Facilitating Location and Use of Socio-economic Data with Minimal User Intervention

by

Jie Song

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
2021

Doctoral Committee:

Professor Hosagrahar V. Jagadish, Chair
Research Professor George Alter
Associate Professor Michael J. Cafarella
Assistant Professor Danai Koutra

Jie Song

jiesongk@umich.edu

ORCID iD: 0000-0002-3433-4522

ACKNOWLEDGEMENTS

First, I am thankful for the support of my advisor, Professor H. V. Jagadish, who has been guiding me through the PhD years to turn from a practical engineer with little research experience to a professional with research mind. He tailored the training to my background and personality and patiently directed me to grasp research knowledge and academic writing and communication skills little by little. When I encountered the biggest challenge in my life at the start of the third year of PhD study, he did much more than what an academic advisor could do and helped me through the time unconditionally. Other than academic achievements, I regard him as a role model to learn from. I will always remember his way of teaching young researchers, leading research teams, supporting women researchers and treating students equally with willingness to give them opportunities regardless of their background.

I would also thank my parents for supporting me through not only the PhD period but my whole life the entire time. They always value my physical health and mental happiness as the first priority. When I look back, they are always there patiently listening to my achievements and confusions as best friends and kindest parents.

I am honored to have the opportunity to work with my committee members, Professor George Alter, Professor Danai Koutra and Professor Michael Cafarella. Professor George Alter is like the second advisor of mine for the past several years. He has been dedicated to the research at ICPSR and I really enjoyed working with him and meet with him weekly. Professor Danai Koutra is always open to discuss new ideas with me and I appreciate her introducing me to other opportunities in the

community. Professor Michael Cafarella is one of the kindest person I have ever met. He is optimistic, humour and smart. I love taking his class EECS 485 and working with him.

I am fortunate to work with Luna Dong and Xian Li at Amazon during internship in 2017, and Yeye He and Surajit Chaudhuri at Microsoft Research in 2019. They broaden my horizon to the research life in the industry and introduced me to apply research to cutting edge problems.

I would like to thank Alex Mueller, Yoko Nagafuchi, Zerui Wei, Ruidong Liu, Tianji Cong, Yin Lin, Zhongjun Jin and other student researchers who have contributed their efforts to the projects in this thesis. Structured Data Transformation Language (SDTL) and the software applications in the C2Metadata workflow were produced by a large team of researchers and developers. I wish to thank the dedicated teams at Colectica (leads Dan Smith and Jeremy Iverson), Metadata Technology North America (leads Jack Gager and Pascal Heus), the Norwegian Centre for Research Data (lead Ørnulf Risnes), and ICPSR (lead George Alter). I am also grateful to my friends, especially Yining Shi, Yingchao Zhong, Jingjing Cao, Xiaojing Du and Mengyang Wang, who are always there together with me through the ups and downs in life.

I am grateful for fellowship support from the National Science Foundation's Data Infrastructure Building Blocks (DIBBs) program under "Continuous Capture of Metadata" NSF ACI-1640575 for work on the C2Metadata Project and SDTA, and Div Of Information & Intelligent Systems (IIS) program under NSF IIS-1250880 for work on GeorAlign.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

ABSTRACT

Multitudes of data sets are available today on almost every topic imaginable. However, given a particular information need, it is not easy to find the right data sets best suited to satisfy the need. Most current dataset search tools perform a keyword search over schema and published metadata. While such searches are easy to specify, they are also very blunt. Many potential data sets are often returned, from which the user has to choose the ones desired. The situation is worse when the user's question can only be answered by combining two or more data sets. A technically proficient user with knowledge of database schema could specify their information need precisely. However, in our scenario, we have to deal with both limited technical expertise and lack of schema knowledge.

In this dissertation, we attempt to facilitate data set location and use by people with little technical proficiency. Our goal is to let the user be "hands-off", providing as little direction as possible. While our techniques are broadly applicable, we particularly focus on datasets with socio-economic data. Specifically, we address two major challenges a user may face: 1) a single dataset is not enough for the task, and 2) the available metadata is inadequate to evaluate the fitness of a dataset for a particular task. To tackle the former challenge, we complement individual dataset search with automatic data integration based on both metadata and data content (`FluxSearch`) for both special-purpose geospatial data (`GeoFlux`) by a multiple reference-based interpolation method (`GeoAlign`), and for more generalized data in a data portal using union and join. For the latter challenge, we consider ancillary information by automatically incorporating data provenance in terms of standardized statistical

transformation (SDTA and SDTL) into metadata (C^2Metadata).

CHAPTER I

Introduction

The consumption of data continues to grow by leaps and abounds. It has been predicted that the Big Data analytics market will soon surpass $200 billion with a compound annual growth rate of 11.7% [52]. With the launch of shared data via open data portals and scientific repositories in addition to traditional data markets, general public, once neglected, as the potential consumer of Big Data has recently drawn more attention of research forces in the field. Answering important socio-economic questions, for instance, is no longer the concern of merely professional data users such as policymakers and data scientists, but also the general public getting more engaged in the game of data.

To gain value from this ocean of data requires the ability to find and make sense of datasets as a first step before data analysis for specific tasks. This pre-analysis step is generally known as *dataset search*. Searching for datasets in principled ways has been researched for decades. Nowadays, dataset search is largely keyword-based over published metadata, within a single data source or over multiple data sources. While such searches are easy to specify, they are also very blunt. Many potential datasets are often returned as ranked results, from which the user has to choose the ones desired.

There are several problems with this approach due to the disconnect between

what datasets are available, what dataset a user needs, and what datasets a user can find and use. Though many communities, particularly in social science and medicine, enforce publishing data with metadata, a large fraction of datasets from other sources such as Web Tables lack metadata. The quality of available metadata also varies. For instance, knowing the original purpose for collecting the data aids interpretation and analysis; while understanding the processing performed before the release of a dataset affects later use of the data (machine learning for example). Available metadata may thus not encompass the actual information a user needs to assess whether the dataset fits a given task as many unique dataset properties are not fully utilized.

The situation is worse when the user's question can only be answered by combining two or more datasets. Firstly, even identifying the datasets can be difficult. Even the datasets are found, they must be integrated through a series of tedious integration steps to obtain a new dataset that meets the desired information need. The representations of datasets and metadata vary from source to source, making the integration process even more complicated.

Though many tools have been developed to facilitate dataset search and data integration respectively, general users with limited or no technical proficiency could not easily use these tools designed for professionals. Moreover, users often need to switch between tools targeting specific sub-steps of search and integration, complicating the use of data even further.

This dissertation studies various aspects of dataset search and preparation to enable users with little technical proficiency to find proper datasets for specific use. To make the problem more tractable, we limit our scope to socio-economic datasets, a large and important class of data concerning general users. Such datasets are generally presented as data tables with or without metadata. Tables in such datasets do not have primary key-foreign key relationships commonly defined in relational databases. Our techniques can be generalized to a broader scope. Specifically, we seek answers

to the following two questions:

- When no single dataset is a proper fit for a specific search task, how could we integrate data from different sources to form a new dataset in an automatic fashion?
- How can we improve metadata quality by including additional aspects of data to effectively evaluate a dataset's fitness for a particular use?

We present key technical challenges and general strategies to address the questions above. We then conclude with the key contributions and outline the work presented in the rest of the dissertation.

1.1 Challenges and Strategies

1.1.1 User Input in Data Integration

Despite extensive research on tools to assist users, data integration remains hard, particularly for users with limited technical proficiency. A large fraction of the tools requires intervention from technical aspects, such as schema alignment, record linkage, and data fusion. Such ability general users often lack. The integration is even more complicated for data from different sources suffering from semantic heterogeneity, integrity constraints, and data-level heterogeneity. A normalization of such data is needed to transform data into a canonical form before integration.

To address this barrier, we study how much we can do with **no user guidance**. Our vision is that the user should merely specify two input datasets to be integrated and get a meaningful integrated result. In general, there are many different ways one could combine two relational tables.

To make the problem tractable, we only consider socio-economic data, typically in tabular form aggregated on a selected segment of the population in some geographic

area and/or time interval. An intuitive way to integrate two such datasets is to join on the geographic unit column, thereby allowing the user to compare information from the two datasets for corresponding geographic units, one per row.

While this sounds easy in principle, there are many obstacles to overcome: (1) data values are not always organized in a standardized way; (2) data may be reported in numerous types of standard geography and there does not always exist a straightforward relationship between any two types; and (3) data can be reported as aggregated data at population level and at individual level.

Leveraging the geographic join key information, we have developed learning-based and heuristic-based methods tackling each of the obstacles above. These methods compose the modular architecture of a systematic approach for automatic data integration for socio-economic datasets. The system can be further extended for joining based on other domain knowledge and other integration operations such as union for more generalized integration scenarios.

1.1.2 Lossy Metadata Information

The common theme of current dataset search strategies, both on the web and within the boundaries of a repository, is the reliance on dataset publishers tagging their data with appropriate information in the correct format. Because current dataset search only uses metadata view of a data, it is imperative that these metadata descriptions are correct and maintained.

Three major disparities arise from this assumption in reality. First, datasets are not always published with metadata. Though many communities are pushing towards this end, the majority of datasets available lack metadata. Even if published with metadata, there is no universal standard for mandatory information in metadata documentation. Moreover, the correctness of the metadata, or metadata quality, could not be guaranteed.

Here we focus on improving the quality of metadata by documenting the data provenance information in terms of data transformation before data publish. These transformation operations are generally performed by various statistical transformation tools. To eliminate manual documentation, we propose to create a system to automatically update (or create) metadata with this transformation information in a standard representation.

1.1.3 Search Over Integrated Datasets and Metadata from Heterogeneous Sources

In addition to heterogeneous data representations, metadata from different sources follow various standards. Metadata descriptions according to standards such as DCAT [70], defines attributes such as title, description, language or license, while "CSV on the Web" documents data types and formats for validation purpose [95]. Moreover, not all datasets are shared with metadata information. Even if they have metadata, the quality of metadata vary. We also noted that there is no universal agreement on how to create (or integrate) metadata for dataset parties involved in data integration. Even if integrated metadata is created, it lacks information regarding the integrated dataset itself.

We created a joint signature for both metadata and data content of the integrated dataset for dataset search over such datasets where no single dataset satisfies the search task. These signatures are precomputed during the offline stage. During the online stage, we efficiently search for data-enhanced metadata using techniques from information retrieval and integrate metadata-enhanced datasets using techniques from the database community. The integrated datasets with metadata documenting the respective metadata of member datasets and the integration process are further evaluated for their relatedness to the search query and the richness of information.

1.2 Summary of Contributions

This dissertation studies the challenges in facilitating the location and use of socio-economic datasets in a hands-off manner for users with little technical proficiency. We limit user contributions to non-technical aspects and address the problem from different perspectives.

To efficiently integrate datasets when no single dataset fits the search purpose, we propose an automatic data integration system that joins datasets leveraging join key knowledge. The system transforms input datasets in a canonical form, identifies the proper join key maximizing data quality, matches entries referring to the same real-world entity before the actual integration. When the aggregation level of datasets differs, we propose an adaptive multi-reference interpolation method that realigns aggregates of respective datasets into a common level to permit further integration.

For metadata not representing the dataset fully in a way a user needs to assess the dataset for a specific search task, we enhance existing metadata with data provenance. For provenance consideration, we standardize the derivation of the dataset in terms of transformational information in a canonical way and automatically updating exiting published metadata with this information in a pipelined system.

Lastly, we developed a dataset search system leveraging the benefit of both systems. We adopt a keyword-based search strategy over automatically integrated datasets with provenance-updated metadata to take advantage of both data content and metadata for a better search.

1.3 Outline of the Dissertation

The rest of the dissertation is organized as follows. In Chapter II, we survey related dataset search techniques and data preparation techniques concerning automatic data integration. In Chapter III, we propose `GeoFlux` as an automatic data

integration system leveraging join key information. In Chapter IV, we propose an adaptive multi-reference interpolation algorithm `GeoAlign` for realigning aggregates over unaligned partitions. In Chapter V, we propose two standard representations `SDTA` and `SDTL` for statistical data transformation, and a system `C`^{2}`Metadata` that automatically updates existing metadata with provenance information in terms of statistical data transformation. `GeoAlign` won the best paper runner-up award at EDBT 2018, and `SDTA` won the best paper award at SSDBM 2021. In Chapter VI, we propose a dataset search system `FluxSearch` that operates over automatically integrated datasets with provenance-aware metadata. In Chapter VII, we conclude the thesis with potential future work.

CHAPTER II

Research Background

2.1 Automatic Data Integration

Data integration has long been recognized as an important problem and much progress has been made to address it but it still remains a challenge [114, 46]. It has been broken down into sub problems, such as data cleaning [87], schema mapping [7], entity matching [60], etc., each of which has been studied extensively. Tools have also been developed to facilitate the usability of database for data integration [85, 24, 9, 73]. Nevertheless, a majority of the data integration work is still conducted manually by hand [51].

Our system limits the scope of data and the scope of integration operations to provide truly completely automatic data integration.

2.2 Data Interpolation

In the GIS community, spatial interpolation has advanced from isoline mapping in cartography to data realignment in different units or grids for multivariate analysis in geographic research [62, 8, 76]. *Realignment*, *crosswalk*, or *regridding*, is commonly used today as a preprocessing step before further data analysis in physics and socioeconomics to interpolate spatial or temporal data distribution from one grid to another

[59]. Since these data are either point or areal based, two categories of methods are proposed for these two types respectively.

Areal interpolation is a subset of the spatial interpolation problem that realigns aggregates. Early methods built upon point-based interpolation, such as point-in-polygon method, do not follow the volume-preserving property such that reconstruction of exactly the original aggregates of each source unit with the transformed value of each target unit is not possible [62, 90]. It has been shown that these methods are not comparable in approximation efficiency with those that do have the property [101, 62]. Later methods thus introduce the property and turn over to the area-based areal interpolation instead [28]. These approaches depend highly on the spatial properties of the data collection area and thus different forms of ancillary data are introduced ever since.

Areal weighting method, one of the early area-based areal interpolation method, makes use of the area ancillary data available in the form of disaggregation partitions between source and target units [72, 29]. This method is widely available in GIS software for general users nowadays. However, it assumes even distribution within units (homogeneity) whereas this assumption hardly stands in reality. Areal weighting has been extended by referring to other single known reference attributes, called dasymetric weighting [3, 89, 74, 42]. These methods are restricted by the assumption of proportionality of the objective attribute to the single reference attribute. Hence the selection of the reference attribute is vital to the prediction accuracy and the methods are not adaptive to different objective attributes.

The regression methods are later introduced as extensions to the dasymetric methods allowing for multiple auxiliary variables. In general, the regression methods involve a regression of the source level data of the objective attribute on the values of the references in target units. For this track of methods, more advanced techniques such as EM algorithm, Monte Carlo simulation, smoothing techniques[62, 97, 107, 25, 23, 94],

etc., are introduced later in the literature. However, they make different assumptions of density distribution within units, some of the mostly used ones are Poisson distribution and binomial distribution, and their performances are rather assumption dependent [61] and auxiliary variable dependent. Recently, more complicated regression models [78, 77, 69] are developed based on domain knowledge such as spatial correlation. However, they lack general applicability to heterogeneous target attributes and are hard to implement for practitioners.

These approaches can also be categorized as extensive or intensive approaches based on their approximation target. Extensive approaches approximate a_o^{st} while intensive ones approximate f_o^{st}. Most approaches for solving the areal interpolation problem are intensive approaches that build spatial statistical models for f_o^{st} in the disaggregation step. These approaches, mostly developed in 2-D space, can be extended to higher dimensions, though these extensions are typically non-trivial. Other major limitations of intensive approaches include narrow scope of application and low robustness to heterogeneous objective attributes.

Current intensive approaches for areal interpolation are not generally applicable for aggregate interpolation due to three main reasons. First, integration of f_o^{st} is computable in 2-D, however, it is computationally intensive in high dimensions with complex f_o^{st}. Second, shape files are indispensable for intensive approaches, and the probability density function for each intersection unit, $f_o^{st}[k]$, is associated with the shape files of source and/or intersection units. Further, attributes in plain tables without handy shape files of target units typically fail re-aggregation. Even if shape files are available, some of them constantly change over time, resulting in approximation inaccuracies. Last but not least, these approaches are not easily approachable for general users, especially those with little technical proficiency in mathematics, statistics and GIS. The $\hat{f}_o^{st}$ model is built upon the spatial knowledge of the objective attribute; however, this knowledge is not available for all users. Further, implemen-

tations of intensive approaches are not publicly available, making them even harder to use.

Another limitation of intensive approaches is that they are not adaptive to new attributes. $\hat{f}_o^{st}$ models are attribute dependent since the true f_o^{st} models for two attributes can be very different. Another point to note is that these approaches make many assumptions of $\hat{f}_o^{st}$. For instance, the distribution model of each intersection unit, the choice of parameters for these distributions and so on. Any change in these assumptions may dramatically influence the accuracy of approximation in some target unit. What is worse, there is no efficient verification of whether they are appropriate or not.

Extensive approaches are more generally applicable than the intensive ones: they can be easily extended to high dimensions, need no unit shape files, and are easy to implement. However, existing extensive approaches make use of a single reference attribute and are still limited in robustness. When the objective attribute and the reference attribute does not share similar spatial distribution, the approximated result can differ substantially from the true aggregates in target units. Further, since they use the same reference attribute irrespective of the objective attribute, they are not adaptive to different objective attributes with heterogeneous spatial distributions.

2.3 Statistical Data Transformation

There is no shortage of algebras for statistical data transformation. Thus in this section, we first broadly discuss related work regarding data transformation in relational databases and statistical data, before further discussion of a common language representation for statistical data transformation.

Though many tools have been developed to facilitate data transformation for users with different technical backgrounds in the database community, they are not well-aligned with, and supportive of, statistical data transformation. SQL, for instance,

is the most widely used data manipulation language for databases. In spite of its extending support of limited data transformations and analytic in recent years, it is primarily used for answering queries regarding data in situ where data can be highly normalized. More recent tools like Potter's Wheel [88], Wrangler [58], Foofah [56] provide an interactive interface for users with less technical proficiency for more specific transformation needs such as data wrangling, data warehousing and data cleaning. However, their emphasis are more on the syntactic transformation than on the preparation for statistical analysis.

More targeted tools are well adopted in the data science community over the past half century. Statistical packages like SPSS® , Stata® , SAS®, R(by packages like dplyr [106], tidyverse [105] and reshape2 [103]) and Python are more popular for statistical data transformation for tabular data. Though simple data conversion (opening data file written in one language by another language using tools like Stat/Transfer[14]) is straightforward, the transformation syntax conversion between these languages is not easily realizable. Attempts for embedding one language in other programming languages is not new for statistical data. Major commercial statistical software such as SAS[54] and SPSS[17] include a built-in interface for calling R. Other attempts [44] have been made to communicate data and results interactively in language interfacing. These attempts, however, are tailored to the characteristics of languages involved, and are often between statistical languages and a more general programming language for possible operation decomposition.

Standard representations of statistical data transformation has been proposed regardless of the actual transformation engine used. The Validation and Transformation Language (VTL) [91] is such a standard language that defined the validation and transformation rules for statistical data at the abstract level. It primarily supports exchange of data validation rules for data quality specification and validation purposes accounting for part of the data transformation purpose but not all. VTL

could not serve as the bridging language for inter-conversion of statistical languages as information losses during conversion.

2.4 Dataset Search

Dataset search involves the discovery, exploration, and return of datasets to an end user. Dataset search has not emerged in isolation, but has built on foundatinal work from other related areas including structured database, keyword based IR, semantic web and tabular search.

A general approach to providing search over datasets is to model the user interface over existing keyword based information retrieval search systems where a user poses a query and a ranked list of existing datasets is returned. Indeed, a majority of data repositories provide this form of interface. The dataset search problem can be addressed at various levels. Services such as Google Dataset Search [41] and DataMed [6] crawl across the web and facilitate global search across all distributed resources. These approaches use tags found in schema.org or DCAT [70] to structure and identify the metadata considered important for datasets. However, the problem also exists at a local level, including open government portals such as data.gov.uk, organizational data lakes, scientific repositories such as Elsevier's [27] and data markets. Across all these systems, users are attempting to discover and assess datasets for a particular purpose.

Here we explore a few aspects of dataset search related to the work in this thesis, namely query languages, query handling, and query results.

2.4.1 Query Languages

Web search has long been a hot topic in the information retrieval community. Web search engines have been used for document, image and video searches. When it comes to searching for structured data such as web tables, relational tables, open

datasets, etc., users have to be technical proficient in structured query languages such as SQL and SPARQL and, at the same time, learn the schema of the data sources to formulate valid query for the search. More recently, to reduce the learning time of users, efforts have been made to enable search queries in more relaxed formats including keyword based, entity based and tabular based query languages. Additional filters may compliment the search query for more specified search. For instance, a query may narrow down the search scope to datasets published before 2012.

2.4.2 Query Handling

Putting our focus on keyword queries, we discuss further related work in the handling of query and data. Most dataset searches operate over the dataset's metadata by building index over the metadata. However, Many datasets have no metadata. [80] pointed out that the quality of the metadata largely affects the discovery and the consumption of the datasets within Open Data Portals. Works like [98, 80, 50] tackles the quality issue of metadata. There are some other efforts [96, 83, 43, 110] that aim to facilitate the search over the data or over the summary profiles or annotations generated from the data to promote the understanding of the datasets for the search engine and the end users.

2.4.3 Query Result Generation

Many of the current dataset search engines evaluates datasets, more specifically their metadata, against the search query and return a ranked list of datasets by different criteria.

Candidate datasets for keyword based queries are evaluated for their relatedness to the query in terms of syntactic relatedness and semantic relatedness similar to keyword search queries in web search [111, 100, 68]. The original datasets are returned as rank result. Entity-based query engines [4, 48, 2] have a different purpose as to

extend the information of the searched entity by datasets. The look up process tries to build links between datasets, known as link discovery, which involves the classic schema mapping and entity matching problems from the database community. The information enriched entities are ranked before presented to the users. Search engines using tabular queries, on the other hand, aim at searching for related datasets. These datasets could either extend the queried table to provide fuller information for entities in the query table by table join, or to present more entities in a similar domain to the query table by table union. The returned result is thus an integrated table such that the query table is enriched with information from related tables. [109, 11] defined three types of tabular search, namely table extension [55, 109, 111], table completion [112, 18] and attribute discovery.

CHAPTER III

GeoFlux: Joining Tables Automatically leveraging Join Key Knowledge

In this chapter, we focus on the design of a hands-off data integration system for socio-economic data minimizing user guidance, called `GeoFlux`. More specifically, we emphasize joining data tables leveraging the geographic information shared among such data. The system tackles multiple challenges user encounter in the joining process in a systematic way.

3.1 Introduction

Social scientists, policy makers, activists, and ordinary citizens all have unprecedented access to a variety of socioeconomic data, with the potential to derive valuable insights. However, in order to discover interesting results, users typically have to integrate data from multiple sources. Unfortunately, data integration is hard, and require more technical training than our target users, even with many helpful tools available today. Our goal in this paper is to see how well we can do with data integration **without any user guidance** at all. Ideally, we want the user only to identify two datasets, and leave it for the system to compute a meaningful integrated result.

In general, there are many ways to integrate information in two tables. We focus

Table 3.1: Monthly HELP (Highway Emergency Local Patrol) Assists: Beginning 2010

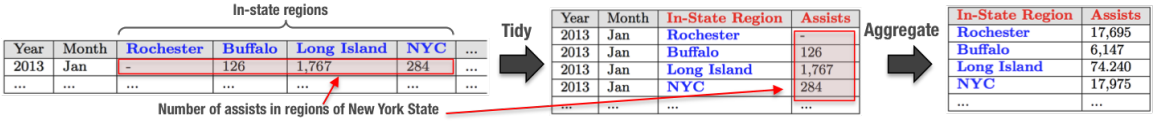
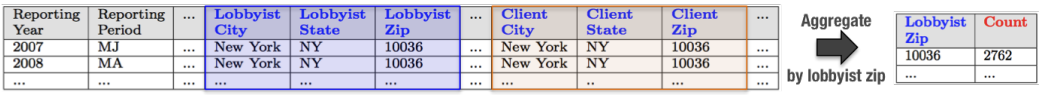


Table 3.2: Registered Lobbyist Disclosures: Beginning 2007



on a specific class that is both large and of practical importance: joining two tables with data aggregated by geography. Socioeconomic data are often reported in the form of tables and 80% of them include geographic information [45] organized in granularities (zip codes, counties, etc) driven by administrative requirements. This geographic information has strategic importance as a link between datasets, and has high administrative and statistical value for governments and data scientists [16].

In the simplest case, we may imagine two tables with two columns each: the first table recording per capita income for each county and the second table recording number of reported crimes per county. An intelligent system could join these two tables on the county name column to get a three-column table that provides insight about the relationship of crime and income. In practice, the tables to be joined are much more complex and may report data in incompatible geographic units. We describe below two real data tables selected at random from data.ny.gov, the official website for open government data for New York State.

Motivating example. *The Monthly HELP (Highway Emergency Local Patrol) Assists data provides the number of motorists assisted by year, month and region, in vehicles on highways since 2010. Table 3.1 is a truncated version of its first row. In Table 3.2, we give a simplified version of a Registered Lobbyist Disclosures table, which contains information about biennial registration and bi-monthly filings by lobbyists to*

the New York State Joint Commission on Public Ethics since 2007. Let's suppose that a social scientist has some interesting hypothesis relating highway assists to lobbyists, and joining these two tables is central to understanding it.

Table 3.1 has three dimension variables (`Year`, `Month` and `In-State Region`) that describe the granularity of the data, and one measure variable (number of motorists assisted) reported summarized with respect to these three dimensions. The `Region` values are spread across columns such that each row is a combination of four assist number observations from four regions. Table 3.2 is well formatted with each column representing a dimension that describes attributes of filings including their lobbyist and client geographic information for city, state and zip code levels; and each row representing one filing as an observation. Unlike Table 3.1, which is reported at population level, Table 3.2 is reported at individual level with each record corresponding to one filing.

As we saw in the example above, data values are often not organized in a standard way. Some tables are structured such that each column corresponds to one variable and each row is an observation; some tables use a partial two-way format in which values of variables are spread as column names; yet other tables may use some other structure.

Furthermore, data may be reported in numerous types of standard geography and there does not always exist a straightforward relationship between any two types. The United States Census Bureau defines legal, administrative and statistical boundaries as shown in Figure 3.1, where lines depict inclusion relationships. For instance, a census tract is defined as a finer granularity region within a single county; however, a zip code may sit in several counties and a county may contain multiple zip codes.

Another problem is that while socioeconomic data are frequently reported as aggregated data at population level to protect the privacy of individual citizens or

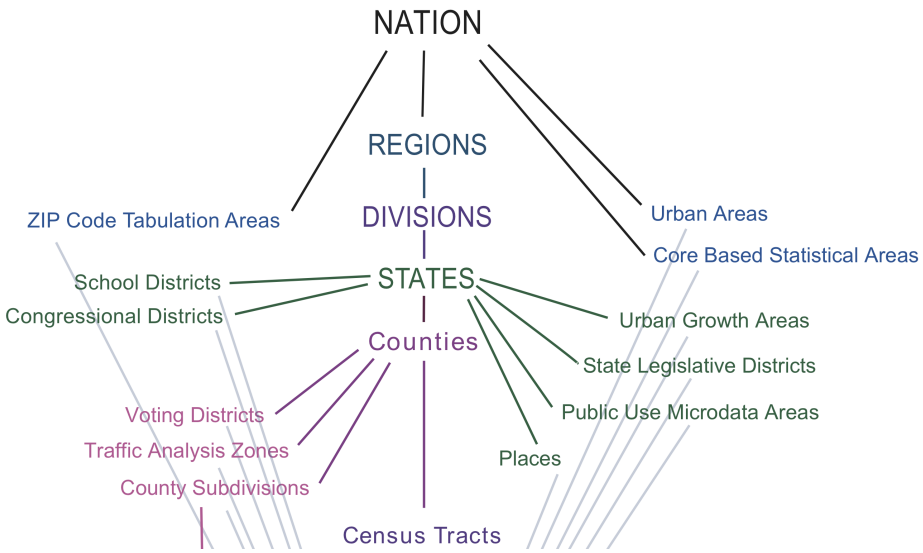


Figure 3.1: (Part of) Hierarchy Diagram of Census Geographic Types from US Census Bureau

survey participants, it can also be collected at individual level with each record describing one observation surveyed. Data collected at different levels cannot simply be integrated.

Motivating Example Continued. *For the Monthly HELP Assists table, we may meld the four region columns into one* `In-State Region` *column and treat region names as its values. The observations for regions will span the new* `Assists` *column so that each row is an observation after transformation as in Table 3.1(middle). After we aggregate the number of assists by* `In-State Region`*, Monthly HELP Assists table is now tidied and aggregated as in Table 3.1(right).*

The Registered Lobbyist Disclosures data are collected at individual level. It is necessary to aggregate the number of filings at some selected geographic type among states, cities and zip codes for both lobbyists and clients. In this example, we assume that the user chooses to aggregate the table by zip codes of lobbyists and store the number of filings in the `Count` *column as in Table 3.2(right).*

Finally, we should join two tables by geographic information, possibly by zip code

or by region. However, there is no way to convert data collected from zip code units to region units accurately or vice versa, as zip code tabulation areas and regions are sibling types (i.e., incompatible) on two line tracts under nation in Figure 3.1. That is, a zip code may overlap with several regions and a region may contain multiple zip codes. In this work, we propose an approximation crosswalk algorithm to convert the distribution of data between the two types, which enables the join.

Our approach. Our goal is to join two tables, based on geographic information, with no user guidance. We accomplish this in three major steps. First, we represent the given data tables in *canonical form*, which we define precisely in §3.2.1. Next, we automatically identify the geographic join column(s). Finally, we "align" the type of geographic aggregation in the join column(s) of the two tables, and then perform the required join. Each of these steps poses its own challenges, which we address as listed below. Our major contributions include:

- We identify a practically important class of data reported by geography, where integration of heterogeneous tables is both feasible and of value (§3.2).
- We incorporate the method into a system, `GeoFlux`, that automatically joins a given pair of tables based on geographic information, and produces a joint table sliced along selected geographic units of target geographic type (§3.3).
- We identify two classes of data tidying transformations that can convert most government data tables into canonical form (§3.4).
- We develop techniques, based on geographic dictionaries and data quality metrics, to identify the geographic join column(s) in any canonical table (§3.5), and to select a target level of geographical aggregation (§3.6).
- We develop a novel multi-crosswalk method for estimating geographic aggregates in any geographic units of chosen type (§4.3). More details are available

in Chapter IV.

- We experimentally evaluate our proposed approach on real government data, and show that almost perfect results can be obtained with no user input (§3.8).

Finally, we present the conclusion in §3.9.

3.2 Problem Statement

In this section, we provide background and notations for terms, followed by a formal definition of our problem.

3.2.1 Preliminaries

Socioeconomic data are typically reported over a target group of *observational units*, such as persons and areas. Every data entry belongs to a *variable*, which contains all values that measure the same underlying variable across observation units, and an *observation*, which contains all values measured on the same observational unit across variables [104].

Variables can be placed in two categories. Some describe *measures*, defined as features whose values are of interest, while others are *dimensions*, which describe the settings under which the measure values are measured, including in particular the location and time period. A common analysis technique in statistics is to discover characteristics of measures over one or more dimensions. For example, how is the `Number of Assists` measure distributed in New York State over the dimension of `In-State Region` in the motivating example. To compare measures from any two such tables, dimensions are commonly used as join keys and the correlation of measures over chosen dimensions is a good starting point of analysis. In this chapter, we choose dimensions containing geographic information as the join keys, but our methods can be generalized to time periods and other hierarchical dimensions.

Table 3.3: State Park Annual Attendance Figures by Facility: Beginning 2003

Year	OPRHP Region	County	Facility	Attendance
2014	Finger Lakes	Tompkins	Allan Treman Marina	210,543
2013	Finger Lakes	Tompkins	Allan Treman Marina	292,200
...	...	...	...	...

We focus on US government data. We denote with $\mathbf{\Sigma}$ the set of geographic types of interest as shown in Figure 3.1. In practice, this is usually a small finite set. For every type $\sigma_i \in \mathbf{\Sigma}, \mathbf{i} \in \mathbb{N}_{[\mathbf{1},|\mathbf{\Sigma}|]}$, there exists a $\mathbf{\Omega_i}$ as the set of geographic units of geographic type σ_i. Each pair of geographic types $(\sigma_i, \sigma_j) \in \mathbf{\Sigma}$ are either related or not, through an inclusion relationship. If σ_j includes σ_i, then for every geographic unit $x \in \mathbf{\Omega_i}$ there exists some geographic unit $y \in \mathbf{\Omega_j}$ that covers the area projected by x entirely. For example, a census tract is entirely within a county. Otherwise, for each $x \in \mathbf{\Omega_i}$, it is possible that the area of x is partially overlapped with the area of some geographic unit $y \in \mathbf{\Omega_j}$. For example, a zip code can include portions of many counties.

Let T be a flat file (SQL table) of government data. We denote the set of columns and the set of rows in T as $\mathbf{C}$ and $\mathbf{R}$, respectively. The set of variables in T is denoted by $\mathbf{V}$ and the set of observations by $\mathbf{O}$. $\mathbf{V}$ can be decomposed as $\mathbf{D} \cup \mathbf{M}$, where $\mathbf{D}$ is the set of dimensions and $\mathbf{M}$ is the set of measures. Let $\mathbf{G}$ be the set of geographic dimensions in T such that $\mathbf{G} \subset \mathbf{D}$, then for every geographic dimension $g_i \in \mathbf{G}$, its data entries are geographic units of some type $\sigma_i \in \mathbf{\Sigma}$ forming a subset of $\mathbf{\Omega_i}$.

Illustrative Example. *Table 3.3 shows a snippet of the government data table for the annual attendance at state parks by facility in New York State. In this example, the geographic domain of interest is New York State. In the complete table, there are 2820 rows in R and 5 columns in* $\mathbf{C}$*. Each row is a facility in the state observed in some year corresponding to an observation in* $\mathbf{O}$*. Each column corresponds to a variable in* $\mathbf{V}$ *for the years, the New York State Office for Parks, Recreation and Historic Preservation (OPRHP) regions of state parks, the counties of state parks,*

the facilities and the number of attendance summarized for facilities of some year, respectively. The first four variables form the set of dimensions $\mathbf{D}$ *and the last variable is the only measure in* $\mathbf{M}$*. Among all dimensions,* `OPRHP Region` *and* `County` *are geographic dimensions in* $\mathbf{G}$*.* `County` *has its data values covering all 62 counties in the set of geographic units* $\mathbf{\Omega_i}$ *for the type of county* $\sigma_i \in \mathbf{\Sigma}$*. Though* `OPRHP Region` *is a geographic dimension, we won't try to match its values with real world geographic units as OPRHP region is not a geographic type in* $\mathbf{\Sigma}$ *and hence cannot be matched.*

We say that a data table is *tidy* or in *canonical form* if

1. There is exactly one column for each variable. That is, $\mathbf{V}$ and $\mathbf{C}$ have a one-to-one correspondence.

2. There is exactly one row for each observation. That is, $\mathbf{O}$ and $\mathbf{R}$ have a one-to-one correspondence.

3. Each type of observational unit forms a table–i.e., observational units that share the same dimension variables are categorized together to form a table.

3.2.2 The Integration Problem

The integration problem is defined for two government datasets. It assumes that both datasets conform to the canonical form. Mathematically, we define the inputs of the integration as two government data tables $T_k, k = 1, 2$.

The problem requires identification of a series of set elements associated with each T_k. We must distinguish the measures in $\mathbf{M_k}$ from the dimensions in $\mathbf{D_k}$ and correctly subset $\mathbf{M_k}$ into the set of geographic dimensions $\mathbf{G_k}$ and the set of non-geographic dimensions $\mathbf{M_k} \setminus \mathbf{G_k}$. T_k is subject to the constraint such that $\mathbf{G_k} \neq \emptyset$ so that there is at least one geographic variable in each data table to permit a join based on geographic information.

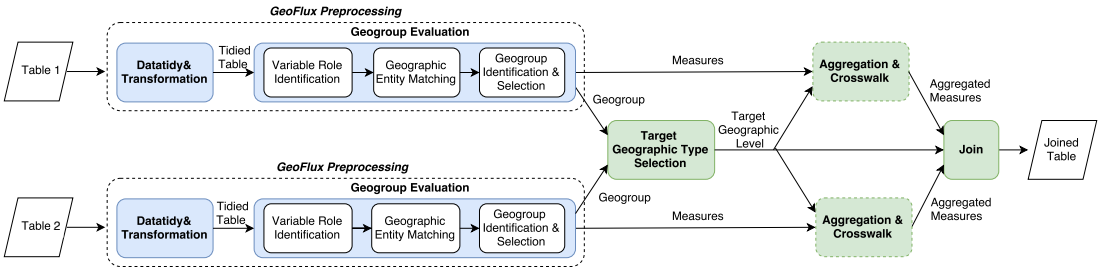


Figure 3.2: `GeoFlux` System Flow Diagram

To enable successful join, we must be able to match geographic dimensions to standard geographic types in $\mathbf{\Sigma}$ and match data entries of each geographic variables to values in $\mathbf{\Omega_i}$ corresponding to its type σ_i. In addition to this, we also need to transform both data tables so that they are aggregated at a common geographic granularity with approximation techniques. Let g_t be the geographic type selected as the proper join key type (or the target geographic type). Each input $T_k, k = 1, 2$ is transformed into T'_k such that $\mathbf{M_k}$ are reevaluated from the source dimensions $\mathbf{D_k}$ to the dimension of type g_t as $\mathbf{M'_k}$. Note that $\mathbf{M_k}$ and $\mathbf{M'_k}$ measure the same underlying measures aggregated over different types of geographic units. The integrated $T_1 \cup_{g_t} T_2$ is a flat table T_{g_t} following the canonical form such that

1. $T_{g_t} = T'_1 \bowtie_{g_t} T'_2$
2. The set of measures $\mathbf{M_{g_t}} = \mathbf{M'_1} \cup \mathbf{M'_2}$.
3. The set of dimensions $\mathbf{D_{g_t}} = \{g_t\}$.

3.3 System Overview

For multiple challenges common in the process of socioeconomic data integration, we propose automatic approaches to solve each of them respectively. Incorporating these ideas, we build a system, called `GeoFlux`, with a pipelined modular architecture that targets the data integration problem without any user intervention.

Figure 3.1 shows the components of `GeoFlux`. Each input data table is first analyzed for integration in the ***`GeoFlux`*** *Preprocessing* step. The optional *Datatidy & Transformation* module transforms a messy table into canonical form. The *Geogroup Evaluation* module then analyzes the statistical types of variables. Entries of geographic dimensions are matched with a standard geographic library to identify the real-world geographic entity they represent before the selection of candidate geographic join variable(s). The *Target Geographic Type Selection* module then determines the best target geography for the join of the two tables. Each table is aggregated by the source geography and crosswalked to the target geography if necessary in the *Aggregation & Crosswalk* module before the actual join (c.f. §3).

For efficiency of processing, all steps are carried out on individual data tables as far as possible. The first two modules depend only on the data being processed. The third module does depend on the pair of tables involved, but the computational complexity is minimal. Though the fourth, aggregation and crosswalk, module depends on the output of the third module, the actual computations are performed individually, on one table at a time. The very last step is the actual join, requiring simultaneous access to both tables.

The system is invoked through a simple user interface. The user merely identifies, or provides, the two input tables, and obtains a joined table as the result. The user optionally has access to intermediate results after each module. She can use this capability to make corrections, and to store the intermediate results, if desired, for later reuse (e.g., if the same table is later joined with something else). The following sections provide more details on the individual modules.

3.4 Datatidy & Transformation

Our formal problem statement calls for tables in canonical form, but many data tables in practice may not be in this form. In our examination of government data,

we found two common classes of structural deviation from canonical form: column headers are values instead of variable names and multiple variables and their values are stored in one column header. In this setion, we describe how to transform such tables into canonical form.

3.4.1 Tidying Challenges

In the motivating example in Table 3.1, for example, the `In-State Region` variable has seven region values as column headers of seven columns; and in the truncated Table 3.4, employment for `Month in Quarter` is spread across 3 columns per `Area`, one column for each month in the quarter. We call such columns *messy*.

In general, one can gain an automated understanding of the structure of a given table based on the schema or the data. In our case, the latter is not likely to be useful, since the data primarily comprise aggregate value entries for measure variables, and the system has no knowledge of value ranges for these. Therefore, messy column recognition has to be performed based solely on schema information, and specifically, column names and value types.

If column names of recognized messy columns are considered entries of unstructured data, existing solutions for Multi-Sequence alignment (MSA) and List segmentation [12, 13, 71] provide various techniques for solving the tidying problem as a multi-variable extraction problem. However, these solutions require domain knowledge in the form of data labels, background corpus, etc., which the system may not have. For geographic data integration, columns involving geographic information are critical. Luckily, there are only a finite number of geographic types of interest in any universe of discourse. We can create a dictionary of valid geographic type names, and values, for our universe, and use this dictionary to advantage. For each geographic type, we have a corresponding dictionary of standard geographic names, as well as "nicknames", abbreviations and other alternative names, for geographic units of the

type. Any complications in the structure of other columns, not involving geography, remain unresolved by this approach. Fortunately, it can usually be propagated from input to output with no major ill-effect.

3.4.2 Datatidy Approach

When column names contain values, there will be a set of columns to represent a (geographic) variable, one column per value. We call such a set of columns a *messy column group*. A single table can have multiple messy column groups, and they can be messy in different ways. Column names in a messy column group are both syntactically and semantically similar, that is, they usually have a common *naming pattern*, in terms of the order of values and variable names, and the terms and term types of values in column headers. For example, in the motivating example in Table 3.1, the columns broken down by region has the names of regions as column headers. They are all values of the `In-State Region` variable. Whereas in Table 3.4, employment of `Month in Quarter` of Albany is spread across 3 columns. Their column headers has the following order: `Albany` for the name of an area, `Month` for the `Month in Quarter` variable, number for the value of `Month in Quarter`, and `Employment` for the number of employment variable. The first step of data tidying is to identify messy columns sharing common naming pattern(s) and value types, and group them together into messy column group(s). Then, we check for the number of underlying variables in each messy column group to determine which class of messy columns it has.

In case of the first class of deviations with one such variable, we apply a *melt* transformation to turn columns into rows. We keep variables in columns that are tidy, *colvar*s for short, unchanged. And then convert the messy column group columns into two new columns: one called `Header` (or the geographic type for messy geographic variables, if we have been able to identify it) that contains values occurring in column

headers and the other called `Value` that contains the reconciled data entries from the columns. We illustrate the technique again with Table 3.1. From the information we learned from the headers and to better represent the roles of the data, `Header` is renamed to `In-State Region` as its values are matched with regions in the region dictionary of New York State and `Value` is renamed to `Assists`.

For the second class of deviations, with multiple underlying variables, `Header` is split after melt is applied such that each variable has a column of its own. The molten table of Table 3.4, for instance, should have a new `Header` column for repeated messy column headers of month in quarter employment of some area and a new `Value` column renamed to `Employment` for the concatenated numbers of employment. The `Header` column is then split horizontally into `Area` and `Month in Quarter` columns.

3.4.3 Transformation Approach

Some input tables may include multiple messy column groups. If one messy column group is transformed and parameterised by the original colvars, the second will be parameterised by the updated colvars: the original colvars and the new colvars obtained from tidying the prior messy column group. The tidied table may then not be meaningful as the variables in each messy column group is described by the original colvars and the misplaced variables in the group, but not the variables in other messy column groups. The underlying problem here is that the observational units are not consistent across the transformed messy column groups.

To address this challenge, we propose to decompose the tidied dataset into multiple subtables linked by a foreign key instead for space compression. We store the original colvars into a new table, the colvars table, and assign a primary key `_Id` to it. Then for new columns created by tidying each messy column group, we have a new table for them referencing the colvars table by the foreign key `_Id`. As in Table 3.4, the colvars are stored in the colvars table with an ascending id number whereas the split

molten table is linked to the colvars table by id.

3.5 Geogroup Evaluation

Before the crosswalk and join are possible, three main preprocessing subtasks are needed: variable role identification, geographic entity matching, and geogroup selection.

3.5.1 Variable Role Identification by Classification

Recall that *measures* are quantitative variables for features of interest and *dimensions* are qualitative variables that characterize and parameterize the *measures*. First off, we have to automatically determine which variables are dimensions and which are measures based on their statistical characteristics. Second, among the variables that represent dimensions, we need to identify the variables that encode geographic information.

Variable types provide rough classification. Variables of integer and double types are likely to be measures whereas variables of string type are dimensions. However, this may not always be the case for numeric variables, especially for integers. `NAICS Code` values in Table 3.4, for instance, are all integers, but this code should be a dimension rather than a measure. A possible improvement is to consider the coherence of values of the variable. Such dimensions tend to have a fixed number of digits whereas the integer measure variables may have more variable number of digits. In addition to these characteristics, we create a small library of words that could be matched with variable names to further improve accuracy.

To automate the process, we formulate the role identification problem as a multi-class classification problem. We represent each variable (e.g., zip code) with a feature vector, the creation of which is based on k values that are selected uniformly at random from the input table. The feature vector representation consists of: (i)

individual-level features that are determined by majority voting, such as number of digits, whether the value can be converted to a numeric value, etc.; and (ii) sample-level features such as unique value rate, coefficient of variation, whether the column is matched with a geographic level, etc. As the feature vector describes a limited number of low-cardinality categorical predictors and continuous predictors at the same time (8 features in total), we use Random Forests [67] which handle such classification efficiently by optimizing the parameters of the weak learner at each split node j of trees **Tr**:

$$\hat{\theta}_j = \arg \max_{\theta_j \in \mathbf{Tt}_j} I_j, \tag{3.1}$$

such that I_j is the objective function in the form of a classical information gain of the distributions of features. In our scenario, we trained the classifier on 159 variables (with $k = 1000$), which were labeled as measures, geographic dimensions and non-geographic dimensions, and found over 94% precision and recall per class and an overall accuracy of 95.6% for all classes.

3.5.2 Geographic Entity Matching

Geographic entities are not always consistently represented in various government datasets. They can be represented by conventional names, abbreviations, geographic codes, etc. The representation of geographic entities introduces ambiguities as different areas may share the same name or abbreviation. `Albany` is the name of a county in New York State and the name of cities in multiple states in the United States. Given `Albany` in city column, we cannot decide which `Albany` city it is based solely on this entry. Rather, we need to look at extra data, such as possibly the state value entered in the next column, to decide. To ensure the accuracy of matching, this submodule matches geographic entities in context.

We define geographic columns/variables describing semantically the same location as a *geogroup*. When matching geographic entities, we not only consider the geographic type of the variable but also the geogroup that contains it and the entities in the column it belongs to. It is thus necessary for us to identify geogroups for geographic variables before matching geographic entities.

From empirical analysis, we found geographic variables in the same geogroup tend to have a common naming pattern. We mean that the token or token types and order of tokens in variable names are the same. Consider the lobbyist geogroup in Table 3.2, the column name strings of variables in the geogroup has the tokens in the following order: the name of the geogroup `Lobbyist` followed by a geographic type. Another geogroup has a similar pattern: the name of the geogroup `Client` followed by a geographic type. It is natural for us to use pattern matching to group geographic variables without additional knowledge of a geogroup.

Once geogroups are determined, we match entities of geographic variables in the context of geogroups. For each geographic type, we define a dictionary of geographic entities in the domain of interest with their names, abbreviations, geographic codes, nicknames, etc as the *geo value reference table*. To match a geographic value with the true area it refers to, we first match it with entities in the *geo value reference table* of the type it belongs to based on text semantic similarities [75]. The entity matching process may return multiple candidate matches if there are ambiguities in naming or representation of the geographic value. Secondly, we verify candidate matches by reviewing geogroup information of the geographic value and eliminate candidates with mismatch. More specifically, if a candidate match is not in the geographic scope defined by the other member variables of geogroup, it is eliminated. Lastly, if a geographic value belongs to no geogroup, we estimate its geographic scope by referring to other values in the same column. If the majority of the geographic entities in the column fall within a certain area, we assume the entity represented by

this geographic value is also in the area. This is illustrated by the matching process again with the `Albany` example. After matching with the geo value reference table for cities, `GeoFlux` returns five cities named `Albany` as matching candidates. If the city column is in the same *geogroup* with a state column and this `Albany` city is coupled with `New York` state, there is one candidate left in the New York State and this completes the matching. However, if the city column exists by itself, `GeoFlux` investigates candidate matches of other geographic values in the city column and finds out that the majority of them are in New York State, then the Albany city in New York State will be assumed to be a better match than other Albany cities in other states.

3.5.3 Geogroup Selection by Learning to Rank

A government table can have multiple geogroups at the same time. To perform a join based on geographic information, we need to decide which location better represents *the* geographic joining location. Not all geogroups are equally important. We assume that the more important a geogroup is, the higher the quality of its data. For example, in the Registered Lobbyist Disclosures data in Table 3.2, if filing by lobbyist location is more important than by client location, then the lobbyist location field is less likely to be left empty and thus its *completeness* should be higher.

Based on this intuition, we evaluate the overall quality of geogroups by five data quality dimensions chosen from data quality assessment criteria [92, 84, 102] and choose the one with the highest quality as the geographic joining location. These five dimensions are intrinsic criteria in the sense that they evaluate data quality in their own right without considering the contextual quality [102] in the context of the joining task. These five dimensions are:

- Completeness: the proportion of stored data against 100% filling of fields.
- Validity: the proportion of data entries that conform to the syntax (format, type, range) defined. Additionally in `GeoFlux`, we check for geographic entries

within the geographic scope, New York State, in our case.

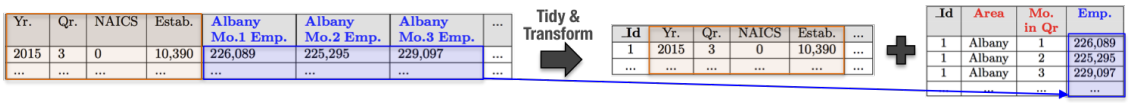
- Uniqueness: the proportion of unique entries among all entries.
- Consistency: the degree to which the same underlying geographic unit is represented with no difference.
- Joint accuracy: the degree to which geographic data in a geogroup correctly describes the real world location jointly with no conflict. We consider the proportion of data correctly matched in the *Geogroup Entity Matching* submodule.

We formulate the geogroup selection problem as a ranking problem that can be solved by learning to rank [66]. The training data consists of data tables and their geogroups. Each table can have multiple geogroups and the importance of each geogroup for the table joining task can vary. The importance is represented as a label for grade evaluated from their rankings of the task. The grade is assigned as the difference of the number of geogroups in a data table and the rank of the geogroup. The higher grade a geogroup has, the more important it is for the task.

Suppose $\mathcal{T}$ is the data table set and $\mathcal{G}$ is the geogroup set. We further suppose that there exists a total ordering between the grades $l \prec l-1 \prec ... \prec 1$ of the grade set $\mathcal{L} = \{1, 2, ..., l\}$. We define $\mathbf{y_i} = \{y_{i,1}, y_{i,2}, ..., y_{i,n_i}\}$ as the label set for the i-th data table $t_i \in \mathcal{T}$ whose geogroup set is $\mathbf{\Gamma_i} = \{\boldsymbol{\gamma}_{i,1}, \boldsymbol{\gamma}_{i,2}, ..., \boldsymbol{\gamma}_{i,n_i}\}$, where n_i denotes the size of $\mathbf{\Gamma_i}$ and $\mathbf{y_i}$. Then $y_{i,j}$ is the grade label representing the importance of geogroup $\boldsymbol{\gamma}_{i,j}$ for the joining task of the i-th data table t_i. A feature vector $f_{i,j}$ is created for each data table-geogroup pair $(t_i, \boldsymbol{\gamma}_{i,j})$ for the five data quality dimensions. The training set is $\{\mathbf{f_i}, \mathbf{y_i}\}_{i=1}^{m}$, where $\mathbf{f_i} = \{f_{i,1}, f_{i,2}, ..., f_{i,n_i}\}'$.

We train a ranking model $\phi(t, \mathbf{\Gamma}) = \phi(f)$ that can assign scores to a given pair of data table d and geogroups $\mathbf{\Gamma}$ based on their feature vector f for their ranking. Since the value of grade label is not numerically meaningful by itself, we adopt the pairwise approach that transforms the ranking into pairwise classification solved by Ranking

Table 3.4: Quarterly Census of Employment and Wages Quarterly Data: Beginning 2000. (Mo.=Month)



SVM [57]. New feature vectors for geogroups of the same data table are generate from the their pairwise differences, e.g., $f_1 - f_2$ and the corresponding label is computed as the sign of the difference, e.g., $sign(y_1 - y_2)$. The training data is thus converted to $\{(f_i' = (f_{i1}, f_{i2}), y_i')\}_{i=1}^{m}$ where $y_i' \in \{+1, 0, -1\}$. Solving the problem is equivalent to solving a Quadratic Programming problem for weights w of features such that

$$\begin{aligned} & min_{w,\epsilon} \frac{1}{2}||w||^2 + C\sum_{i=1}^{m} \epsilon_i \\ \text{s.t. } & y_i\langle w, f_{i1} - f_{i2}\rangle \geq 1 - \epsilon_i \\ & \epsilon_i \geq 0 \text{ for i=1, 2, ..., m} \end{aligned} \tag{3.2}$$

Due to limited access to data tables with multiple geogroups, the model is trained with 20 data tables (with 2-5 geogroups each) labeled by 3 trained labelers (who had high agreement, Kendall's $\tau > 0.9$). In the few cases of disagreement, the final labels were obtained via majority voting. As we care for the highest ranked geogroup only, we evaluate the mean reciprocal rank (MRR) that computes the inverse position of the most important geogroup in our case. A 3-fold cross-validation is computed with averaged MRR 97.6% that proves the feasibility of the model.

3.6 Target Geo-Type Selection

To integrate two datasets, we must select a target geographic type for aggregation that can serve as the join key. The selection procedure takes the geogroup of the highest data quality in both datasets as input and the target geographic type for joining as output. Since each geogroup may have geographic dimensions of multiple geographic types, this target geographic type cannot be simply determined by eye-

Algorithm 1: Target Geographic Type Selection

Input: Two geogroups of the highest overall quality
$\boldsymbol{\gamma}_k = \arg\max_{\boldsymbol{\gamma}_{k,j} \in \boldsymbol{\Gamma}_k} \hat{y}_{k,j} = \{\gamma_{k,1}, \gamma_{k,2}, ..., \gamma_{k,n_k}\}$ from data tables T_k, where $k = 1, 2$; n_k is the number of geographic dimensions in $\boldsymbol{\gamma}_k$ such that $n_k > 0$.

Output: Target geographic type g_t and the pair of geographic dimensions p to be crosswalked

$S_{st}, S_{pc}, S_{cr} \leftarrow \emptyset$;
foreach $\gamma^{(1)} \in \boldsymbol{\gamma}_1$ **and** $\gamma^{(2)} \in \boldsymbol{\gamma}_2$ **do**
  if $Acc(\gamma^{(1)}) < \alpha$ **or** $Acc(\gamma^{(2)}) < \alpha$ **then**
    continue;
  else if $GeoType(\gamma^{(1)}) == GeoType(\gamma^{(2)})$ **then**
    S_{st}.add($\langle \gamma^{(1)}, \gamma^{(2)} \rangle$);
  else if $IsParentChild(\gamma^{(1)}, \gamma^{(2)})$ **then**
    S_{pc}.add($\langle \gamma^{(1)}, \gamma^{(2)} \rangle$);
  else if $ExistApproxCrosswalk(\gamma^{(1)}, \gamma^{(2)})$ **then**
    S_{cr}.add($\langle \gamma^{(1)}, \gamma^{(2)} \rangle$);
if $S_{st} \neq \emptyset$ **then**
  $p \leftarrow \arg\max_{p_i \in S_{st}} Depth(p_i[\gamma^{(1)}])$;
  $g_t \leftarrow GeoType(p[\gamma^{(1)}])$;
else if $S_{pc} \neq \emptyset$ **then**
  $p \leftarrow \arg\min_{p_i \in S_{pc}} \frac{|Depth(p_i[\gamma^{(1)}]) - Depth(p_i[\gamma^{(2)}])|}{\min(Depth(p_i[\gamma^{(1)}]), Depth(p_i[\gamma^{(2)}]))}$;
  $g_t = GeoType(\arg\max^{i=1,2}_{p[\gamma^{(i)}]}(Depth(p[\gamma^{(i)}]))$;
else
  $p \leftarrow \arg\min_{p_i \in S_{cr}} \frac{|Depth(p_i[\gamma^{(1)}]) - Depth(p_i[\gamma^{(2)}])|}{Depth(p_i[\gamma^{(1)}]) + Depth(p_i[\gamma^{(2)}])}$;
  $g_t = GeoType(p[\gamma^{(2)}])$;

balling. The main idea is that geographic types in lower levels in the hierarchy (Figure 3.1) are preferred and the number of dataset crosswalks should be minimized. To be more specific, we would like to have a target geographic type with small area units such that at most one dataset needs to be crosswalked to it. These two requirements preserve both areal variation and statistical accuracy of the data.

Algorithm 1 describes the selection procedure for target geographic type in three steps. For each pair of geographic dimensions from the two geogroups γ_1 and γ_2 selected by §3.5 from two tables T_1 and T_2, we first rule out geographic dimensions with low quality, that is, variables with too many missing or unmatchable entries, computed by the $Acc(\cdot)$ and constrained by the threshold α. These variables are not taken into consideration as reliability of the data is hurt by lost measure values

associated with their invalid entries in propagation. We take common geographic types in two geogroups as preemptive target geographic type candidates to avoid possible crosswalk. Thus in the second step, we check for common geographic types in the remaining geographic variables and prefer the one in the lowest level of the standard geography in Fig. 3.1 with the depth of the geographic type computed by $Depth(\cdot)$. If there are no such candidates, then one of the data tables has to be crosswalked from a source geographic type to a target geographic type. In the last step, we choose the target geographic type for such data tables. According to the areal hierarchy of geographic types, the aggregation of data collected from lower level type units to its ancestor type units is statistically accurate though it loses information—the fewer the levels between types, the less information is lost. Hence crosswalk is accurate between a pair of geographic dimensions whose types are on the same line track of the hierarchy with the fewest links in lower level; its upper level type is the target geographic type. However, it is possible to have no pairs on the same line track, in which case crosswalk with approximation is needed. For pairs with crosswalk approximation algorithms, we choose the target geographic type of the pair with the minimal difference-over-sum objective for their levels.

3.7 Crosswalk with `GeoAlign`

Once a target geographic unit has been selected, as described above, we must convert one or both tables to represent aggregates at the desired level, through a process of *reaggregation* before finalizing the join. If individual level data are available, they can be reaggregated directly; if the target geographical type is an ancestor of the source type in Figure 3.1, then we can simply aggregate the data in hand appropriately, with the assistance of the hierarchical geographic relationship files publicly available[99]. In general, if neither of these two situations holds, we have to estimate values for the measure variables at the desired target level of aggregation. For ex-

ample, if one table is aggregated by county and another by zip code, they cannot be joined as the boundaries of these two sets of units rarely coincide. For such incompatible geographies, the approximation of aggregated data from the source geography to the target geography, also called *crosswalk* or *interpolation*, is an inevitable task to permit a successful join.

To better discuss the problem, in Chapter IV, we define the interpolation problem, present the challenges, and finally propose the `GeoAlign` crosswalk algorithm that outperforms the state-of-the-art methods.

3.8 Experimental Evaluation

We experimentally evaluated our current implementation of `GeoFlux`. The system is developed in Python with embedded MySQL DBMS version 14.14 used for data storage. All experiments were performed on a 2.3 GHz Intel Core i7 with 8 GB memory and a 7200 rpm SATA disk.

We evaluated the feasibility of the system from two crucial aspects: whether the system can correctly complete the join of two tables based on geographic information in an automated manner (effectiveness), and whether the runtime of the system is fast enough (efficiency). We also studied the performance of individual modules since their performances are inter-dependent. Finally, we report on additional experiments that examined the performance of specific components with special considerations: the *Datatidy & Transformation* and *Geogroup Evaluation* modules at the front-end, and the `GeoAlign` algorithm in the *Aggregation & Crosswalk* module at the back end.

Table 3.5: Independent Module Effectiveness Indicators

Dataset	Datatidy ACC	Geogroup					Pre-join Data Entity Correctness
		Mea. & Dim. ACC	Geo. & Non-geo. Dim. ACC	Geo. Dim.	Geo. Entity Matching ACC		
					Unique Match	Nonunique Match	
NYC Asian Subgroups Population	1	1	1	borough	1	1	1
DYCD After-school Programs	1	1	1	borough_community	1	1	1
State Park Facility	1	1	1	county	0.83	0.95	0.95
Health Coalitions	1	1	1	city	0.48	0.65	0.65
				state	1	1	1
				zip_code	1	1	1
Public Determinations on Judicial Conduct	1	1	1	county	1	1	1
Sporting License Issuing Agents	1	1	1	county	0.04	0.05	0.05
				city	0.11	0.13	0.13
				state	1	1	1
				zip	1	1	1
Adult Care Facility	1	0.86	1	county	1	1	1
School Progress	1	0.75	0.93	county_name	0.97	0.97	0.97
BUSNET Operator	1	1	0.80	city	0.06	0.20	0.14
Quarterly Census Employment and Wages	1	0.91	1	state_fips	1	1	1
				county_fips	0.98	0.97	0.97
				county_name	0.97	0.97	0.97
Index Crimes	1	1	1	county	0.98	0.98	0.98
DSNY Graffiti	1	0.83	1	borough	1	1	1
Hospital Inpatient Prevention Quality	1	1	1	patient_zip_code	1	1	1
Oil and Gas Production	1	0.94	1	county	0.97	1	1
				town	0.95	1	1
Spill Incidents	1	0.95	0.84	county	0.74	0.99	1
				zip_code	0.95	0.09	0.09

3.8.1 `GeoFlux` Prototype Evaluation

3.8.1.1 Workload Selection

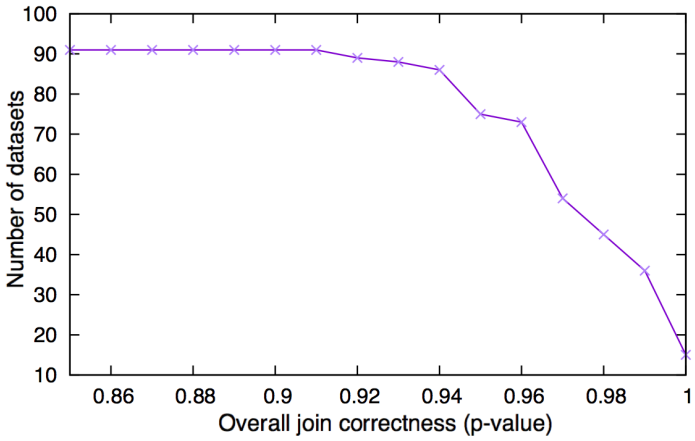
Users may join any pair of tables and synthetic data may not cover unknown problematic conditions worth understanding. Therefore, it is important to test the effectiveness of the system over real datasets. We worked with OPEN-NY, which is the web platform for up-to-date open government data for New York State covering socioeconomic topics. As of July 20th, 2016, there were 1977 datasets available on OPEN-NY and 1752 of them were in tabular format. We chose 40 tabular datasets randomly; Of these 40 tables, 21 had no geographic information and 4 had geographic types not known to `GeoFlux`. The remaining 15 datasets each had at least one geographic type maintained by `GeoFlux`. The size of these data files ranges from 1KB to 99 MB and the number of columns they contain ranges from 7 to 23.

The evaluation was performed on 105 joins using every pairing of the selected datasets in CSV format.

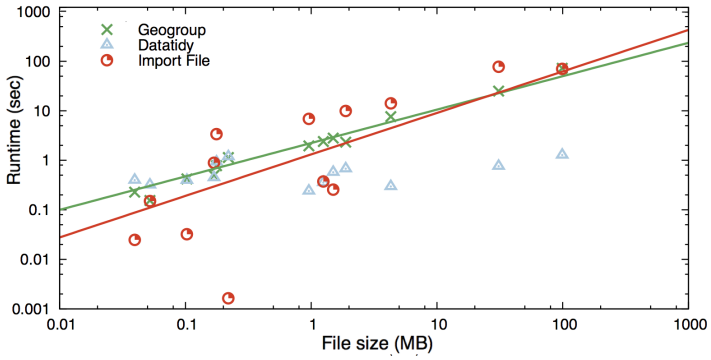
3.8.1.2 Overall Effectiveness

When we consider overall effectiveness of the join, we care about whether data entries are correctly represented for both types of variables in the join result: geographic dimensions and measures. Among all geographic dimensions in each input dataset, one is selected as the source geographic type before joining with the other dataset after possible crosswalk. In each record, if the entry of the variable of the source geographic type is correctly matched with a geographic entity in the corresponding geo value reference table, the measure entries of the record can be successfully realigned via crosswalk to its target geographic unit. We report the entity matching quality of entries of the selected geographic variable of the source type and the proportion of measure entries that can be realigned for each input.

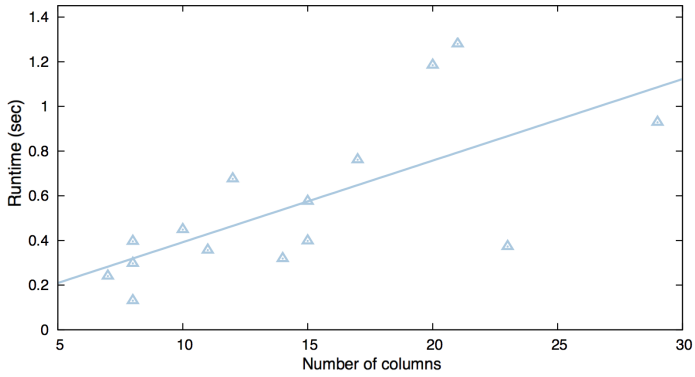
We computed an indicator of overall correctness of the joined result as the product of results from the pre-join evaluation metric that computes data entry correctness for each input table. We defined the metric as the sum, over all measure entries, of correctly identified measure entries whose geographic value of the source geographic type is correctly matched standard geographic entity. By correctly identified measure entries, we mean data entries of true positive measures, which are measure variables correctly identified as measures, in the classification of measures and dimensions. The range of the overall join correctness indicator is between 0 and 1. As shown in Figure 3.3(a), if the overall join correctness (p-value) increases, the number of pairwise joins whose overall join correctness is greater or equal to p-value will gradually decrease until it drops dramatically around p-value= 0.91. We found that 91 out of 105 pairwise joins has an overall join correctness over 0.91. In other words, for 91/105 pairs, `GeoFlux` produced a join result that was more than 91% correct, demonstrating its overall effectiveness.



(a) Pairwise joins with overall join correctness $\geq$ p-value.



(b) Log-log plot of runtime of pre-join modules over dataset size



(c) Datatidy & transformation runtime over number of attributes.

Figure 3.3: Correctness of join and runtime performance of pre-join modules.

3.8.1.3 Independent Module Effectiveness

We further investigate the effectiveness of *Datatidy & Transformation* and *Geogroup Evaluation* modules and their contributions to the overall join correctness.

We computed accuracy (ACC) of binary classification tests [93] as the indicator of effectiveness. A binary classification test defines two outcomes, positive if examples belong to a class and negative otherwise. ACC calculates the sum of the number of correctly recognized class examples (true positives) and the number of correctly recognized examples that do not belong to the class (true negatives), over all examples. The detailed results are listed in Table 3.5.

Datatidy & Transformation Effectiveness. This module checks and tidies mis-structured columns when necessary to place the table in canonical form. We define the outcome of mis-structured column identification as positive (classifying the columns as in need of tidying) or negative (classifying the columns as in no need of tidying). For all 15 datasets, `GeoFlux` accurately recognized them as not in need of tidying.

In addition, we conducted a standalone test for another 6 datasets that require data tidying. 155 out of 157 messy columns in these datasets were correctly identified and tidied, among which 149 are misplaced values of geographic variables. Two of them raised false negative errors due to the lack of information for conventional regions that are not formally defined by Census Bureau. One of the geographic areas that we failed to recognize is `Long Island`, which is not a standard geographic unit and comprises four counties: Kings, Queens, Suffolk and Nassau. Local people sometimes colloquially use it to refer to the latter two counties exclusively. Terms like this are not encoded in `GeoFlux`, and even if they are, their meaning is undetermined without extra knowledge. In short, the two classes of messy problems we defined covered every example we found in our datasets.

Geogroup Evaluation Effectiveness. In this module, variables, defined as columns after possible transformation in *Datatidy & Transformation* module, are tested for effectiveness in terms of accuracy (ACC) in three tasks: classification of variables

into measures and dimensions, classification of dimensions into geographic and non-geographic variables, and entity matching result of geographic entities.

The overall accuracy of the task for 218 variables in all test datasets was 93.6%. The accuracy showed high dependence on data types and success in variable name pattern matching. `GeoFlux` assigns any variable that contains qualitative, discrete information (for example, fields where the values are strings and Boolean values) to dimensions unless it has domain knowledge to suggest that the variable is measures. Variable names including `Month`, for instance, are recognized as dimensions in most cases. However, `GeoFlux` failed in classifying `Months in Production` in the Oil and Gas Production dataset. The variable is actually a measure that describes the number of months spent in the production of the merchandise. Another example is that the `Grade` variable, whose values are letter grades as strings in the `School Process` dataset, should be a measure rather than a dimension if considered in context.

In the test of geographic dimension classification effectiveness, the overall accuracy for 144 dimensions in all test datasets was 96.5%. Failure of classification in this task is also restricted by variable name pattern matching ability of the system with limited domain knowledge. As an example, the Spill Incidents dataset has low accuracy in this task as `street 1`, `street 2` and `locality` are misclassified as non-geographic variables. This is because `GeoFlux` has not learned that `street` and `locality` keywords refer to geographic areas.

We also evaluated the entity matching quality of data entries of geographic variables in 15 datasets. For each geographic variable, we consider the matching quality in terms of the rate of correctly matched unique geographic entities. Among 24 matchable geographic variables, 18 of them have roughly 95% unique values correctly matched with real world geographic entities. We further investigate datasets with low correctness rate. There are two major reasons for mismatches: geographies change over time and the data itself can be incorrect or inconsistent. Geographies don't stay

the same. Counties, as an example, are created and deleted or altered in boundaries by U.S. Census Bureau to adapt to modern data collection challenges. `GeoFlux` can not recognize geographic entities defined earlier if they do not exist in the 2010 Census Bureau Geography reference tables. Data quality is the other factor that influences matching quality. A common mistake in government data is that data entities does not match the geographic type setting of a variable. The `Sporting License Issuing Agents` dataset, for instance, has a `county` variable with 814 unique values in 1197 records, whereas only 30 of the unique values are actually counties in 61 records and the rest are halmets, villages, towns, etc.

3.8.1.4 `GeoFlux` Efficiency

Recall from Figure 3.2, the join operation itself is just a standard relational join. The special processing performed by `GeoFlux` is pre-join, and applied to one table at a time. Since the standard join remains unchanged, our focus is on the cost of the pre-join operations we perform on each table.

Figure 3.3(b) shows the log-log plot of runtime in seconds (y-axis) of pre-join operations over the file size of test datasets in MB (x-axis). The execution time of *Import File* and *Geogroup Evaluation* are both roughly proportional (note that the slopes of the trend lines for both modules are about to be 1 in the log-log plot) to the size of the file instances to be imported. This is intuitively obvious for *Import File* as its performance is dominated by the time it takes to read through files and to create tables to store files in database, which are both proportional to file size. In *Geogroup Evaluation*, the system scans over all geographic data entries in the table to match them with standard geographic entities. For a given dataset, let n, n_{col} and n_{geocol} be the number of data entries, number of columns and number of geographic variables respectively. The execution time for geographic entity matching is $O(n \times \frac{n_{geocol}}{n_{col}})$ or $O(n)$, when $n_{col} = n_{geocol}$ in the worst case. As for *Datatidy & Transformation*, the

runtime is not related to the file size but rather to the number of variables in each instances as depicted in Figure 3.3(c).

3.8.2 `GeoAlign` Evaluation

We used county and zip code as the two geographic types of interest to evaluate the performance of `GeoAlign` from two perspectives: effectiveness and efficiency. We further consider the scalability and robustness of the algorithm as real datasets can be large and erroneous. The details of the evaluation are discussed in Chapter IV.

3.9 Conclusions and future work

In this chapter, we introduced `GeoFlux` as an automatic data integration system that joins government data tables based on geographic information. We showed that high quality automated data integration is possible, at least for some limited contexts, still of great practical importance. We also introduced a `GeoAlign` crosswalk algorithm that presents better performance than state-of-the-art crosswalk algorithms, of which more details are discussed in Chapter IV.

Our central vision is to have the system automatically figure out how to combine information from two different tables identified by the user. For a focused, but practically important, class of applications, involving government data reported over geographic regions, we have realized our vision.

We recognize our approach of performing automatic integration is limited in the sense that the approach is domain-knowledge dependent. Prior knowledge of geographical level hierarchy and entities for each geographical level, for instance, are essential for performing target geo-type selection and crosswalk. Also, a quantified quality indicator for the integrated data is not visible to the end user and user input is voided in the entire pipeline. In this sense, our proposed integration might not satisfy user intentions. The user may prefer integrated data joined on county level

while the system reckon joining on zip code level as more statistically meaningful. Instead of complete automation, we may minimize user input in technically intense tasks while providing recommendations for non-technical tasks by suggesting a ranked list of options for user to choose.

We believe that our methodology can be extended to apply to other applications, such as tables with temporal information. Our next goal is to generalize our solution to work in a broader context.

CHAPTER IV

GeoAlign: Interpolating Aggregates over Unaligned Partitions

In this chapter we introduce `GeoAlign`, a novel multi-reference crosswalk algorithm that *estimates* aggregates in desired target units. This crosswalk algorithm helps realigning aggregate values from one set of units to another in an adaptive manner, permitting the joining of aggregated columns when the aggregation levels are incongruent with each other.

4.1 Introduction

Data are often found in silos, created independently. For example, administrative agencies and governments collect a great deal of data about their domain, most of which are then published in aggregate form. The primary purpose of the data collection is administrative, and the choice of data representation and structure is made by each agency for its own purpose. These data can be invaluable for understanding many social issues, particularly in conjunction with other data sources. However, most administrative agencies are not concerned with interoperability with other agencies, therefore standardization is unlikely. On the other hand, agencies value the privacy of individual citizens, and do not want any benefits from public

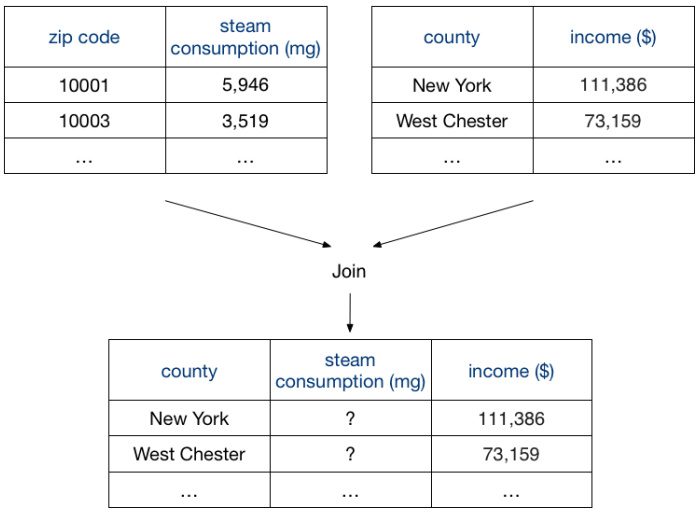


Figure 4.1: Join two tables for steam consumption (mg) and per capita income ($) in New York State together by county

data release to hurt their primary administrative mission. Therefore, in many cases, they will release data only in aggregate form. Similar reasoning applies in many other contexts as well. For example, Google Trends data is aggregated by geographical unit and time period, to avoid disclosing information about individual queries.

Data integration [65, 47] has been extensively studied, since there is often great benefit from joining multiple datasets. The bulk of the work on this topic addresses structural discrepancies, through schema mapping [5, 86, 49], and identification of individuals across datasets, through entity matching [60, 37]. One challenge not addressed in data integration is the case of data reported as aggregates over incompatible geographical/temporal units. This is a practical problem faced by government data center, NGOs, social scientists, and the general public when trying to related socioeconomic data to drive decision making processes, approximately 80% of which are related to a geographical location [30]. Even if the intention of joining such aggregated data based on their spatial or temporal properties seems to be the reasonable action of practice, these aggregates cannot easily be realigned accurately.

Motivating example. *Let us consider two tables shown in Figure 4.1 – one table*

has the steam consumption amount aggregated by zip code and the other has the per capita income reported by county. A sociologist wants to study the correlation of energy consumption with income in order to plan for future energy supply arrangement. Valuable insight could be obtained by joining these two tables. However, this is not straightforward since the data are reported on incompatible aggregate units, since one zip code may intersect several counties and one county may contain or overlap with multiple zip codes.

This challenge can be addressed by realigning one or both datasets to a common geographic type (target type) before performing the join. Let the intended target type be county, by which the per capita income is already reported. However, we only know the steam consumption amount by zip code, and have to estimate the number for each county. This estimate is obtained as a form of interpolation. Finding a good estimate of steam consumption per county is the challenge we need to address.

This problem of estimating aggregate values for geographic areas arises in many contexts, and has been extensively studied. *Areal Interpolation*, in Geographical Information Systems (GIS), is the process of aligning an attribute from one areal unit system (the source type of a set of polygons) to another spatially incongruent system (the target type of another set of polygons) [39, 64, 40, 62, 28]. It is more commonly known as *crosswalk*, or the *modifiable areal unit problem* in socioeconomic fields. If the attribute is uniformly distributed in space, then the interpolation can be performed in a straightforward way based on area. For example, if 70% of the area of a zip code lies in county A and 30% in county B, then we could estimate that 70% of the crimes reported in the zip code occurred in county A and the remaining 30% in B.

This uniform distribution assumption or homogeneity assumption rarely holds in practice. If we know something about the distribution, that can be taken into

account in the interpolation. For example, if we know that more crimes occur in densely populated urban areas than in sparsely populated rural areas, we can take this into account. The mathematics can be tricky depending on exactly what we know about the distribution of the attribute of interest, so there has been a stream of research in the literature towards solving the problem based on different assumptions.

In the data integration scenario, we often do not know much about an attribute of interest. Therefore, we may be unable to develop good rules for how it should be distributed. Even so, we can do better than make an unrealistic uniformity assumption, if we have access to additional data. In particular, if we can find a *reference* attribute, for which we know the detailed distribution, we can use it to perform a crosswalk from source units to target units of aggregation. For example, we may have detailed distribution available for population, with fine granularity aggregates giving us the population in every intersection of county and zip code. If we believe the crimes are distributed similarly to population (or at least more similarly to population than to area), then we can exploit our knowledge of population distribution to estimate the desired aggregates for number of crimes. In particular, consider a zip code with a population of 25,000 people. Suppose this zip code intersects two counties A and B, with the population in the intersections being 10,000 and 15,000 respectively. Suppose that we know there were 100 reported crimes in this zip code last year. We can estimate that 40 of these crimes occurred in county A and 60 occurred in county B, following the same ratio as the population. This approach makes no assumptions about the probability distribution of the reference attribute or the attribute of interest. It can work well if the attribute of interest is distributed similarly to the reference attribute. To the extent the distributions differ, the estimates will be off.

Our goal is to solve this data alignment problem through the use of more data. We often may have access to more than one candidate reference attribute, each with its own distribution. We may not have domain knowledge enough to understand

which reference is most similar to our variable of interest. Even if we found the best reference, its distribution may still not be close enough. Is there some way we can combine the information in the multiple reference attributes to do better? And at the same time, more adaptively predicts the estimates to new attributes of interest than using a single reference.

We develop `GeoAlign`, a technique that does just this. The idea is to weight their relative contributions to the final estimate so that the most similar reference attributes have the greatest impact on the estimate.

The intellectual contributions of the chapter are as follows:

- We define the general aggregate interpolation problem over unaligned partitions in one or more dimensions, which is an important problem in data integration (§4.2).

- We propose `GeoAlign`, an adaptive multi-reference crosswalk algorithm that solves the areal interpolation problem by realigning aggregates from source units to target units by learning distribution similarities between the attribute of interest and the reference attributes (§4.3). We show that `GeoAlign` can be used not just in two-dimensional maps but also for spaces with arbitrary numbers of dimensions.

- We evaluate the performance of `GeoAlign` against real data from data.ny.gov and Esri data in 2-dimensional space. These experiments show that `GeoAlign` outperforms the state-of-the-art single reference crosswalk approach in accuracy (§4.4). It is, at the same time, efficient, scalable and robust to noisy references even when limited references are available.

We conclude with future work (§4.5).

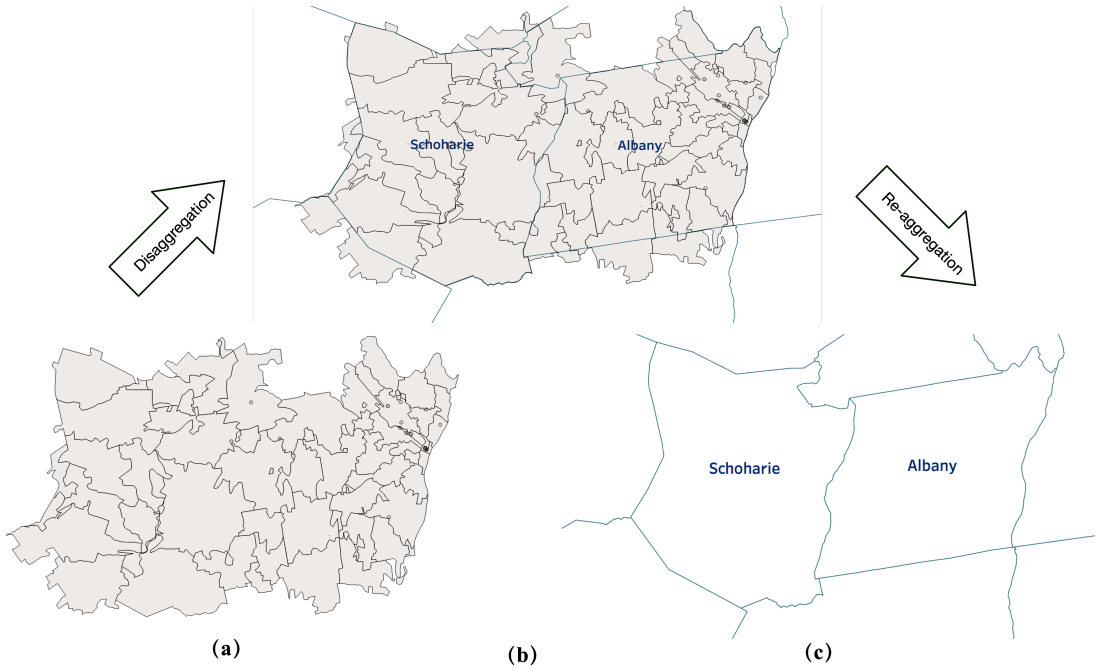


Figure 4.2: Examples of units in the partial map of New York State for aggregate interpolation: (a) zip code units (source units), (b) zip code and county intersection units and (c) county units (target units).

4.2 Problem Statement

In this section, we first introduce the terms we use throughout this chapter before we formally define the aggregate interpolation problem in multi-dimensional space. We then illustrate, with examples, the aggregate interpolation problem in 2-D and in other dimensions.

4.2.1 Preliminaries

In Geometry, an n-dimensional universe $\Omega \subset \mathbb{R}^n$ can be partitioned into some *unit system* γ^y composed of a set of *units* $U^y = \{u_1^y, u_2^y, ...\}$, where $\forall_{u_i^y \in U^y} u_i^y \subset \mathbb{R}^n$. Units in U^y satisfy

$$\forall_{u_i^y, u_j^y \in U^y, ij} u_i^y \cap u_j^y = \emptyset, \tag{4.1}$$

that is any pair of units in U^y is disjoint with each other since they have no spatial overlap in n dimensions. Suppose that an attribute of interest α_x exists, then we

denote its *aggregate vector* as $a_x^y = [a_x^y[1], a_x^y[2], ..., a_x^y[|U^y|]]$ such that $a_x^y[i]$ is the aggregate of α_x in the ith unit of U^y.

As an example in 2-D space, in the universe of New York State Ω, county partitions compose a unit system γ^y. They share no areal intersection such that they are spatially incongruent with each other. Steam consumption, which is the attribute of interest α_x, has its data in Figure 4.1 collected from such a set of county units U^y. Another possible unit system is zip code partitions. We can view the steam consumption column in the table as its aggregate vector a_x^y for the county unit system. Each entry of the vector represents the amount of steam consumption in some county.

4.2.2 The Aggregate Interpolation Problem

We define the following terms for the aggregate interpolation problem in $\mathbb{R}^n$:

- $U^s = \{u_1^s, u_2^s, ...\}$, *source units* of the *source unit system* γ^s in the universe Ω.
- $U^t = \{u_1^t, u_2^t, ...\}$, *target units* of the *target unit system* γ^t in the same universe.
- $a_o^s = [a_o^s[1], a_o^s[2], ..., a_o^s[|U^s|]]$, aggregate vector of the objective attribute α_o in source units. $a_o^s[i]$, the i^{th} aggregate of a_o^s, is collected from source unit u_i^s.
- $a_o^t = [a_o^t[1], a_o^t[1], ..., a_o^s[|U^t|]]$, aggregate vector of the objective attribute α_o in target units. $a_o^t[j]$, the j^{th} aggregate of a_o^t, is collected from target unit u_j^t.

Given U^s, U^t and a_o^s, *aggregate interpolation* approximates a_o^t as $\hat{a}_o^t = [\hat{a}_o^t[1], \hat{a}_o^t[2], ..., \hat{a}_o^t[|U^t|]]$.

Aggregate Interpolation Problem in 2-D When it comes to a 2-dimensional space $\mathbb{R}^2$, units are *simple polygons* consisting of straight, non-intersecting edges forming a closed path by pair-wise join. A unit in 2-dimensional space can be denoted by

$$u_i = (V_{u_i}, E_{u_i}) \text{ where } |V_{u_i}| = |E_{u_i}| = n_i, \tag{4.2}$$

where V_{u_i} is a set of vertices in $\mathbb{R}^2$ and E_{u_i} is a set of edges connecting the vertices in V_{u_i} such that every vertex is shared by exactly two edges. Then, u_i is the closed area formed by connecting n_i vertices in V_{u_i} by n_i edges in E_{u_i}.

This problem is referred to as the *areal interpolation* problem in the GIS community. The 2-dimensional space is the map; and the unit system, also recognized as feature layer in GIS, is composed of partitions delimited by boundaries of some geographic type. Some of the most widely used geographic types in demographic data are county, zip code, and more. For instance, as shown in Figure 4.2, U^s is the feature layer for zip code in (a); U^t is the other feature layer for counties in (c). Given the aggregates of steam consumption in zip codes a_o^s shown in Figure 4.1 from the motivating example, the aggregate interpolation problem in 2-D approximates the steam consumption in counties, $\hat{a}_o^t$.

Aggregate Interpolation Problem in other dimensions In the 1-dimension setting of the problem, units are *intervals* or *line segments* between two points such that

$$u_i = [u_{i_1}, u_{i_2}], \tag{4.3}$$

where u_{i_1} and u_{i_2} are two points on the real line $\mathbb{R}$. We may illustrate the problem as interpolation of population histogram aggregates for two sets of age intervals as depicted in Figure 4.3. In this case, we can treat the set of narrow bins of age in (a) as U^s, the set of wide bins of age in (b) as U^t, for the same range of age as the universe of interest Ω. Given the population histogram for narrow age bins, a_o^s, the aggregate interpolation problem in 1-D predicts the population histogram for wide age bins $\hat{a}_o^t$.

Unit system overlapping also exist in 3-D or higher dimensions. One example is 3-D GIS data, such as the distribution of disease, evaluated for cubic units of different size scales. Another example is the data collected for 4-D space (3D) and time

systems, such as environmental exposures, crosswalked to another system incongruent in both space and time units. For both cases, areal interpolation is the bridge to map the data across unit systems to enable side-by-side comparison with data from incompatible units.

4.3 Aggregate Interpolation by `GeoAlign`

In this section, we first introduce some additional definitions and notations used throughout the rest of the chapter and a general two-step solution solving the aggregate interpolation algorithm. We then lay the groundwork for the assumptions made by `GeoAlign` before exploring the details of the algorithm.

4.3.1 `GeoAlign` preliminaries

Before introducing the general steps to solve the aggregate interpolation problem, we further define the set of *intersection units* for the intersection unit system γ^{st} as $U^{st} = \{u_1^{st}, u_2^{st}, ...\}$, where $\forall_{u_k^{st} \in U^{st}}, u_k^{st} \subset \mathbb{R}^n$. Each intersection unit is a subregion within some source unit and some target unit, that is

$$\forall u_k^{st} \in U^{st}, \exists u_i^s \in U^s \wedge u_j^t \in U^t, u_k^{st} \subseteq u_i^s \text{ and } u_k^{st} \subseteq u_j^t. \tag{4.4}$$

It can be thus deduced that $|U^{st}| \geq max(|U^s|, |U^t|)$.

The aggregate vector of the intersection units for some attribute α_x is denoted as $a_x^{st} = [a_x^{st}[1], a_x^{st}[2], ..., a_x^{st}[|U^{st}|]$.

In the simplest case, the intersection units are the n-dimensional spatial intersections of source and target units. For instance, for the areal interpolation problem in Figure 4.2, U^{st} is the set of intersection areas between zip codes and counties in (b); and for the histogram realignment problem in Figure 4.3, U^{st} is the set of age intersection intervals between source and target bins. More fine-grained partitions of

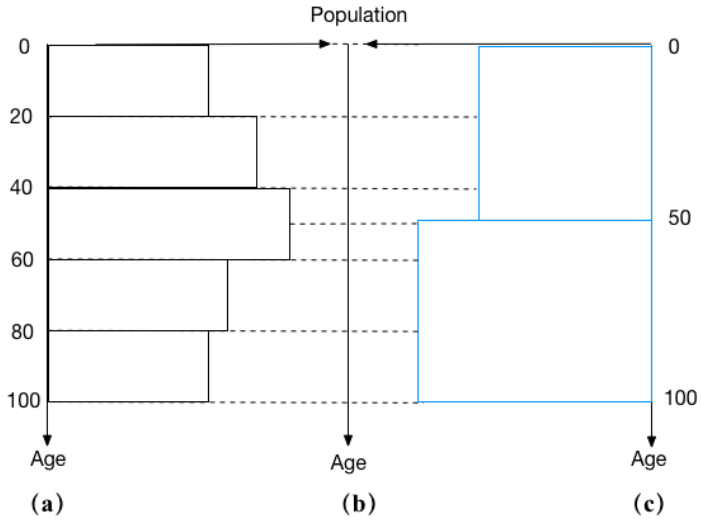


Figure 4.3: Realign population histogram in two sets of age intervals by transforming aggregates from (a) narrow bins to (c) wide bins. The dotted lines separate the age range into a set of tentative intersection units as in (b).

intersection units may be introduced if necessary when disparate spatial properties of the attribute in these partitions are introduced by auxiliary data.

Assume that the probability density function of attribute α_x for γ^{st} is a piecewise function, denoted as

$$f_x^{st}(z) = \begin{cases} f_x^{st}[1](z) & , z \subset u_1^{st} \\ f_x^{st}[2](z) & , z \subset u_2^{st} \\ \dots & \\ f_x^{st}[|U^{st}|](z) & , z \subset u_{|U^{st}|}^{st} \end{cases} \tag{4.5}$$

is known, then its aggregate in the source units and target units follows:

$$\begin{aligned} a_i^s &= \sum_{\forall u_k^{st} \in U^{st}, u_k^{st} \subseteq u_i^s} a_k^{st} \\ &= \sum_{\forall u_k^{st} \in U^{st}, u_k^{st} \subseteq u_i^s} \int_{z \subset u_k^{st}} f_x^{st}[k](z) dz, \end{aligned} \tag{4.6}$$

and similarly

$$a_j^t = \sum_{\forall u_k^{st} \in U^{st}, u_k^{st} \subseteq u_j^t} a_k^{st}$$
$$= \sum_{\forall u_k^{st} \in U^{st}, u_k^{st} \subseteq u_j^t} \int_{jz \subset u_k^{st}} f_x^{st}[k](z)dz. \quad (4.7)$$

Alternatively speaking, the aggregate in each source/target unit is equivalent to the sum of aggregates of all intersection units within it.

Two-step Approximation. We use a two-step solution to solve the aggregate interpolation problem for objective attribute α_o. In our solution, we first compute the approximate a_o^{st} (a_o^{st} is the aggregate vector for the intersection units). We then aggregate these approximate intersection unit aggregates to determine the approximate target unit aggregates. The two steps in our solution are described below:

1. **Disaggregation**: Split the aggregates in each source unit to its intersection units. Mathematically speaking,

$$\hat{a}_o^{st}[k] = \mathcal{B}(a_o^s[i], ...), \text{ s.t. } u_k^{st} \subseteq u_i^s, \quad (4.8)$$

   where the disaggregation function $\mathcal{B}(a_o^s[i], ...)$ computes the approximated $\hat{a}_o^{st}[k]$ of $a_o^{st}[k]$. Note that ... denotes the ancillary data that contribute to the approximation. Some of the most commonly used ancillary data are shape files of u_i^s and u_k^{st}, etc. More advanced approximation function may use external ancillary data. For instance, the distribution of a reference attribute that is positively related to the distribution of α_o.

2. **Re-aggregation**: Aggregate the approximated intersection unit aggregates for

the target unit they reside in, or equivalently

$$\hat{a}_o^t[j] = \sum_{\forall u_k^{st} \in U^{st}, u_k^{st} \subseteq u_j^t} \hat{a}_o^{st}[k]. \tag{4.9}$$

General Solution Properties. Regardless of the types of ancillary data available, some constraints are widely adopted in the existing two-step approximation solutions. We name two of them here.

One of these constraints is the *volume preserving* property [97, 62]. This property ensures that every source aggregate is preserved by the total of approximated aggregates in its intersection units, or

$$a_o^s[i] = \sum_{\forall u_k^{st} \in U^{st}, u_k^{st} \subseteq u_i^s} \hat{a}_o^{st}[k]. \tag{4.10}$$

The property is improving the estimation in that greater fidelity is given to the approximation in the intersection units, which propogates to a more accurate estimation in target units. It has been shown experimentally that methods following the volume preserving property make comparatively better predictions [62].

Homogeneity is also often used to compensate for the absence of information. Mathematically, for some attribute α_x, its probability density function in a given unit is constant. In other words, its aggregate on any sub-unit of the given unit is proportional to the area of the sub-unit. However, the assumption of homogeneity is rarely met in the real world [108].

4.3.2 `GeoAlign` Assumptions

We often have access to multiple reference attributes, no one of which perfectly matches the objective attribute we wish to estimate. It would appear advantageous for us to use all of them instead of using a single reference attribute as the current ex-

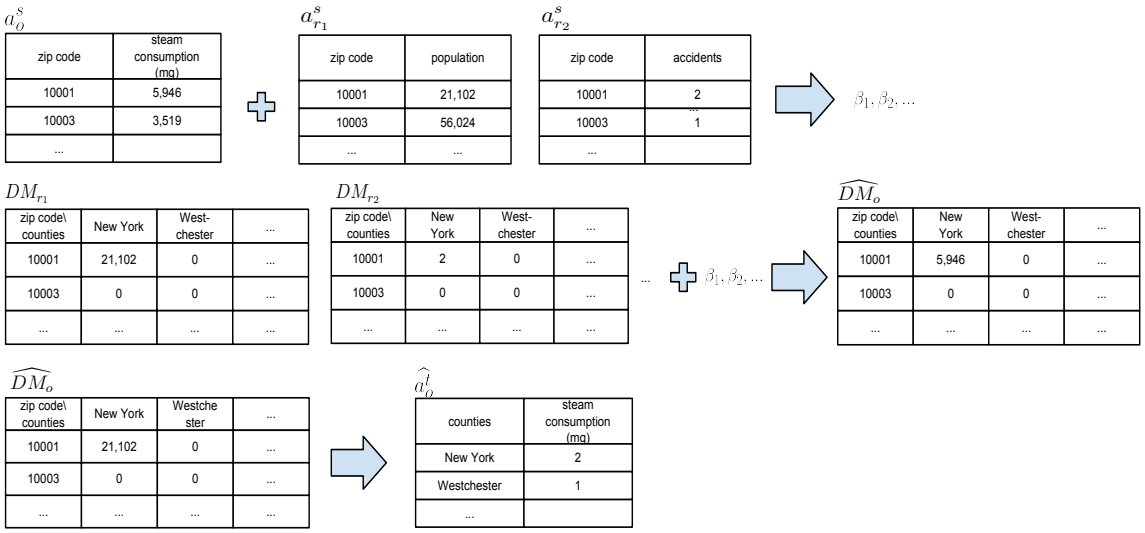


Figure 4.4: `GeoAlign` interpolation for the objective steam consumption data in Figure 4.1 from zip codes to counties using two reference attributes: population and accidents, in three steps: weight learning, disaggregation and re-aggreagtion.

tensive approaches described above. To this end, we propose `GeoAlign`, an aggregate interpolation algorithm that realigns aggregated data by learning from a combination of reference attributes to best predict the actual aggregates of the objective attribute in target units. `GeoAlign` leverages the advantages of extensive approaches and is, at the same time, robust to various objective attributes.

An intuitive idea could be to model the objective attribute aggregates as a function of multiple reference attributes aggregates in source units, evaluate coefficients with estimation methods and substitute reference attributes in target units for prediction. However, this is not applicable for the aggregate interpolation algorithm since training samples (objective attribute aggregates in source units) and test samples (objective attribute aggregates in target units) are not randomly drawn from the same population and the test samples are constrained by the training samples.

To address the linkage between two sets of samples and to account for the scale variations of reference attributes, in `GeoAlign`, the realignment of the objective attribute is related to that of the reference attributes through a statistical model for re-aggregation. In order to make the problem tractable, we assume that different

attributes are independent across source units, and that every attribute is correlated in its distribution between source and target units. We will loose the independence assumption of references later as shown in experiments in §4.4.4.2.

4.3.3 Disaggregation Matrix

Since we study the partition of aggregates in intersection units, in the disaggregation step, $\mathcal{B}(a_o^s[i], ...)$ can be reformulated as

$$\hat{a}_o^{st}[k] = \frac{\omega_o^{st}[k]}{\omega_o^s[i]} a_o^s[i]$$
$$\text{subject to} \sum_{\forall u_k^{st} \in U^{st}, u_k^{st} \subseteq u_i^s} \omega_o^{st}[k] = \omega_o^s[i], \tag{4.11}$$

where $\frac{\omega_o^{st}[k]}{\omega_o^s[i]}$ is the share of aggregate in the k-th intersection unit ($\omega_o^{st}[k]$) over that in the i-th source unit ($\omega_o^s[i]$) it resides in. Intuitively, the re-aggregation step sums up the weighted share of all intersection units in all source units that overlap with the target unit. Alternatively speaking,

$$\hat{a}_o^t[j] = \sum_{\forall u_i^s, u_i^s \cap u_j^t \neq \emptyset} \frac{\sum_{\forall u_k^{st} \subseteq u_i^s \cap u_j^t} \omega_o^{st}[k]}{\omega_o^s[i]} a_o^s[i]. \tag{4.12}$$

Rather than approximating a_o^{st} in the disaggregation step, we can instead infer $\frac{\omega_o^{st}[k]}{\omega_o^s[i]}$, $\omega_o^{st}[k]$ or $\sum_{\forall u_k^{st} \subseteq u_i^s \cap u_j^t} \omega_o^{st}[k]$. This choice often depends on the type of ancillary data available. The most widely used ancillary data is the true disaggregation of a reference attribute between source and target units. For instance, for the population reference mentioned in the introduction, the population aggregates in intersection units of counties and zip codes. We denote the *disaggregation matrix* of some attribute α_x between two unit systems γ^{y_1} and γ^{y_2} as $DM_x^{y_1,y_2}$, where $DM_x^{y_1,y_2}[i,j]$ is

Table 4.1: Notations in §4.2 and 4.3

Notation	Description
Ω	an n-dimensional universe of interest
γ^y	a unit system in Ω, for example γ^s at source level
$U^y = \{u_1^y, u_2^y, ...\}$	the set of units in γ^y
α_o	the objective attribute
$A_r = \{\alpha_{r_1}, \alpha_{r_2}, ...\}$	the set of reference attributes
$\alpha_x \in \alpha_o \cup A_r$	an attribute of interest
$a_x^y = [a_x^y[1], a_x^y[2], ...a_x^y[\lvert U^y \rvert]]$	the aggregate vector of α_x in units of U^y
f_x^y	the probability density function of α_x for γ^y
$\mathcal{B}(a_o^s[i], ...)$	the disaggregation function
ω_x^y	the weighted share vector of α_x for γ^y
$a_x'^y$	the normalized a_x^y
$DM_x^{y_1,y_2}$	the dimension matrix of α_x, where $DM_x^{y_1,y_2}[i,j]$ is the aggregate of α_x in the intersection of $u_i^{y_1}$ and $u_j^{y_2}$
$\beta = [\beta_1, \beta_2, ...\beta_{\lvert A_r \rvert}]$	weights computed from Equation (4.15)

its aggregate in the intersection area of $u_i^{y_1}$ and $u_j^{y_2}$. For γ^s and γ^t,

$$DM_x^{s,t}[i,j] = \sum_{\forall u_k^{st} \subseteq u_i^s \cap u_j^t} a_x^{st}[k] \tag{4.13}$$

The disaggregation matrix of the reference attribute between source and target units is often wrapped up in a crosswalk relationship file. When the disaggregation matrix of only one reference attribute α_r is available, we can substitute $\sum_{\forall u_k^{st} \subseteq u_i^s \cap u_j^t} \omega_o^{st}[k]$ for $DM_r^{s,t}$ to complete the approximation of the objective attribute in target units. This type of method is named as the *dasymetric method* [64, 63, 107]. A special case of it is the areal weighting method [61], using the disaggregation matrix of area as the reference. Dasymetric methods are widely employed in socioeconomic data realignment by general users [26].

Since we only consider the disaggregation matrix between source and target units,

from now on, we use DM_x for $DM_x^{s,t}$.

4.3.4 `GeoAlign` Algorithm

In the real world, the disaggregation matrix of more than one references attributes is often available. `GeoAlign` is a volume-preseving method that leverages the distribution similarity of the objective attribute with reference attributes at the source level and predicts the dimension matrix of the objective as a weighted combination of the dimension matrices of the references. We will first extend some of the notations in Section 4.2, and then describe our proposed algorithm in detail.

Notation. Let $A_r = \{\alpha_{r_1}, \alpha_{r_2}, ...\}$ be the set of reference attributes available. The aggregate vectors of these reference attributes in source units are represented as $a^s_{r1}, a^s_{r2}, \ldots, a^s_{r_{|A_r|}}$, where $a^s_{r_k} = [a^s_{r_k}[1], a^s_{r_k}[2], ..., a^s_{r_k}[|U^s|]]$ for the kth reference attribute. Similarly, the aggregate vectors of these reference attributes in target units are represented as $a^t_{r_1}, a^t_{r_2}, ..., a^t_{r_{|A_r|}}$, where $a^t_{r_k} = [a^t_{r_k}[1], a^t_{r_k}[2], ..., a^t_{r_k}[|U^t|]]$.

We assume that the ancillary data available is the disaggregation matrix of all the reference attributes. We denote the disaggregation matrix of the kth reference attribute as DM_{r_k}.

To avoid variation in scale, we normalize the objective attribute and the references at the source level, adjusting their values measured on different scales to a notionally common scale. This is reasonable in two ways. First, `GeoAlign` is dependent on the distribution similarity between the objective attribute and the references across source units rather than their actual value similarity. Second, `GeoAlign` jointly considers the similarity of the objective with multiple references. The magnitude of the references should not be a contributing factor.

The normalized $a^s_{r_k}$ is denoted by $a'^s_{r_k}$ for $k = 1, 2, ..., |A_r|$, and is computed as $a'^s_{r_k} = a^s_{r_k} / \max_{i, i \leq |U^s|} a^s_{r_k}[i], a^s_{r_k}[i] \geq 0$.

The aggregate vector of the objective attribute in source units a^s_o is also normalized

similarly, and is denoted as a'^{s}_{o}.

`GeoAlign` Steps In the disaggregation step, `GeoAlign` computes $\widehat{DM_o}$, which is the estimated weighted dimension matrix of the objective attribute. Our intention is to best predict $\widehat{DM_o}$, and at the same time, preserve its volume preserving property. We propose

$$\widehat{DM_o}[i,j] = \begin{cases} \frac{\sum_{k=1}^{|A_r|} \beta_k \times DM_{r_k}[i,j]}{\sum_{k=1}^{|A_r|} \beta_k \times a^s_{r_k}[i]} \cdot a^s_o[i], & \sum_{k=1}^{|A_r|} a^s_{r_k}[i] \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{4.14}$$

where $\beta = [\beta_{\mathbf{1}}, \beta_{\mathbf{2}}, ..., \beta_{|\mathbf{A_r}|}]$ is the learned weight vector and $\sum_{i=1}^{|A_r|} \beta_i = 1$.

Our preliminary experiments lay the ground work of our assumption such that the higher the similarity between two attributes at the source level, the more likely their distribution in the intersection level are similar. We can thus express the objective attribute as linearly associated with the reference attributes for both aggregate vector in source units and disaggregation matrix. The weights are obtained by solving a constrained linear least squares programming problem with objective function:

$$\begin{gathered} \min_{\beta} \frac{1}{2} ||\mathbf{A}\beta - \mathbf{b}||^{\mathbf{2}} \\ \text{subject to } \sum\nolimits_{k=1}^{|A_r|} \beta_k = 1 \\ \text{where } \beta_k \geq 0, \text{ for } k = 1, 2, ..., |A_r| \end{gathered} \tag{4.15}$$

where $\mathbf{A}$ is the column-wise concatenation of $a'^{s}_{r_k}$ for $k = 1, 2, ...,$
$|A_r|$ and $\mathbf{b}$ is a'^{s}_{o}. Instead of computing $\widehat{DM_o}$ by directly applying the weights to DM_{r_k}s, we adapt it to the scale of reference attributes and insert back the weights to Eq. (4.14) to get an adjusted $\widehat{DM_o}$.

The approximated disaggregation matrix of the objective attribute satisfies the

Algorithm 2: `GeoAlign`

Input: aggregate vectors of reference attributes in source units $a^s_{r_1}, a^s_{r_2}, ..., a^s_{r_{|A_r|}}$; corresponding disaggregation matrices $DM_{r_1}, DM_{r_2}, ..., DM_{r_{|A_r|}}$; and the aggregate vector of the objective attribute in source units a^s_o.

Output: estimated aggregates of the objective attribute in target units $\hat{a}^t_o$

1 **Step 1. Weight Learning:** Compute weights, β, by solving the least squares problem in Equation (4.15)

2 **Step 2. Disaggregation:** Compute the estimated weighted disaggregation matrix of the objective attribute, $\widehat{DM_o}$, using Equation (4.14)

3 **Step 3. Re-aggregation:** Re-aggregate to estimate the aggregates of the objective attribute in target units, $\hat{a}^t_o$, using Equation (4.17)

volume preserving property such that

$$\widehat{DM_o}[i,j] \geq 0 \qquad \text{and} \qquad \sum\nolimits_{j=1}^{|U^t|} \widehat{DM_o}[i,j] \approx a^s_o[i]. \tag{4.16}$$

The estimated aggregates of the objective attribute in target units are computed in the reaggregation step as

$$\widehat{a}^t_o[j] = \sum\nolimits_{i=1}^{|U^s|} \widehat{DM_o}[i,j] \tag{4.17}$$

Following the pseudocode in Algorithm 2, we further illustrate the algorithm by the motivating example in Figure 4.1, with the steps depicted in Figure 4.4. Assume that `GeoAlign` is crosswalking the steam consumption objective from zip codes to counties. Moreover, assume that the aggregate vectors, $a^s_{r_1}$ and $a^s_{r_2}$, and the disaggregation matrices, DM_{r_1} and DM_{r_2}, for two reference attributes, population and accidents, are readily available. Maximizing the distribution similarity across units between the normalized objective, a'^s_o, and the normalized references, $a'^s_{r_1}$ and $a'^s_{r_2}$, the objective attribute is first optimized as a weighted combination of the references at the source level (zip code level). The weights, β_1 and β_2, are then reassigned to the disaggregation matrices of the references DM_{r_1} and DM_{r_2}, and adjusted to predict

an approximated disaggregation of the objective $\widehat{DM_o}$. The approximated disaggregation matrix is eventually re-aggregated to derive an approximate of the objective at the target county level ($\hat{a}_o^t$).

It can be easily shown that `GeoAlign` is applicable to any dimension since the algorithm involves no dimension dependent information or computation. Rather, the only information needed is the true partition of reference attributes in source and target intersection units regardless of dimension or dimension-related information, such as spatial correlation for geospatial data. Alternatively, if true partition of references in finer granularity is available, the data can be aggregated to the level of source and target intersection as a reference attribute.

4.4 Experimental Evaluation

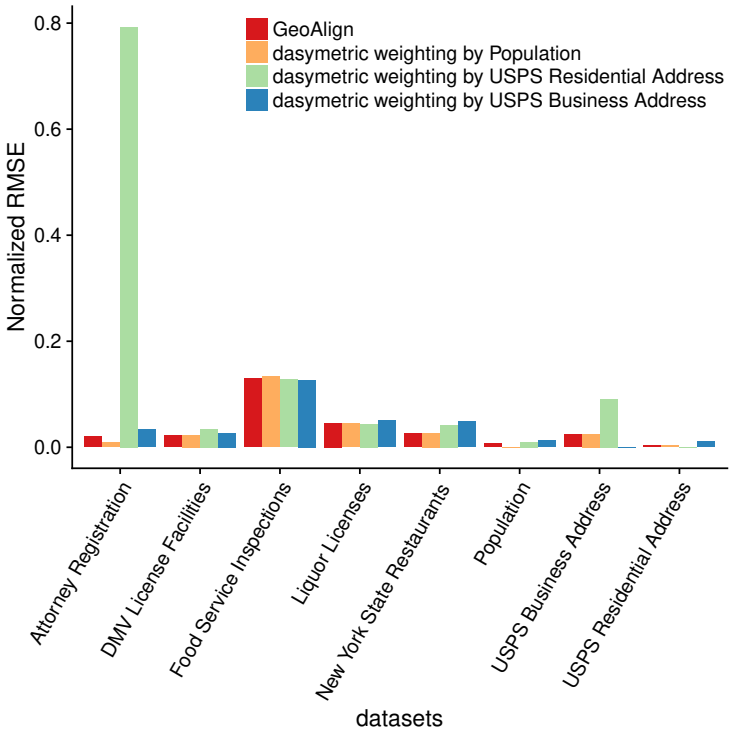
We evaluated the feasibility of the `GeoAlign` algorithm from two crucial aspects: whether the algorithm can correctly complete the realignment task (effectiveness), and whether the runtime of the algorithm is fast enough (efficiency). Additionally, we consider runtime scalability when larger datasets are involved and the robustness of the algorithm when low quality or limited reference attributes present.

We compare the performance of `GeoAlign` with that of areal weighting method [62] and dasymetric method [107, 64, 63] that utilizes three reference attributes separately.

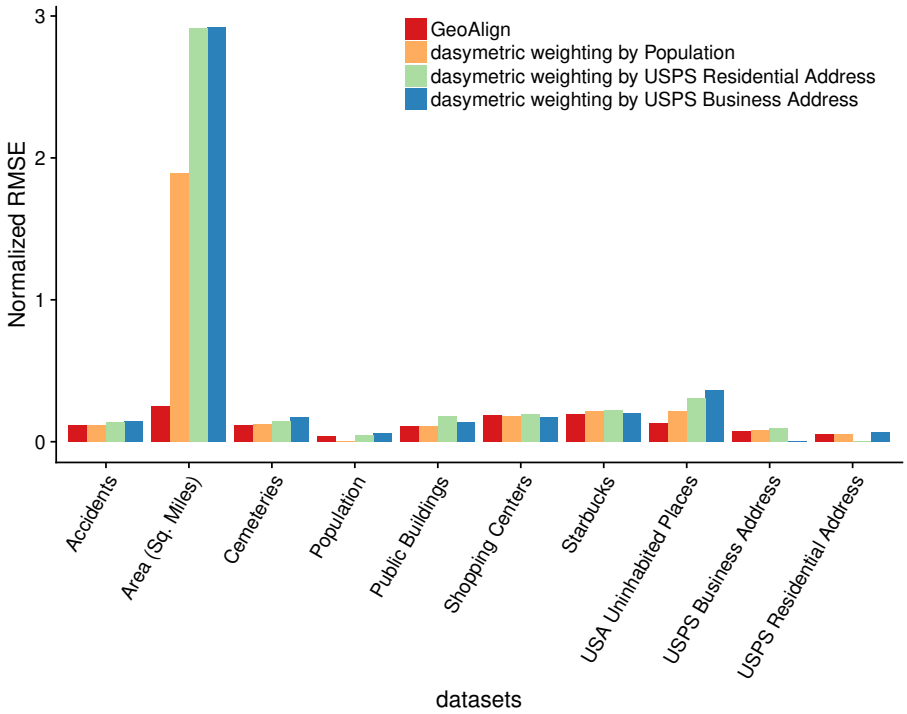
4.4.1 Experimental Setup

We developed the `GeoAlign` algorithm in Python. All experiments were performed on a 2.3 GHz Intel Core i7 with 8 GB memory and a 7200 rpm SATA disk.

We evaluated `GeoAlign` for 2-D areal interpolation. We used county and zip code as the two geographic types of interest, and focused on data from two different universes, New York State and the United States. Most of the New York State data ere collected from `data.ny.gov`, populated in tabular form. Three population level



(a) New York State



(b) the United States

Figure 4.5: `GeoAlign` prediction performance (NRMSE) compared with dasymetric methods. Since a better prediction yields a lower NRMSE, `GeoAlign` is making comparable or better predictions than the dasymetric methods for tests in New York State and the Unite States.

demographic datasets have been used as reference data for the single crosswalk algorithm, namely the population data from United States Census Bureau [10], the aggregated USPS residential address data and the aggregated USPS business address data [81]. In addition, we also selected five large individual level datasets (The New York State Restaurants dataset is generated by selecting unique restaurants in the Food Service Inspections dataset) with geographic information and aggregated their number of records for the intersection area of the two geographic types to form their disaggregation matrices [19, 20, 21, 22]. Thus we obtained a total of eight reference datasets with accurate distributions by zip code and by county, and their disaggregation matrices from zip codes to counties.

Besides the three population level Census data, which cover the entire nation including New York State, other data for the United States were collected from Esri, where the Maps and Data group provides publicly available geocoded GIS data. Six individual level GIS data [32, 33, 34, 35, 31, 36] were aggregated based on their geospatial information for zip code and county levels and their intersections using ArcGIS Pro [53]. We also computed the area of units at these three levels, which is later used as the reference attribute by the areal weighting method, yielding 10 datasets in total for the universe of the United States.

There are more datasets with attributes for which the aggregate vectors are available for both zip code and county for New York State or for the United States. However, we did not use them as reference attributes due to two reasons. First, it was not clear whether these aggregates are accurate or approximate. In §4.4.4.1, we further discuss the impact of the reference approximates on the prediction. The other reason is that several attributes do not have their disaggregation matrices publicly accessible and such attributes cannot be used as reference attributes. In case of limited reference attributes, we show in §4.4.4.2 that `GeoAlign` makes reasonable predictions even when the references are poorly selected.

Since the number of datasets with accurate disaggregation matrix is limited, we adopted the cross-validation evaluation method that deals with the problem well. We conducted two series of experiments, one for each universe. More specifically, for each universe, we picked one of the datasets as the test dataset, in turn, and used the remaining datasets to develop crosswalks in `GeoAlign` whose combined weighted performance is then evaluated for the test dataset. The performance of `GeoAlign` is compared with the base-line single reference crosswalk method that redistributes by a disaggregation matrix of some known attribute. More specifically, `GeoAlign` is compared with the areal weighting method and the dasymetric algorithm referencing the three population level datasets. Note that when one of the population reference datasets or the area dataset is used as the test dataset, the performance of both methods referencing this dataset is not evaluated.

4.4.2 `GeoAlign` Effectiveness

To evaluate the effectiveness of `GeoAlign`, we adopted root mean square error (RMSE) as the evaluation criterion that computes the deviation of estimated aggregates from true aggregates of the attribute in counties. To ease the comparison across datasets of heterogeneous scales, in Figure 4.5, we show the RMSE normalized by the mean of the measured data (NRMSE).

The NRMSE of `GeoAlign` is compared with that of the dasymetric method using three population level datasets and the areal weighting methods for both New York States (Figure 4.5(a)) and the United States (Figure 4.5(b)), using eight and ten datasets respectively. The performance of areal weighting method is not shown in the figure since it makes poor predictions for all test datasets: over 15 times of the NRMSE of `GeoAlign` for New York State experiments and over 50 times of the NRMSE of `GeoAlign` for the United States experiments.

The NRMSE of `GeoAlign` is less then 0.13 for New York State experiments and

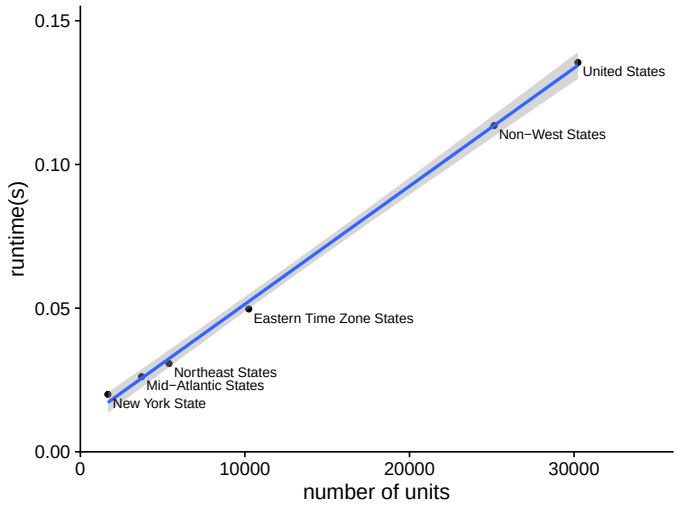
less than 0.26 for the United States experiments. Though three dasymetric methods have comparable error on most datasets, for these datasets, `GeoAlign` is making equal or better predictions. It should also be noted that no one of these three methods is predicting uniformly well for all datasets as `GeoAlign` does, in whichever universe. For instance, the dasymetric method referencing the population data presents much higher error than the other methods when predicting for attorney registration and USPS Business Address counts for counties in New York State; all three dasymetric methods fail in accuracy for both area and USA uninhabited places datasets in the United States.

Except the USPS business address dataset, the rest three are individual level datasets with limited number of observational units that are sparsely distributed in the universe. Also, they do not align well with demographic attributes as those in the areal weighting and dasymetric methods. We observe that `GeoAlign` accounts for sparsity and heterogeneous distributions with flexibility.

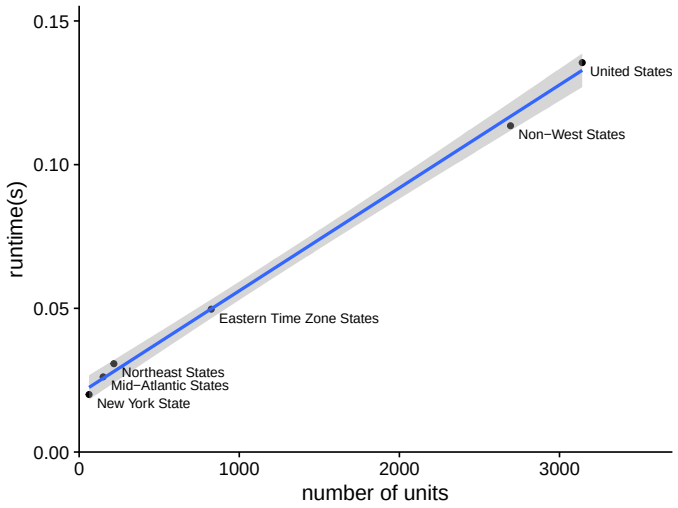
4.4.3 `GeoAlign` Efficiency and Scalability

We evaluated the efficiency of `GeoAlign` in terms of algorithm runtime. Apart from the horizontal efficiency comparison across cross-validated tests for a given universe, we also considered the scalability of `GeoAlign` runtime. This is realized by comparing `GeoAlign` efficiency vertically across the universes of different scales.

In addition to New York State and the United States, new universes were selected as a set of states whose boundaries are congruent with any other state in the universe. The selection is a greedy process that ensures the states in a universe are tightly connected from a geospatial perspective. These four new universes include Mid-Atlantic division and Northeast region defined by Census Bureau, states contained entirely in the Eastern Time Zone and all states excluding the ones in the Census West Region (non-West). They form a spatial coverage hierarchy preventing the inter-state



(a) Zip Code Level



(b) County Level

Figure 4.6: `GeoAlign` runtime scales linearly with respect to the number of units in source level and target level

influence of randomly selected universes.

Moreover, for factor control purpose, instead of collecting more datasets for new universes, for each universe, we subset the ten datasets covering the United States, keeping the entries collected from units within the universe as inputs.

To avoid random error, we averaged the runtime across ten trials for the cross-validated experiments in each universe.

Experimental results show that `GeoAlign` runtime is stable across experiments for the same universe. This is consistent with our claim that the complexity of `GeoAlign` is not related to the magnitude of the count data. The majority of the runtime, over 90%, is spent on computing the disaggregation matrix after the weights are estimated. Note that the aggregate vectors of the objective attribute in source geographic units has the same size for all the different datasets (the size is $|U^s|$). Similarly the aggregate vectors of the reference attributes in source geographic units are all of the same size (all of size $|U^s|$), the aggregate vectors of the reference attributes in target geographic units are all of the same size (all of size $|U^t|$). Further, all the disaggregation matrices are all of the same size as well. The reason for the minor difference in `GeoAlign` runtime for different datasets is because of the difference in the number of non-zero entries in the disaggregation matrix, which is stored as sparse matrix, of reference attributes. For the disaggregation matrix, sparse datasets, such as cemeteries, have less non-zero entries, while dense datasets, such as population, have more non-zero entries. Matrix operations involving sparse matrices are influenced by this factor in SciPy package.

As for cross-universe comparison, we ploted `GeoAlign` runtime versus the number of zip codes (source units) and the number of counties (target units) in Figure 4.6. These two plots show that `GeoAlign` is fast: it runs for less than 0.15 second even for crosswalk between 30238 zip codes and 3142 counties in the United States universe. They also prove the linear relationship between `GeoAlign` runtime with the number of units in source and target levels since the dominating disaggregation matrix construction operation is linearly related to these two factors.

4.4.4 `GeoAlign` Robustness

As mentioned earlier in §4.4.1, during the reference attribute collection process, we encountered two difficulties: the undetermined accuracy of reference attributes at

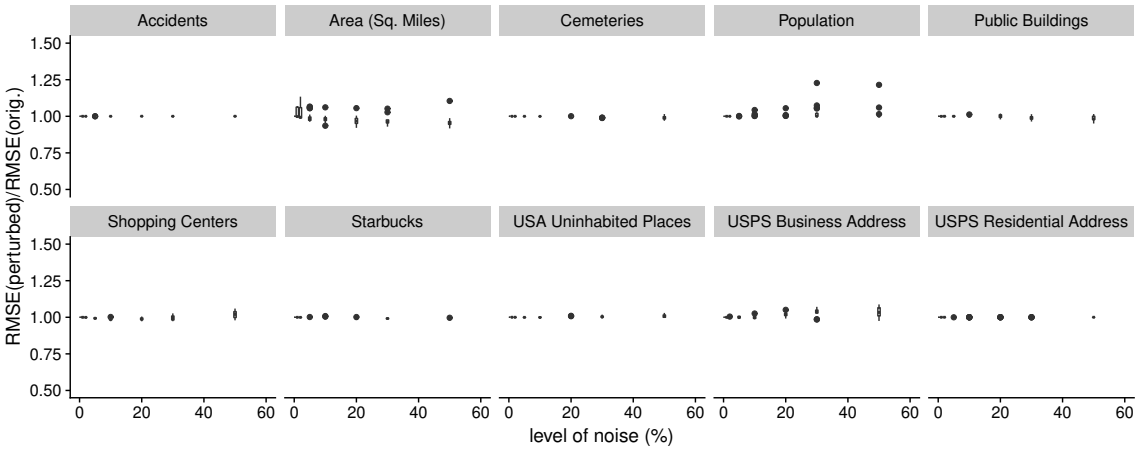


Figure 4.7: When noises are introduced in references, the prediction deviation is evaluated as the ratio of the RMSE using the perturbed references to the RMSE using the original references. The closer the ratio is to 1, the more invariant `GeoAlign` is to reference noises. For up to 50% level of noise, most experiments have the prediction deviation around 1 indicating the robustness of `GeoAlign` to noisy references.

the source level, and the limited availability of datasets with disaggregation matrix. We conducted two series of experiments evaluating the robustness of `GeoAlign` with respect to these two problems respectively.

4.4.4.1 Inaccurate Reference Attributes

Public aggregated data can be derived in multiple ways. They can be aggregates of individual level data, approximates derived from some crosswalk algorithm, etc. Without the raw data and the transformation information available, the accuracy of these aggregates are unknown. It is thus hard to determine whether the data can be used as references.

To quantitatively evaluate the influence of the accuracy of reference attributes on `GeoAlign`, we artificially introduced "noise" to the reference attributes. We define *noise* as the deviation from the actual value. Noise is measured by "levels" such that a x% level of noise for y is $\pm x * y/100$. The noise-polluted y is thus $(1 + x/100) * y$. For each of the ten cross-validated experiments in United States, we synthetically generated noisy reference attributes at the source level with 1%, 2%, 5%, 10%, 20%,

30% and 50% degrees of noises for all references. Each experiment is replicated 20 times to account for random error due to randomness of positive or negative noises. We quantify the prediction deviation as the ratio of the RMSE using perturbed noisy reference attributes to the RMSE using the original reference attributes. The closer the prediction deviation is to 1, the smaller the impact of the noises is. `GeoAlign` is making better prediction with the perturbed reference attributes if the ratio is higher than 1; whereas a less than 1 ratio indicates worse prediction with the perturbed reference attributes.

In Figure 4.7, we show the box plot of the prediction deviation with respect to different levels of noise. The prediction performance of `GeoAlign` is stable across experiments. For each experiment, `GeoAlign` is making robust predictions for all levels of noise. Though for the area and population datasets, higher levels of noise resulted in higher prediction error, the mean prediction deviation for these levels is still small (less than 1.1).

4.4.4.2 Limited Reference Attributes

In general, we cannot predict how many reference attributes will be available. We may have very few, or we may have very many. In the process of reference attribute selection, there are two questions to consider: whether `GeoAlign` can make reasonable predictions with limited number of reference attributes, and how to select the reference attributes when more than one is available.

To answer these two questions, we chose multiple subsets of reference attributes among all reference attributes and repeated the cross-validated experiments for datasets in the United States. The subset of reference attributes were chosen based on their relationship with the target attribute of each test dataset. We adopted the leave-n-out metric such that $n = 1, 2$ for reference attributes with the highest (or lowest) correlation with the target attribute at the source level. The NRMSEs of these four

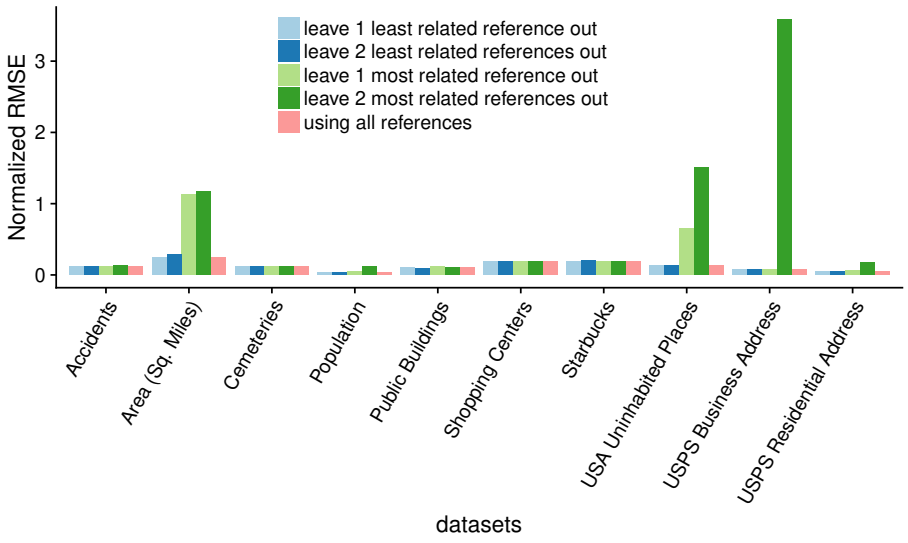


Figure 4.8: `GeoAlign` is robust to the choice of reference attributes. Though extra reference attributes do not create any loss, reference attributes with higher correlation with the objective are preferred.

series of experiments are compared with experiments using all reference attributes in Figure 4.8.

For 7 out of 10 tests, `GeoAlign` is making robust predictions regardless of the subset of reference attributes used. As for the series of experiments leaving 1 or 2 least target-attribute-related reference(s) out, the performance of `GeoAlign` is almost identical to using all reference attributes. This is in accordance with `GeoAlign`'s ability of assigning little weights to reference attributes loosely related to the target attribute.

Leaving out the most target-related attributes out can have an impact on accuracy. This does impact three of our attributes: area, USA uninhabited places and USPS business address datasets. None of the references are closely related to the area and the USA uninhabited places datasets at the source level (correlations less than 0.25). Apart from the two references left out, the rest of the references have even lower correlation with the target attribute (less than 0.2 and 0.05 respectively). According to the assumption basis of `GeoAlign`, the distribution of the target attributes

is thus poorly related to the distribution of these attributes, leading to increased prediction error. We also found that leaving out the reference most related to the target attribute has almost no impact on the prediction for the USPS business address dataset; while leaving out top two such references dramatically worsens the situation. Further analysis reveals that these two references are highly correlated with each other at the source level ($\approx 96\%$), the weight assigned to the reference most related to the target attribute is reassigned to the other when the former is left out. This verifies that similar attributes at the source level are also similarly distributed in the intersection units, as the predicted disaggregation matrix of the target attribute is almost the same regardless of using the reference most related to the target attribute or not.

These experiments give us more insight into `GeoAlign` reference attribute selection. `GeoAlign` prefers reference attributes highly related to the target attribute at the source level. For reference attributes poorly related to the target variable, it is able to weigh their contributions accordingly. The reference attributes are not necessarily independent of each other and the reference attributes are not necessarily accurate at the source level. From the user's perspective, `GeoAlign` is able to make reasonable predictions by simply given all available reference attributes.

4.5 Conclusions and Future Work

In this chapter, we formally define the problem of aggregate interpolation in multi-dimensional space and propose `GeoAlign`, an adaptive multi-reference algorithm that realigns aggregates better than state-of-the-art approaches for real socioeconomic datasets. Unlike existing areal interpolation algorithms, `GeoAlign` requires no knowledge of spatial properties or dasymetric maps of source and target units and is thus generally applicable for plain aggregate tables. Our experiments show that `GeoAlign` is making better predictions in a reasonably short time. Its runtime scales linearly

with the number of units in source and target levels, and is robust to noisy references even when limited references are available.

There are several limitations of `GeoAlign` that we do not emphasize. First, the evaluation of `GeoAlign` is performed on zip code and county levels due to limited access to reference attributes. The performance of `GeoAlign` on other source and target level pairs are not fully demonstrated. For an interpolation from a source level whose units each covers a large number of units in the target level, the algorithm may perform poorly failing the distribution assumption. The approximated distribution of the target attribute leveraging reference attributes at the source level could not capture the delicate distributional characteristics of it at the target level. For instance, if two attributes distribute similarly at the state level, their distributions may vary a lot at county level.

CHAPTER V

SDTA: Standardizing Statistical Data Transformation by a Structured Algebra

In this chapter, we focus on improving the quality of metadata for better dataset search. Here we consider for socio-economic data the provenance information in terms of statistical data transformation in data documentation. In addition to proposing two structured representations for statistical data transformation, we have also implemented an online system to automatically update existing metadata with the transformation in the proposed representation.

5.1 Introduction

Statistical data transformation is the process of converting statistical data from one format or structure to another format or structure. It involves not only structural transformation, but also statistically intensive tasks such as data checking, outlier adjustment, data summary for visualization. It is an inevitable step to prepare the data for statistical analytic by data scientists, researchers, survey companies, government or even general public to reveal the unknown value of raw data in crucial fields like social science, marketing, health, etc. Nowadays, more data portals are asking for publishing data with its metadata, of which the transformation information is a vi-

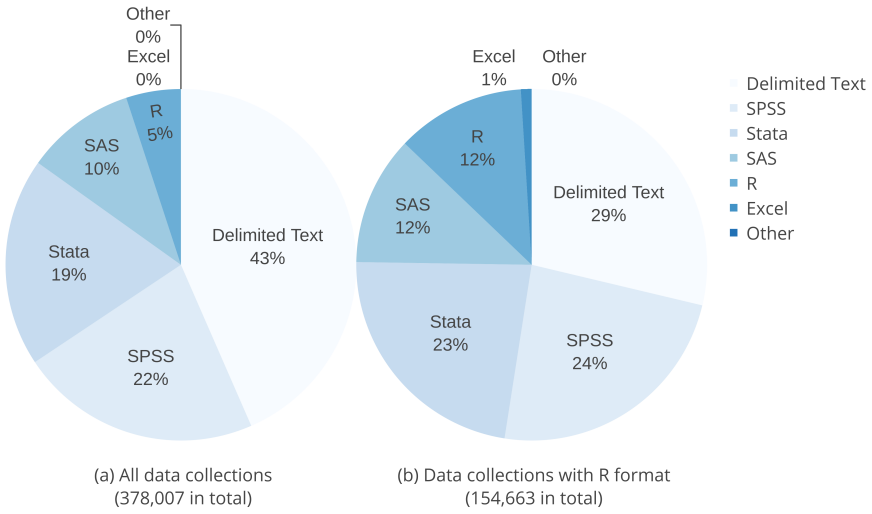


Figure 5.1: ICPSR Data Downloads by Format (September 4, 2015 to March 4. 2016). Tab- or comma-delimited ASCII files may be analyzed in other statistical packages or other types of software, like relational databases.

tal part for understanding problems related to data profiling, data provenance, data integration, data reproduciblility, etc.

These data transformation operations are generally represented by domain-specific transformation language (DSTLs) associated with transformation tools, the preference of which varies by fields. For clinical trials data, SAS is the dominant choice. While for social science data, users have a more balanced choice. Fig. 5.1 is such an example showing the number of data downloads by format from Inter-university Consortium for Political and Social Research (ICPSR). In the database community, relational query language is largely accepted over the others [56, 58, 88].

However, no universal agreement on the representation of statistical data transformation is available. Each DSTL has its own data model, semantic and syntactic representations and respective scope of transformation operations supported. In the classical relational database theory, a relation is a set of tuples, the ordering of which is not defined. Statistical data tables, on the other hand, have both ordered rows and ordered columns, which are essential for pattern tracking along the dimensions. Piv-

TableUnpivot

Age	Gender	Total
<20	F	30
<20	M	60
>20	F	50
>20	M	20

pivot

TablePivot

Age	GenderF	GenderM
<20	30	60
>20	50	20

Stata:

```
reshape wide Total, i(Age), j(Gender)
```

SQL Server using pivot (dynamic creation):

```
DECLARE @Columns as VARCHAR(MAX)

SELECT @Columns =
COALESCE(@Columns + ', ','') + QUOTENAME(Gender)
FROM
   (SELECT DISTINCT Gender
    FROM   TableUnpivot
   ) AS T
ORDER BY T.Gender
```

Generate pivoted column names based off data during runtime

```
DECLARE @SQL as VARCHAR(MAX)
SET @SQL = 'SELECT Age, ' + @Columns + '
FROM TableUnpivot
PIVOT
(
    SUM(Total)
    FOR Gender
    IN (' + @Columns + ')
) AS TablePivotinform
ORDER BY Age'

EXEC(@SQL)
```

R using reshape2 and data.table libraries:

```
dcast(data=TableUnpivot,
      formula=Age ~ paste0("Gender", Gender),
      value.var="Total")
```

Figure 5.2: An example of pivoting table using functionally equivalent commands in Stats, SQL Server and R

oting data, as another example, is a common action of transformation that transposes data from multiple rows into columns of a single row providing a better summary of data. It is widely used for data exploration and plotting.

Motivating example. *In Figure 5.2, we use three alternative DSTLs, namely Stata, SQL Server and R, to create the identical two-dimensional pivot data* `TablePivot` *for the number of people in four age and gender groups from* `TableUnpivot`*. Here we display the data in tabular format.*

- *The embedded* `reshape` *function in Stata could easily transforms the long table* `TableUnpivot` *into a wide one by specifying the* `wide` *keyword, the row variable* `i` *(*`Age`*), the column variable* `j` *(*`Gender`*) and the name of the column(*`Total`*) storing values spanning the pivot table.*

- *There was no corresponding command in the basic SQL. SQL Server [] introduced the* `pivot` *operation in 2005. Since the unique values of the column variable* `Gender` *could not be captured by the pre-defined* `pivot` *command,* `Columns` *variable is first declared to dynamically select these values at runtime before pivoting. Note that the new column variable values are updated as the concatenation of* `Gender` *with unique* `Gender` *values to align with the* `TablePivot` *illustration in the figure.*

- *Unlike high level languages designed specifically for data transformation, R is a more powerful low level general-purpose programming language with libraries developed for data transformations and statistical manipulations. This example leverages* `dcast` *function in* `reshape2` *library to create a pivot table by specifying the input data table (*`TableUnpivot`*), the cast formula (composed of* `Age` *and* `Gender`*) and the name of the column (*`Total`*) storing values spanning the pivot table.*

These disparities have raised the communication and manipulation barrier between languages. Given a complex transformation task, users may either compose a series of transformations by a chosen tool or reuse existing transformation scripts. However, it is not easy for general users to master a transformation tool since the representations are not intuitively understandable. The learning cost in terms of time and money is thus a non-negligible concern. The limited scopes of transformations covered by different tools complicates the problem even further. Users may need to use multiple tools in order to complete the task if it is not easily realizable by any one of them, or by the ones users are familiar with. On the other hand, code reuse seems to be a better option if there exists a solution for an identical or similar transformation task. The reuse of existing code requires the understanding of the language of the code, the ability of which the user may lack. If there exists a conversion mechanism between two languages to translate the code in the source language to a target language that the user is familiar with, the reuse of code could then realized.

Moreover, the implementation problem is not fully emphasized by existing tools. If a large dataset undergo a long transformation flow, the efficiency of transformation operations cannot be neglected. For database queries, effective optimization of relational query processing relying on relational algebra has been demonstrated by past experiences.

Is it possible to have a unified representation of statistical data transformation in a *simple* yet computationally efficient way covering the majority of transformations? Inspired by relational algebra and relational query language for databases, we tackle these problems by developing a simple algebra, called *Structured Data Transformation Algebra* (`SDTA`), for manipulating statistical data transformation in a generic data model; and a predominant declarative transformation language, called *Structured Data Transformation Language* (`SDTL`) [1], that works for the same data model. Both DSTLs are realizations of a generic data transformation model defined for a

generic data model providing a unified view of the statistical data transformation domain. Here we address two challenges: (i) how to define a generic data model that is compatible with both relational data model and statistical data model, and allows information prorogation along the transformation flow, and (ii) how to define the operators in the algebra that could benefit from relational algebra optimization techniques.

The intellectual contributions of the chapter are as follows:

- Based on the design considerations in §5.2, we build the fundamentals of statistical data transformation by defining a generic data model (§5.3) and transformation model (§5.4) for statistical data transformation.
- We define `SDTA` as an algebraic realization of the transformation model and further illustrate `SDTA` operators in §5.5.
- We define `SDTL` as a declarative statistical transformation language in §5.6.
- In §5.7, we showcase that `SDTA` (and `SDTL`) are useful in many scenarios, including data documentation and language translation. We have developed `C`^{2}`Metadata` (http://c2metadata.org) as a pipelined system for automatic transformation documentation as provenance-aware metadata for scientific data, available for four major statistical languages. Using `SDTA` and `SDTL` as bridges, we have also built a mapping between `SDTL` and the same statistical languages for efficient language translation.

We finalize the chapter by a conclusion section in §5.8.

5.2 Design Considerations

The heart of relational data model is the algebraic operations over collections of tuples, also referred to as *relation*s (to distinguish from *statistical table*s), for efficient

bulk access and implementation. Relying on declarative queries built on top of relational algebra, the data model is receiving more popularity by putting user focus on query needs rather on execution details.

If one is to perform bulk manipulation on statistical data, it's intuitive to create a relational-like algebra - considering table layout of relations analogous to that of statistical data, and a majority of operations on relations commonly applied to statistical data. Their differences between the underlying data models and operations over the data models, however, prevents the direct transfer of relational algebra to an algebra for statistical data.

One key issue lies in the mathematical foundation of the data model from the two domains. Both database relations and statistical tables are commonly represented in tabular format as collections of rows and columns. Relational theory regards relations as sets of tuples (often displayed as rows) without duplication or ordering. Each tuple is uniquely identified by primary keys. Statistical data, on the contrary, have looser constraints on duplication as in multi-sets. A multi-set extended relational algebra as in [] is an appropriate adaption to ameliorate the difficulty by retaining duplicates. This approach does not emphasize the orderings of rows and columns in statistical data that are sometimes enforced properties for statistical comparison and pattern finding. Some aggregations in groups, for instance, rely on the order of tuples in group. Using `TableUnpivot` as an example, one may try to find the first entry whose `Total` is greater than 30 for each `Age` group. In some other cases, users specify where exactly to insert or update a row. The ordering information thus needs to propagate through a series of data transformations. Relational operations should be extended to preserve the ordering of result. At the same time, additional identifiers are essential to uniquely identify duplicates without modifying the original data table.

Another key issue to note is the fundamental unit of operation and the operations performed are quite different between the two. Relational algebra operates on

tuples/rows. Basic relational operations are column operations working horizontally over tuples such that a target tuple is generated from one or more source tuples. There are more freedom in statistical data transformation. Such data sometimes have their rows and columns manipulated alike by operations like aggregation, group, insertion and deletion. In a more extreme case, the rows and columns in `TablePivot` of Figure 5.2, both of which are realignment of values of attributes (`Age` and `Gender`) of `TableUnpivot`, are semantically the same for further manipulation. Operations could also be performed on a specific cell, on a fraction of data using operations like loops, or on metadata of different levels.

An alternative idea is to borrow the data model and operations from a statistical data transformation language and create an algebra for the model. Though this idea is seemingly applicable, statistical data transformation languages differ in data model, operation representation and scope of operations supported. None of these languages is easily compatible with the others. A majority of these languages, such as SPSS, SAS and Stata, are specialty languages designed for different statistical analysis needs at different times and they themselves evolve over versions. Operations once not exist may be defined in later versions and simple operations have more varieties for convenient usage as add-ons. R is a more flexible functional programming language with advanced statistical routines, while Python is a general purpose objected-oriented programming language with statistical transformation functionalities (using packages like Panda and Statsmodels). They are given more flexibility in defining new operations and accessing content of data (such as cell access and blocks of data access using for loops).

We are thus inspired to summarize the majority of the operations for statistical data at an structural-wise abstracted level and propose a model accordingly leveraging the structural constructs of relational databases and statistical languages. We propose to construct a data model and a transformation model, both of which are

generic for statistical data transformation. The data model, referred to as Meta-Table, is able to preserve the information during transformation along the workflow, and at the same time, is compatible with data models of other DSTLs. We introduce additional metadata as counterpart of the original data table at different levels of abstraction. Moreover, the data model permits provenance tracking of data elements across transformations. Operating on the generic data model is the generic transformation model permitting multiple realizations of transformations. In this chapter, we introduce an algebraic realization `SDTA` and a declarative realization `SDTL`. We regard the fundamental unit of operation as the table.

5.3 Generic Data Model

This section provides a logical model of the structure of the data over domains of statistical data transformation, inspired by the data models in major DSTLs, the data model compatibility need across languages and the empirical analysis of transformation need on statistical data, permitting the tracking of data table elements across transformations.

As mentioned in the previous section, database relational and statistical tables are commonly represented in tabular format. However, there are discrepancies between these data models across languages especially in the interpretation of the data model roles serving for heterogeneous computations and transformations targeted. Here we list a few common problems encountered in the empirical comparison of statistical data models in relational DBMSs and statistical languages:

- Duplicate rows: The same row can appear no more than once in relational model but not necessarily in statistical models.
- Anonymous columns: Every attribute need to be named and reference-able in the relational model. An attribute in some statistical model can be unnamed.

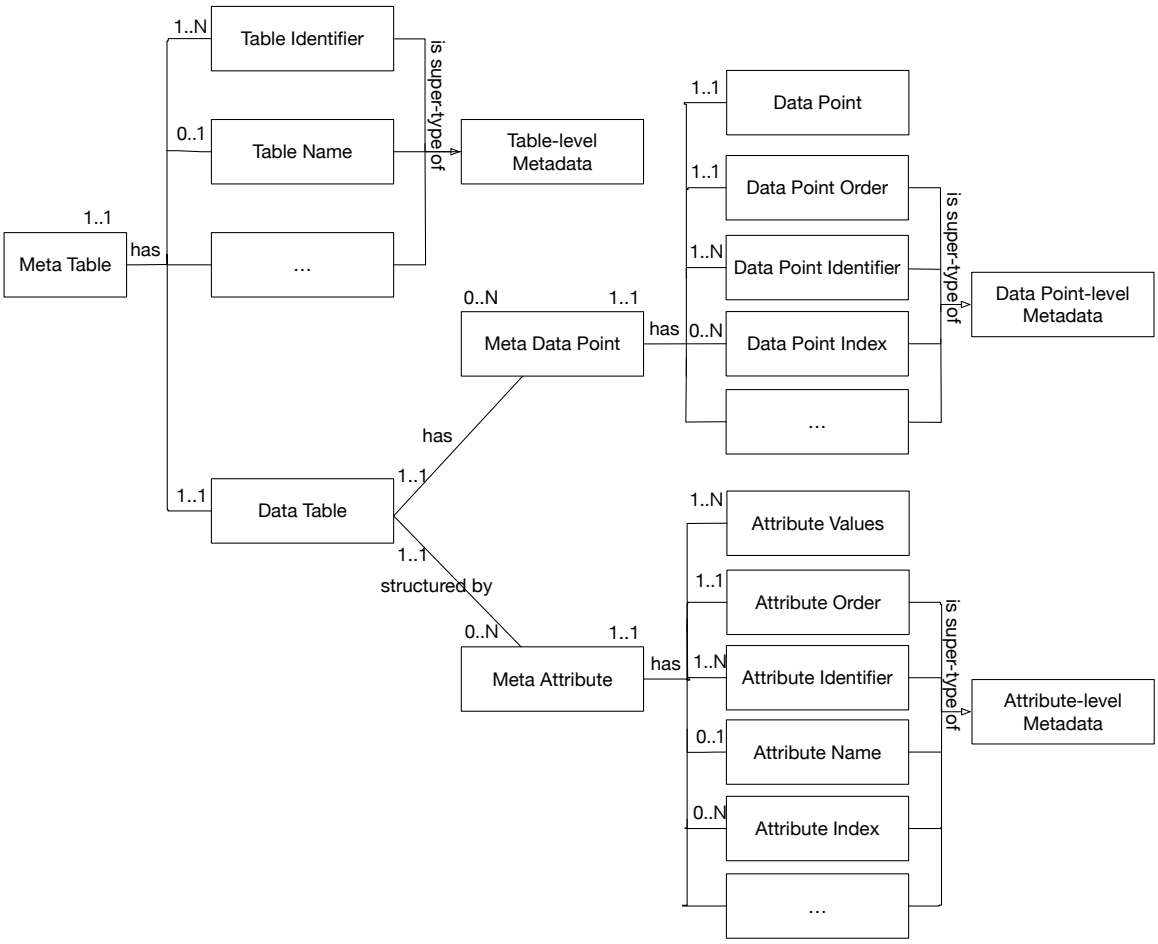


Figure 5.3: Meta Table, the Generic Data Model for Statistical Data Transformation

- Duplicate column names: Every attribute need to have a unique name in the relational model. Attribute names can be duplicated in some statistical model.

- Column order and row order: The order of columns is significant while that of the rows is immaterial in the relational model. The order of columns and/or orders are enforced in some statistical model. Columns and/or rows can be referenced by the order index.

- Missing values: Missing values can be represented by NULL in the relational model. In some statistical models, more than one type of missing value can be specified and the handling of missing values in logical statements are disparate in different statistical models.

- Metadata information: Besides the body of the table, in the relational model, table has name and attributes have names and types, etc. Statistical models allow more metadata information for convenient analysis. Metadata can be at different levels with more than one instance per type and extended by user-defined metadata, and even meta-metadata.

To tackle the compatibility issues of data models, we propose *Meta Table* as the logical data model for statistical data transformation. This concept is our counterpart to the concept of a classical statistical table, permitting the conversion between a Meta Table and a DSTL-specific data model without information loss.

We regard the classical statistical table as a named table composed of table body with data aligned in rows and columns. We extend the classical statistical table body by metadata types at different levels: table-level, attribute-level and data point-level to ensure the preservation of information during conversion. Meta Table is depicted in Fig. 5.3.

A Meta Table is composed of a Data Table and Table-level Metadata, instances of which include Table Identifier, Table Name, etc. A Data Table has Meta Data

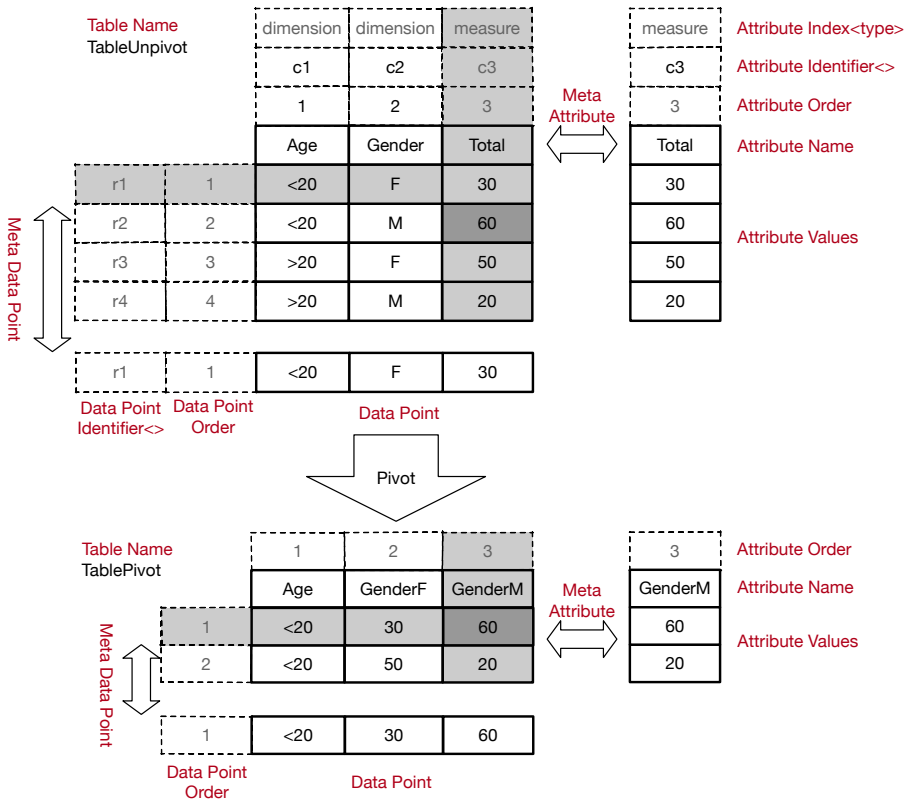


Figure 5.4: A Meta Table illustration of the pivoting table example in Fig. 5.2

Points structured by Meta Attributes. Each Meta Data Point(or Meta Attribute) is a Data Point (or Attribute) described by Data Point-(or Attribute-) level metadata, such as Order, Identifier, Index, etc. The instances of the metadata is subject to the conversion need.

Motivating example continued. *In Fig. 5.4, we show a Meta Table representation of the tables in the pivoting example in Fig. 5.2. In this example, Attribute/Data Point Order can also be regarded as Attribute/Data Point Identifier. For each table, the solid line-bordered cells coincides with the table body. The dashed line-bordered columns left to the table body represents Data Point-level Metadata, while the dashed line-bordered rows up to the table body represents Attribute-level Metadata.*

One of the key points to notice is the reference-ability of data model roles. Instead of requiring a name for each data model role, Meta Table references by identifiers. At each level, at least one unique identifier is required, ensuring the provenance of data elements over transformations. Another metadata to be noted is the Order of Data Points/Attributes which can be viewed as a special case of the identifier. The order information is a required metadata which is vital for validating the equality of Meta Tables. When we compare Meta Tables, we generally compare the body of the table, namely the Data Points and the Attributes excluding the optional metadata. When the Order is specified, We regard Data Points/Attributes as an ordered array rather than a set. The comparison is thus similar to list comparison for the type of artefacts. Other types of equality comparison can be specified otherwise. All other metadata are optional. Metadata of the same type are differentiated by names.

In Table 5.1, we show the access syntax for required and optional metadata with examples from Meta Table `TableUnpivot` in Fig. 5.4.

It should be noted that Meta Table is a logical representation of datasets, not

Table 5.1: Meta Table Metadata Access Syntax

Required or Optional	No. of Instances: Lower Bound	No. of Instances: Upper Bound	Metadata Role	Named or Not	Access Syntax	Access Example (TableUnpivot in Fig. 5.4)
Required	1..	1	Attribute Order, Data Point Order	No	Role::Value	$AttributeOrder :: 3$
		N	Table Identifier, Attribute Identifier, Data Point Identifier	Yes if >2 instances	Role$\langle Name \rangle$::Value	$AttributeIdentifier\langle\rangle :: c3$ (optional name for single instance)
Optional	0..	1	Table Name, Attribute Name, Data Point Name	No	Role::Value	$AttributeName :: Total$
		N	Data Point Index, Attribute Index	Yes if >2 instances	Role$\langle$Name$\rangle$::Value	$AttributeIndex\langle type \rangle :: measure$

necessarily corresponding to physical datasets and physical data structures.

5.4 Generic Transformation Model

The generic data transformation model provides a user-oriented view of the definition of algorithms by expressions over the generic data model for statistical data transformation. We define *Transformation* as the specification of an algorithm to derive the target Meta Table(s) from one or more source Meta Tables.

The general form of a Transformation is the following:

$$\mathcal{T}_t := expression(\mathcal{T}_s)$$

indicating that the outcome of the transformation *expression* on Meta Table(s), $\mathcal{T}_s$, in the right-hand side is assigned to the Meta Table(s), $\mathcal{T}_t$, in the left-hand side. $\mathcal{T}_s$ and $\mathcal{T}_t$ are both ordered lists of Meta Tables.

An *expression* can be of two types: (i) a direct transformation expression on Meta Table(s), and (ii) a nest transformation expression that takes the return value (Meta Table(s)) of an *expression* as input. A simple example of Transformation of the first type is

$$T_t := TableUnpivot,$$

where Meta Table T_t is assigned by Meta Table $TableUnpivot$ without changes (i.e. copied).

An example of the second type using `SDTA` operators for *expression* is

$$T_t := \kappa^r_{(Order::1,TRUE)} \left(\alpha^c_{(Identifier<>::c_3/10,4)} \left(TableUnpivot \right) \right).$$

`SDTA` is a realization of Transformation denoting *expression* as transformation oper-

ators (e.g. κ^r, α^c) on operands (e.g., Meta Table(s)), to produce the intended result following composition rules. In this example, the expression in the inner brackets first computes a temporary Meta Table as $TableUnpivot$ adding a fourth column as c_3 divided by 10. The result of the outer transformation dropping the first row of the temporary Meta Table is then assigned to T_t. The details of the transformation operators in `SDTA` are described in the next section.

This expression is a generic representation of transformation on Meta Table, regardless of the language used as long as the language is compliant with the specification above for statistical data transformation. In this chapter, both `SDTA` and `SDTL` defines mathematical expressions for statistical data transformation using this model.

5.5 Standard Data Transformation Algebra (`SDTA`)

We formally define a set of primitive operators composing Standard Data Transformation Algebra (`SDTA`) as the basic component of transformations on the generic data model (defined in §5.3) focusing on the operational perspective of statistical data transformations. The algebra is inspired by Relational Algebra and extended to the analysis and transformation needs of general statistical operations. `SDTA` could contribute to the optimization of execution depending on the execution engine and data storage structure.

It should be noted that statistical transformation workflow and relational query workflow do not work in the same way. A series of statistical transformations are generally performed over the data in a pipelined workflow, using the result of the previous step as (part of) the input of the next. The resulting data of each step are newly created at the abstract level, if not at the physical level. Relational algebra, however, has some of the operators read the data only, composing a Data Query Language (DQL), and some read and write at the same time, composing a Data Manipulation Language (DML). The write operations change the data in place.

Table 5.2: SDTA Operators

Component	Operator	Notation
Column	AddCol	$\alpha^{c}_{(f_1,i_1),(f_2,i_2),...,(f_n,i_n)}(T)$
	DropCol	$\kappa^{c}_{(c_1,p_1),(c_2,p_2),...,(c_n,p_n)}(T)$
	KeepCol	$\sigma^{c}_{(c_1,p_1),(c_2,p_2),...,(c_n,p_n)}(T)$
	OrderCol	$\omega^{c}_{f}(T)$
	AggrCol	${}_{c_1,c_2,...,c_m}\gamma^{c}_{f_1,f_2,...,f_n}(T)$
	Join	$S \bowtie_{C,f^c,f^r} T$
Row	AddRow	$\alpha^{r}_{(f_1,i_1),(f_2,i_2),...,(f_n,i_n)}(T)$
	DropRow	$\kappa^{r}_{(r_1,p_1),(r_2,p_2),...,(r_n,p_n)}(T)$
	KeepRow	$\sigma^{r}_{(r_1,p_1),(r_2,p_2),...,(r_n,p_n)}(T)$
	OrderRow	$\omega^{r}_{f}(T)$
	AggrRow	${}_{r_1,r_2,...,r_m}\gamma^{r}_{f_1,f_2,...,f_n}(T)$
Column & Row	Reshape	$\lambda_{(str_1,...,str_n),(c_1,...,c_n),f}$
Metadata	InsertMeta	$\alpha^{m}_{Role<Name>}(T)$
	DeleteMeta	$\kappa^{m}_{i}(T)$
	UpdateMeta	$\rho^{m}_{i,v}(T)$

Transformation on statistical data mainly manipulates three structural components: columns, rows and metadata. The transformation is more analytical computation intense rather than structural transformation intense. In this sense, statistical computations manipulates structural components independently most of the time. Alternatively speaking, it hardly make changes to multiple structural components at the same time.

The set of basic `SDTA` operators focuses on four aspects: the independent manipulation of (i) columns and (ii) rows, (iii) the manipulation involving the change of the table structure (both columns and rows), and (iv) metadata manipulation. Due to page limitation, we will give a brief introduction of the operators in the following paragraphs.

5.5.1 Add, Drop, Keep And Order Rows and Columns

Unlike relational theory, the columns and rows of statistical tables are ordered such that many operations on rows and columns are alike. Here we define AddCol,

DropCol, KeepCol and OrderCol for columns and AddRow, DropRow, KeepRow and OrderRow for rows.

Definition V.1. AddCol (α^c). Let T be the input Meta Table, f_x be the statistical functions over columns of T and i_x be the column position index for $x = 1, ..., n$. AddCol creates the output Meta Table R by adding n column(s) f_x at position i_x to T as

$$R = \alpha^c_{(f_1,i_1),(f_2,i_2),...,(f_n,i_n)}(T).$$

Definition V.2. DropCol (κ^c). Let T be the input Meta Table, c_x be a unique identifier of the column of T and p_x be the drop condition predicate for $x = 1, ..., n$. DropCol creates the output Meta Table R by dropping n column(s) identified by c_x when the corresponding drop condition p_x is true from T as

$$R = \kappa^c_{(c_1,p_1),(c_2,p_2),...,(c_n,p_n)}(T).$$

Definition V.3. KeepCol (σ^c). Let T be the input Meta Table, c_x be a unique identifier of the column of T and p_x be the keep condition predicate for $x = 1, ..., n$. KeepCol creates the output Meta Table R by keeping n column(s) identified by c_x when the corresponding keep condition p_x is true from T as

$$R = \sigma^c_{(c_1,p_1),(c_2,p_2),...,(c_n,p_n)}(T).$$

Definition V.4. OrderCol (ω_c). Let T be the input Meta Table, f be an ordering function of columns of T. OrderCol creates the output Meta Table R by reordering the columns of T by f as

$$R = \omega^c_f(T).$$

Similarly, we define AddRow, DropRow, KeepRow and OrderRow for rows.

Definition V.5. AddRow (α^r). Let T be the input Meta Table, f_x be the statistical

functions over rows of T and i_x be the row position index for $x = 1, ..., n$. AddRow creates the output Meta Table R by adding n row(s) f_x at position i_x to T as

$$R = \alpha^r_{(f_1,i_1),(f_2,i_2),...,(f_n,i_n)}(T).$$

Definition V.6. DropRow (κ^r). Let T be the input Meta Table, c_x be a unique identifier of the row of T and p_x be the drop condition predicate for $x = 1, ..., n$. DropRow creates the output Meta Table R by dropping n row(s) identified by c_x when the corresponding drop condition p_x is true from T as

$$R = \kappa^r_{(c_1,p_1),(c_2,p_2),...,(c_n,p_n)}(T).$$

Definition V.7. KeepRow (σ^r). Let T be the input Meta Table, c_x be a unique identifier of the row of T and p_x be the keep condition predicate for $x = 1, ..., n$. KeepRow creates the output Meta Table R by keeping n row(s) identified by c_x when the corresponding keep condition p_x is true from T as

$$R = \sigma^r_{(c_1,p_1),(c_2,p_2),...,(c_n,p_n)}(T).$$

Definition V.8. OrderRow (ω_r). Let T be the input Meta Table, f be an ordering function of rows of T. OrderRow creates the output Meta Table R by reordering the rows of T by f as

$$R = \omega^r_f(T).$$

5.5.2 Column Aggregation and Row Aggregation

In statistical analysis, other than the original data, summary data information is a primary feature achievable by aggregation functions such as MAX, AVG, MEAN. In addition to the common vertical aggregation (column aggregation) in relational

algebra, horizontal aggregation (row aggregation) is also a common practice for statistical data. Here we define the two types of aggregation with an optional grouping clause.

Definition V.9. AggrCol (γ^c). Let T be the input Meta Table, $c_1, c_2, ..., c_m$ be a list of identifiers of columns on which to group (can be empty), f_x be an column-wise aggregation function for $x = 1, ..., n$. AggrCol creates the output Meta Table R by aggregating columns of T by m aggregation functions f_x over groups specified by unique value combinations of n columns identified by c_x as

$$R =_{c_1,c_2,...,c_m} \gamma^c_{f_1,f_2,...,f_n}(T).$$

Definition V.10. AggrRow (γ^r). Let T be the input Meta Table, $r_1, r_2, ..., r_m$ be a list of identifiers of columns on which to group (can be empty), f_x be an column-wise aggregation function for $x = 1, ..., n$. AggrCol creates the output Meta Table R by aggregating columns of T by m aggregation functions f_x over groups specified by unique value combinations of n columns identified by c_x as

$$R =_{r_1,r_2,...,r_m} \gamma^r_{f_1,f_2,...,f_n}(T).$$

5.5.3 Join

The join operation is similar to the join in relational algebra, extended by the ordering preservation feature. To preserve the ordering of rows in the input Meta Tables, the join operation in `SDTA` is in place join with a natural ordering, if not specified to be reordered after the join.

Definition V.11. Join ($\bowtie$). Let T and S be the input Meta Tables, C as a set of identifiers of column, regarded as join keys (by default is the column intersection of T and S), and f^c and f^r be the optional reordering functions of rows and columns.

Consider each pair of rows r_t from T and r_s from S by looping over rows in T and then in S in order, if r_t and r_s have the same value on each of the columns specified by C, add a row r to the output Meta Table R, where r has the same value as r_t on T and r has the same value as r_s on S. The columns and rows of the output Meta Table can be reordered by f^c and f^r if specified. By default, $f^c = (C, T \backslash C, S \backslash C)$. The Join operation can be represented by the following equation.

$$R = S \bowtie_{C, f^c, f^r} T.$$

5.5.4 Metadata Manipulation

To manipulate the metadata of Meta Table, we define three operators: InsertMeta, DeleteMeta and UpdateMeta.

Definition V.12. InsertMeta (α^m). Let T be the input Meta Table. InsertMeta creates the output Meta Table R by inserting metadata role *Role* with an optional name $Name$ as the identifier of the new metadata to T as

$$R = \alpha^m_{Role<Name>}(T).$$

Definition V.13. DeletaMeta (κ^m). Let T be the input Meta Table. DeleteMeta creates the output Meta Table R by deleting the metadata specified by identifier i from T as

$$R = \kappa^m_i(T).$$

Definition V.14. UpdateMeta (ρ^m). Let T be the input Meta Table. UpdateMeta creates the output Meta Table R by updating the value v of the metadata identified by i to T as

$$R = \rho^m_{i,v}(T).$$

5.6 Standard Data Transformation Language (SDTL)

We further propose a declarative language `SDTL` for statistical data transformation based on the algebraic operators defined for `SDTA`.

`SDTL` is defined by the Convention-based Ontology Generation System (COGS) [15] information model. It offers multiple representations such as XML, JSON, GraphQL under one specification. It supports the most basic and widely used data transformation operations in the four main statistical languages from major data production projects (the General Social Survey (GSS)[1], the American National Election Study (ANES)[2], and the National Survey of Family Growth (NSFG)[3]), ICPSR, DataOne[4] and sample scripts provided to journals in conjunction with replication datasets. Each operation is bundled with a natural language interpretation template to better illustrate the associated operation.

The majority of `SDTL` commands are composite operations derived from nested `SDTA` operations, providing shortcuts for commonly used transformation of statistical data permitting the reuse and sharing of definitions. `SDTL` defines facilitating operators in multiple categories as an extendable function library for better statistical manipulation. In this section, we briefly introduce `SDTL`. More details are available at `http://c2metadata.gitlab.io/sdtl-docs/master/`.

There are two base types in `SDTL`: TransformBase and ExpressionBase. `SDTL` Transform commands extends TransformBase and inherits general transformation properties including the input data consumed, the output data generated, the type of the transformation command. Expressions, on the other hand, are passed to and evaluated by Transforms.

Currently, there are five categories of major Transform commands manipulating:

[1] http://gss.norc.org/
[2] https://electionstudies.org/
[3] https://www.cdc.gov/nchs/nsfg/index.htm
[4] https://www.dataone.org/

Table 5.3: Major transform commands supported by SDTL

dataset level	load, save, rename, create, merge, match, transpose, format display
column/variable level	recode, rename, aggregate, compute, delete, label, sort, missing value, join
row/case level	select, sort, add, delete, aggregate
cell level	update, label
procedural	if-then, do repeat loop

(i) input/output, (ii) data structure, (iii) data contents, (iv) change of metadata and (v) procedural control. Here we list a few commands for each category in Table 5.3.

An important Expression in SDTL is FunctionCallExpression calling functions defined in the function library. Current function library defines string operators, Boolean operators, validation operators, conditional operators, statistical operators, etc. Statistical functions span a wide variety of commonly adopted in statistical analysis used in major statistical languages. Some of them work on individual data cells, such as missing_value that convert the cell value into missing value. Aggregation functions are generalized to make vertical and block aggregation possible, in addition to the usual vertical aggregation in SQL. An example of the horizontal aggregation function working on rows is row_first, finding the first non-missing value among values of a range of attributes (columns) for each row. Another set of popular statistical function is the collapse function creating summary statistics of block data. col_sd is a simple collapse function that computes the standard deviation of multiple attributes (columns) within group.

To permit support of functionalities not currently available, SDTL allows user-defined commands and functions for extensibility consideration. User-defined transformations or expressions should extend TransformBase or ExpressionBase type with additional properties necessary. Statistical computation functions can be added to the function library in a simple manner.

5.7 Use Cases

SDTA and SDTL are not merely simple, optimizable and extendable, they are applicable in a variety of use cases. Here we emphasize two application scenarios: data documentation and language translation.

5.7.1 Automatic Data documentation of Statistical Transformation by C^{2}Metadata

As the research community responds to increasing demands for public access to scientific data, the need for improvement in data documentation has become critical. Accurate and complete metadata is essential for data sharing and for interoperability [38]. However, the process of describing and documenting scientific data has remained a tedious, manual process even when data collection is fully automated.

Researchers in many fields use statistics packages for data management as well as analysis. These packages, however, lack tools for documenting variable transformations in the manner of a workflow system or even a database. At best, the operations performed by the statistical package are described in a script, which more often than not is not even available to future data users. Different statistics packages differ in data model, transformation representation and scope of transformations covered; thereby further complicating the understanding of the transformation process.

We have developed C^{2}Metadata (http://c2metadata.org) as a pipelined system for automatic transformation documentation as provenance-aware metadata for scientific data, available for four major statistical languages: SPSS, Stata, SAS and R.

The pipelined modular architecture of C^{2}Metadata comprising four modules, as shown in the grey area in Figure 5.5.

The workflow in C^{2}Metadata consists of four modules: SDTL Parser, Pseudo-code Generator, XML Updater and Codebook Formatter.

SDTL Parser. In the Script Parser module, C^{2}Metadata expresses data transfor-

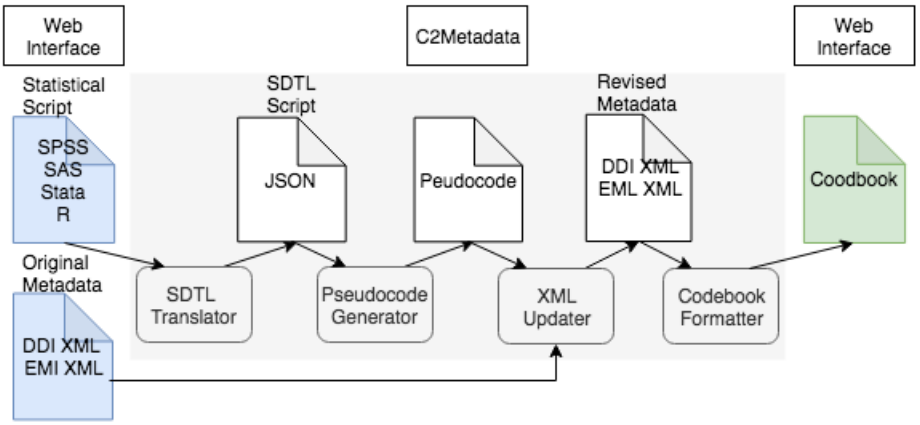


Figure 5.5: workflow

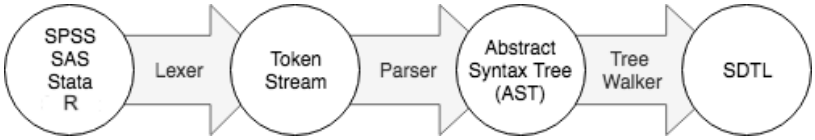


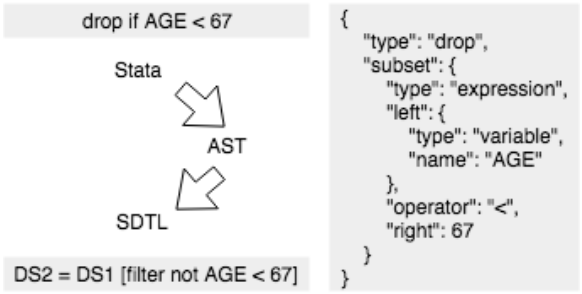
Figure 5.6: SDTL Parser Components

mations in SDTL, independent of the language used. The components of the module are shown in Figure 5.6.

Script Parser is customized for each statistical package since they each use a different scripting language. It takes a command script written in scripting languages as input and uses standard compiler techniques to parse input, obtain a syntax tree, and then generate SDTL code. The syntax Lexer transforms the raw SPSS/Stata/SAS/R syntax code by a lexer grammar for each language into a stream of tokens. The ANTLR-based [82] Parser then parses the token stream into an *abstract syntax tree* (AST), simple or nested. The Tree Walker finally walks over the AST and refers to a mapping between statistical languages and SDTL for SDTL translation. Below is the translation of an example of a simple command in Stata by Stata Script Parser.

Since R is a more dynamic and open language than SPSS, Stata or SAS, we limit our scope of translation to the set of transformation-based functions in the base and tidyverse [105] packages.

The SDTL Translator issues an error when illegal or unrecognized commands are



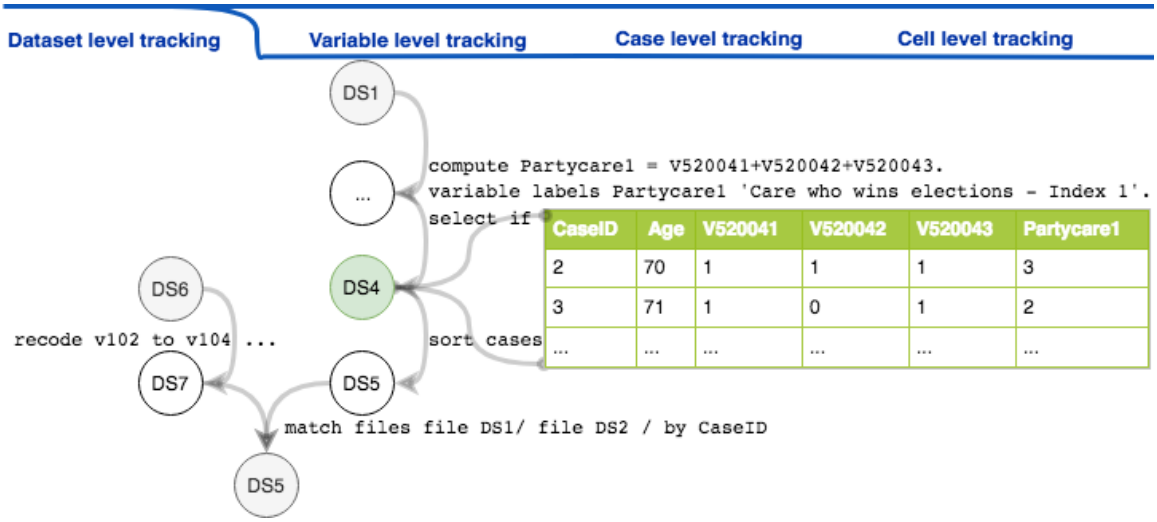
found in inputs or intermediate steps.

Pseudocode Generator. The translated`SDTL` script is then converted to human readable text for a more user-friendly illustration of the transformations included.

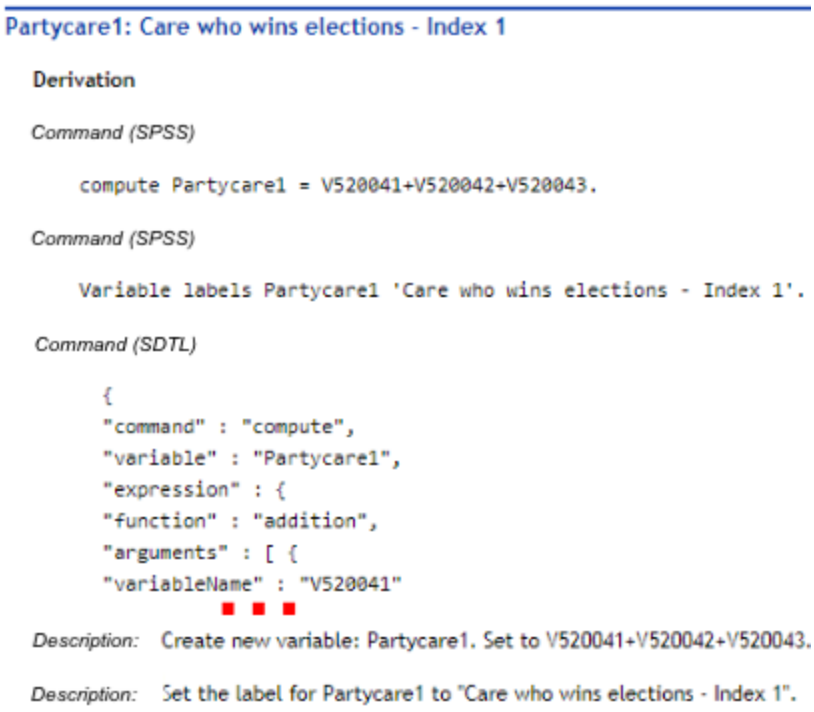
XML Updater. The original metadata in DDI or EML standards are updated with both file level and data element level transformations including the original transformation script, the natural language description of transformations and the `SDTL` equivalent in XML format. (DDI and EML are both based on XML).

Codebook Formatter. An HTML codebook is generated from the revised metadata describing the contents, structure and layout of the revised data.

Apart from the revised metadata, `C`^{2}`Metadata` allows tracking of the processed transformations at four different levels: dataset level, variable level, case level and cell level, in multiple settings and transformation representations. Figure 5.7(a) shows the tracking of the transformation of the sample at the dataset level in a graphical setting, where each node is a dataset after one step of transformation. Here we present the transformation by the original SPSS representation. By clicking on the node, the intermediate transformation result will expand for inspection. Variable-, case-level and cell-level give more provenance information of the object of interest at the current stage or along the transformation trace. An example of the tracking of variable Partycare1 is shown in Figure 5.7(b) in a codebook setting. In this senario, The transformations applied specifically to this variable are represented by the original language (SPSS), `SDTL` and a human-readable natural language description.



(a) Graphical visualization at dataset level



(b) Cookbook setting at variable level

Figure 5.7: An example of dataset level transformation lineage visualization and codebook level variable derivation

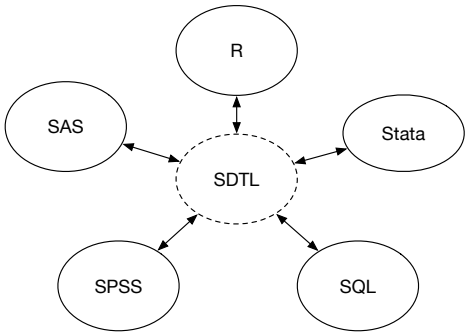


Figure 5.8: Language translation using `SDTL` as the bridge

`C`^{2}`Metadata` is deployed as a Docker container since multiple collaborators contribute to the development of the project using technologies including .NET, closure, Java, XSLT, COGS, among others.

5.7.2 Language Translation

In language translation, a *pivot language* (or *bridge language*), can be used as an intermediary language for translation between many different language pairs. Using `SDTL` (or `SDTA`) as the bridge, we have built a mapping between `SDTL` and major statistical languages (SPSS®, SAS®, Stata® and R) as shown in Figure 5.8 for efficient language translation between statistical languages. Such a translation mechanism provides chances for multiple tasks, for instance, code reproducibility and execution efficiency when the execution of some operations are more efficient in one language than in another.

5.8 Conclusion

To denote statistical data transformation in a standard manner, we have presented a generic data model and a generic transformation model for the purpose. Inspired by relational algebra and query language, we also propose `SDTA` and `SDTL` as two language realizations of the transformation model. `SDTA` has only a few operators covering the majority of operations in statistical transformation. Both languages are simple,

extendable, reproducible and optimize-able. They can serve as the standard language for documenting the provenance information of data transformation in metadata and the intermediate language for statistical language inter-conversion.

Currently, the mappings are manually built between statistical languages and SDTA/SDTL, which is tedious and time consuming. It is not possible to map all operations with this approach, leading to a certain number of unsupported operations not translatable to SDTA/SDTL, which degrades the power of SDTA/SDTL for transformational language representation and language translation. Also, SDTA/SDTL do not support analysis operations including ML-based model learning operations such as regressions and classifications, and statistical testings. Since analysis operations compose an important part of the operations on statistical data, we plan to extend SDTA and SDTL in this direction in the future.

CHAPTER VI

FluxSearch: Searching Datasets Leveraging Both Metadata and Data Content

This chapter focuses on the design of `FluxSearch` system that facilitates keyword search of integrated datasets on data portals. This search is error-prone, as we will see with prior literature. To minimize error, we will use both the data tables themselves and the metadata describing the dataset. To push for data sharing, many data portals nowadays require data depositors to share data with metadata that describes the data. These data portals often provide a dataset search engine for dataset search or discovery, where users compose keyword queries to look for related datasets leveraging a query index built on metadata. However, metadata quality is often low since metadata generation takes much manual work, and it lacks the contribution from the depositors or the verification and curation from the data portals. Metadata can be missing, incomplete, or not descriptive of the data, causing the search quality to be under expectation. Even though the search engine can find datasets related to the query, it is possible that no single dataset best fits the query. A massaged dataset integrated from two or more datasets partly matching the query might be a better answer. One may use schema mapping, entity matching, transformation, and integration techniques, such as union and join, from the database community to integrate data tables in the datasets, though the integrated table has no metadata that cor-

rectly describes it. We identify two problems not well addressed in the current dataset search engines to search for integrated datasets: the lack of data attention for search using metadata and the lack of metadata attention for integration using data. By enriching metadata with data and enriching data with metadata, `FluxSearch` finds better datasets related to the query, smartly proposes the integration strategy of datasets partly match the query, generates the integrated dataset ready to be shared, and ranks the integrated datasets by their validity of integration and relatedness to the query. Currently, `FluxSearch` focuses on the integration of two datasets.

6.1 Introduction

To generate value from data in the open data era, finding datasets suitable for a specific task is an inevitable yet vital problem to address. Unlike web tables that scatter around the web pages with limited ancillary information to help with its interpretation, datasets shared on open data portals are better documented with metadata, maintained by centralized management, and searchable by an embedded dataset search engine. Opendatasoft, in a recent post, lists over 2600 open data portals around the globe, including domain-dependent data portals for open government data, GIS data, biomedical data, etc., and data portals host data from multiple domains. Some data portals host data themselves, while others hold data from different sources, organizing them under subsets of categories to make it easier for users to find.

By dataset, we mean data, mainly in tabular format, with metadata that explains the data, possibly with ancillary materials. Metadata can be categorized into three types: descriptive metadata, structural metadata, and administrative metadata. Descriptive metadata is defined for dataset identification and retrieval, such as basic information including title, author, and abstract, and in-depth information including provenance, annotation, works citing the dataset, etc. Structural metadata docu-

<codeBook> 0.0 Codebook

- Mandatory
- Not Repeatable
- Attributes:

Description:
Every element in the DDI DTD/Schema has the following attributes:
ID - This uniquely identifies each element.

<docDscr> 1.0 Document Description

<citation> 1.1 Bibliographic Citation

<titlStmt> 1.1.1 Title Statement

<stdyDscr> 2.0 Study Description

<fileDscr> 3.0 Data Files Description

<dataDscr> 4.0 Variable Description

<var> 4.3 Variable

- Optional
- Repeatable
- Attributes: ID, xml:lang, source, name, wgt, wgt-var, weight, qstn, files, vendor, dcml, intrvl, rectype, sdatrefs, methrefs, pubrefs, access, aggrMeth, measUnit, scale, origin, nature, additivity, temporal, geog, geoVocab, catQnty

<location> 4.3.1 Location

- Optional
- Repeatable
- Attributes: ID, xml:lang, source, StartPos, EndPos, width, RecSegNo, fileid, locMap

<labl> 4.3.2 Label

- Optional
- Repeatable
- Attributes: ID, xml:lang, source, level, vendor, country, sdatrefs

<catgry> 4.3.18 Category

<catValu> 4.3.18.1 Category Value

<labl> 4.3.18.2 Label

<otherMat> 5.0 Other Study-Related Materials

Figure 6.1: Part of XML-based DDI Schema Tag Library, version 2.1

ments relationships within and among objects in the data, such as data profiling and data schema. Some of the common granularity levels described in such datasets' metadata include dataset-level, table-level, and variable-level metadata. Administrative metadata manages information resources such as version number, archiving data, etc. Metadata may follow different standards defining the types/categories of fields and the structure of metadata, and possibly controlled vocabularies and name authorities for data value standards, and data content standards guiding what values to input into metadata fields. Metadata standards evolve from domain-specific formats such as MARC library offering limited text-based download formats to libraries with limited standardization. The more recent libraries offer metadata with less proprietary formats (e.g., RDF, XML, JSON) as a part of the open data initiatives. In Figure 6.1, we show part of the Schema Tag Library of metadata standard DDI. It defined five top-level fields for codebook `<codebook>`, document `<docDscr>`, dataset `<stdyDscr>`, data table `<fileDscr>`, variable `<dataDscr>` and other materials `<otherMat>` in a dataset. Metadata of the same dataset following different standards, such as JSON-based schema.org utilized by Google Dataset Search and XML-based EML, DCAT utilized by CKAN, contain similar information.

A large number of data portal internal search engines and external vertical dataset search engines, such as Google Dataset Search, discover datasets by populating keyword queries over *metadata* and rank datasets based on their matching score. Though many challenges are foreseen in the field, one of the not well-addressed ones is that no single dataset can fully answer the search query in many situations. Two, or more, datasets may partly answer the query from different facets.

This reminds us of keyword search in RDBMS that also stores data tables. The integration of IR and DB benefits the user to search unstructured information using keywords based on scoring and ranking over structured information in databases without studying query languages, such as SQL and SPARQL. Existing approaches

leverage database schemas, mainly primary-key foreign-key references (as edges), to build links between tuples (nodes) that contain part of the keywords in different relations for an integrated structure covering all the keywords searched, e.g., a directed/undirected graph. As for datasets shared on data portals, unlike in RDBMS, there may not be clear relationships identified between them since these datasets are collected individually and shared for heterogeneous purposes using different schemas and terminologies.

To mind the gap between user search intention and the availability of datasets, we aim to simply ask the user to express their full intention as keyword queries and save users from the tedious work of finding relevant datasets and constructing the best match integrated dataset from datasets partly matching. To make the problem more tractable, we narrow our scope to datasets with one or more data tables and semi-structured metadata and integrate data from at most two datasets to satisfy the search query. We mainly focus on metadata in XML-based DDI format, a general metadata standard widely adopted in the social science community. However, our ideas equally apply to other semi-structured metadata standards following a similar schema.

There are multiple steps in a classic dataset search model for a single dataset lookup. Datasets are indexed offline to allow for fast online retrieval for datasets related to the query. These datasets are then ranked for a final query result based on their relatedness to the query. To search for integrated datasets, we emphasize three perspectives, namely the search, the integration, and the query result generation, which are not fully explored with current techniques for the problem we identify.

Problems with the search. Since queries are populated to the backend for search on query indexes built on metadata of datasets, the quality of metadata largely determines whether a dataset related to the query can be found; thus affecting the quality of the search result. Poor quality metadata can be missing or incomplete, or lack

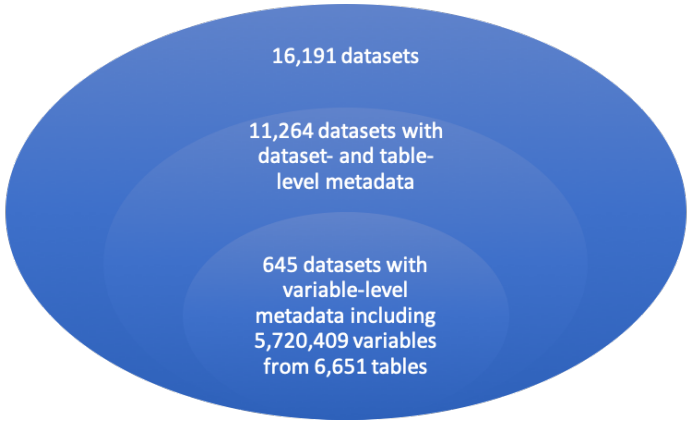


Figure 6.2: ICPSR Studies (Datasets) by metadata granularity (March 24. 2021)

descriptive power.

Many datasets from the past have no metadata. They may have ancillary summary files in text, word, or pdf formats describing the data but no structured metadata feasible for indexing for the search. Though more and more data portals are demanding structured metadata for recent dataset deposits, the quality of the metadata varies across depositors and data portals and is not guaranteed to be without missing information. In Figure 6.2, we show the datasets from ICPSR by metadata granularity. Approximately 70% of datasets from ICPSR have dataset- and table-level metadata. Only around 4% of all datasets have variable-level metadata for 5.7 million variables in 6,651 tables from these datasets, the generation of which takes more than 20 years of curation work of data depositors from ICPSR. Data portals like ICPSR have portal-end data processors to systematically generate and validate the metadata for a high-quality guarantee, while many other data portals, such as Dataverse, have no such quality control mechanism. Metadata generation, either by the depositor or by the data portal, is mostly manual work with little automation.

Many relevant data tables may be missed in this search process. A dataset may have data entries that match the search query, but its low-quality metadata may not match. Such a dataset is not searchable for the query with the current indexing system. Some recent efforts have been made to provide summary annotation of data

as ancillary information for the search. However, the power of data is not fully recognized in the process.

We consider both metadata and data as important components to decide whether the dataset match (part of) the query. Intuitively, one option is to build the query index on both data and metadata. However, structured data and semi-structured metadata are defined for different schemas and cannot be easily indexed together. Another option is to build two indexes on data and metadata separately and populate the query to both indexes before comparably combining the matches. Through an empirical analysis of datasets on ICPSR, we found that table names, variable names, and variable values in data can be encoded for ease of storage and privacy protection using short text or numbers for categories. Their true values are masked and accessible from the metadata if documented. Among 881,626 numeric variables with attribute `representationType` of field type `var`, 483,061 (54.8%) are categorical variables whose values in the data table are encoded as numbers. An index built on the masked values of the data is infeasible for a meaningful search. Both options require modification to the indexing mechanism. We seek to summarize information from data and enrich metadata with such information for more complete metadata to index the enriched metadata for better search. This approach takes advantage of the original indexing mechanism. As a by-product, it generates part of the metadata in an automated way saving the effort of manual specification, which can be later used for other analysis or research. With this idea in mind, we identify several problems to tackle in the process.

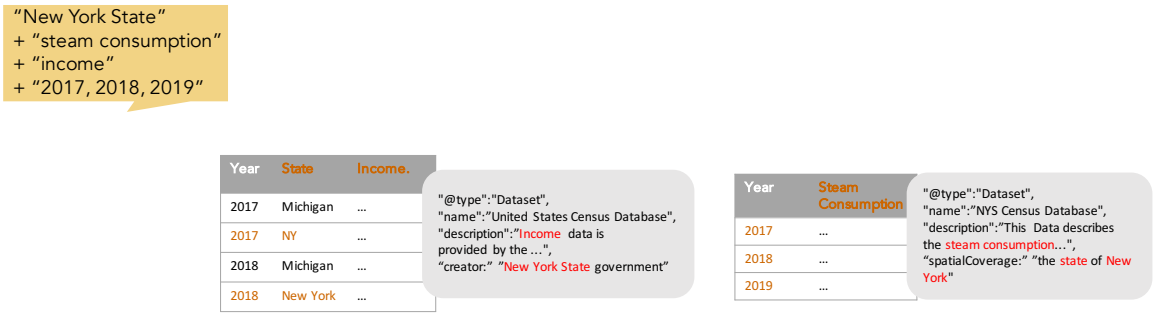
By empirical analysis of datasets from ICPSR and Dataverse, we observe that datasets do not usually share the same set of metadata fields. There are over 500 fields defined in the DDI standard for metadata. The fields in the DDI standard are organized in a hierarchy. For each field, the standard defines its parent field, whether it is mandatory or not, the cardinality, the optional attributes describing the field,

and a detailed description of the field. Among the five top-level metadata fields in the DDI standard shown in Figure 6.1, a small fraction of datasets in ICPSR has high-quality variable-level metadata which requires heavy manual curation. Only datasets with complementary materials such as survey questionnaires and images may have metadata defined for these materials.

To create metadata documentation, a user must first understand the fields and field hierarchy of metadata schema and select the appropriate fields and attributes to include for different datasets. This is not an easy task considering the scale of the metadata schema. Also, the importance of the fields is different. We regard mandatory fields as important fields. Many of the optional fields, however, are incompletely documented or ignored by depositors for effort saving. The definition of metadata field importance varies by data portals. Summary statistics documented by field type `<sumStat>`, a child field of `<var>` for variables, for example, is at times missing for variables in datasets with variable-level metadata from Dataverse. Dataverse is an open data-sharing community that does not involve the level of curation and validation that ICPSR enforces. Depositors share data from different sources, and they have different criteria for which fields are important and when a field should be defined. Certain types of `<sumStat>` with attributes such as `type="mean"` and `type="stdev"` are defined for variable whose `representationType` attribute of `<var>` is `"numeric"`. While for ICPSR, the majority of numeric valued variables have the full set of summary statistics documented for variables, including variables of number-encoded categorical variables with representation type `"code"`. Also, the presence of some fields and the value of attribute depend on the presence of other fields and/or the value of other attributes, mainly its ancestor, descendants, and sometimes siblings, along the metadata schema hierarchy. The value of `<labl>`, for instance, is a shorter description of the parent field, defined for multiple parent fields such as variable `<var>`, category group `<catgryGrp>`, record group `<recGrp>` differentiated

by attribute `level` values `"variable"`, `"category group"` and `"record group"`, respectively. Variables with attribute `representationType="text"` only have child fields `<sumStat type="vald">` and `<sumStat type="invd">` for number of valid and invalid values, while numeric variables have more types of summary statistics for distribution-related information. The last problem to consider is what is the value of fields and field attributes. We need to identify fields that are not accessible without prior knowledge of how the dataset is generated. While for data and metadata-related fields, we may systematically determine its value and attribute values based on the context.

Problems with the Integration. The problem of finding related tables or tabular search has been a sub-topic of dataset search, where the query is a table, and the purpose is to manipulate and extend them with other tables. The main challenge is to find latent links between the query table and the candidate tables. Different similarity metrics evaluate how closely related two variables from two tables are to assess the possibility of table integration further. These metrics consider data content mostly, if not including variable names and table names [79, 55]. The power of metadata, not generally available for web tables, is not fully utilized. A latent variable described in the metadata of one table may not be considered for integration with variables in another table. A possible integration candidate may fail to be considered in this case. Also, there are multiple types of possible integration. Past works mostly consider table union or table join separately, but rarely the two jointly. Table union identifies variables from the same underlying domain to extend the table with more entities. Table join takes one step further to identify variables from the same domain sharing common entities to extend existing entities with richer descriptive information. The main questions to be answered are how to choose between union and join for candidate datasets wisely and how to rank the integrated datasets from both union and join



(a) An example of search for integrated dataset

(b) Metadata enriched Data for integration for the dataset on right in (a)

Figure 6.3: A motivating example for the search of integrated datasets

reasonably. For table join, [113] considers joining on one variable. However, two datasets may share multiple variables from the same domain, including variables from the data and latent variables from the metadata. It is necessary to discover all shared latent variables and prioritize multi-variable join over single-variable join in the ranking step for a more statistically valuable join.

We discuss these problems from the integration perspective with the following motivating example.

Motivating example. *Search for integrated datasets is not easy to solve due to the lack of links between data tables from different datasets. Basically, we are given arbitrary tables. To make the problem more tractable, we consider two tables for now. We intend to augment data tables with richer metadata for more links to promote reasonable or valid integration. Here we consider two integration scenarios: table join and table union. More specifically, for table join, we join aggregated measure variables of the two tables by their common dimension variables, also known as the*

join keys.

To approach the problem, pairs of dimensions, one from each table, are evaluated for their joinability as the degree of data value overlaps. Multiple existing methods could be applied, and these methods rely on the fact that the true values of entities are stored in the data table. However, some of the data values in tables of data portals are encoded as surface or masked values to hide the true values. Only with the metadata could the true value be decoded for interpretation. For instance, many attribute names for tables collected from surveys are encoded as V1, V2, ... such that each variable corresponds to a survey question and values of which are answers of the question by survey respondents. Since the survey questions are generally long to be stored as attribute names, they are stored as variable-level metadata, `<labl type="variable">` *for instance, for each variable in the ideal case. Similarly, survey question answers can also be encoded into categories in tables, and their true values are stored as multiple variable-level metadata* `<labl type="category">` *for each variable. Hence, to evaluate the joinability of variables in the dataset, we need to uncover the true values of entities in data tables from metadata. The question to ask is then which fields contain the true value for different data elements.*

Another critical problem is that a valid join depends on the ability to discover all common dimensions shared by the tables, whereas latent dimensions hidden in the metadata are once neglected during the process.

Let us continue with the motivating example in Figure 6.3. With two given tables, we first identify `Year` *and* `State` *as dimensions and* `Income` *as measure in the first table, and* `Year` *as dimension and* `Steam Consumption` *as measure in the second table. As* `Year` *in both tables share many values in common, it can be regarded as the join key or unionable variable.*

Let us consider join first. An intuitive join on `Year` *generates a joined table should contain four attributes:* `Year` *(T1, T2),* `State` *(T1),* `Income` *(T1) and* `Steam`

Consumption (T2). This is not a valid table since State is describing the Income but not Steam Consumption. Thus State could not be regarded as the dimension of the integrated table. An alternative approach is to aggregate measures for the join key in each table before the join. Then Income and Steam Consumption are aggregated for Year respectively before the aggregated tables join by Year. The joined table has three columns: Year (T1, T2), sum(Income) (T1), sum(Steam Consumption) (T2). This is a seemingly valid approach if we have no access to the metadata. Given the metadata, we know that the spatial coverage of T2 is New York State. sum(Income) for multiple states in one year is not strongly comparable with Steam Consumption for New York state in the same year. If we take one step back, at least we need to state the derivation processes of the joined table in its metadata for responsible data sharing. If a user analyses the joined table, she knows the geographical coverage of the two aggregated variables by tracing back to its source tables.

We thus seek to discover latent variables in the metadata before integratability (unionability/joinability) evaluation. Since metadata is defined for multiple granularities, such as dataset-level, table-level, and variable-level metadata, data table elements could be augmented accordingly and propagate downwards along the hierarchy. In our motivating example, since spatial coverage is a dataset-level metadata element, the spatial coverage constraint is likely to apply to all data tables in the dataset, which also applies to all tuples in these tables. It can be treated as a virtually populated dimension variable spatialCoverage with constant value "the state of New York" as shown in T2'. Continue with join key discovery and integration, we will get a joined table with four attributes: Year (T1, T2'), State (T1)/SpatialCoverage (T2'), sum(Income) (T1) and sum(Steam Consumption (T2') such that both Year and State/SpatialCoverage are table-level dimensions jointly describing Income and Steam Consumption.

Leveraging the unionable/joinable variable pairs Year (T1, T2) and State (T1)/

`SpatialCoverage` (T2'), We now have six integrated tables such that four join/union on either Year (T1, T2) or `State` (T1)/`SpatialCoverage` (T2'), and two join/union on both variables. For a given search query, the search engine need to rank these integrated tables wisely based on their relatedness to the query and the validity of the integration.

The integrated dataset. As discussed in the motivating example, integrated tables lack metadata description. There is no standard defining what information should be documented in the metadata for such a table to make it a valid dataset for sharing. To make data sharing with metadata sustainable, we find it necessary to define the integration process in the metadata of the integrated dataset, including the source tables and the datasets they belong to, latent variables used for integration, integrability of variable pairs used for integration, and the integration transformations, for better understanding of the derived datasets. Provenance and metadata manipulation techniques should be used for this purpose. Since this is not the main focus of this chapter, we will not dive deep into this problem in detail.

Tackling the problems above, for dataset search engines using keyword-based queries where two datasets may partly answer the query from different facets, we developed `FluxSearch` to generate a constructive dataset by coupling dataset search with data integration providing dynamic task-oriented datasets. `FluxSearch` enriches metadata with data for the search by generating missing metadata information for important metadata fields. `FluxSearch` employs a fusion evaluation algorithm that efficiently chooses between the common integration techniques, namely union and join, for properly selected union or join variable pairs from data and metadata. `FluxSearch` eventually returns a ranked list of integrated datasets with the integration process documented in the metadata by `SDTL`.

In the rest of this chapter, we formally define the problems we tackle (§6.2),

introduce the system overview of `FluxSearch` (§6.3), discuss the design of metadata enriched search (§6.4) and data enriched integration (§6.5), evaluate its performance using real data from ICPSR (§6.6), and finalize with the conclusion (§6.7).

6.2 Problem Definition

This section introduces the terminology we use in the rest of the chapter and defines the problems we solve. We first present the notations for terms we use.

Definition VI.1 (Metadata Standard)**.** Metadata standard $S = (S_t, f_S)$ defines the schema of metadata, where S_t is the set of metadata field types defined for describing different aspects of data, and $f_S(t), t \in S_t$ is the ruling function of metadata field type t that constraints how t should be defined in the metadata by a set of rules including where t should be defined, whether t is mandatory or optional, the cardinality of t, the optional or mandatory attributes of t, the value ranges of t's attributes, etc.

In this chapter, we consider semi-structured metadata (e.g. RDF, XML, JSON-based), XML-based in particular, where fields are constrained by parent-to-child relationship such that $t\uparrow$, the parent field type of t, is uniquely determined by t. We denote $t\cdot$ as a sibling field of t with no ordering constraint. Note that some field types may appear in multiple places in the metadata. For such field types, the field type and its attribute values $t^* = (t, A_t)$ uniquely determine its parent field type. For instance, `<labl type="category">` and `<labl type="variable">` are children field types of `<category>` and `<variable>`, respectively.

Metadata is the data that describes data. The data can be a dataset, a data table, a data entry, a variable, ancillary data, etc. Metadata of data can be in different formats or following different standards. The simplest metadata can be a text description of the abstract of the data. Richer metadata may contain more information such as descriptive metadata, technical metadata, administrative metadata, etc.

Here, we consider metadata that follows the same metadata standard only.

Definition VI.2 (Metadata)**.** Metadata $M_S(D) = (M_f, f_M)$ of data D, M for short, following metadata standard S, contains a set of metadata fields M_f defined for (part of) field types in S_t such that $\{type(f) : f \in M_f\} \subset S_t$. M_f is constrained by the ruling function f_M such that the set of rules defined for a field in M_f is more strict if not equal to the rules defined for its field type in the metadata standard S, denoted by $f_S(type(f)) \subset f_M(type(f))$, where $type(f) \in S_t$.

Here we distinguish metadata field type $type(f)$ from metadata field f since metadata field type may have multiple instances of fields in the metadata if the ruling function allows. Fields of the same type can have different sets of attributes with different values.

Definition VI.3 (Dataset)**.** A Dataset $T^S = (R, M, \cdot)$, or $T = (R, M)$, is composed of a set of tables $R = \{R_1, R_2, ...\}$ and metadata $M_S(T)$, or M, describing the dataset following metadata standard S and possibly a set of ancillary materials $\cdot$. Each table R_i contains a list of column variables V_i and a list of row entities E_i. We further define functions $name_{tbl}(R_i)$ for the surface table name of R_i and $name_{var}(v)$ for the surface variable name of $v \in V_i$. We denote $M = M_{dat} \cup M_{tbl} \cup M_{var} \cup M_{ent} \cup M_{etc}$ as the union of metadata at different granularities including dataset-level, table-level, variable-level (for table columns) and entity-level (for table rows) metadata, and others respectively.

Good metadata describing a dataset should contain metadata describing the overall dataset and metadata describing data elements in the datasets at different granularities. Unfortunately, in reality, such good metadata is not feasible and constrained by the metadata standard chosen. DDI, for example, does not have entity-level metadata. Moreover, only a small set of metadata fields is required. Whether to define a large number of optional fields and attributes in the metadata and how to define

them are data-dependent, domain-dependent, depositor-dependent, and data portal-dependent. We assume that the metadata of the datasets we study are valid such that they follow the constraints of the metadata standard defined by the ruling function.

Next, we define the problem of keyword search for integrated datasets and the subproblems we study in later sections.

Definition VI.4 (Keyword search for integrated datasets)**.** Given keyword search query $q = \{w_1, w_2, ..., w_n\}$ and a large set of datasets $\mathcal{T}$ such that each dataset $T = (R, M) \in \mathcal{T}$ has its metadata M following a standard $S \in \mathcal{S}$, a search engine SE returns a ranked list of top-k datasets $\mathcal{T}_k^{Rnk} = [T_1^{Rnk}, T_2^{Rnk}, ..., T_k^{Rnk}]$ ordered by the scoring function $score(\cdot)$ such that $score(T_i) > score(T_j), T_i, T_j \in \mathcal{T}_k^{Rnk}$. A dataset $T \in \mathcal{T}_k^{Rnk}$ is either a dataset from $\mathcal{T}$ or an integrated dataset, denoted by $T \in \mathcal{T} \cup \mathcal{T}^{Int}$. $\mathcal{T}^{Int}$ is the set of integrated datasets derived from tables $\mathcal{R} = \cup_{R \in \mathcal{T}} R$ in $\mathcal{T}$, such that $\mathcal{T}^{Int} = \{generateIntegratedDataset(integrate(\mathcal{R}_x, p)) : \mathcal{R}_x \subset \mathcal{R}\}$, where $p = [\mathcal{R}_{x1} = op_1(\mathcal{R}_x, \cdot), \mathcal{R}_{x2} = op_2(\mathcal{R}_{x1}, \cdot), ...]$ and $op_i \in P_i \cup P_t$. p is the integration plan represented by an ordered list of integration operations in P_i interleaved by data transformation operations in P_t.

In real data portals, a dataset may be described by metadata in multiple standards. The datasets on ICPSR, for instance, have metadata following the Dublin Core standard and the DDI standard, a subset of which is also described by DATS (JSON-based) and DCAT standards. To make our problem more tractable, we assume that all datasets in $\mathcal{T}$ follow the same standard, such that $|\mathcal{S}| = 1$. The integrated dataset is derived from two tables such that $|\mathcal{R}_x| = 2$. The integration operation is either union or join, and the transformation operations are aggregation, select, and crosswalk (discussed in Chapter IV). k is chosen as a reasonable value for user inspection of the datasets in $\mathcal{T}_k^{Rnk}$. The problem is thus defined as

Definition VI.5 (Keyword search for two-way integrated datasets)**.** Given keyword search query $q = \{w_1, w_2, ..., w_n\}$ and a large set of datasets $\mathcal{T}$ such that each dataset

$T \in \mathcal{T}$ has its metadata M following some standard S, a search engine SE_2 returns a ranked list of top-k datasets $\mathcal{T}_k^{Rnk} = [T_1^{Rnk}, T_2^{Rnk}, ..., T_k^{Rnk}]$ ordered by the scoring function $score(\cdot)$ such that $score(T_i) > score(T_j), T_i, T_j \in \mathcal{T}_k^{Rnk}$. $k < \alpha_k$, where α_k defines the upper bound of k. A dataset $T \in \mathcal{T}_k^{Rnk}$ is either a dataset from $\mathcal{T}$ or an integrated dataset, denoted by $T \in \mathcal{T} \cup \mathcal{T}_2^{Int}$. $\mathcal{T}_2^{Int}$ is the set of integrated datasets derived from integrating two tables from all tables in T, denoted by $\mathcal{R} = \cup_{R \in \mathcal{T}} R$, such that $\mathcal{T}_2^{Int} = \{generateIntegratedDataset(integrate(\mathcal{R}_x, p)) : \mathcal{R}_x \subset \mathcal{R}, |\mathcal{R}_x| = 2\}$, where $p = [\mathcal{R}_{x1} = op_1(\mathcal{R}_x, \cdot), \mathcal{R}_{x2} = op_2(\mathcal{R}_{x1}, \cdot), ...]$, such that $op_i \in P_i \cup P_t$ and $\cdot$ is the ancillary information such as metadata. p is the integration plan represented an ordered list of transformation operations in P_t finalized by an integration operation $op_{|p|} \in P_i$ such that $P_t = \{aggregation, select, crosswalk\}$ and $P_i = \{union, join\}$.

To tackle the problem of keyword search for two-way integrated datasets, we propose to enrich metadata with data for better search and enrich data with metadata for better integration. We define the two problems next.

Definition VI.6 (Enrich Data with Metadata)**.** Given a dataset $T = (R, M)$ and a large set of datasets $\mathcal{T}$ from the same source as T, whose metadata follow metadata standard S, $enrich_m(M, R, \mathcal{T}) = M^+$ generates an enriched metadata $M^+ = (M_f^+, f_M^+)$ following metadata standard S such that $M_f \subset M_f^+$ and $f_M \subset f_M^+$.

Leveraging the data and other datasets from the same source as T, $enrich_m(\cdot)$ generates metadata that better describes T by defining some missing fields in the original metadata M. It is also possible to raise an alarm for filled fields where the proposed value of the field is different from its original value for human inspection. Since we aim to automate the system as much as possible, we do not explore the validation use case. In §6.4, we discuss the problem in more depth.

Definition VI.7 (Enrich Metadata with data)**.** Given a dataset $T = (R, M)$ and a large set of datasets $\mathcal{T}$ from the same source as T, whose metadata follow metadata

standard S, $enrich_d(R_i, M, T) = R_i^+, R_i \in R$ generates an enriched data table R_i^+ such that the values of data elements in R_i are true values $R_i^+ = decode(R_i, M, \mathcal{T})$ and the set of variables in R_i is $V_i^+ = V_i \cup latent(R_i, M, \mathcal{T})$, where $decode(\cdot)$ decodes the true value of data in R_i and $latent(\cdot)$ extract latent variables related to R_i from metadata M.

Note that we do not execute $enrich_d(\cdot)$ operation. We are presenting a virtual view of the enriched data table. The integratability of two data tables is evaluated based on their true values and all variables describing the data leveraging summary statistics stored in precomputed union and join indexes. We will give a more detailed explanation in §6.5.

6.3 System Overview

In this section, we show the pipelined workflow of `FluxSearch`, shown in Figure 6.4, broadly introduce the modules involved before diving deeper into major modules next.

For efficient search result generation, `FluxSearch` is divided into two stages: the offline stage for preprocessing in support of the online stage and the online stage for runtime query search. The offline stage takes a large set of datasets with data tables and metadata as input and produces three indexes, namely query index, join index, and union index for efficient online search. Each pair of tables and the metadata of the dataset it belongs to is independently preprocessed first. The table is enriched with metadata related to it as virtual metadata enriched data table containing true values and latent variables. The Messy Group Identification module identifies the structural messiness of the table, which is then passed to the Transformation module to produce tidied data table in canonical format. Variable Role Identification then try to identify measure and dimension variables. Each table and its metadata pass

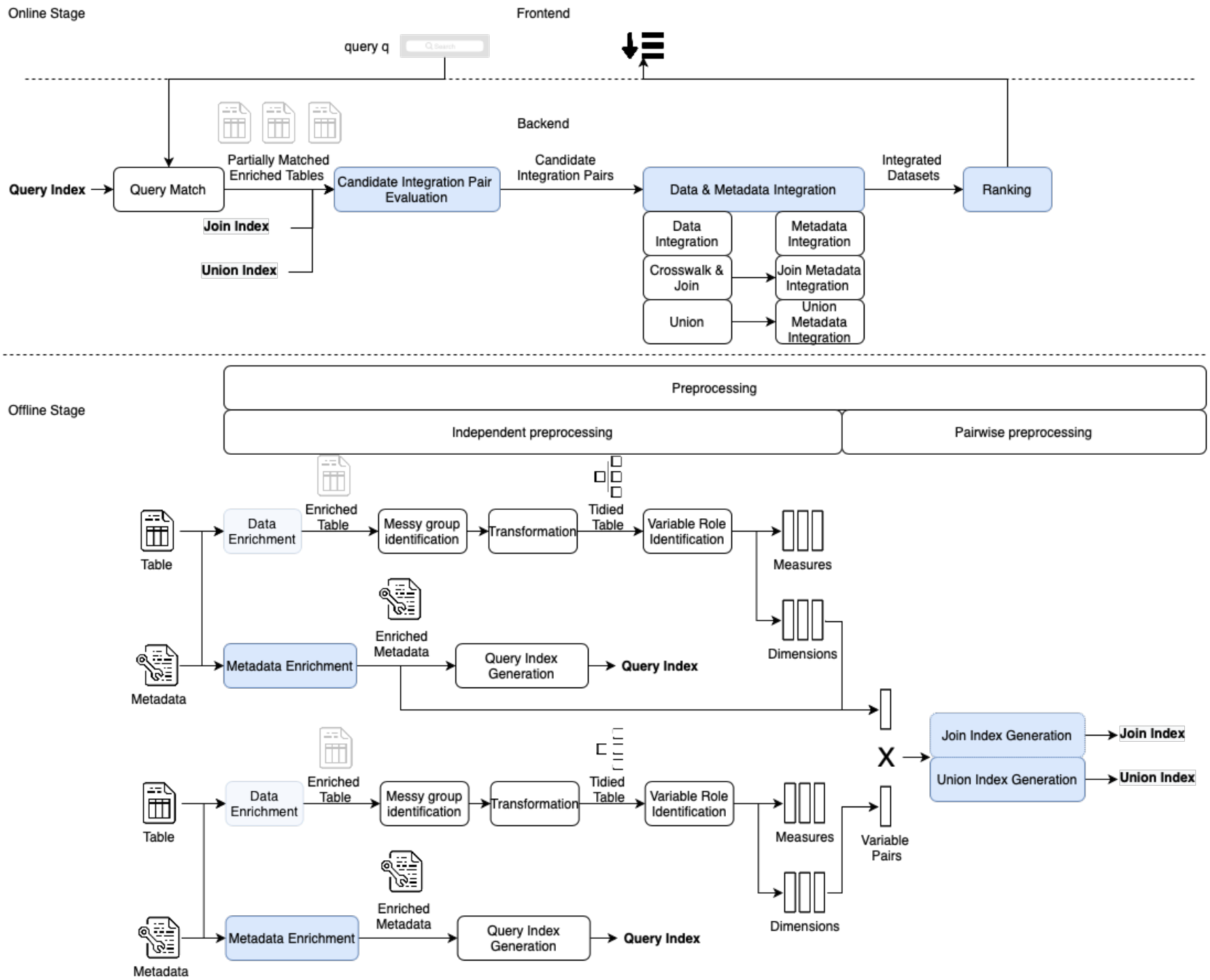


Figure 6.4: `FluxSearch` System Pipeline

through the Metadata Enrichment module to generate data enriched metadata, later used to generate query index for fields in the enriched metadata. After each table is preprocessed individually, enriched tables are compared in pairs. Variable pairs, one from each table in the pair, are evaluated for joinability and unionability in join index Generation and union index Generation modules to produce the join index and the union index, respectively. For a large number of enriched tables, optimization techniques such as blocking approaches have been employed to reduce pairwise comparisons in these two modules.

Both the Independent Preprocessing step and the Pairwise Preprocessing step may run in parallel in a distributed system setting for datasets and dataset pairs, respectively. The offline stage may run for the initial set of datasets. For newly deposited datasets, `FluxSearch` may update the indexes periodically such that each new datasets go through the independent preprocessing step whose resulted variables then pair with variables in other enriched tables for pairwise preprocessing.

The online stage is similar to general dataset search engines composed of a frontend interface where the users enter a keyword-based query in the search bar. The query is then populated to the backend for query result generation, and a ranked list of integrated datasets is returned to the frontend for user exploration. At the backend, the Query Match module matches the search query against the query index to retrieve enriched tables whose enriched metadata matches a subset of terms in the keyword query. These tables are then grouped as candidate integration pairs if they share integratable variable pairs by referring to join index and union index, the union or join of which is scored for relatedness to the keyword query and integration validity. The Ranking module finally returns top-k integrated datasets back to the frontend as the final output.

6.4 Metadata Enrichment for Search

There are a considerable amount of datasets in a data portal, of which the amount of metadata is substantial in size. It takes more effort to generate the metadata than to generate the data for large datasets. A large fraction of the datasets from the old days are published with little to no metadata. The quality of the metadata varies as data portals have limited maintenance over the metadata. In reality, metadata may follow different standards. In this section, we identify a few problems presented in keyword search over such metadata and propose solutions to improve the quality of metadata by enriching metadata by data.

To save the effort of metadata generation, `FluxSearch` aims to automatically enrich metadata by extracting descriptions from data. Given a pool of datasets with metadata, `FluxSearch` selects the important fields to include in metadata, matches the granularity levels between fields in metadata and structural elements in data, and performs updates in metadata for enriched metadata.

6.4.1 Metadata Field Type Importance

Given metadata standard S with a set of field types S_t and a pool of datasets $\mathcal{T}$ with data and metadata following the standard, we claim that not all fields in the metadata are equally important. The importance of metadata fields is data portal-dependent. One field commonly documented in datasets from one data portal may hardly be defined in metadata from another portal. We aim to narrow down metadata fields to a subset of important fields and consider possible metadata enrichment for important fields only.

For well-documented metadata, the presence of field type t is conditional on the presence of its parent field and parent attribute values. We thus define the importance of a field type t to $\mathcal{T}$ as the maximum of the conditional probability of the presence of t on the presence of its parent $t\uparrow$ and the parent's attribute values $A_{t\uparrow}$.

Algorithm 3: getImportantFields

Input: metadata standard tree S with root t_0, a set of metadata $\mathcal{M} = \{M | M \in \mathcal{T}\}$ and importance parameter α_{Imp}
Output: A set of important fields *result*

```
result ← ∅;
if T ≠ ∅ then
    foreach t ∈ t_0.getChildren() do
        ns, ds = {}, {};
        M' ← ∅;
        foreach M ∈ M do
            ds[M.getAttributesValues()] += 1;
            foreach child ∈ M.getChildren() do
                if child.getType() == t then
                    ds[M.getAttributesValues()] += 1;
                    break;
            M' ← M' ∪ M
        imp = 0;
        foreach x ∈ ns.keys() do
            if ns[x]/ds[x] > α^Imp then
                imp = ns[x]/ds[x];
        if imp > α^Imp then
            result.add(t);
            result.add(getImportantFields(t, M', α^Imp));
return result;
```

Definition VI.8 (Metadata Field Type Importance)**.** Given metadata field type $t \in S_t$ from metadata standard $S = (S_t, f_S)$ and a pool of datasets $\mathcal{T}$ whose set of metadata is $\mathcal{M}^{\mathcal{T}}$, the importance of t with respective to $\mathcal{T}$ is

$$\begin{aligned} Imp_{\mathcal{T}}(t) &= max_{x \in A_{t\uparrow}} P(t | t\uparrow, x) \\ &= \sum_{M \in \mathcal{M}^{\mathcal{T}}} \frac{\#\text{ of co-occurrence of } t,\ t\uparrow \text{ and } x \text{ in } M}{\#\text{ of co-occurrence of } t\uparrow \text{ and } x \text{ in } M}. \end{aligned}$$

For an importance threshold $\alpha^{Imp} \in (0, 1]$, We regard field t such that $Imp_{\mathcal{T}}(t) > \alpha^{Imp}$ as an important field. In reality, in some data portals, many of the fields are not included due to lack of documentation even though they are relatively important. We may choose a smaller α^{Imp} for a reasonable set of important fields to consider for enrichment. In Algorithm 3, we show function $getImportantFields()$ for retrieval of

important fields of S for a truncated metadata standard S' including important fields only.

6.4.2 Schema Mapping between Data and Metadata

`FluxSearch` performs a descriptive schema mapping from the schema of datasets $\mathcal{T}$ to the truncated metadata standard S' which includes only important fields. This is similar to the traditional schema mapping but not entirely the same as we are looking for field types in the metadata standard that describes corresponding data elements and data-related statistics. We select a subset of datasets $\mathcal{T}'$ whose metadata fields cover a majority of fields in the truncated metadata standard, constrained by the coverage threshold α^{Cov}.

Instead of simple value-based mapping, we want to accommodate the structural relation between data elements for disambiguity, more specifically the parent-to-child linkage between data elements such as dataset-table, table-variable, table-entity, variable-variable, variable statistics (such as min, max, mean, variance, standard deviation, occurrence, etc.) and entity-entity value statistics. We thus define a simplified DTD schema S_0 for these relationships and translate each data table R in $\mathcal{T}'$ to this schema using $M_0 = translate(D, S_0)$. Basically, we are manually generating metadata based solely on the data available. We then build the mapping from S_0 to S' in a top-down manner leveraging the source-to-target tree binding constrained by the parent-to-child relationship and the sibling relationship. Suppose a field type $t \in S_0$ is mapped to field type $t' \in S'$, a child field type $t\downarrow$ of t is either mapped to some descendant field type of t' or some descendant field type of the root such that the path from the root to the mapped field type is clear of mapped field types. A sibling field type $t\cdot$ of t could map to any field types in S' other than the field types on the path from the root to t nor t's descendants, the path from the root to whom may only contain mapped field types of t's ancestors. The mapping between t and t' is

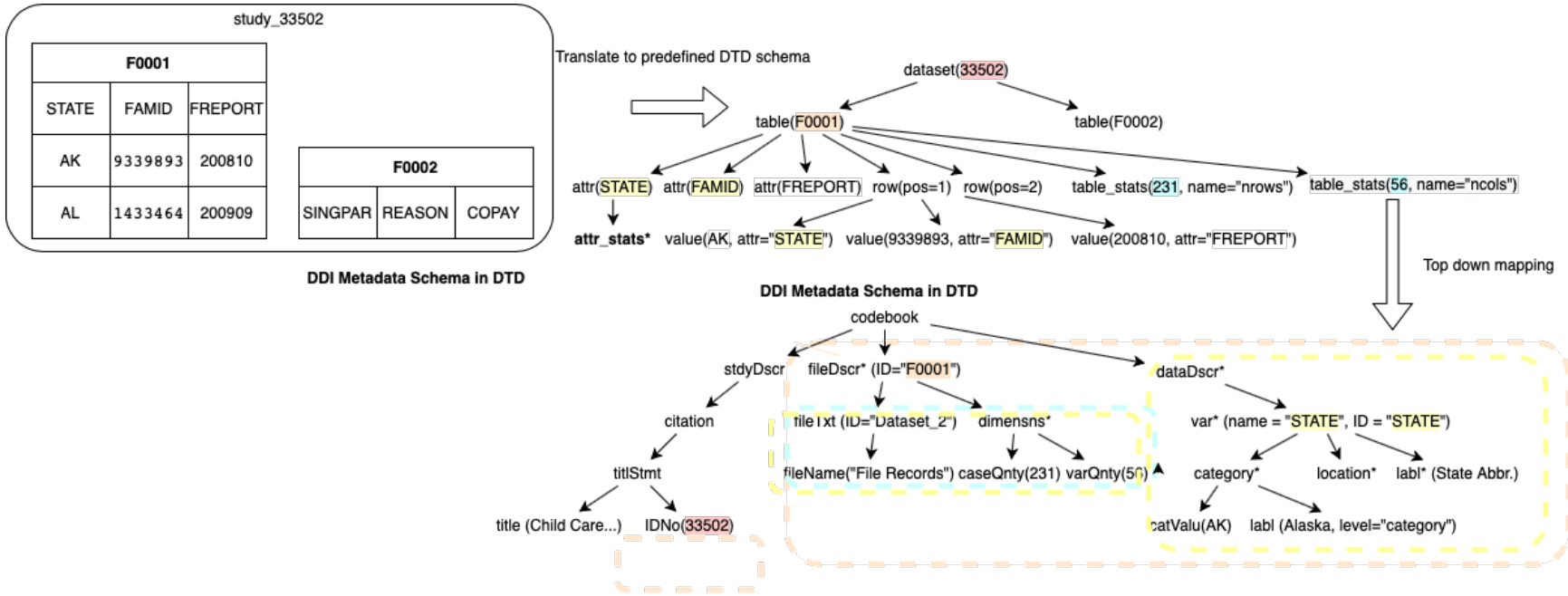


Figure 6.5: An example of schema mapping between data table and metadata

determined by the conjunctive match score of datasets in $\mathcal{T}'$, defined as the expected Jaccard similarity of value matching, denoted by

$$score^{Map}(t, t', M_0, M) = E[\frac{U(t, M_0) \cap U(t', M)}{U(t, M_0) \cup U(t', M)}] \tag{6.1}$$

, where $U(t, M_0)$ is the set of unique values of field type t in M_0 and $U(t', M)$ is the set of unique values of field type t' in M for each dataset $(D, M) \in \mathcal{T}'$. Here we evaluate the syntactic value matching. In more advanced settings, alternative approximated semantic value matching or embedding-based field matching can be adopted.

In Figure 6.5, we present the conversion from data tables in study 33502 to the simplified schema and how it is further mapped to the truncated DDI schema. For the top-down mapping, we start from top-level field `dataset`, considering jointly with other tables in $\mathcal{T}'$ whose data tables are converted to S_0. Say we have 100 datasets, the Jaccard similarity of value matching between dataset type in S_0 and `IDNo` in S' is 0.97, which is the highest among all matchings for `dataset`. Suppose that `dataset` is mapped to `IDNo`, we move on to `table` where two tables F0001 and F0002 exist in the dataset. We repeat the mapping process for the descendants of `IDNo` or other branches of `codebook`. We colored the field and the corresponding mapping search space of the field (dotted area) by the same color. Note that we could not find a mapping for the row, but we could for `table_stats` which may document the number of rows and columns in the table as mapped to `caseQnty` and `varQnty` in DDI schema.

6.4.3 Enrich Metadata with Data

For metadata field types in S' that are important but not documented initially in S_0, we inspect the schema description of the descendant fields under the mapped fields of `table` in S_0. For fields that can be computed from the original data, we define statistical equations and conditional constraints as rules for field value evalua-

tion to enrich metadata. Note that attribute values of some fields depend on domain knowledge of the data, which may not be available without user inspection of ancillary materials in the same dataset. If such ancillary materials are not available, we could not derive a concrete interpretation. For example, a variable with integer values may have attribute `representationType` value equals to `"code"` or `"numeric"`. Without referring to the survey questionnaire for the survey question corresponding to this variable, it is hard to decide the value of `representationType`. Since we are making `FluxSearch` an automatic system and trying to enrich metadata with our best effort, we take advantage of profiling information such as the value distribution of the variable and the values' format for further prediction. Datasets in data portals have portal-dependent characteristics. ICPSR has a large number of datasets from surveys such that integer variables are often encoded categorical variables, whereas dataverse has many datasets from biomedical domains with scientific data such that integer variables have a larger probability of being true values. We thus consider the likelihood of attribute values leveraging the population of attributes in a similar condition from the same data portal.

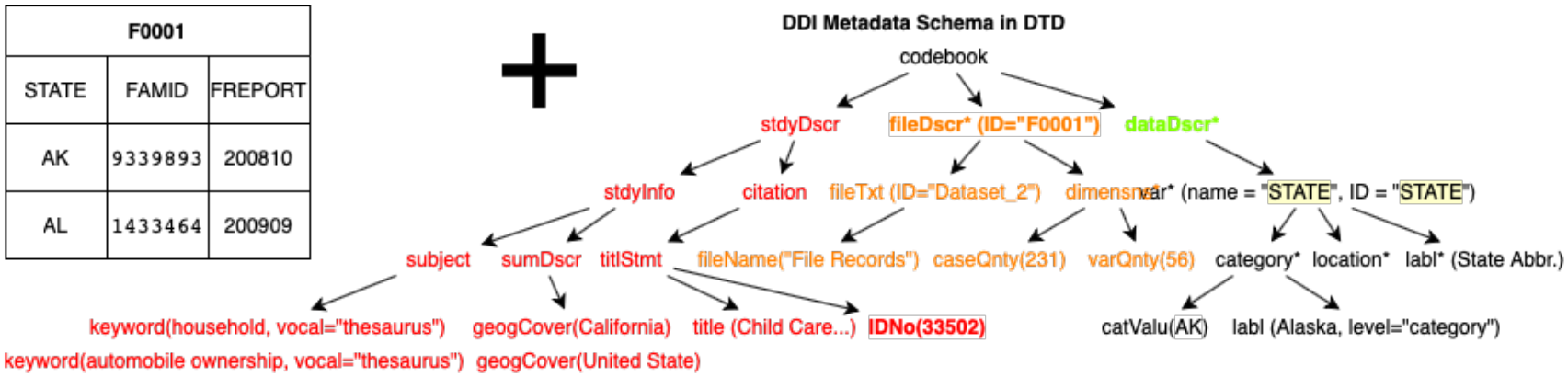
Recall that not all fields are presented in all metadata. Field presence and attribute values in metadata are related to its ancestors' presence and attribute values (mostly the parent), the siblings of the same or different types, and the descendants. `<sumStat type="vald">` and `<sumStat type="invald">`, for instance, are sibling fields of the same type that often occur together as compliments. `<sumStat>` and `<cargry>` are also sibling fields, but of different types that often occur together jointly, providing summary and details information about variable value distribution. We thus regard field presence and their attribute values as the target and the related features as its context and evaluate the co-occurrence of the target and the features by a log-linear regression model for each important field for a large sample of metadata in $\mathcal{T}$. The presence of field is regarded as a binary feature while each

field attribute is a multi-class feature, the number of classes of which depends on the number of values present in the sample and the presence of the attribute. For a large number of features, we use the latent class model, a feature reduction approach from PCA adapted for categorical features, to combine a small set of initial features that captures a large proportion of total variance optimally. Then for each metadata in $\mathcal{T}$, for each important field and its attributes, we test the likelihood of the target with given context features to decide whether to enrich the metadata with the field or the field attribute before performing the actual enrichment.

An important problem to consider is that a field or attribute can be missing due to lack of documentation (the true missing) or not applicable, which we could not tell due to a large amount of lack of documentation in real data portals. To overcome the obstacle, we choose a well-curated sample of metadata for training and validation, assuming no missing metadata fields exist in the training data.

6.5 Data Enrichment for Integration

In a general dataset search system using keyword queries, the query is populated to the backend at online query time and matches with the query index for fast retrieval of datasets relevant to the query. Now with data enriched metadata, a better query index is built for better quality search. While for the search of integrated datasets, these datasets are further evaluated for integration possibility or *integratability*, and the integrated datasets are re-evaluated for their relevance to the query. Instead of performing all the comparisons and evaluation tasks at the online stage, we precompute the join index and the union index for variable pairs from two tables at the offline stage to facilitate online time integration. This section explores the techniques adopted for Join Index Generation and Union Index Generation modules at the offline stage and Candidate Integration Pair Generation and Ranking modules at the online stage.



F0001			dataset-level metadata enriched data			table-level metadata enriched Data
State Abbr.	Family ID	Frequency Reported	stayDscr. srdyInfo. subject. keyword (0.76)	stayDscr. srdyInfo. subject. keyword (0.81)	stdyDscr. citation. titlStmt. title (1.00)	...
Alaska	9339893	200810	household	automobile ownership	Child Care...	
Alabama	1433464	200909	household	automobile ownership	Child Care...	

Figure 6.6: An example of the virtual view of metadata enriched data table

6.5.1 Virtual View of Metadata Enriched Data

As the last modules of the preprocessing step of the offline stage, pairs of tables with enriched metadata are fed to the join index and union index generation modules for a fast online query search. Both modules take pairs of data variables, one from each table in the table pair, as input. In this subsection, we first illustrate how the data tables are virtually enriched with metadata to derive a virtual view of variables with true values and latent variables available before presenting the derivation of the two indexes.

If we look at only data tables in datasets, there are two major problems we identify here during integration. First, since the metadata is describing the data, metadata may contain descriptive information that is not shown in the data. Latent attributes derived from metadata may serve as join keys or unioned with other attributes in other tables for integration. We have also shown in the introduction that it is necessary to identify all possible join keys between data tables for a more statistically meaningful join. It is thus an inevitable task to retrieve latent variables from metadata. Secondly, we find that data encoding is common in real-world datasets for reasons like space saving and privacy protection. All table elements, namely, dataset name, table name, attribute name, and attribute value can be encoded. Without referring to the metadata for their true values, it is not possible to identify whether two variables from two data tables are from the same domain or contain similar values for union or join. In Figure 6.6, we show an example of a dataset, which is synthesized based on a real dataset from ICPSR, including part of the metadata for all levels and table F0001, and the virtual view of the attributes in the metadata enriched table. According to the metadata, we know that dataset 33502 is related to household and automobile ownership according to the `keyword`. This information, however, is not clear when looking at the data only. We know from the metadata that this data table with encoded name F0001 contains summary records according to `fileName`.

Similarly, its attribute names and the values of attribute STATE are also encoded.

To tackle the two problems, we identify latent variables from metadata and decode true values of data elements for a metadata enriched table. Instead of performing the operation physically, we present a virtual view of such a table to facilitate integration next. Intuitively, we regard table-level metadata fields as propagatable to the corresponding table such that these fields can be treated as latent variables to be added to these tables. However, dataset-level metadata fields are not equally propagatable to all data tables in the dataset. Some dataset-level metadata fields are strongly describing some tables in the dataset, while weakly describing others. We define the descriptivity of metadata field to data elements of a table as the correlation between the metadata field value and the values of data elements in the table. Descriptivity measures how likely a metadata field and its value can be added to a table as its variable and values.

Definition VI.9 (Descriptivity)**.** Given a large pool of datasets $\mathcal{T}$, we denote $P^{\mathcal{T}}_{t_f,t_e} = \{(x,y) : x \in M, y \in R_i, R_i \in R, T = (R,M), T \in \mathcal{T}\}$ as the set of pairs of metadata field of type t_f and data element of t_e from the same dataset in $\mathcal{T}$. The descriptivity of a metadata field f to a data element e from the same dataset $T \in \mathcal{T}$ is

$$score^{Dscr}_{\mathcal{T}}(f,e,\mathcal{T}) = \prod_{w_i \in W_f, w_j \in W_e} \frac{p^{\mathcal{T}}_{t_f,t_e}(w_i,w_j)^2}{p^{\mathcal{T}}_{t_f}(w_i)p^{\mathcal{T}}_{t_e}(w_j)}, \tag{6.2}$$

where W_f is the set of words in f and W_e is the set of words in e.

$$p^{\mathcal{T}}_{t_e}(w) = \frac{|\{x : w \in x, x \in P^{\mathcal{T}}_{t_e}\}|}{|P^{\mathcal{T}}_{t_e}|} \text{ and } p^{\mathcal{T}}_{t_y}(w) = \frac{|\{y : w \in y, y \in P^{\mathcal{T}}_{t_y}\}|}{|P^{\mathcal{T}}_{t_y}|} \tag{6.3}$$

such that $P^{\mathcal{T}}_{t_m}$ is the set of metadata fields (or data elements) of type t_m in $P^{\mathcal{T}}_{t_f,t_e}$ where

$m \in \{f, e\}$, and

$$p^{\mathcal{T}}_{t_f,t_e}(w_i, w_j) = \frac{|\{(f, e) : w_i \in f, w_j \in e, (f, e) \in P^{\mathcal{T}}_{t_f,t_e}\}|}{|P^{\mathcal{T}}_{t_f,t_e}|} \tag{6.4}$$

Note that descriptivity is only applicable to (f, e) pair where the granularity of the data that f describes is higher than the granularity of the data element e and f describes e. That is, table-level metadata can describe a variable in the corresponding table but not a variable in another table.

For data enrichment, we only need to compute the descriptivity score for dataset-level metadata types to data tables. We precomputed an offline index for $p^{\mathcal{T}}_{t_f}$ and $p^{\mathcal{T}}_{t_e}$ for a sample of datasets from $\mathcal{T}$ and built an inverted index on it as the descriptivity index.

Recall that in 6.4, we identify metadata fields for different levels. These fields can be viewed as latent variables to enrich data tables in the dataset such that the attribute name is the path from the root to the field, and the attribute values are equivalent to the value of the field. Note that we only care for string-valued fields. Fields with numeric values are mostly used for the documentation of statistics. As for encoded data elements, we decode the true values from corresponding fields based on domain knowledge. In Figure 6.6, we show the virtual view of an enriched data table. We identify `fileName`, `var↓ labl` and `var↓ category↓ labl` as fields containing true value of table names, attribute name and attribute value, respectively. The table is enriched by latent variables from `keyword`, `title` fields. Since there are two dataset-level `<keyword>` fields, we evaluate the descriptivity of their values to F0001 and show the descriptivity score in the enriched columns corresponding to these variables. This table is more related to "automobile ownership" compared to "household." We perform the same evaluation for two `<geoCoverage>` fields.

6.5.2 Generate join index and union index

We define metadata enriched table R^+ of table R with variables $V^+ = V \cup V'$, where V is the set of variables from R and V' is the set of latent variables from the metadata M. For each pair of enriched data tables R_1^+ and R_2^+, we perform the joinability and unionability evaluation for pairs of variables from them during offline preprocessing.

Two variables are unionable if their values are retrieved from the same domain. We consider three types of variable unionability, namely syntactic unionability, semantic unionability, and natural language unionability, based on their syntactic value overlap, semantic value overlap leveraging ontologies, and semantic value similarity even if their syntactic or semantic values do not overlap, respectively. The ensemble unionability takes the highest score computed from the three metrics for the best possible unionability. The definition of table unionability inspires this idea in [79]. Since variables from the same domain may be encoded differently, we refer to the true value of variables in metadata to perform syntactic unionability and semantic unionability evaluation using YAGO ontology. We further perform a natural language unionability evaluation on relevant metadata of variables. We choose to adopt GloVe to generate weighted word embedding vectors for selected metadata fields describing variable v including variable values and related metadata including descendant fields of the corresponding `var` denoted by S^{att}, table-level metadata of the corresponding table R denoted by S^{tbl} and dataset-level fields S^{dat} of the corresponding dataset T, eliminating statistical metadata fields. As not all variables are equally related to a table, especially latent variables, we model the relatedness of a latent variable to the data table as its descriptivity score and use it as the weight for word embedding vector computation.

Similarly, two variables are joinable if they are from the same domain and their values have a large number of overlaps. If variables are syntactically or semantically

unionable, they are syntactically or semantically joinable. We thus use the same metric to evaluate syntactic and semantic joinability and unionability. The ensemble method for joinability evaluation takes the higher score between the two metrics as the output.

Instead of comparing all variable pairs in V_1^+ and V_2^+, we take several optimization techniques to reduce the number of comparisons. We define $V_1'(t)$ as the set of latent variables in V_1' derived from metadata fields of type t in truncated metadata standard S'. Then $V_{1\cap 2}' = \cup_{t\in S', V_1(t)\neq\emptyset, V_2(t)\neq\emptyset} V_1'(t)$ is the set of latent variables in V_1' such that there exists latent variables in V_2 derived from the same type of metadata in S'. The latent variables derived from metadata fields whose type is exclusive to V_1' is denoted by $V_{1\setminus 2}'$. Naturally, latent variables $V_1(t)'$ and $V_2'(t)$ derived for the same type of field t have a one-to-one mapping for domain matching as each field of metadata defines the domain of the corresponding latent variables. Then for variable unionability evaluation, we extract variable pairs from $V_1 \times V_2$, $V_1 \times V_{2\setminus 1}'$ and $V_2 \times V_{1\setminus 2}'$. As for joinability evaluation, we evaluate variable pairs from $V_1 \times V_2$, $V_1 \times V_{2\setminus 1}'$, $V_2 \times V_{1\setminus 2}'$ and $V_{1\setminus 2}' \times V_{2\setminus 1}'$.

For syntactic and semantic value overlap, we evaluate the unionability/joinability using Jaccard Similarity. While for natural language unionability, we use the Cosine similarity of the mean vectors of variables' embedding vectors. To process a large number of variable pairs, we construct an LSH index over minhash of the true values of variables and the class annotations of variables to approximate syntactic and semantic value overlap and filter out variables with few overlaps. For natural language unionability, we apply simhash LSH to variable pairs to find variable pairs with high Cosine similarity. We take data enriched metadata documenting the true values, the cardinality of values, and ancillary information of variables as input for the evaluation, saving the effort of another round of data scan. We may process the three indexes for selected variable pairs in a distributed system and map indexes for the

same variable pairs for ensemble unionability and ensemble joinability computation. We store variable pairs with ensemble unionability and ensemble joinability higher than the union threshold α^{Uno} and join threshold α^{Joi} in union index and join index, respectively.

6.5.3 Candidate Integration Pair Generation

During online query phase, a keyword query $q = \{w_1, w_2, ..., w_n\}$ is populated to the backend Query Match module to first retrieve a set of related data tables $\mathcal{R} = \{R_1, R_2, ...\}$ by $getMatchedTables(q, \mathcal{T})$ ranked by their relevance to the query by $rlvScr(R, q, \mathcal{T})$ for data table $R_i \in \mathcal{R}$ in $\mathcal{T}$. We use a traditional web search engine applying tf-idf for ranking and index generation on metadata enriched data tables such that

$$tf(w_i, R_j) = \frac{\text{frequency of } w \in R_j}{|\{w|w \in R_j\}|} \tag{6.5}$$

$$idf(w_i, \mathcal{R}) = \log \frac{|\mathcal{R}|}{|\{R \in \mathcal{R} : w_i \in R\}|} \tag{6.6}$$

$$tfidf(w_i, R_j, \mathcal{R}) = tf(w_t, R_j) \times idf(w_i, \mathcal{R}). \tag{6.7}$$

Other more advanced techniques for ranking such as [55] can be adopted for keyword search for tables specifically. We will not dive into details since this is not the contribution of our work.

Suppose that $getMatchedTerms(R, q)$ returns the set of keywords that R matches in q as q_i. We want to find top-k integrated data tables. An integrated table can be individual tables $R \in \mathcal{R}$ or a joined or unioned table from two tables $R_1, R_2 \in \mathcal{R}$. Since we aim to find integrated tables matching as many keywords as possible, we formulate this as a Maximum Coverage Problem (MCP) such that we select at most $m = 2$ tables from $\mathcal{R}$ such that the maximum number of keywords in q are covered. Note that during integration, keywords covered in source tables may no longer exist

Algorithm 4: generateIntegratedDatasets

Input: Two data tables X and Y, the integration method $m \in \{U, J\}$, the integration index $I_m \in \{I_u, I_j\}$, the target number of matched term n_0, the number of top integrated datasets k, the ranking threshold α^{Rnk}, the query q and the query index I_q

Output: A list $result$ of lists of the ranking score and integrated dataset $(score, R^+)$ whose index is the number of terms in query q matched in R^+

$result \leftarrow []$;
foreach $h_c \in cAlgns(X, Y, m, I_m, q, I_q)$ **do**
    $score_{upper} = scrUpper(X, Y, h_c, m, I_m, q, I_q)$;
    if $score_{upper} > \alpha^{Rnk}$ *and* $(|result[n_0]| < k$ *or* $score_{upper} >= minScr(result[n_0]))$ **then**
        $n_{0true}, R^+, score = integrate(X, Y, h_c, m, I_m, q, I_q)$;
        $result[n_{0true}].append((score, R)).top(k)$;

in the integrated table. Thus we first approximate the upper bound of the number of keywords covered in an integrated table by the number of keywords in the merged keywords set covered by the pair of tables before computing the true keyword coverage after integration. The MCP problem is NP-hard. It can be approached by a greedy algorithm such that at each stage, choose a data table that contains the largest number of uncovered keywords. We optimize alternatively since we have a small $m = 2$ and an upper bound for the number of result k. Since we are looking for top-k integrated tables, we start by searching for candidate integration pairs whose keyword coverage upper bound is n as $\mathcal{P}_n$, evaluate the true keyword coverage of the integrated table, compute the ranking score of the integrated tables that covers n keywords and add to the rank result ordered by descending ranking score. Next, for candidate integrated pairs whose keyword coverage upper bound is $n-1$, we repeat the previous procedure until k integrated tables are found.

6.5.4 Ranking

After a pair of candidate integration tables $R_1, R_2 \in \mathcal{R}$ are retrieved, we need to decide how to integrate it and evaluate its relevance to the keyword query. For table union and table join, given the unionability/joinability of variable pairs from

two tables computed at the offline stage, we need to determine a one-to-one mapping for a subset of variables among the two tables before performing the integration on these variables.

[79] proposed the idea of c-alignment such that for the set of variables X, Y, there is a one-to-one mapping $h_c : X' \to Y'$ such that $X' \subset X, Y' \subset Y$, and $|X'| = |Y'| = c$. We briefly review their idea here. Since there might be multiple alignment for some c, the set of all c-alignment is denoted by $A^c(X, Y)$. The unionability score of a c-alignment $h_c : X \to Y$ is the joint unionability of variable pairs in the alignment $U(X, Y, h_c) = \prod_{i \in [1,c]} U(X_i, h_c(X_i))$, where $X_i \in X$ and $U(X_i, h(X_i))$ is the variable unionability of X_i and X_i's mapping $Y_j \in Y$. The c-unionability of X, Y, denoted by $U_c^+(X, Y)$, is the maximum unionability for all c-alignment between X and Y.

Similarly, we define the c-joinability of table pairs $J_c^+(X, Y)$ as the maximum joinability of their c-alignment for variable join. Note that for table join, we evaluate the c-joinability on dimension variables only, identified by the Variable Role Identification Module during offline preprocessing. The aligned variables serve as join keys. During the actual join, measure variables are aggregated by join key for each data table before the actual join.

We regard the integrated relevance score of an integrated table as dependent on the validity of union/join and the relevance of the integrated table to the query. We define the *integrated relevance score* of the integrated table to query q covering n_0 words in q as the maximum product of the unionability/joinability m_c and relevance of the integrated table $integrate_c^m$ for all c-alignment h_c of all $c \leq \alpha^c$ through a integration method $m \in \{U, J\}$ as

$$
\begin{aligned}
&score_{n_0,m}^{Rnk}((X, Y), q, \mathcal{T}) = \\
&\qquad max\{m_c(X, Y, h_c) \times score(integrate_c^m(X, Y, h_c), q, \mathcal{T}) : \\
&\qquad |matchTrms(integrate_c^m(X, Y, h_c)| == n_0, c <= \alpha^c, h_c \in H_c\}
\end{aligned}
$$

, where H_c is the set of c-alignment using integration method m and α^c is the upper bound for the number of variables in a c-alignment.

To compute the relevance score for integrated tables, we could easily retrieve the unionability and joinability score of aligned variable pairs from the join index or union index in O(1) time. To save the effort of integrating all c-alignment of a table pair, we approximate its upper bound and only compute the true score when needed. Recall that we are using tf-idf for relevance score evaluation. For a unionable pair of tables, the upper bound of term frequency for a word $w \in q$ in a table unioned from X, Y by c-alignment h_c is $\sum_{i\in[1,c]} tf(w, X_i) + tf(w, h_c(X_i))$. Similarly, for a joinable pair of tables, the upper bound of term frequency is $\sum_{i\in[1,c]} tf(w, X_i) \times tf(w, h_c(X_i))$. Since term frequencies are also available in query index, we can easily compute the upper bound of relevance score of a candidate integration pair and eliminate those whose relevance score is below α^{Rlv}.

6.6 Experiments

Since there is no closely related previous work tackling the problem for datasets with metadata following a well-defined semi-structured schema, we evaluate the performance of `FluxSearch` for the search and integration separately.

6.6.1 The Data

We have crawled 16,191 datasets from ICPSR, among which 11,264 datasets have well-curated dataset- and table-level metadata documented in XML-based DDI format. These are publicly available metadata. We requested variable-level metadata from ICPSR with restricted access due to privacy protection reasons. It covers 5.7 million variables in 6,651 tables from 645 datasets. Each variable-level DDI file contains variable-level metadata for one table. ICPSR has a plan of releasing variable-level metadata, which requires further privacy-related masking. By merging the metadata

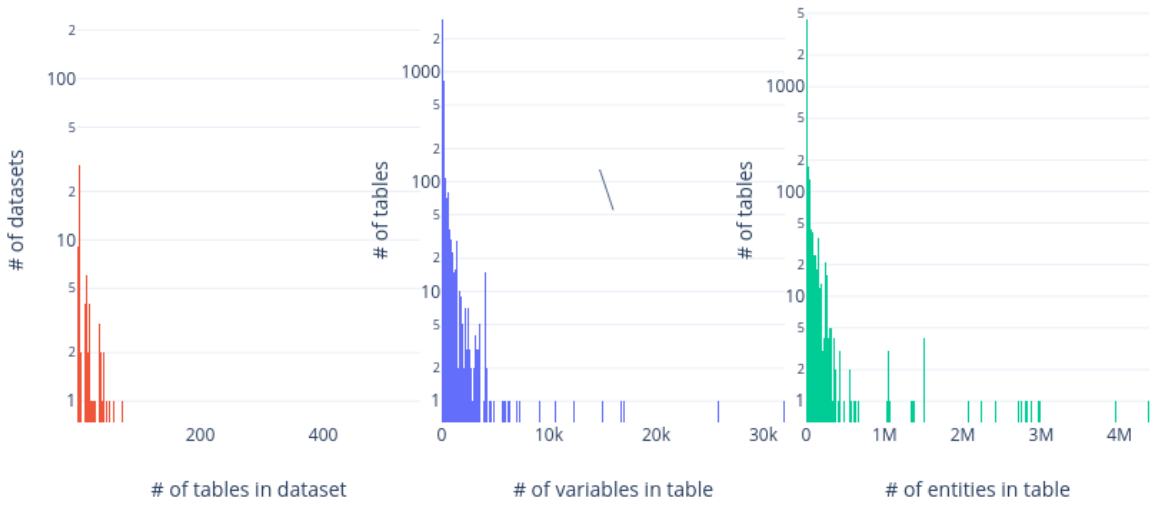


Figure 6.7: Histograms of the number of tables in a dataset (left) and the number of variable in a table (right) for 654 datasets with variable- and table-level metadata

for each dataset, we generated one DDI file per dataset with all metadata information. In Figure 6.7, we show the distribution of the number tables in a dataset and the number of variables (columns) and entities (rows) in a table for 645 datasets with full metadata. For these 645 datasets, a dataset has 8.7 tables on average and a maximum of 413 tables, while a table has 232 variables and 28,102 entities on average and a maximum of 31,941 variables and 17,873,828 entities. If we consider all datasets from ICPSR, the size of a dataset can be over 200GB. For dataset and data tables at this scale, it is hard to manually generate table-level and variable-level metadata manually and curate without automation.

6.6.2 Enriched Data Quality

We specifically evaluate the quality of data enriched metadata which is indexed for searching. We assume that the 645 datasets with all levels of well-curated metadata as the golden standard benchmark, denoted by B^m. Given $\alpha^{Imp} = 0.25$, among 158 field types defined in the metadata of these datasets, `FluxSearch` identified 58 as important. We inspected these fields and defined derivation rules for 34 of them whose values are computed from data and other fields to the best of our knowledge

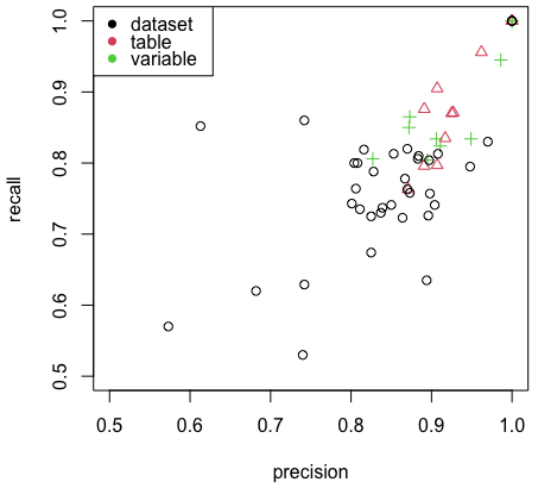


Figure 6.8: Precision vs. Recall for Metadata Enrichment with 10% fields take-out

Table 6.1: Average precision and recall by field take-out ratio

take-out ratio	avg. precision	avg. recall
10%	0.875	0.812
20%	0.867	0.807
30%	0.845	0.793

without referring to other ancillary materials possibly in the datasets. For other data fields that may require domain knowledge or more advanced techniques or more labor efforts to specify, we may send an alarm to the curator for further inspection. Since this is not the contribution of our work, we focus on data-related fields only.

We randomly selected 40% of the datasets as the training data B^m_{train}. To generate the testing data B^m_{test}, for each field type, we randomly selected $10\%, 20\%$ and 30% of all fields of the type in the rest 60% datasets of the benchmark and eliminated these fields from the corresponding metadata. We evaluate whether `FluxSearch` could correctly decide the presence of a field and its attribute value leveraging information of the other datasets and its co-occurrence with other fields, their attributes, and data in terms of precision and recall. In Figure 6.8, we show the precision and recall

Table 6.2: Search queries with relevant datasets returned by ICPSR search engine

Data	Track Name/ID	query
TREC	Federated Web Search Track	detroit riot
		Asian culture
		earthquake
		reinforcement learning
		severed spinal cord
		salmonella
		causes of the cold war
		constitution of italy
		cat movies
		weather in nyc
	Web Track	game theory
		fidel castro
		identifying spider bites
		benefits of running
	Entity Track	NCAA
		Astra Zeneca
		British Royal Monarchy
		United Parcel Service (UPS)
		National Book Foundation
INEX	2007 Ad Hoc iTrack	travelling
		fuel efficient cars
		home heating solar panels
		food additives physical health risk grocery/store labels
	Ad Hoc + Multimedia Track	motor car
		Hurricane satellite image
		classic furniture design chairs

Table 6.3: Relatedness evaluation for 125 queries in benchmark B^q

Search engine	No result	No relevant result	With relevant result (avg. relatedness for top-k results k=3, 5, 10)
$ICPSR_{syn}$	9	98	26(0.590/0.469/0.338)
FluxSearch	6	47	72(0.866/0.789/0.743)

plot for 10% fields taken out. It occurs that there exist fields/variables for all three levels with precision = 1.0 and recall= 1.0. This is intuitive as some fields for these levels always present, for instance, the root field for the three levels. The prediction of variable-level field/attribute is the most accurate, while the prediction of dataset-level field/attribute is the least accurate. Since table-level and variable-level fields often co-occur with other fields of the same type, they are likely to be program generated and thus consistent in presence patterns. This matches what we observe in the data. Dataset-level metadata, however, largely depends on the data collector and the depositor. The collection methods, for instance, are not always deposited together with the data, making it hard for the data curators at ICPSR to define metadata for the corresponding field.

In Table 6.1, we show the precision and recall averaged for all fields by take-out ratio during B_{test} generation. The average precision and recall drop slightly as the take-out ratio increase. Through a careful analysis of the data, we found that the take-out ratio does not affect variable-level and table-level metadata much when the take-out ratio is relatively small, as they are systematically defined. It is much harder to predict dataset-level metadata the definition of which varies by dataset.

6.6.3 Query Result Relatedness

It is not easy to select a set of keyword queries for testing `FluxSearch`. In the ideal case, we can collect such queries from real-world user query logs. With no easy access to such data, we leverage INEX and TREC data widely adopted as golden standards

for information retrieval evaluation. It is not guaranteed that ICPSR datasets and these queries are from the same domain. Many queries may have no results returned. We thus compare the relative performance of a synthesized ICPSR search engine, denoted by $ICPSR_{syn}$ and `FluxSearch`, especially for queries where both return related results for a subset of 5,000 datasets from the crawled datasets smaller than 10GB in sizes. The original ICPSR search engine is built on top of Solr/Lucene such that Solr index is built on metadata and text layer streamed from documentation files in PDF format. In $ICPSR_{syn}$, we built the index on metadata only, eliminating additional information not accessible to `FluxSearch`.

For tracks related to web search and data discovery, we randomly selected 25 queries per track from 3 TREC tracks: Federated Web Search Track, Web Track, and Entity Track; and 2 INEX tracks: 2007 Ad Hoc iTrack and Ad Hoc + Multimedia Track. This benchmark, denoted by B^q, includes 125 queries, which are used for query result relevance evaluation. For each query, we populate the query against the search engine on $ICPSR_{syn}$ and evaluate the relatedness of the top-10 dataset to the query on a 0-1 scale. We have two primary labelers who are experienced with socio-science data and data analytics are involved in scoring the relatedness. When they do not agree on the relatedness of a dataset to a query, the scoring result from a secondary labeler is regarded as the relatedness score. Among 125 queries, 9 of them return no results, 98 return no relevant datasets, while 26 return at least one relevant dataset, among which one query has only three results. In Table 6.2, we show the 26 queries with relevant datasets from $ICPSR_{syn}$ search results.

In Table 6.3, we show the number of queries and the average relatedness score for each case, if not 0. Among 9 queries for which $ICPSR_{syn}$ returns no result, `FluxSearch` returns result for 3 of them. This is contributed by enriching data with metadata as some keywords are matched with data in these datasets once not contained in the metadata. After examining the 6 queries that both engines systems

could not handle, we found that they are either not in English or from a domain not covered by the data available. It is also possible that relevant information is available from ancillary materials of datasets which is not accessible to the search engine yet. These materials can be in heterogeneous formats, including pdf and images, some of which are of low resolution from more than 40 years ago, that can not be processed easily. For queries (98) that $ICPSR_{syn}$ returns no relevant result, `FluxSearch` returns relevant results for more than half of them (51). There are cases where both systems fail. Queries like "images of tsunami", though some of the keywords are covered by a dataset returned, are not related to the semantic meaning of the query. The top results returned by both engines cover "tsunami", "image" or "tsunami image," but are not close to the image of a tsunami. This kind of query needs natural language interpretation or additional constraints on the search.

`FluxSearch` returns relevant results for more queries than $ICPSR_{syn}$ (72:26), and the average relatedness of the returned results is much higher (0.743:0.338) for top-10 results. Metadata enrichment contributes to finding keywords from data for this experiment. For queries like "types of bridges vehicles water ice", `FluxSearch` finds more individual datasets whose data match part of the keywords, "bridges vehicles" and "water ice," for instance. The integration process then generates integrated datasets that cover more keywords in the query. Both steps return more datasets that cover more keywords than $ICPSR_{syn}$.

6.6.4 Efficiency

Additionally, we evaluate the efficiency of offline query index, union index and join index generation, and online time query result generation.

To build a query index, we first enrich metadata with data. Since only 645 out of 11k dataset has variable level metadata, the majority of time for metadata enrichment is spent on reading data (5TB) for variable metadata generation. Instead of

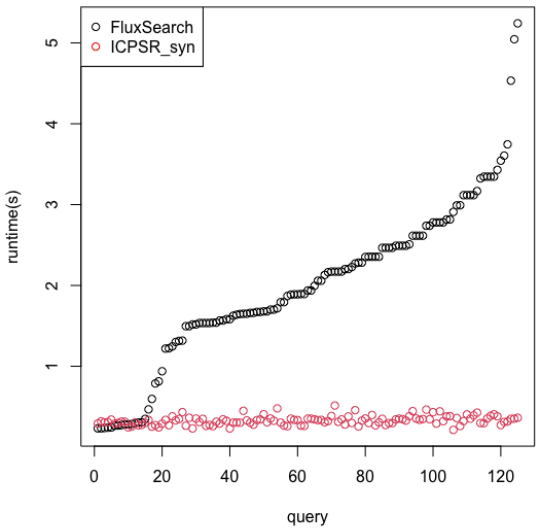


Figure 6.9: Runtime for 125 queries in B_q ordered by ascending runtime of `FluxSearch`

enriching metadata for all variables not documented yet, we enrich for non-numeric variables. We first read the top-N (N=100) value of each variable to predict its type before performing the actual enrichment. We spend seven hours enriching metadata and build the query index in a distributed setting. During offline union index and join index building, we use LSH indices that bucketize variables into highly unionable/joinable pairs. Variables do not appear in the same bucket are assumed to have no unionability/joinability. This can be used as a blocking scheme so that we only compute variable pair unionability/joinability for variables in the same index bucket. We took a similar approach to compute table unionability during the online stage. It took us five hours of parallel computation to compute the union and join index. The majority of time is spent on unionability/joinability distribution estimation using 100,000 randomly selected table pairs with around 3M variable pairs.

During the online stage, $ICPSR_{syn}$ generates top-10 query results for all queries in less than 1s, taking advantage of Lucune of Solr index. The query result generation time of `FluxSearch` varies by query with the mean of 1.971s and the standard

deviation of 1.038s. In Figure 6.9, we depict the runtime efficiency of `FluxSearch` and $ICPSR_{syn}$ ordered by ascending `FluxSearch` runtime. For queries with fewer keywords and a small number of candidate integration pairs, its runtime is comparable to $ICPSR_{syn}$. The majority of time is spent on performing the actual integration and storing the data back to the database.

6.7 Conclusion

In this chapter, we proposed `FluxSearch` as a system performing a keyword-based search for the integrated dataset from a data portal for data with semi-structured metadata following a predefined schema. `FluxSearch` complements metadata by data for the search and complements data by metadata for the integration, saving the effort of manual data integration for finding better data that match the query. Through an experimental evaluation using real socioeconomic datasets with metadata defined in DDI format, we demonstrate the feasibility of using `FluxSearch` for an effective and efficient search of top-k integrated datasets covering as many keywords in the query as possible.

There are several limitations of `FluxSearch`. The quality of metadata is low in the real world for most data portals without portal-side amendment. Many datasets have missing metadata fields that should be defined. Making a training set of datasets with high-quality metadata for metadata field recommendation requires human effort for metadata generation. Also, in metadata enrichment, we reckon the set of important metadata fields defined for the data portal as the scope of metadata fields to consider for enrichment assuming a single-domain data portal. However, for a multi-domain data portal, the fields and attributes in metadata included in a dataset are also domain-dependent, which is not considered in the current implementation of `FluxSearch`. For integrated dataset generation, we currently consider two tables for integration using join or union. In reality, more tables might be integrated leveraging

both join, union, and other transformations. Lastly, we find that some datasets have multiple tables with identical schema collected for entities of the same dimensions, which is often reflected in table titles. These tables can be easily unioned as a long table. There are also wide tables chunked for ease of storage as smaller tables in a dataset. Such in-dataset table relationship is not considered in `FluxSearch`.

CHAPTER VII

Conclusion

In this thesis, we introduced the problem of the search and use of socio-economic data for users with limited technical proficiency. We presented `SDTL/SDTA` and `C`^{2}`Metadata` to improve metadata quality for better search in a hands-off manner. By defining a standard representation for statistical data transformation in `SDTL/SDTA`, we developed `C`^{2}`Metadata` that automatically integrates data transformation information in the metadata. For the use of datasets, we presented `GeoAlign`, `GeoFlux`, and `FluxSearch` to permit the search of integrated datasets. `GeoAlign` algorithm and `GeoFlux` system lay the foundation of automatic data integration leveraging geographical information. On top of these works, `FluxSearch` offers keyword query search over datasets integrated by join or union. The work presented in this thesis contributes to making datasets more viable to users looking to find more and better datasets without user intervention. It aims to shrink the gap between data and the general public to make data more usable.

7.0.1 Future Work

There are several possible directions to pursue as future steps of this thesis. To make dataset search and use more applicable for users with little to no technical proficiency, we expect *more high-quality* datasets with metadata that could satisfy user

expectations. Here we identify three directions we consider valuable in this sense: i) to present a unified view for datasets across different sources by standardizing both data and metadata, ii) to promote high-quality metadata generation minimizing human effort in the tedious process, and iii) to extend `FluxSearch` with a human-in-the-loop interactive integration feature. All three directions tackle different perspectives of effort-saving dataset search and use as they consider the scope, the quality, and the understanding of datasets for both dataset creators and dataset explorers.

Similar to unified data in the database, we find it necessary to create a unified view of metadata and data for datasets from different sources to make more datasets searchable. Instead of schema, metadata of datasets follow diverse metadata standards with a defined thesaurus. The choice of the metadata standard to follow depends on the community of the data collection and the preference of the data creators. It also depends on the community of the data collection and the preference of the data creators. Many socio-science data repositories have adopted the Data Documentation Initiative (DDI) standard for microdata, and statistical organizations use Statistical Data and Metadata eXchange (SDMX), which is more suited to aggregate data. Ecological Metadata Language (EML) is one of several standards used for ecological research data, and the biomedical community has several standards, including Observational Medical Outcomes Partnership (OMOP) and Fast Healthcare Interoperability Resources (FHIR). They each define a different set of fields. Also, different metadata standards have different values, named as thesaurus, for the same or similar fields. The United States in the thesaurus of `CollectionLevel` field of one standard may be represented by USA in the thesaurus of `GeographicCoverage` of another standard. Though the two fields may represent a similar set of real-world entities in their respective thesaurus, their syntactic representations are different. Even if they are semantically similar, in the data, they may be coded differently. Without referring to the metadata, one could hardly find the true value. In addition to applying existing

techniques from schema mapping and entity matching, leveraging the definition of metadata standards and their hierarchical field structure could better promote the mapping and matching quality. With the popularization of the standardization of statistical data transformation using `SDTL` and `SDTA`, we realize the urge to define a unified standard for metadata and data standards and a conversion mechanism to convert datasets represented in one standard to another. Datasets following different standards could thus be virtually or physically transformed to the standard to broaden the scope of datasets to search.

With a unified view of the datasets, we then consider the generation of datasets following the standard. Currently, data creators are the main contributor to the metadata. However, in many circumstances, they are passively generating metadata per requests of data sharing for publication. For data hubs with no metadata validation, such as dataverse, the quality of metadata is very low. For data hubs with metadata verification mechanisms, such as ICPSR, data processors from the data hub is manually viewing the deposited data and complementary materials to generate the metadata before asking the data depositor to verify the correctness of the metadata. There are different teams of data processors responsible to fill in the information of different fields. For instance, some data curators classify the topics of the datasets while some others look up related publications. The process can take up to weeks to fully describe a dataset with high-quality metadata. Our work of `C`^{2}`Metadata` has automated the generation of data transformation in metadata, the other parts of the metadata, however, remain manually generated. We consider the development of a dataset deposit interface for data depositors with metadata field value recommendation and data value standardization so that data depositors could verify the metadata and data during deposit in an interactive setting. We identify error detection and privacy protection for data preparation, field tagging, and bibliography generation as four major steps of such an interface. Each of them leverages different learning

techniques based on past rules/values defined for these steps.

Lastly, given more and better datasets for search, how can we manipulate these datasets for more valuable results through integration? `FluxSearch` eliminates humans in the process as we consider users as the general public without much technical background. Also, we are considering data integration for unionable or joinable datasets. Richer massaged datasets can be obtained by the union and join mixture integration for more than two datasets. There are many ways of combining union and join for candidate integration datasets, the choice of which can be system decision or user decision depending on user intention. Given the user feedback stream, we could model user preference by a probabilistic model and learn to optimize in candidate integration dataset generation and integration rank scoring.

BIBLIOGRAPHY

[1] G. Alter, D. Donakowski, J. Gager, P. Heus, C. Hunter, S. Ionescu, J. Iverson, H. Jagadish, C. Lagoze, J. Lyle, et al. Provenance metadata for statistical data: An introduction to structured data transformation language (sdtl). *IASSIST Quarterly*, 44(4), 2020.

[2] S. Auer, L. Bühmann, C. Dirschl, O. Erling, M. Hausenblas, R. Isele, J. Lehmann, M. Martin, P. N. Mendes, B. van Nuffelen, C. Stadler, S. Tramp, and H. Williams. Managing the Life-Cycle of Linked Data with the LOD2 Stack. In *The Semantic Web – ISWC 2012*, pages 1–16. Springer, Berlin, Heidelberg, Nov. 2012.

[3] B. Bajat, N. Krunić, and M. Kilibarda. Dasymetric mapping of spatial distribution of population in timok region. In *Proceedings of International conference Professional practice and education in geodesy and related fields, Klavodo-Djerdap, Serbia*, 2011.

[4] K. Balog, E. Meij, M. De Rijke of the 3rd international semantic search, and 2010. Entity search: building bridges between two worlds. *dl.acm.org*.

[5] Z. Bellahsene, A. Bonifati, E. Rahm, et al. *Schema matching and Mapping*, volume 20. Springer, 2011.

[6] bioCADDIE. Datamed biomedical data search engine. `https://datamed.org/`, 2018.

[7] A. Bonifati and Y. Velegrakis. Schema matching and mapping: From usage to evaluation. In *EDBT/ICDT*, pages 527–529. ACM, 2011.

[8] P. A. Burrough, R. McDonnell, R. A. McDonnell, and C. D. Lloyd. *Principles of geographical information systems*. Oxford University Press, 2015.

[9] M. J. Cafarella, A. Halevy, and N. Khoussainova. Data integration for the relational web. *Proc. VLDB Endow.*, 2(1):1090–1101, Aug. 2009. ISSN 2150-8097.

[10] U. S. Census. 2010 census data [data file]. Available from `https://www.census.gov/2010census/data/`, 2010. Accessed: 2014-08-14.

[11] A. Chapman, E. Simperl, L. Koesten, G. Konstantinidis, L.-D. Ibáñez-Gonzalez, E. Kacprzak, and P. Groth. Dataset search: a survey. *arXiv.org*, Jan. 2019.

[12] X. Chu, Y. He, K. Chakrabarti, and K. Ganjam. TEGRA - Table Extraction by Global Record Alignment. *SIGMOD Conference*, 2015.

[13] S.-L. Chuang, K. C.-C. Chang, and C. Zhai. Context-Aware Wrapping - Synchronized Data Extraction. *VLDB*, pages 699–710, 2007.

[14] I. Circle Systems. Stat/transfer (version 14), 2015. URL `https://stattransfer.com`.

[15] Colectica. Convention-based ontology generation system (cogs) 1.0. http://cogsdata.org/docs/, 2017.

[16] M. Craglia and H. Onsrud. *Geographic Information Research: Transatlantic Perspectives*. Taylor & Francis, 2004. ISBN 9780203211397.

[17] C. Dalzell. Calling r from spss, 2013. URL https://developer.ibm.com/tutorials/ba-call-r-spss/.

[18] A. Das Sarma, L. Fang, N. Gupta, A. H. P. o. the, and 2012. Finding related tables. *dl.acm.org*.

[19] N. O. Data. Nys attorney registrations (attorney registration) [data file]. Retrieved from https://data.ny.gov/Transparency/NYS-Attorney-Registrations/eqw2-r5nb, 2013. Accessed: 2017-05-01.

[20] N. O. Data. Facilities licensed by the department of motor vehicles (dmv license facilities) [data file]. Retrieved from https://data.ny.gov/Transportation/Facilities-Licensed-by-the-Department-of-Motor-Veh/nhjr-rpi2, 2013. Accessed: 2017-05-01.

[21] N. O. Data. Food service establishment inspections: Beginning 2005 (active) (food service inspections) [data file]. Retrieved from https://health.data.ny.gov/Health/Food-Service-Establishment-Inspections-Beginning-2/2hcc-shji, 2013. Accessed: 2014-08-14.

[22] N. O. Data. Liquor authority quarterly list of active licenses (liquor licenses) [data file]. Retrieved from https://data.ny.gov/Economic-Development/Liquor-Authority-Quarterly-List-of-Active-Licenses/hrvs-fxs2, 2013. Accessed: 2014-08-14.

[23] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[24] H.-H. Do and E. Rahm. Coma: A system for flexible combination of schema matching approaches. In *VLDB*, pages 610–621. VLDB Endowment, 2002.

[25] C. L. Eicher and C. A. Brewer. Dasymetric mapping and areal interpolation: Implementation and evaluation. *Cartography and Geographic Information Science*, 28(2):125–138, 2001.

[26] C. L. Eicher and C. A. Brewer. Dasymetric mapping and areal interpolation: Implementation and evaluation. *Cartography and Geographic Information Science*, 28(2):125–138, 2001.

[27] Elsevier. scientific repository. https://datasearch.elsevier.com/, 2018.

[28] R. Flowerdew and M. Green. *Developments in areal interpolation methods and GIS*, pages 73–84. Springer Berlin Heidelberg, Berlin, Heidelberg, 1993. ISBN 978-3-642-77500-0.

[29] R. Flowerdew, M. Green, and S. Fotheringham. Areal interpolation and types of data. *Spatial analysis and GIS*, 121:145, 1994.

[30] C. Franklin. An introduction to geographic information systems: Linking maps to databases. *Database*, 15(2):12–21, Apr. 1992. ISSN 0162-4105.

[31] E. A. Gallery. Starbucks [map]. Retrieved from `https://services.arcgis.com/nzS0F0zdNLvs7nc8/arcgis/rest/services/Starbucks/FeatureServer`, 2014. Accessed: 2017-10-02.

[32] E. A. Gallery. Accidents reported to national highwat traffic safety administration in 2011 (accidents) [map]. Retrieved from `http://services.arcgis.com/0TU5BETrBlnIhvOx/ArcGIS/rest/services/NHTSAAccidents2011/FeatureServer`, 2017. Accessed: 2017-10-02.

[33] E. A. Gallery. Usa cemeteries (cemeteries) [map]. Retrieved from `http://www.arcgis.com/home/item.html?id=5b08fa8bb5a64ea7848dc5188e47994a`, 2017. Accessed: 2017-10-02.

[34] E. A. Gallery. Usa public buildings (public buildings) [map]. Retrieved from `http://www.arcgis.com/home/item.html?id=d5d5b513a40145ffa60b67d9c7ab9680`, 2017. Accessed: 2017-10-02.

[35] E. A. Gallery. Us shopping centers 2015 (shopping centers) [map]. Retrieved from `https://services1.arcgis.com/6kyLQ3wRvoPKn52I/arcgis/rest/services/US_Shopping_Centers_2015/FeatureServer`, 2017. Accessed: 2017-10-02.

[36] E. A. Gallery. Usa uninhabited places [map]. Retrieved from `http://www.arcgis.com/home/item.html?id=5f0a5776cbaf4b34b9600809bf791d69`, 2017. Accessed: 2017-10-02.

[37] L. Getoor and A. Machanavajjhala. Entity resolution: theory, practice & open challenges. *Proceedings of the VLDB Endowment*, 5(12):2018–2019, 2012.

[38] B. Glavic and K. R. Dittrich. Data provenance: A categorization of existing approaches. In *BTW*, volume 7, pages 227–241, 2007.

[39] M. F. Goodchild, N. S. N. Lam, and U. of Western Ontario. Dept. of Geography. *Areal interpolation: a variant of the traditional spatial problem*. London, Ont.: Department of Geography, University of Western Ontario, 1980.

[40] M. F. Goodchild, L. Anselin, and U. Deichmann. A framework for the areal interpolation of socioeconomic data. *Environment and planning A*, 25(3):383–397, 1993.

[41] Google. Blog: Google dataset search. https://developers.google.com/search/docs/data-types/dataset, 2018.

[42] I. Gregory. The accuracy of areal interpolation techniques: standardising 19th and 20th century census data to allow long-term comparisons. *Computers, Environment and Urban Systems*, 26(4):293–314, 7 2002. ISSN 0198-9715.

[43] S. Gupta, P. Szekely, C. A. Knoblock, A. Goel, M. Taheriyan, and M. Muslea. Karma: A system for mapping structured sources into the semantic web. In *Extended Semantic Web Conference*, pages 430–434. Springer, 2012.

[44] E. Haghish. Seamless interactive language interfacing between r and stata. *The Stata Journal*, 19(1):61–82, 2019.

[45] S. Hahmann, D. Burghardt, and B. Weber. "80% of all information is geospatially reference" towards a research framework: Using the semantic web for (in) validating this famous geo assertion.

[46] A. Halevy, A. Rajaraman, and J. Ordille. Data integration: The teenage years. In *VLDB*, pages 9–16. VLDB Endowment, 2006.

[47] A. Halevy, A. Rajaraman, and J. Ordille. Data integration: the teenage years. In *Proceedings of the 32nd international conference on Very large data bases*, pages 9–16. VLDB Endowment, 2006.

[48] T. Heath and C. Bizer. Linked Data - Evolving the Web into a Global Data Space. *Synthesis Lectures on the Semantic Web*, 2011.

[49] M. A. Hernández, R. J. Miller, and L. M. Haas. Clio: A semi-automatic tool for schema mapping. *ACM SIGMOD Record*, 30(2):607, 2001.

[50] P. Heyvaert, P. Colpaert, R. Verborgh, E. Mannens, and R. Van de Walle. Merging and enriching dcat feeds to improve discoverability of datasets. In *European Semantic Web Conference*, pages 67–71. Springer, 2015.

[51] E. Hovy, J. L. Ambite, and A. Philpot. Addressing a bottleneck in data integration using automated learning techniques.

[52] I. D. C. (IDC). Worldwide semiannual big data and analytics spending guide. https://www.idc.com/getdoc.jsp?containerId=IDC_P33195, 2018.

[53] E. Inc. Arcgis pro 2.0 [computer software]. Available from https://pro.arcgis.com/en/pro-app/, 2017.

[54] S. I. Inc. Calling functions in the r language, 2016. URL https://support.sas.com/rnd/app/studio/statr.pdf.

[55] C. J, HalevyAlon, and KhoussainovaNodira. Data integration for the relational web. *Proceedings of the VLDB Endowment*, Aug. 2009.

[56] Z. Jin, M. R. Anderson, M. J. Cafarella, and H. V. Jagadish. Foofah - Transforming Data By Example. *SIGMOD Conference*, 2017.

[57] T. Joachims. Training linear SVMs in linear time. *KDD*, 2006.

[58] S. Kandel, A. Paepcke, J. M. Hellerstein, and J. Heer. Wrangler - interactive visual specification of data transformation scripts. *CHI*, 2011.

[59] K. Kemp. *Encyclopedia of geographic information science*. Sage, 2008.

[60] H. Köpcke and E. Rahm. Frameworks for entity matching: A comparison. *Data & Knowledge Engineering*, 69(2):197–210, 2010.

[61] N. S.-n. Lam. An evaluation of areal interpolation methods. In *Proceedings, Fifth International Symposium on Computer-Assisted Cartography (AutoCarto 5)*, volume 2, pages 471–479, 1982.

[62] N. S.-N. Lam. Spatial interpolation methods: A review. *The American Cartographer*, 10(2):129–150, 1983.

[63] M. Langford. Obtaining population estimates in non-census reporting zones: An evaluation of the 3-class dasymetric method. *Computers, environment and urban systems*, 30(2):161–180, 2006.

[64] M. Langford, D. J. Maguire, and D. J. Unwin. The areal interpolation problem: estimating population using remote sensing in a gis framework. *Handling geographical information: Methodology and potential applications*, pages 55–77, 1991.

[65] M. Lenzerini. Data integration: A theoretical perspective. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 233–246. ACM, 2002.

[66] H. LI. A Short Introduction to Learning to Rank. *IEICE TRANSACTIONS on Information and Systems*, E94-D(10):1854–1862, Oct. 2011.

[67] A. Liaw and M. Wiener. Classification and regression by randomforest.

[68] LiJian and DeshpandeAmol. Ranking continuous probabilistic datasets. *Proceedings of the VLDB Endowment*, Sept. 2010.

[69] X. H. Liu, P. C. Kyriakidis, and M. F. Goodchild. Population-density estimation using regression and area-to-point residual kriging. *Int. J. Geogr. Inf. Sci.*, 22 (4):431–447, Jan. 2008. ISSN 1365-8816.

[70] F. Maali, J. Erickson, and P. Archer. Data catalog vocabulary (dcat). *W3c recommendation*, 16, 2014.

[71] I. R. Mansuri and S. Sarawagi. Integrating Unstructured Data into Relational Databases. *ICDE*, 2006.

[72] J. Markoff and G. Shapiro. The linkage of data describing overlapping geographical units. *Historical Methods Newsletter*, 7(1):34–46, 1973.

[73] H. A. Maurer and R. Mehmood. Merging image databases as an example for information integration. *CEJOR*, 23(2):441–458, 2015.

[74] J. Mennis and T. Hultgren. Intelligent dasymetric mapping and its application to areal interpolation. *Cartography and Geographic Information Science*, 33(3): 179–194, 2006.

[75] R. Mihalcea, C. Corley, and C. Strapparava. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*, pages 775–780, 2006.

[76] L. Mitas and H. Mitasova. Spatial interpolation. *Geographical information systems: principles, techniques, management and applications*, 1:481–492, 1999.

[77] A. S. Mugglin, B. P. Carlin, and A. E. Gelfand. Fully model-based approaches for spatially misaligned data. *Journal of the American Statistical Association*, 95(451):877–887, 2000.

[78] D. Murakami and M. Tsutsumi. A new areal interpolation method based on spatial statistics. *Procedia-Social and Behavioral Sciences*, 21:230–239, 2011.

[79] F. Nargesian, E. Zhu, K. Q. Pu, and R. J. Miller. Table union search on open data. *Proceedings of the VLDB Endowment*, 11(7):813–825, Mar. 2018.

[80] S. Neumaier, J. Umbrich, and A. Polleres. Automated quality assessment of metadata across open data portals. *Journal of Data and Information Quality (JDIQ)*, 8(1):2, 2016.

[81] O. of Policy Development and R. P. . R). Hud usps zip code crosswalk files [data file]. Available from `https://www.huduser.gov/portal/datasets/usps_crosswalk.html`, 2010. Accessed: 2014-08-14.

[82] T. J. Parr and R. W. Quong. Antlr: A predicated-ll (k) parser generator. *Software: Practice and Experience*, 25(7):789–810, 1995.

[83] R. Pimplikar and S. Sarawagi. Answering table queries on the web using column keywords. *Proceedings of the VLDB Endowment*, 5(10):908–919, 2012.

[84] L. L. Pipino, Y. W. Lee, and R. Y. Wang. Data quality assessment. *Communications of the ACM*, 45(4):211–218, 2002.

[85] L. Qian, M. J. Cafarella, and H. V. Jagadish. Sample-driven schema mapping. In *SIGMOD*, pages 73–84, New York, NY, USA, 2012. ACM.

[86] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, 2001. ISSN 0949-877X.

[87] E. Rahm and H. H. Do. Data cleaning: Problems and current approaches. *IEEE Data Engineering Bulletin*, 23:2000, 2000.

[88] V. Raman and J. M. Hellerstein. Potter's Wheel - An Interactive Data Cleaning System. *VLDB*, 2001.

[89] M. Reibel and M. E. Bufalino. Street-weighted interpolation techniques for demographic count estimation in incompatible zone systems. *Environment and Planning A*, 37(1):127–139, 2005.

[90] Y. Sadahiro. Accuracy of count data estimated by the point-in-polygon method. *Geographical Analysis*, 32(1):64–89, 2000. ISSN 1538-4632.

[91] SDMX. Validation and transformation language (vtl), 2018. URL `https://sdmx.org/?page_id=5096`.

[92] F. Sidi, P. H. S. Panahy, L. S. Affendey, M. A. Jabar, H. Ibrahim, and A. Mustapha. Data quality: A survey of data quality dimensions. In *Information Retrieval & Knowledge Management (CAMP), 2012 International Conference on*, pages 300–304. IEEE, 2012.

[93] M. Sokolova and G. Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427 – 437, 2009.

[94] G. Q. Tabios and J. D. Salas. A comparative analysis of techniques for spatial interpolation of precipitation. *JAWRA Journal of the American Water Resources Association*, 21(3):365–380, 1985. ISSN 1752-1688.

[95] J. Tennison. CSV on the web: A primer. `Http://www.w3.org/TR/2016/NOTE-tabular-data-primer-20160225/`, 2016.

[96] P. Thomas, R. Omari, and T. Rowlands. Towards searching amongst tables. In *Proceedings of the 20th australasian document computing symposium*, page 8. ACM, 2015.

[97] W. R. Tobler. Smooth pycnophylactic interpolation for geographical regions. *Journal of the American Statistical Association*, 74(367):519–530, 1979.

[98] J. Umbrich, S. Neumaier, and A. Polleres. Quality assessment and evolution of open data portals. In *2015 3rd International Conference on Future Internet of Things and Cloud*, pages 404–411. IEEE, 2015.

[99] United States Census Bureau. Geographic Relationships Between Entity Types [Data File]. Available at `https://www.census.gov/geo/Maps-data/data/relationship.html`.

[100] G. Van, de RijkeMaarten, and KanoulasEvangelos. Neural Vector Spaces for Unsupervised Information Retrieval. *ACM Transactions on Information Systems (TOIS)*, June 2018.

[101] P. R. Voss, D. D. Long, and R. B. Hammer. *When census geography doesn't work: Using ancillary information to improve the spatial interpolation of demographic data*.

[102] R. Y. Wang and D. M. Strong. Beyond Accuracy - What Data Quality Means to Data Consumers. *J. of Management Information Systems*, 1996.

[103] H. Wickham. Reshaping data with the reshape package. *Journal of Statistical Software*, 21(12):1–20, 2007. URL `http://www.jstatsoft.org/v21/i12/`.

[104] H. Wickham. Tidy data. *The Journal of Statistical Software*, 59, 2014. URL `http://www.jstatsoft.org/v59/i10/`.

[105] H. Wickham. *tidyverse: Easily Install and Load the 'Tidyverse'*, 2017. URL `https://CRAN.R-project.org/package=tidyverse`. R package version 1.2.1.

[106] H. Wickham, R. François, L. Henry, and K. Müller. *dplyr: A Grammar of Data Manipulation*, 2018. URL `https://CRAN.R-project.org/package=dplyr`. R package version 0.7.6.

[107] J. K. Wright. A method of mapping densities of population: With cape cod as an example. *Geographical Review*, 26(1):103–110, 1936.

[108] Y. Xie. The overlaid network algorithms for areal interpolation problem. *Computers, environment and urban systems*, 19(4):287–306, 1995.

[109] M. Yakout, K. Ganjam, K. C. P. o. t. 2012, and 2012. Infogather: entity augmentation and attribute discovery by holistic matching with web tables. *dl.acm.org*.

[110] S. Zhang and K. Balog. Ad hoc table retrieval using semantic similarity. In *Proceedings of the 2018 World Wide Web Conference*, pages 1553–1562. International World Wide Web Conferences Steering Committee, 2018.

[111] S. Zhang, K. B. P. o. t. . w. w. w. conference, and 2018. Ad hoc table retrieval using semantic similarity. *dl.acm.org*, .

[112] S. Zhang, K. B. P. o. t. t. I. A. SIGIR, and 2017. Entitables: Smart assistance for entity-focused tables. *dl.acm.org*, .

[113] E. Zhu, D. D. 0001, F. Nargesian, and R. J. Miller. JOSIE - Overlap Set Similarity Search for Finding Joinable Tables in Data Lakes. *SIGMOD Conference*, 2019.

[114] P. Ziegler and K. R. Dittrich. *Conceptual Modelling in Information Systems Engineering*, chapter Data Integration — Problems, Approaches, and Perspectives, pages 39–58. Springer, 2007. ISBN 978-3-540-72677-7.