] Lise Getoor and Ashwin Machanavajjhala. Entity resolution for Big Data. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 1527–1527, New York, NY, USA, 2013. ACM.

[62] Jeremy Ginsberg, Matthew H. Mohebbi, Rajan S. Patel, Lynnette Brammer, Mark S. Smolinski, and Larry Brilliant. Detecting influenza epidemics using search engine query data. *Nature*, 457, 2009.

[63] Parke Godfrey, Ryan Shipley, and Jarek Gryz. Algorithms and analyses for maximal vector computation. *The VLDB Journal*, 16(1):5–28, 2007.

[64] Ryan G. Gomes, Peter Welinder, Andreas Krause, and Pietro Perona. Crowdclustering. In J. Shawe-Taylor, R.S. Zemel, P.L. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 558–566. Curran Associates, Inc., 2011.

[65] Hector Gonzalez, Alon Y. Halevy, Christian S. Jensen, Anno Langen, Jayant Madhavan, Rebecca Shapley, and Warren Shen. Google fusion tables: data management, integration and collaboration in the cloud. In *Proceedings of the 1st ACM Symposium on Cloud Computing, SoCC 2010, Indianapolis, Indiana, USA, June 10-11, 2010*, pages 175–180, 2010.

[66] Luis Gravano, Panagiotis G. Ipeirotis, Nick Koudas, and Divesh Srivastava. Text joins in an rdbms for web data integration. In *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, pages 90–101, 2003.

[67] Todd J. Green and Val Tannen. Models for incomplete and probabilistic information. In *Proceedings of the 2006 International Conference on Current Trends in Database Technology*, EDBT'06, pages 278–296. Springer-Verlag, 2006.

[68] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220, June 1993.

[69] Stephen Guo, Aditya Parameswaran, and Hector Garcia-Molina. So who won?: Dynamic max discovery with the crowd. In *Proceedings of the 2012 ACM SIG-*

MOD International Conference on Management of Data, SIGMOD '12, pages 385–396, New York, NY, USA, 2012. ACM.

[70] Alon Halevy, Michael Franklin, and David Maier. Principles of dataspace systems. In *Proceedings of the Twenty-fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '06, pages 1–9, 2006.

[71] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD, 2000.

[72] Oktie Hassanzadeh, Ken Q. Pu, Soheil Hassas Yeganeh, Renée J. Miller, Lucian Popa, Mauricio A. Hernández, and Howard Ho. Discovering linkage points over web data. *PVLDB*, 6, 2013.

[73] Katherine A. Heller, Krysta M. Svore, Angelos D. Keromytis, and Salvatore J. Stolfo. One class support vector machines for detecting anomalous windows registry accesses. In *In Proc. of the workshop on Data Mining for Computer Security*, 2003.

[74] Joseph M. Hellerstein, Michael Stonebraker, and Rick Caccia. Independent, open enterprise data integration. *IEEE Data Eng. Bull.*, 22(1):43–49, 1999.

[75] James F Heltshe and Nancy E Forrester. Estimating species richness using the jackknife procedure. *Biometrics*, pages 1–11, 1983.

[76] Thomas N. Herzog, Fritz J. Scheuren, and William E. Winkler. *Data quality and record linkage techniques*. Springer, 2007.

[77] Joaquín Hortal, Paulo AV Borges, and Clara Gaspar. Evaluating the performance of species richness estimators: sensitivity to sample grain size. *J. of Animal Ecology*, 75(1):274–287, 2006.

[78] Ting Hua, Chang-Tien Lu, Naren Ramakrishnan, Feng Chen, Jaime Arredondo, David Mares, and Kristen Summers. Analyzing civil unrest through social media. *Computer*, 46(12):80–84, 2013.

[79] Wen-Han Hwang and Tsung-Jen Shen. Small-sample estimation of species richness applied to forest communities. *Biometrics*, 66(4):1052–1060, 2010.

[80] Ihab F. Ilyas, Walid G. Aref, and Ahmed K. Elmagarmid. Supporting top-k join queries in relational databases. *The VLDB Journal*, 13(3):207–221, September 2004.

[81] Abhay Jha, Dan Olteanu, and Dan Suciu. Bridging the gap between intensional and extensional query evaluation in probabilistic databases. In *Proceedings of the 13th International Conference on Extending Database Technology*, EDBT '10, pages 323–334, New York, NY, USA, 2010. ACM.

[82] Abhay Jha and Dan Suciu. Knowledge compilation meets database theory: Compiling queries to decision diagrams. In *Proceedings of the 14th International Conference on Database Theory*, ICDT '11, pages 162–173. ACM, 2011.

[83] Lili Jiang, Yafang Wang, Johannes Hoffart, and Gerhard Weikum. Crowdsourced entity markup. In *CrowdSem*, 2013.

[84] Xin Jin, Nan Zhang, and Gautam Das. Attribute domain discovery for hidden web databases. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 553–564, New York, NY, USA, 2011. ACM.

[85] Colleen B. Jonsson, Luiz Tadeu Moraes Figueiredo, and Olli Vapalahti. A global perspective on hantavirus ecology, epidemiology and disease. *Clinical Microbiology Review*, 23(2), 2010.

[86] Varun Kacholia, Shashank Pandit, Soumen Chakrabarti, S. Sudarshan, Rushi Desai, and Hrishikesh Karambelkar. Bidirectional expansion for keyword search on graph databases. In *Proceedings of the 31st International Conference on Very Large Data Bases*, VLDB '05, pages 505–516, 2005.

[87] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. Wrangler: Interactive visual specification of data transformation scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 3363–3372, 2011.

[88] E. L. Kaplan and Paul Meier. Nonparametric estimation from incomplete observations. *JASA*, 53:457–481, 1958.

[89] Gjergji Kasneci, Maya Ramanath, Mauro Sozio, Fabian M. Suchanek, and Gerhard Weikum. Star: Steiner-tree approximation in relationship graphs. In *Proceedings of the 2009 IEEE International Conference on Data Engineering*, ICDE '09, pages 868–879, 2009.

[90] Shehroz S. Khan and Michael G. Madden. One-class classification: taxonomy of study and review of techniques. *Knowledge Eng. Review*, 29(3):345–374, 2014.

[91] Christoph Koch. Approximating predicates and expressive queries on probabilistic databases. In *Proceedings of the Twenty-seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '08, pages 99–108. ACM, 2008.

[92] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[93] S. Kumar Kondredi, P. Triantafillou, and G. Weikum. Combining information extraction and human computing for crowdsourced knowledge acquisition. In *30th IEEE International Conference on Data Engineering*, ICDE, 2014.

[94] Donald Kossmann, Frank Ramsak, and Steffen Rost. Shooting stars in the sky: An online algorithm for skyline queries. In *Proceedings of the 28th International Conference on Very Large Data Bases*, VLDB '02, pages 275–286, 2002.

[95] Jon Lee, Vahab S. Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. Non-monotone submodular maximization under matroid and knapsack constraints. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 323–332. ACM, 2009.

[96] Kalev Leetaru and Philip Schrodt. Gdelt: Global data on events, language, and tone, 1979-2012. *Inter. Studies Association Annual Conf.*, 2013.

[97] Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '02, pages 233–246. ACM, 2002.

[98] Hector J. Levesque and Ronald J. Brachman. Expressiveness and tractability in knowledge representation and reasoning1. *Computational Intelligence*, 3(1):78–93, 1987.

[99] Xian Li, Luna Dong, Kenneth Lyons, Weiyi Meng, and Divesh Srivastava. Scaling up copy detection. In *Proceedings of the 2015 IEEE International Conference on Data Engineering*, ICDE '15, 2015.

[100] Xian Li, Xin Luna Dong, Kenneth Lyons, Weiyi Meng, and Divesh Srivastava. Truth finding on the deep web: is the problem solved? In *Proceedings of the 39th international conference on Very Large Data Bases*, PVLDB'13, pages 97–108. VLDB Endowment, 2013.

[101] Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. Annotating and searching web tables using entities, types and relationships. *PVLDB*, 3(1):1338–1347, 2010.

[102] J P Linge, R Steinberger, T P Weber, R Yangarber, and E van der Goot. Internet surveillance systems for early alerting of health threats. *Eurosurveillance*, 14(13), 2009.

[103] Larry M. Manevitz and Malik Yousef. One-class SVMs for document classification. *J. Mach. Learn. Res.*, 2:139–154, March 2002.

[104] Theofrastos Mantadelis, Ricardo Rocha, Angelika Kimmig, and Gerda Janssens. Preprocessing boolean formulae for BDDs in a probabilistic context. In *JELIA*, 2010.

[105] Adam Marcus, Eugene Wu, David Karger, Samuel Madden, and Robert Miller. Human-powered sorts and joins. *Proc. VLDB Endow.*, 5(1):13–24, September 2011.

[106] Adam Marcus, Eugene Wu, Samuel Madden, and Robert C. Miller. Crowdsourced databases: Query processing with people. In *CIDR*, pages 211–214, 2011.

[107] Yasuko Matsubara, Yasushi Sakurai, Christos Faloutsos, Tomoharu Iwata, and Masatoshi Yoshikawa. Fast mining and forecasting of complex time-stamped events. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 271–279, New York, NY, USA, 2012. ACM.

[108] Weiyi Meng, Clement Yu, and M. Tamer Ozsu. *Advanced Metasearch Engine Technology*. Morgan & Claypool Publishers, 2010.

[109] George A. Mihaila, Louiqa Raschid, and Mara esther Vidal. Using quality of data metadata for source selection and ranking. In *Vossen (Eds.), Proceedings of the Third International Workshop on the Web and Databases, WebDB*, pages 93–98, 2000.

[110] Andrew Cameron Morris, Viktoria Maier, and Phil Green. From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition. In *INTERSPEECH*, 2004.

[111] H. B. Newcombe, J. M. Kennedy, S. J. Axford, and A. P. James. Automatic linkage of vital records: Computers can be used to extract ”follow-up” statistics of families from files of routine records. *Science*, 130(3381):954–959, 1959.

[112] Feng Niu, Christopher Ré, AnHai Doan, and Jude Shavlik. Tuffy: Scaling up statistical inference in Markov logic networks using an RDBMS. *Proc. VLDB Endow.*, 4(6):373–384, March 2011.

[113] Feng Niu, Ce Zhang, Christopher Ré, and Jude Shavlik. Elementary: Large-scale knowledge-base construction via machine learning and statistical inference. *Int. J. Semant. Web Inf. Syst.*, 8(3):42–73, July 2012.

[114] Natalya F. Noy. Semantic integration: A survey of ontology-based approaches. *SIGMOD Rec.*, 33(4):65–70, December 2004.

[115] Dan Olteanu and Jiewen Huang. Using OBDDs for efficient query evaluation on probabilistic databases. In *Proceedings of Second International Conference on Scalable Uncertainty Management*, SUM, 2008.

[116] Dan Olteanu, Jiewen Huang, and Christoph Koch. SPROUT: Lazy vs. eager query plans for tuple-independent probabilistic databases. In *Proceedings of the 2009 IEEE International Conference on Data Engineering*, ICDE ’09, pages 640–651. IEEE Computer Society, 2009.

[117] Dan Olteanu, Jiewen Huang, and Christoph Koch. Approximate confidence computation in probabilistic databases. In *Proceedings of the 26th International Conference on Data Engineering, ICDE 2010, March 1-6, 2010, Long Beach, California, USA*, pages 145–156, 2010.

[118] Aditya Pal, Vibhor Rastogi, Ashwin Machanavajjhala, and Philip Bohannon. Information integration over time in unreliable and uncertain environments. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 789–798, New York, NY, USA, 2012. ACM.

[119] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 79–86, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[120] Aditya G. Parameswaran, Stephen Boyd, Hector Garcia-Molina, Ashish Gupta, Neoklis Polyzotis, and Jennifer Widom. Optimal crowd-powered rating and filtering algorithms. *PVLDB*, 7(9):685–696, 2014.

[121] Aditya G. Parameswaran, Hector Garcia-Molina, Hyunjung Park, Neoklis Polyzotis, Aditya Ramesh, and Jennifer Widom. CrowdScreen: Algorithms for filtering data with humans. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 361–372, New York, NY, USA, 2012. ACM.

[122] Hyunjung Park and Jennifer Widom. Crowdfill: Collecting structured data from the crowd. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 577–588, 2014.

[123] Jon Parker, Yifang Wei, Andrew Yates, Ophir Frieder, and Nazli Goharian. A framework for detecting public health trends with Twitter. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM, 2013.

[124] Michael Paul and Mark Dredze. You are what you tweet: Analyzing Twitter for public health. 2011.

[125] Leo L. Pipino, Yang W. Lee, and Richard Y. Wang. Data quality assessment. *Commun. ACM*, 45(4):211–218, 2002.

[126] Ravali Pochampally, Anish Das Sarma, Xin Luna Dong, Alexandra Meliou, and Divesh Srivastava. Fusing data with correlations. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 433–444, 2014.

[127] Franco P. Preparata and Michael Ian Shamos. *Computational Geometry*. Springer, 1985.

[128] Jay Pujara, Hui Miao, Lise Getoor, and William Cohen. Knowledge graph identification. In *International Semantic Web Conference (ISWC)*, 2013.

[129] Guo-Jun Qi, Charu C. Aggarwal, Jiawei Han, and Thomas Huang. Mining collective intelligence in diverse groups. In *Proceedings of the 22Nd International Con-*

ference on World Wide Web, WWW '13, pages 1041–1052. International World Wide Web Conferences Steering Committee, 2013.

[130] Alexander J. Quinn and Benjamin B. Bederson. Asksheet: Efficient human computation for decision making with spreadsheets. CSCW, 2014.

[131] Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D. Manning. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 248–256, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[132] Christopher Re, Nilesh Dalvi, and Dan Suciu. Efficient top-k query evaluation on probabilistic data. In *Proceedings of the 23rd International Conference on Data Engineering, ICDE*, ICDE. IEEE, 2007.

[133] Theodoros Rekatsinas, Amol Deshpande, Xin Luna Dong, Lise Getoor, and Divesh Srivastava. SourceSight: Enabling effective source selection. 2015.

[134] Theodoros Rekatsinas, Amol Deshpande, and Lise Getoor. Local structure and determinism in probabilistic databases. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 373–384. ACM, 2012.

[135] Theodoros Rekatsinas, Amol Deshpande, and Aditya G. Parameswaran. Crowdgather: Entity extraction over structured domains. *CoRR*, abs/1502.06823, 2015.

[136] Theodoros Rekatsinas, Xin Luna Dong, Lise Getoor, and Divesh Srivastava. Finding Quality in Quantity: The Challenge of Discovering Valuable Sources for Integration. *CIDR*, 2015.

[137] Theodoros Rekatsinas, Xin Luna Dong, and Divesh Srivastava. Characterizing and selecting fresh data sources. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 919–930. ACM, 2014.

[138] Theodoros Rekatsinas, Saurav Ghosh, Sumiko Mekaru, Elaine Nsoesie, John Brownstein, Lise Getoor, and Naren Ramakrishnan. SourceSeer: Forecasting rare disease outbreaks using multiple data sources. In *the SIAM International Conference on Data Mining*, SDM, 2015.

[139] Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, and Padhraic Smyth. The author-topic model for authors and documents. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, UAI '04, pages 487–494. AUAI Press, 2004.

[140] Scott Sanner and David McAllester. Affine algebraic decision diagrams (AADDs) and their application to structured probabilistic inference. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, pages 1384–1390. Morgan Kaufmann Publishers Inc., 2005.

[141] Bernhard Schölkopf, John C. Platt, John C. Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the support of a high-dimensional distribution.

Neural Comput., 13(7):1443–1471, July 2001.

[142] Felix Naumann Sebastian Kruse, Paolo Papotti. Estimating data integration and cleaning effort. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*, 3 2015.

[143] Prithviraj Sen and Amol Deshpande. Representing and querying correlated tuples in probabilistic databases. In *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007*, pages 596–605, 2007.

[144] Prithviraj Sen, Amol Deshpande, and Lise Getoor. PrDB: managing and exploiting rich correlations in probabilistic databases. *The VLDB Journal*, 18, 2009.

[145] Prithviraj Sen, Amol Deshpande, and Lise Getoor. Read-once functions and query evaluation in probabilistic databases. *Proc. VLDB Endow.*, 3(1-2):1068–1079, September 2010.

[146] T. Shen, A. Chao, and C. Lin. Predicting the number of new species in further taxonomic sampling. *Ecology*, 84(3), 2003.

[147] Cheng Sheng, Nan Zhang, Yufei Tao, and Xin Jin. Optimal algorithms for crawling a hidden database in the web. *PVLDB*, 5(11):1112–1123, July 2012.

[148] Victor S. Sheng, Foster Provost, and Panagiotis G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 614–622, New York, NY, USA, 2008. ACM.

[149] Fabio Somenzi. CUDD: CU Decision Diagram Package. `http://vlsi.colorado.edu/fabio/CUDD/`.

[150] Ingo Steinwart, Don Hush, and Clint Scovel. A classification framework for anomaly detection. *J. Mach. Learn. Res.*, 6:211–232, December 2005.

[151] Michael Stonebraker, Daniel Bruckner, Ihab Ilyas, George Beskales, Mitch Cherniack, Stan Zdonik, Alexander Pagan, and Shan Xu. Data curation at scale: The data tamer system. In *Proceedings of CIDR'13*, 2013.

[152] Alexander Strehl, Er Strehl, Joydeep Ghosh, and Raymond Mooney. Impact of similarity measures on web-page clustering. In *In Workshop on Artificial Intelligence for Web Search (AAAI 2000)*, pages 58–64. AAAI, 2000.

[153] Fabian Suchanek and Gerhard Weikum. Knowledge harvesting in the big-data era. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 933–938, 2013.

[154] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A large ontology from wikipedia and wordnet. *Web Semant.*, 6(3):203–217, September 2008.

[155] Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. Probabilistic databases. *Synthesis Lectures on Data Management*, 3(2):1–180, 2011.

[156] Olivier Teytaud, Sylvain Gelly, and Michèle Sebag. Anytime many-armed bandits. In *CAP*, 2007.

[157] Beth Trushkowsky, Tim Kraska, Michael J. Franklin, and Purnamrita Sarkar. Crowdsourced enumeration queries. In *Proceedings of the 2013 IEEE International Conference on Data Engineering (ICDE 2013)*, ICDE '13, pages 673–684, Washington, DC, USA, 2013. IEEE Computer Society.

[158] Prasang Upadhyaya, Martina Unutzer, Magdalena Balazinska, Dan Suciu, and Hakan Hacigumus. Affordable analytics on expensive data. In *Proceedings of the First International Workshop on Bringing the Value of "Big Data" to Users (Data4U 2014)*, Data4U '14, 2014.

[159] Guy Van den Broeck and Adnan Darwiche. On the role of canonicity in knowledge compilation. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI Conference on Artificial Intelligence, Austin Texas, USA, January 2015*. AAAI Press, January 2015.

[160] Petros Venetis, Alon Y. Halevy, Jayant Madhavan, Marius Pasca, Warren Shen, Fei Wu, Gengxin Miao, and Chung Wu. Recovering semantics of tables on the web. *PVLDB*, 4, 2011.

[161] Daisy Zhe Wang, Yang Chen, Sean Goldberg, Christan Grant, and Kun Li. Automatic knowledge base construction using probabilistic extraction, deductive reasoning, and human feedback. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, AKBC-WEKEX '12, pages 106–110, 2012.

[162] Daisy Zhe Wang, Eirinaios Michelakis, Minos N. Garofalakis, and Joseph M. Hellerstein. BayesStore: managing large, uncertain data repositories with probabilistic graphical models. *PVLDB*, 1(1):340–351, 2008.

[163] Jiannan Wang, Tim Kraska, Michael J. Franklin, and Jianhua Feng. CrowdER: Crowdsourcing entity resolution. *Proc. VLDB Endow.*, 5(11):1483–1494, July 2012.

[164] Xianggang Wang and Eric Grimson. Spatial latent dirichlet allocation. In *Advances in Neural Information Processing Systems*, volume 20, 2007.

[165] Xuerui Wang and Andrew McCallum. Topics over time: A non-markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 424–433. ACM, 2006.

[166] Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 515–526, New York, NY, USA, 2014. ACM.

[167] Michael L. Wick, Andrew McCallum, and Gerome Miklau. Scalable probabilistic databases with factor graphs and MCMC. *Proc. VLDB Endow.*, 3(1):794–804, 2010.

[168] Frank Wilcoxon. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80–83, December 1945.

[169] William E. Winkler. Improved decision rules in the fellegi-sunter model of record linkage. Technical report, Statistical Research Division, U.S. Census Bureau, Washington, DC, 1993.

[170] William E. Winkler. Methods for record linkage and bayesian networks. Technical Report Statistical Research Report Series RRS2002/05, U.S. Bureau of the Census, Washington, D.C., 2002.

[171] Xiaoxin Yin, Jiawei Han, and Philip S. Yu. Truth discovery with multiple conflicting information providers on the web. *IEEE Trans. on Knowl. and Data Eng.*, 20(6):796–808, June 2008.

[172] Jeffrey Xu Yu, Lu Qin, and Lijun Chang. *Keyword Search in Databases*. Morgan and Claypool Publishers, 1st edition, 2010.

[173] Nevin L. Zhang and David Poole. On the role of context-specific independence in probabilistic inference. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'99, pages 1288–1293. Morgan Kaufmann Publishers Inc., 1999.

[174] Bo Zhao, Benjamin I. P. Rubinstein, Jim Gemmell, and Jiawei Han. A bayesian approach to discovering truth from conflicting sources for data integration. *Proc. VLDB Endow.*, 5(6):550–561, February 2012.

[175] Hongwei Zhu, Stuart E. Madnick, Yang W. Lee, and Richard Y. Wang. Data and information quality research: Its evolution and future. In *Computing Handbook,*

Third Edition: Information Systems and Information Technology, pages 16: 1–20. 2014.