An Ontology of Analytic Knowledge and Processes for Automated Data Science

Anonymous submission

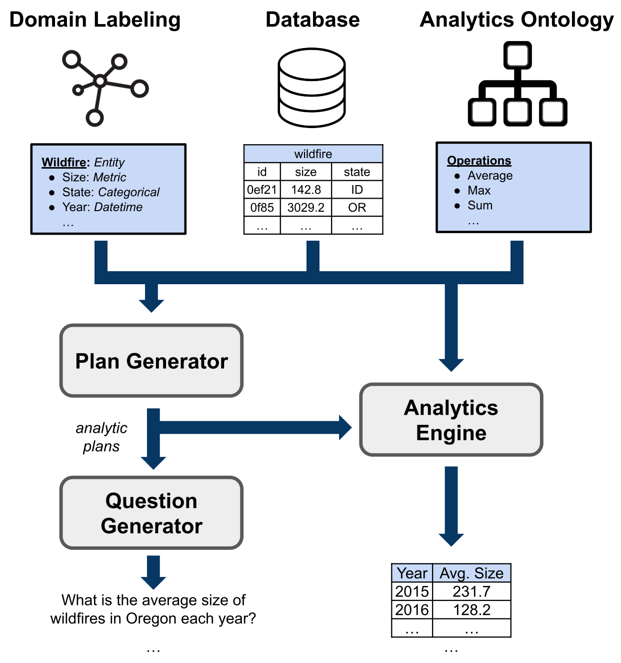
Abstract

The principal goal of data science is to derive meaningful information from data. To do this, data scientists develop a space of analytic possibilities and from it reach their information goals by using their knowledge of the domain, the available data, the operations that can be performed on those data, the algorithms/models that are fed the data, and how all of these facets interweave. In this work, we take the first steps towards automating key aspects of the data science pipeline. We present an extensible ontology of data analytic operations that scopes across domains and data, as well as a method for codifying domain-specific knowledge that links this analytics ontology to actual data. We validate the functionality of our analytics ontology by implementing a system that leverages it, alongside domain-specific labelings for an initial set of 8 domains with high social impact, to automatically generate a space of answerable questions and associated queries. In this way, we produce information spaces over data that enable complex analyses and search over this data and pave the way for applying automated search techniques over this space.

1 Introduction

Data science is a vast and interdisciplinary field whose principal goal is to derive meaningful information from data. Data scientists use domain knowledge, the available data, the operations and algorithms that can be used to derive information from that data, and how all of these facets interweave to develop a space of analytic possibilities and reach their information goals. Due to the level of technical skills required, the process of performing good data science can be a very demanding task, and it often takes a significant amount of training to do it well.

Yet, with the vast amounts of data being generated on a daily basis, it is critical that more people have access to meaningful information derived from their data. This is particularly true for domains in which high quality and accurate information is crucial for good outcomes, such as journalism, healthcare, and policy making. While data scientists are employed across a wide range of such domains, it is not feasible for every organization to hire the requisite number of data scientists due to both cost constraints as well as an ever increasing demand for data scientists. Furthermore, not every organization or individual has the technical skills required to do it themselves. These challenges greatly limit people's abilities to access high quality information to guide their decision making.



Wildfire: *Entity*
- Size: *Metric*
- State: *Categorical*
- Year: *Datetime*
- …

wildfire		
id	size	state
0ef21	142.8	ID
0f85	3029.2	OR
…	…	…

Operations
- Average
- Max
- Sum
- …

Year	Avg. Size
2015	231.7
2016	128.2
…	…

Figure 1: The domain labeling encodes knowledge of the entities present in a database while the analytics ontology encodes knowledge on core analytic information and processes which provide a conceptual framework for performing analytics on these entities. This process flow outlines a practical way in which these knowledge representations can surface meaningful information from data by generating analytic plans, mapping them to the data via the analytics ontology and domain labeling, and ultimately executing them to derive the information.

One way to enable broader access to high quality information is to build a system that can automatically surface such information from data and make it available to people in a form which is understandable to them: natural language. The first step to doing this is to codify the knowledge and processes a data scientist relies on when analyzing data.

This entails developing a method for representing the data scientist's knowledge of the domain, what the data means in the context of this domain, and how analytics map onto these data. The second step is to determine how to develop analytic plans against this representation, such that they can be used to surface the space of analytic possibilities in clear natural language.

In this paper, we lay the foundations for automating data science by building representations for the internal knowledge and processes a data scientist utilizes when deriving information from data. We present a domain-independent (i.e. can scope across different data disciplines) and expandable (i.e. capabilities can be augmented) ontology that captures the knowledge needed to automate the main functions of a data scientist. We also present a method for generating a complementary labeling of domain-specific knowledge that links data to the analytics ontology. With this *domain labeling*, the tables and columns of a relational database are mapped to entities, attributes, and the relationships between them. We put a particular emphasis on ensuring that the domain-level expansion process is simple, with the aim of enabling non-technical users to bring in data and explore the space of information it encodes. This also entails labeling each of the entities and attributes with nicely worded language (a "nicename") and providing units where applicable.

In order to validate the functionality of the analytics ontology and domain labeling, we develop a system that utilizes them when deriving information from real-world data. We also developed an analytic plan representation that enables complex queries to be composed in terms of the entities and attributes present in the domain labeling. This representation is similar to SQL, but is execution agnostic and abstracts away technically complex details like table joining in order for the resulting plans to be more interpretable for people without a data science background. To ensure such plans can be executed against the data, we build an analytics engine that automatically constructs an object-relational mapping for the database according to the domain labeling. This mapping enables the analytic plans to be converted to SQL queries and thus executed against data.

With this work, we seek to bring out the realizable space of information based on the available data and analytics. Towards this end, we develop a planning component that can generate a wide variety of useful analytic plans for a given domain labeling. Since analytic plans are not the most intuitive way for people to understand what information can be derived from the data, we build a language generation component which can produce the question this analytic plan is meant to answer. In this way, the space of analytic possibilities can be surfaced to any user, regardless of their technical ability, and they are empowered to choose which questions they want answered. Figure 1 shows the process flow for exposing the information space. The ability to generate this plan space paves the way for the future development of automated search techniques.

To demonstrate the totality of our approach, we create 8 domain labelings for domains of high social impact to ensure that our approach generalizes well to a variety of fields and is domain-agnostic. These domains are healthcare, urban housing, criminal justice, environmental sustainability, education, legal and judicial, socioeconomics, and business. We use these in conjunction with our planner to showcase the kinds of useful information our approach can present to a user.

The primary contributions of this paper are:

- An extensible ontology of data analytic operations that scopes across domains and data.
- A method for producing domain-specific labels that links this analytics ontology to actual data and provides linguistic constructs that enable improved communication of any derived information.
- An execution agnostic plan representation that enables the retrieval and analysis of data according to the entities in the domain labeling.
- A method for leveraging the analytics ontology and domain labeling in order to build searchable information spaces over data that surfaces useful information in natural language, making complex analyses and search over this data easily available to non-technical human experts.

2 Related Work

Formalizing world knowledge with structured representations in the form of an ontology has long been a way to imbue systems with an understanding of the entities and features of the domain they operate in. Such knowledge representations provide the structure necessary for a wide range of reasoning and inference processes to function. Ontologies such as Wikidata (Vrandečić and Krötzsch 2014) and ConceptNet (Speer, Chin, and Havasi 2017) seek to codify general world knowledge, while others such as Cyc (Lenat 1995) are designed to encode commonsense rules as well. Oftentimes the knowledge within these general-purpose ontologies is too broad, with conceptual classes too vague and coverage of specialized knowledge too sparse for practical usage within more focused domains. As a result, highly specialized ontologies continue to be developed for a diverse range of areas such as medicine (Salvadores et al. 2013), law (Casellas 2011), food (Kamel Boulos et al. 2015), chemical engineering (Marquardt et al. 2010), and biological environments (Buttigieg et al. 2013).

Along with these efforts for codifying domain knowledge, ontologizing data science has emerged as an area of interest. For instance, (Patterson et al. 2019) have semantically enriched data science scripts with the goal of successfully modeling computer programs. However, their work focuses more on supporting automated reasoning about data science software rather than encoding core analytic knowledge and processes that can be used when mapping analytics onto data in a domain-agnostic fashion. (Sicilia et al. 2018) presents a way to describe the data transformations that can be applied as part of a data science pipeline. Our work includes this as a subcomponent of a model for describing how analytics map onto data and how formalized domain knowledge can be utilized to determine the kinds of analyses that will result in meaningful information. In an effort to codify knowledge around data, as well as to increase interoperability and reusability of methodologies, multiple ontologies have been created. Examples of these include DMOP for data mining processes (Keet et al. 2015), PMML for machine learning models (Guazzelli, Stathatos, and Zeller 2009), SDMX (Capadisli, Auer, and Ngonga Ngomo 2015) and STATO (Gonzalez-Beltran and Rocca-Serra 2016) for statistics, and even some ontologies for modeling data transformation and workflows (Bowers and Ludäscher 2004) (Barker and Hemert 2007).

In addition to automation through explicit data knowledge we have seen a sustained history of efforts in improving, and sometimes automating, the data science pipeline through supplemental tools and workstreams. This includes analyses of tabular data (e.g. Excel and other spreadsheet variants), data cleaning (Petrova-Antonova and Tancheva 2020), programming-based analytics computation (e.g. Pandas, Matlab and R), and data visualization (e.g. Tableau and Power BI). However, even the tools with ease of use or more interactive experiences, such as notebook-style interfaces (Kluyver et al. 2016) (Pérez and Granger 2007), are built for data scientists and require significant data science knowledge. Furthermore, these lack an ability to directly integrate semantic knowledge with available analyses, requiring users to bring that knowledge themselves and manually constrain the analyses to what's appropriate to the domain (Beg 2021).

3 Data Analytics Ontology

In order to automate the data analysis process, a representation of analytic operations, their functionality, and their inputs and outputs must be specified. Such a representation can be leveraged to determine what analysis can be performed on any given piece of data, how this data will be transformed as a result of this analysis, and how this result can be used as input to other analytic operations. The analytics ontology we present is shown in Figure 2.

Analytic Operations

Not all analytic operations specified in the ontology can or should be applied to all types of data. For instance, it does not make sense to perform arithmetic operations like sum on categorical variables, like gender. A key goal of the analytic ontology we present is to provide a mapping between analytic operations and the types of data they can be applied to. By defining such a mapping, it is possible to constrain the data analysis to those analyses which will result in meaningful and useful information. This is done by specifying an *attribute type* for the inputs and outputs of each analytic operation.

Attribute Types

We define six main attribute types which enable the data to be mapped to higher level concepts that can be utilized when determining what analytics will result in useful information given the data. For example, it makes sense to perform a *max* operation on a column of a patient's heart rate measurements, but does not make sense to perform a *sum* operation on this column as it serves no practical purpose. The six main attribute types are:

- **Arithmetic**: numerical values for which mathematical operations make sense
- **Categorical**: discrete values typically denoting a class or category
- **Datetime**: values designating a date and time
- **Document**: values containing free text
- **Identifier**: values meant to be used as a unique identifier for an entity
- **Metric**: values meant to be used a measure

For example, assume there is a dataset comprising business reviews with a *Business* entity which has attributes *number of reviews* and *rating*. They would both be considered *Arithmetic* attributes, while only *rating* is a *Metric*, as it can be used as a measure of the entity in the context of this dataset. These attribute types are leveraged to constrain the space of possible analytics that can be performed on the entities specified in a domain labeling.

There is also a set of attribute types reserved for *derived attributes*, which are those attributes that result from executing an analytic operation. These include:

- **Entity**: value that results from retrieving an entity
- **Attribute**: value that results from retrieving an attribute
- **AttributeCollection**: values that result from collecting many attributes
- **Group**: value that results from grouping by an attribute
- **Filter**: value that results from specifying a filter
- **Sort**: value that results from specifying a sorting order over the results
- **Limit**: value that results from specifying a limit over the results
- **String**: value that results from specifying a particular string to use

For each of the operations, each of the input and output attributes are constrained to have one or more attribute types. For example, the *average* operation would have exactly one mandatory input argument with an attribute type of either *Arithmetic* or *Metric*. It would have a second optional argument with the *Group* derived attribute type. The output of this operation would be a single derived attribute with a type of either *Arithmetic* or *Metric*. For a full listing of the operations, operation type, and their input and output attribute types, please see Appendix A.

4 Domain Labeling

In order to effectively apply the operations specified in the analytics ontology to data, we produce a lightweight mapping called a *domain labeling* which specifies the entities that comprise the data, their attributes, and the relationships between these entities. Each of their attributes has one or more attribute types which allows them to be linked to the analytics ontology. Importantly, these domain labelings are simple to specify relative to conventional domain ontologies and the process of creating them is more similar to tagging than the knowledge engineering that is applied when producing an ontology. In this section, we present further details of how this domain labeling is specified.

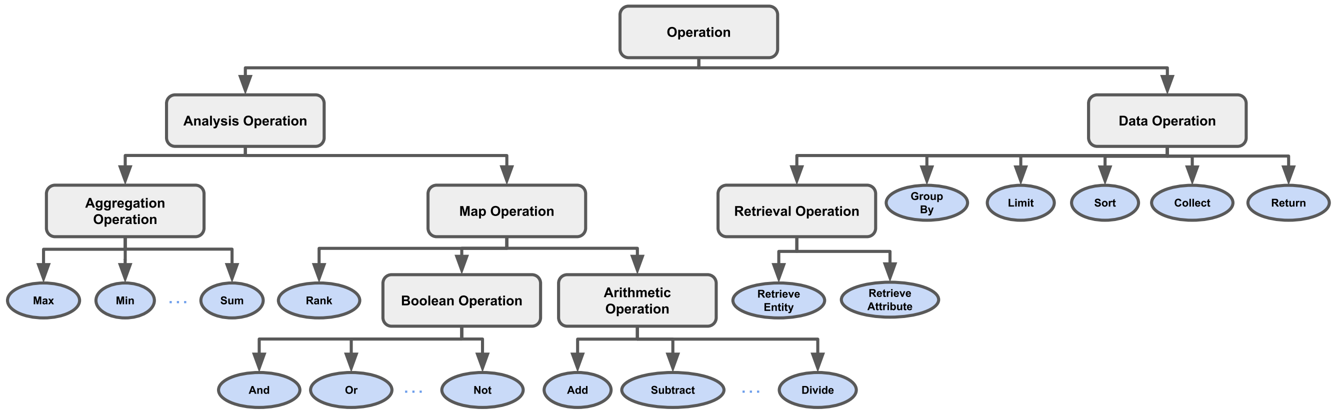


Figure 2: The analytics ontology which encodes knowledge of data analytic operations and processes. It is designed to be extensible and, in addition to those shown in here, currently comprises 11 aggregation operations, 9 boolean operations, and 6 mathematical operations. This ontology specifies a conceptual mapping between these operations and the types of data that makes sense for them to operate upon, much in the same way a data scientist identifies the appropriate analysis to carry out on any given data.

Entities

Entities defined by a *domain labeling* specify one of the entities represented in the corresponding database. An entity is a concept that groups relevant data together, potentially spanning multiple underlying data tables, into a single unified grouping. This entity contains attributes (which map to columns of the data that comprise the entity) and has relationships with other entities (which map to one or more joins across the data tables). Each entity has a name which is used to refer to it when performing analysis upon its attributes. An example of this can be seen for a subset of instances in the healthcare domain in Figure 3.

Entities defined by a domain labeling specify one of the entities represented in the corresponding database. For example, in a dataset detailing information about visits to the emergency room, entities are likely to include patients, stays, tests/diagnostics performed on the patient, and ER centers (if the dataset includes multiple).

Attributes

Each entity has one or more attributes, each of which maps to a column of one of the underlying database tables comprising the entity. Each attribute has a data type (e.g. integer, float, string) and one or more ontological types (e.g. categorical, metric, arithmetic). The former is used when producing the object-relational mapping for the database and the latter is used to connect this attribute to the operations from the analytics ontology that can be applied to this attribute.

Each attribute also has a ”nicename” that provides a more descriptive label for the attribute than the column name, as well as the units for this attribute, if any. For example, the attribute ”dbp” could have the nicename of ”Diastolic Blood Pressure” and units ”mmHg”. These two properties of the attribute are used by the language generator described in Section 6.

Relationships

Relationships define the connection between two entities in the labeling. These relationships can be one-to-one, one-to-many, or many-to-many. In the specification of the domain labeling, they are represented as an abstraction of SQL joins. Adding relationships between entities forms a *domain graph schema*, and each provides relevant metadata (e.g. one *Subject* can have many *Emergency Department Stays*). For instance, with a one-to-many relation, the entity on the many side can be grouped by the entity on the one side. This allows for aggregations to be applied to the entity on the *one* side. For example, the system knows it is possible to compute the average of stay duration grouped by subject ID based on the type of relationship between the two entities these attributes come from.

5 Analytic Plan Representation

In order to effectively use the domain labeling and analytic ontologies, we require an expressive and compositional plan representation. Existing representations like SQL are inadequate since the analytics ontology scopes beyond the operations supported within this query language. Having an execution agnostic representation such as this allows us to seamlessly expand beyond SQL operations in the future. To satisfy this representational need, we define an analytic plan representation which allows for the specification of plans in which the entities and attributes defined in the domain labeling are retrieved and analyzed with the operations defined by the analytics ontology.

Plan Structure

Plans are represented as a directed acyclic graph that specifies an ordered series of steps to carry out, wherein operations are chained together in order to retrieve and analyze data. Each node of the graph represents an operation whose

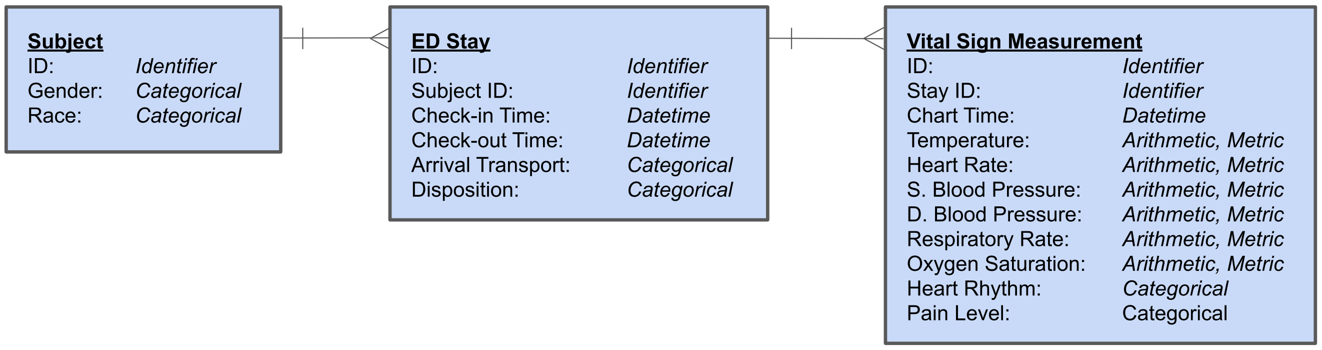


Figure 3: An example of a domain labeling which specifies entities, their attributes, and how they relate to each other for a dataset in the healthcare domain. The entity name is given in bold, attributes are listed on the left, and their corresponding attribute types are shown on the right. Each of these attributes maps to specific column of the underlying database, and each of the relationships provides the joins between the underlying tables. This formulation enables analytic plans to be specified in terms of the entity attributes without regards to how the tables should be joined, thus abstracting out implementation details that can be automatically handled with an execution engine.

output is fed to later steps which require the results. In this way, arbitrarily complex plans can be composed to satisfy any information goal that can be described with the available data and analytics. An example of an analytic plan can be seen in Figure 4.

Leveraging the analytics ontology allows this representation to support standard SQL operations (retrieval, aggregation, groupby, filtering, sorting, etc.). Entities and attributes described by the domain labeling can be retrieved using the *retrieve_entity* and *retrieve_attribute*. Analytic operations take these attributes as input and produce *derived attributes* (i.e. ones which are not present in the domain labeling), which can be used as input to subsequent operations. In this way, arbitrarily complex analysis can be composed by chaining operations together. Data manipulation operations such as *sort* and *limit* can be added, as can filtering operations like comparison and boolean operators.

Both entity attributes and derived attributes can be passed to the *collect* operation in order to stage them for output in the final results. The *return* operation takes in the attributes to be collected, along with any *sort*, *filter*, and *limit* operations that were specified. Each *return* operation denotes the end of a plan and results in a structure which is analogous to a single SQL query. Subsequent queries can retrieve attributes collected by prior queries. In this way, subplans, which are analogous to nested queries or ”subqueries” found in SQL, can be produced.

6 A Practical Implementation of the Data Analytics Ontology

To illustrate how the analytics ontology can be operationalized as a real system, we develop a prototype system that utilizes the operation ontology and a domain labeling to produce a set of useful analyses and information that can be derived from data. These analyses are presented as natural language questions that enable anyone to manually search through the list of questions and pick ones they want the answer to. Each question has a corresponding analytic plan that can be executed by this system in order to derive these answers.

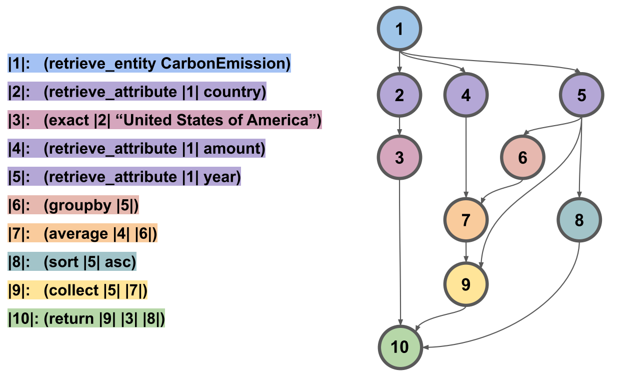

|1|: (retrieve_entity CarbonEmission)
|2|: (retrieve_attribute |1| country)
|3|: (exact |2| “United States of America”)
|4|: (retrieve_attribute |1| amount)
|5|: (retrieve_attribute |1| year)
|6|: (groupby |5|)
|7|: (average |4| |6|)
|8|: (sort |5| asc)
|9|: (collect |5| |7|)
|10|: (return |9| |3| |8|)
1
2
3
4
5
6
7
8
9
10

Figure 4: The analytic plan in textual and graph forms are shown for the question: “What is the average carbon emissions grouped by year in ascending order for the United States of America?”

Analytics Engine

Execution of analytic plans requires that they first be converted to a query format that is native to the datasource (e.g., SQL for relational databases) and then executed to retrieve results. This is where the analytics engine comes in.

Plan Parsing and Execution Conversion of the graph-structured analytics plan into a query language is done by first breaking the graph into ”subplans”, where the result of one subquery functions as a data source for subsequent queries. From each subplan, the necessary information to form an executable query, including entities, their attributes, analytics operations, and filters, is then extracted. For example, in the plan in Figure 4, the attributes *country*, *amount*, and *year* are retrieved from the *CarbonEmission* entity. The *average* operation is applied to *amount*, grouped by *year*. Lastly, the results are sorted by *year* and restricted such that *country* has value "*United States of America*". From this information, the query is constructed using a query abstraction library (for relational databases we use the SQLAlchemy Python package (Bayer 2012)). In this last step, the domain labeling is leveraged to convert the entity and relationship abstractions to tables and joins.

SQL Object Relational Mapping While the system's analytics engine is designed to be easily extendable to a variety of data source types, it is currently only configured to execute queries against relational databases. Upon initialization, for each selected domain labeling, a corresponding object relational mapping (ORM) is built (using SQLAlchemy). The ORM provides a programmatic interface between the information defined in the domain labeling, and the data stored in a relational database.

The ORM is constructed using configuration mappings defined in the domain labeling, specifically, the tables, columns, and joins between tables. Unlike objects defined in the domain labeling, the ORM objects hold a direct one-to-one correspondence with database objects; ORM entities correspond to tables, attributes correspond to columns, etc.

Implicit Joins A major benefit provided by the abstractions of the domain labeling is that, for a given domain, relationships between tables need only be defined once (when the labeling is defined). No join information of any kind is required in plan definitions. Instead, the system leverages the joins and relations defined in the domain labeling to determine which SQL joins to use when attributes are selected corresponding to columns of different tables.

First, all necessary joins between tables within an entity are identified. These intra-entity joins are necessary when multiple attributes are specified as belonging to the same entity, but correspond to columns from different tables. Then, all joins between each pair of entities in a given plan are identified by collecting joins along the shortest path of relationship links between those entities.

Domains

The analytics engine is equipped with a domain-independent analytic ontology and code-level implementations of these operations to perform on various datasets. To demonstrate a variety of applications of our approach in a wide range of domains, we evaluate our system using 8 high social impact domains. We specify a domain labeling for each of these domains. Each labeling determines the domain-relevant features of the data and how they map to real world concepts and other in-domain entities. Below is a list of domains and datasets for which we seek to surface the space of information possibilities.

- **Education**
  - Illinois State Board of Education's per pupil expenditure report for 2021-2022 academic year. (Max Larkin 2020)
- **Criminal Justice**
  - Center for Homeland Defense and Security's Shooting Incidents at K-12 Schools between January 1970 and June 2022 (Center for Homeland Defense and Security 2023)
- **Legal and Judicial**
  - Systematic Content Analysis of Litigation Events Open Knowledge Network (SCALES - OKN) (SCALES OKN 2021)
- **Healthcare**
  - Medical Information Mart for Intensive Care (MIMIC)-IV-ED database (Johnson et al. 2023)
- **Business**
  - Yelp Open Dataset: A subset of Yelp's businesses, reviews, and user data (Yelp Inc. 2023)
- **Environmental Sustainability**
  - Air Data: Air Quality Data Collected at Outdoor Monitors Across the US (United States Environmental Protection Agency 2015)
  - Spatial Wildfire Occurrence data for the United States, 1992-2020 (Short 2022)
- **Urban Housing**
  - Zillow Observed Rent Index (Zillow Group, Inc. 2023)
- **Socioeconomic**
  - Income and Poverty Dataset for Illinois (U.S. Bureau of Economic Analysis 2022; U.S. Bureau of Labor Statistics 2023; U.S. Census Bureau 2022b,a,c)

Generating Information Spaces

The definition of the domain labelings makes possible the generation of an *information space*: a space of domain-independent information goals that can be used to answer relevant questions about the data.

Plan Generation We produce analytic plan templates corresponding to these information goals which have fillable slots, allowing for executable plans to be generated and executed within the target domain by filling in these slots. Slot types include entity, attribute, and analysis operation slots.

To generate the space of possible plans, all combinations of the objects in the domain labeling that match a slot type, are used for filling that slot. For example, for a labeling defining a legal domain, the entity slot may be filled by a *Judge* or a *Case* entity. An attribute slot may be filled by a *name* or *age* attribute (if the entity slot has been filled by the *Judge* entity); or *case name*, *duration*, or *year* (if the entity slot has been filled by the *Case* entity). In this way, the attribute slot is constrained by the entity it is being drawn from. Attribute slots are also constrained according to their attribute type. For example, a *Metric* attribute slot may only be filled with attributes of this type (e.g., *salary*, *duration*, etc.), and a *Categorical* attribute slot may only be filled by categorical attributes (e.g., *case type*, *year*, etc.). This separation of types is particularly useful for plan templates that incorporate grouping aggregations, as the categorical attribute will generally be the attribute to group on,

Domain	Questions
Environmental Sustainability	What is the max air quality index grouped by year for state of Washington? What is the average fire size grouped by year for state of California?
Healthcare	What is the disease for stay id of 31945330? What is the count of stay id for subject id of 10023239?
Urban Housing	For date sorted in descending order and limited to the top results, what is the average rent for region name of United States? What is the average rent for region name of San Francisco, CA?
Criminal Justice	What is the count of unique incident id for weapon type containing ”handgun”? Is the count of unique incident id for weapon type containing ”handgun” greater than count of unique incident id for weapon type containing ”rifle”?
Education	What is the correlation between total per pupil expenditure and percentage of students with an sat-math scores that exceed standards for county of DuPage County? What is the average student enrollment for county of Adams and school type of HIGH SCHOOL?
Legal and Judicial	What is the average case duration grouped by case type? What is the average case duration grouped by year for name of colleen kollar-kotelly?
Socioeconomic	What is the average personal income grouped by year? What is the estimated people below 17 in poverty for county of Wayne County?
Business	What is the average star rating for business of Lou Malnalti’s? What is the count of business for city of Philadelphia?

Table 1: A suite of example questions which can be surfaced via the plan generator. These represent a small subset of the useful information that can be derived for each of the domain labelings given the operations defined in the analytics ontology. Each question corresponds to an analytic plan which can be executed with the engine. In this way, meaningful information can be easily derived from any given dataset by specifying the lightweight domain labeling.

whereas the metric will generally be the attribute to aggregate. Finally, an analysis operation slot may be filled by any analysis operation whose inputs matches the plan structure at hand. In many cases, this is an aggregation (e.g., sum, max, min, average, etc.).

Additionally, the information space includes plans that answer questions about specific instances of entities in the domain labeling, rather than questions about aggregations over all entities in the data (as the information goals described above do). This is done by adding a filler slot to the plan that limits it to a specific entity instance. Plans of this type are generated by filling this slot with each possible unique instance of this entity in the data. For example, for a `judge = [name]` filter, a different plan is generated by substituting `[name]` with each of the possible judge names in the data. Together, by filling these templates with possible combinations of values from the domain labeling, analytic ontology, and database, a vast, yet tractable space of useful analytic plans (corresponding to particular information goals) may be surfaced to the user.

Question Generation Since graph-based plan representations are not the most intuitive way for lay-people to understand what kind of information can be derived from the data, we build a template-based language generator which can produce a corresponding question for each of the analytic plans. The plan graph is traversed in reverse order starting from the terminal *return* operation node. The attributes to collect are determined by examining the *collect* statement it references, and all filters to express for each attribute are gathered by looking at the ancestor nodes of the collected attributes.

The analytic ontology and domain labeling specify language templates for the analysis operations and nicenames for the entity attributes, respectively. The analytic language templates are filled in using the attribute nicenames in order to produce simple language that can be used to express the meaning of each step of the analytic plan. These simple language statements can be chained as necessary (for nodes which have extensive ancestry) and joined together (when there are many nodes being utilized in the final output). The result of this is a natural language question for the corresponding analytic plan. Examples of questions for each of the 8 domains that were generated with this method can be seen in Table 1.

7 Conclusion and Future Work

In this paper, we introduce an analytics ontology that encodes knowledge of analytic operations and how they map onto data in order to replicate key pieces of knowledge that data scientists use to derive meaningful information from data. To complement this knowledge representation, we showcased a system for surfacing quality information from datasets in a wide range of domains. In the future, we will seek to encode more data science knowledge by identifying new attribute types and increasingly granular categories of types. For instance, metrics could be further broken down into diagnostic and ranking metrics to distinguish between measures that can be used for competitive comparisons (e.g. sales performance) versus those that should not (e.g. heart rate of patients in a hospital). We will also be exploring ways in which the information derived by the analytics engine can be better communicated (e.g. via documents and visualizations). This work lays the foundation for further work towards automating data science and providing systems that allow all people, regardless of their technical background, to derive meaningful insights from their data.

A Analytic Ontology Operations

A listing of the operations currently available within the analytics ontology can be seen in Table 2. This ontology was designed with extensibility in mind, and currently implements most major SQL operations.

References

Barker, A.; and Hemert, J. v. 2007. Scientific workflow: a survey and research directions. In *International Conference on Parallel Processing and Applied Mathematics*, 746–753. Springer.

Bayer, M. 2012. SQLAlchemy. In Brown, A.; and Wilson, G., eds., *The Architecture of Open Source Applications Volume II: Structure, Scale, and a Few More Fearless Hacks*. aosabook.org.

Bowers, S.; and Ludäscher, B. 2004. An ontology-driven framework for data transformation in scientific workflows. In *International Workshop on Data Integration in the Life Sciences*, 1–16. Springer.

Buttigieg, P. L.; Morrison, N.; Smith, B.; Mungall, C. J.; and Lewis, S. E. 2013. The environment ontology: contextualising biological and biomedical entities. *Journal of biomedical semantics*, 4(1): 1–9.

Capadisli, S.; Auer, S.; and Ngonga Ngomo, A.-C. 2015. Linked SDMX data. *Semantic Web*, 6(2): 105–112.

Casellas, N. 2011. *Legal ontology engineering: Methodologies, modelling trends, and the ontology of professional judicial knowledge*, volume 3. Springer Science & Business Media.

Center for Homeland Defense and Security. 2023. Shooting incidents at K-12 schools (Jan 1970-Jun 2022) - CHDS School Shooting Safety Compendium.

Gonzalez-Beltran, A.; and Rocca-Serra, P. 2016. Statistics Ontology (STATO).

Guazzelli, A.; Stathatos, K.; and Zeller, M. 2009. Efficient deployment of predictive analytics through open standards and cloud computing. *ACM SIGKDD Explorations Newsletter*, 11(1): 32–38.

Johnson, A.; Bulgarelli, L.; Pollard, T.; Celi, L. A.; Horng, S.; Mark, R.; and of Technology [MIT], M. I. 2023. MIMIC-IV-ED Demo v2.2.

Kamel Boulos, M. N.; Yassine, A.; Shirmohammadi, S.; Namahoot, C. S.; and Brückner, M. 2015. Towards an "Internet of Food": food ontologies for the internet of things. *Future Internet*, 7(4): 372–392.

Keet, C. M.; Ławrynowicz, A.; d'Amato, C.; Kalousis, A.; Nguyen, P.; Palma, R.; Stevens, R.; and Hilario, M. 2015. The Data Mining OPtimization Ontology. *Journal of Web Semantics*, 32: 43–53.

Kluyver, T.; Ragan-Kelley, B.; Pérez, F.; Granger, B.; Bussonnier, M.; Frederic, J.; Kelley, K.; Hamrick, J.; Grout, J.; Corlay, S.; Ivanov, P.; Avila, D.; Abdalla, S.; and Willing, C. 2016. Jupyter Notebooks – a publishing format for reproducible computational workflows. In Loizides, F.; and Schmidt, B., eds., *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, 87 – 90. IOS Press.

Lenat, D. B. 1995. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11): 33–38.

Marquardt, W.; Morbach, J.; Wiesner, A.; Yang, A.; and CAPE, O. 2010. *A Re-Usable Ontology for Chemical Process Engineering*. Springer.

Max Larkin. 2020. Per Pupil Exp. by district - dataset by jmlarkin.

Patterson, E.; Baldini, I.; Mojsilovic, A.; and Varshney, K. R. 2019. Teaching machines to understand data science code by semantic enrichment of dataflow graphs. arXiv:1807.05691.

Pérez, F.; and Granger, B. E. 2007. IPython: a System for Interactive Scientific Computing. *Computing in Science and Engineering*, 9(3): 21–29.

Petrova-Antonova, D.; and Tancheva, R. 2020. Data cleaning: A case study with OpenRefine and Trifacta Wrangler. In *Quality of Information and Communications Technology: 13th International Conference, QUATIC 2020, Faro, Portugal, September 9–11, 2020, Proceedings 13*, 32–40. Springer.

Salvadores, M.; Alexander, P. R.; Musen, M. A.; and Noy, N. F. 2013. BioPortal as a dataset of linked biomedical ontologies and terminologies in RDF. *Semantic web*, 4(3): 277–284.

SCALES OKN. 2021. Home - SCALES OKN.

Short, K. C. 2022. Spatial wildfire occurrence data for the United States, 1992-2020.

Sicilia, M.-Á.; García-Barriocanal, E.; Sánchez-Alonso, S.; Mora-Cantallops, M.; and Cuadrado, J.-J. 2018. Ontologies for data science: On its application to data pipelines. In *Research Conference on Metadata and Semantics Research*, 169–180. Springer.

Speer, R.; Chin, J.; and Havasi, C. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-first AAAI conference on artificial intelligence*.

United States Environmental Protection Agency. 2015. Air Data.

U.S. Bureau of Economic Analysis. 2022. Personal income in Cook County, IL.

U.S. Bureau of Labor Statistics. 2023. Unemployed persons in Cook County, IL.

U.S. Census Bureau. 2022a. Estimate of median household income for Cook County, IL.

U.S. Census Bureau. 2022b. Estimate of people age 0-17 in poverty in Cook County, IL.

U.S. Census Bureau. 2022c. Estimate of people of all ages in poverty in Cook County, IL.

Vrandečić, D.; and Krötzsch, M. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10): 78–85.

Yelp Inc. 2023. Yelp Open Dataset.

Zillow Group, Inc. 2023. Housing data - Zillow Research.

Operation	**Operation Type**	**Input**	**Output**
Average	Aggregation	[1,1]: [Arithmetic, Metric] [0,1]: [Grouping]	[1,1]: [Arithmetic, Metric]
Correlation	Aggregation	[2,2]: [Arithmetic, Metric, Datetime] [0,1]: [Grouping]	[1,1]: [Arithmetic, Metric, Datetime]
Count	Aggregation	[1,1]: [Arithmetic, Metric] [0,1]: [Grouping]	[1,1]: [Arithmetic, Metric]
Count Unique	Aggregation	[1,1]: [Arithmetic, Metric] [0,1]: [Grouping]	[1,1]: [Arithmetic, Metric]
Get One	Aggregation	[1,1]: [Arithmetic, Metric, Datetime] [0,1]: [Grouping]	[1,1]: [Arithmetic, Metric, Datetime]
Max	Aggregation	[1,1]: [Arithmetic, Metric, Datetime] [0,1]: [Grouping]	[1,1]: [Arithmetic, Metric, Datetime]
Median	Aggregation	[1,1]: [Arithmetic, Metric, Datetime] [0,1]: [Grouping]	[1,1]: [Arithmetic, Metric, Datetime]
Min	Aggregation	[1,1]: [Arithmetic, Metric, Datetime] [0,1]: [Grouping]	[1,1]: [Arithmetic, Metric, Datetime]
Standard Deviation	Aggregation	[1,1]: [Arithmetic, Metric] [0,1]: [Grouping]	[1,1]: [Arithmetic, Metric]
String Aggregation	Aggregation	[1,1]: [Arithmetic, Metric, Datetime] [0,1]: [Grouping]	[1,1]: [Arithmetic, Metric, Datetime]
Sum	Aggregation	[1,1]: [Arithmetic] [0,1]: [Grouping]	[1,1]: [Arithmetic, Metric]
And	Boolean	[1,inf]: [Filter]	[1,1]: [Filter]
Contains	Boolean	[1,1]: [Attribute] [1,1]: [Metric]	[1,1]: [Filter]
Exact	Boolean	[2,2]: [Arithmetic, Metric, Categorical, String, Datetime, Identifier]	[1,1]: [Filter]
Greater Than	Boolean	[2,2]: [Arithmetic, Metric]	[1,1]: [Filter]
Greater Than Equal	Boolean	[2,2]: [Arithmetic, Metric]	[1,1]: [Filter]
Less Than	Boolean	[2,2]: [Arithmetic, Metric]	[1,1]: [Filter]
Greater Than Equal	Boolean	[2,2]: [Arithmetic, Metric]	[1,1]: [Filter]
Not	Boolean	[1,inf]: [Filter]	[1,1]: [Filter]
Or	Boolean	[1,inf]: [Filter]	[1,1]: [Filter]
Add	Arithmetic	[2,inf]: [Arithmetic, Metric, Datetime]	[1,1]: [Arithmetic, Metric, Datetime]
Divide	Arithmetic	[2,inf]: [Arithmetic, Metric, Datetime]	[1,1]: [Arithmetic, Metric, Datetime]
Multiply	Arithmetic	[2,inf]: [Arithmetic, Metric, Datetime]	[1,1]: [Arithmetic, Metric, Datetime]
Percent Change	Arithmetic	[2,2]: [Arithmetic, Metric]	[1,1]: [Arithmetic, Metric]
Square Root	Arithmetic	[1,1]: [Arithmetic, Metric, Datetime]	[1,1]: [Arithmetic, Metric, Datetime]
Subtract	Arithmetic	[2,inf]: [Arithmetic, Metric, Datetime]	[1,1]: [Arithmetic, Metric, Datetime]
Collect	Data Operation	[1,inf]: [Attribute]	[1,1]: [AttributeCollection]
Groupby	Data Operation	[1,inf]: [Attribute]	[1,1]: [Group]
Limit	Data Operation	[1,1]: [Attribute]	[1,1]: [Limit]
Return	Data Operation	[1,1]: [AttributeCollection] [0,1]: [Filter] [0,1]: [Sort] [0,1]: [Limit]	[1,1]: [Entity]
Sort	Data Operation	[1,inf]: [Attribute] [1,1]: [String]	[1,1]: [Sort]
Retrieve Attribute	Retrieval	[1,1]: [Entity] [1,1]: [String]	[1,1]: [Attribute]
Retrieve Entity	Retrieval	[1,1]: [String]	[1,1]: [Entity]

Table 2: The set of all operations implemented as part of the analytics ontology. This includes operations used for retrieval, analysis, filtering, and data transformations. Inputs and outputs for each operation are defined as [min_num, max_num]: [attribute_types].