LinkIt

Making Sense of Disparate Data

1. Background

The increasing value of data, data analytics, and machine learning is undeniable. Data is like raw ore that can be mined, refined, and used to fuel the economy. Mining and refining real ore is hard and the same is true for data, albeit for different reasons. In the case of data, the greatest value is derived when multiple data sets are used together to create new insights. For example, imagine a customer calls a support line at a company. The moment the support person answers the phone, they ought to know whether there have been previous support calls, what products the customer has purchased, how they rated the products, the sentiment of any reviews they have written, and whether they have been offered any promotions. Armed with this information, the support person has the greatest chance of providing a speedy and successful outcome.

That may not sound too difficult, but unfortunately, that data will be housed in at least 4 systems (sales, support, product, marketing)[1]. Those systems almost certainly have incompatible data organization, even if they were developed by the same organization! A company could just mandate that all the different systems converge on a single data model, but repeated failures show that such projects take years, cost millions, and are likely to start rotting almost as soon as the project is complete.

Now consider that the company may also want to mine and integrate data from external sources. What has the customer said about the product/company on Twitter? What's the weather like in the customer's region (might affect some products or the ability to correct problems)? What do we know about other people like that customer – how did we resolve their issues? Even more now than before, we are guaranteed we will be working with lots of different schemas.

Integrating data is hard which is why there is a multi-billion dollar industry trying to address it.

2. Vision

LinkIt is a tool that automates the most labor-intensive manual tasks associated with data integration by examining disparate data, rationalizing the schema, and finding natural linkages. *LinkIt* provides an extensible framework so that the range of data it understands can be expanded over time – perhaps even by non-programmers.

The Basic Problem

Consider two datasets from the example above: a database of customer information and a database of sales information. The customer table might look like this (greatly simplified):

FName	LName	Phone	Street	City	…
Joe	Blow	555-679-1212	520 Main St	Mytown	…

[1] In a real world scenario there could be over 10,000 tables that make up the data assets in a business – sometimes 10's of thousands.

Jane	Doe	555-345-5432	6751 Plum St	Midvale	…
Sven	Fishman	555-947-2659	123 Able Dr	Monterey	…

The sales information might look like this:

Name	Mobile	Addr1	…	Purchased	Credit Score
Ed Smith	555-808-2020	123 Taft Ave		Laptop	620
Jane Doe	555-650-3991	6080 Elm St		Electric Razor	749
Sven Fishman	555-415-3947	4098 4th		AirPod Pro	530

By examining the column names and a little of the data, a human would have no problem understanding the meanings of the columns and how the data in one table compares to the other. They could further identify data that exists in one table but not in the other, thereby identifying the total scope of the aggregate data. By contrast, this is not a standard feature of databases. As a result, humans spend a lot of time performing this task. Data Scientists often say that they spend 80% of their time just wrangling data[2].

LinkIt will free them from this drudgery while also making integrated data available sooner, with higher value. The basic *LinkIt* workflow involves the following phases:

1. **Loading** a small sample of data from each of their data sets
2. **Classifying** the data and metadata to determine the semantic meaning of each column of data (e.g name, address, phone number, social security number, etc.) and cataloging the results.
3. **Finding linkages** of columns of data across datasets (e.g. there is a customer name in both of these tables)
4. **Viewing/Editing** the Catalog to allow a user to understand all of their data and fix any errors or omissions in the automated process.

When the process is complete, users will be able to refer to the catalog to generate rich queries over the composite data set. One can imagine that the Catalog (in machine readable form) could be used by another piece of software to automatically create federated queries across the disparate underlying data sets, but that is beyond the scope of this project.

3. Learning Objectives

1. **Requirements Collection and System Design:** This project will require students to create/gather detailed requirements and develop a system design that spans all the phases mentioned above, including architecture, data analysis, rule-based matching[3], and UI/UX.
2. **Elements of Extensibility**: A key aspect of this system is extensibility. The design must allow for new classifiers to be added to detect/infer new types of data. Students must examine how an extensible architecture impacts the implementation and performance characteristics of the system and understand how APIs should be built to provide for extensibility.

[2] They spend the other 20% of their time complaining about it.
[3] Machine learning could be used, but is by no means necessary

3. **Data Schema Fundamentals**: Students will need to understand the fundamentals of how data sets are typically organized and what is involved in querying across them.
4. **System Integration and Test**: Students must understand the process of integrating components developed by other team-members and testing the integrated system.
5. **Communications**: The team must be able to describe the problem being solved, how they solved, and why it is valuable (business value).

4. Major Components

The major components of *LinkIt* correspond to elements of the workflow listed above: Load, Analyze, Catalog, and Edit.

- Classifier:
  - The classifier analyzes the data samples to determine the likely type of the data. This must be an extensible architecture consisting of a framework and plug-in type detectors. See Figure 1 – Classifier Framework and Plugins.
  - It is the classifier's job to populate the catalog
- Linker: Once the classifier is complete, the linker examines the different data sets to determine whether there are columns that can be linked. It updates the catalog with these linkages
- Catalog
  - The catalog is the persistent data structure that maintains the data description that has been discovered by the Classifier and which will be editable by the editor.
  - It is not an active part of the workflow, but rather a repository.
- Viewer/Editor
  - The editor allows the user to view the Catalog as created by the Classifier and make any additions or corrections as they see fit.
  - The editor will show both data classifications (e.g. Column A from dataset X is a Social Security Number) and linkages (e.g. column A from dataset X is the same type as column H from dataset Y).

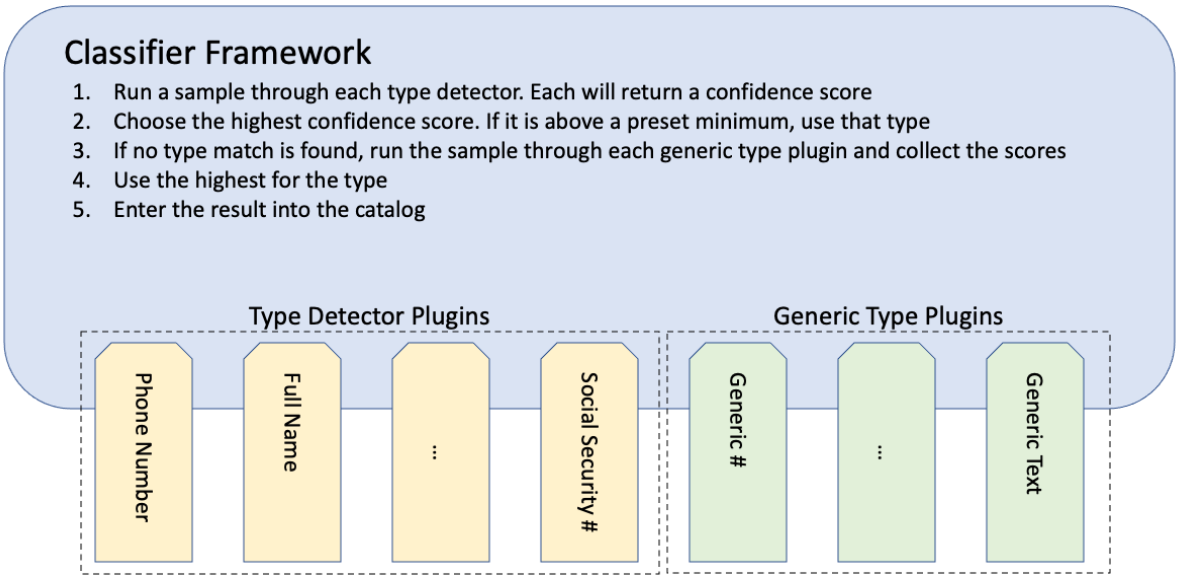

Figure 1 – Classifier Framework and Plugins

5. Deliverables

Deliverables are broken into Core Features, Completeness Features, and Stretch Goals. The Core Features are required to prove the concept. The Completeness Features. Are needed for a minimum viable product (MVP), and the Stretch Goals are nice-to-have features or follow-on work.

All code and documents related to the project must be maintained in github or another group accessible common repository.

Core Features

Feature	Description
Data Architecture	The team must develop and agree upon data formats for the loaded sample data/metadata, and for the Catalog. These may be stored in a database or in flat files (e.g. JSON)
Classifier Framework	A framework that can process the loaded sample data and populate the catalog. It must be able to execute the workflow shown in Figure 1. It must also be able to find and load the classifier plugins.
Minimal Classifier Plugins	Data detector plugins must be created for a small handful of common data types. They need not be optimized, and they can use existing classifier code if desired. Generic data detectors must also be provided.
Reporter	If the Editor is not included in the project, there must be a way of reporting on the results of the classification process. E.g. which types were discovered and which linkages.

Completeness Features

Feature	Description
Viewer	A UI (built with any set of desired technology) that allows the user to see the results of the Classifier as represented in the catalog.

Stretch Goals

Feature	Description
Editor	An extension of the Viewer that allows users to edit the Catalog. [Appropriate when 3 or 4 teams are working on the overall project]
Sensitive Information Identification	An extension to the Catalog and Data Detectors that allows certain types to be classified as sensitive information (e.g. PII, health data, financial information, etc.)

6. Sponsor Commitment

The sponsor is committed to providing students with a valuable learning experience, starting with a kickoff session to explain the project, including motivation and goals, in more detail so that the students have a deep understanding before they launch their work. On an ongoing basis, the sponsor will be available for in-person or video sessions for a minimum of one hour per week for the duration of the project. The sponsor can aid in all aspects of the project including requirements, architecture, data design, software development, testing, and integration. Further, the sponsor will provide detailed explanations of any concepts with which the students are unfamiliar.

7. Summary of Requirements and Expectations

The table below provides a summary of the proposal's requirements and expectations.

Requirement or Expectation	Description
Real World Character	This project is intended to give the students an experience of creating an extensible architecture which is an important real-world skill. It also exposes them to important aspects of the data challenges faced in many real-world projects. Furthermore, it requires multiple components to be integrated which is the case in every real-world project.
Strong Sponsor Commitment	The sponsor is available for a minimum of weekly one-hour meetings in person or by video and is happy to go beyond that as appropriate.
Project Size and Scope	The project scope is flexible and can be adjusted to meet the requirements / constraints of the class as discussed in *Section 5 Deliverables.*
Language/Platform	Students may use which ever languages, frameworks, and platforms they find most expedient. Different components need not make the same choices as long as they are interoperable. Students are strongly encouraged to use existing open source components anywhere it makes sense.
Project Management	There is no specific toolset suggested to define/track/communicate Epics/Stories. GitHub (or similar) should be used for source code management, as well as issue and task tracking.
Interaction Support	The sponsor can provide technical advice and direction as well as refinement of requirements and changes to scope as necessary. The sponsor can also provide guidance on the overall architecture and how similar problems are solved in industry.
GUI Interface	In its minimal form, not GUI is required, but in that case a reporting function is needed (see Deliverables)
IP / NDA	All IP will be labeled under a commercially acceptable open-source license such as Apache 2.0 or MIT. No NDA is required. Publishing all designs and artifacts is preferred.

8. Appendix

Data Detectors

It is suggested that the following data detectors be provided as a baseline, but more types should be added as time permits:

- Name
  - First Name/Last Name
  - Full Name
- Phone Number
  - Number
  - Type (mobile/home/office/etc)
- Address
  - Street Address
  - City
  - State
  - Postal Code
  - Country Codes
- Social Security Number
- Email address
- URL
- Social media handle (e.g. twitter, Instagram, facebook, mastodon, etc.)
- Currency (global)
- Credit Card Information
  - Number
  - Expiration
  - CVV
- Generic Text
- Generic Number
- Generic Date / Time
- Generic ID: Often datasets contain columns that are simply IDs used to relate between data sets. It can be difficult to identify these, but sometimes the column name can give a hint.

Note that some of these, like “First Name” may be region dependent. For this project is fine to make them work for the US only.

All data detectors should be plugins rather than some as plugins and some built into the classifier.

Data detectors are provided the column name for each data sample and may use it in developing their confidence score.

The implementors must decide whether data detectors are given all the samples for a column at once or called multiple times with one sample at a time.

Team Composition Suggestions

This project does not have a networking component so may not be of as much interest to students with that concentration. CS and Software Engineering students would probably be best suited to it – particularly any who are interested in databases, analytics, or data management.

Depending on how many students are available, the project may be scaled back to two teams of 4 focused on the Classifier and Plugins, or scaled out to 3-4 teams to tackle more of the components. One possible breakdown for a 4-team project could look like this:

Team 1: Classifier Framework

- Define the Catalog format (*shared task*)
- Create the classifier framework
- Develop and test a dynamic loading framework for the plugins
- Provide an ability to run the classifier and update the catalog
- Integration and test of all components (*shared task*)

Team 2: Plugins

- Define the Catalog format (*shared task*)
- Develop and test plugins for generic content types
- Develop and test type-specific plugins
- Ability to designate a field or group of fields as Personally Identifiable Information (PII)
- [Flesh out]
    - Multi-value columns: Don't know what this is, but there a limited number of the same few options (e.g. daily, weekly, monthly)
    - Composite Fields (e.g. these four fields are an address)
- Integration and test of all components (*shared task*)

Team 3: Linker

- Define the Catalog format (shared task)
- Examine data from the catalog to find linkages and potential linkages
- Update the catalog with the linkage information
- Use data from across data sets to refine catalog contents

Team 4: Viewer/Editor

- Define the Catalog format (shared task)
- Write the app that allows the user to view the catalog contents
- Add editing functionality to allow the catalog to be updated.
- Integration and test of all components (*shared task*)

A 3-team version could be created by dropping the Linker component or the Viewer/Editor component. A 2-team version would drop both of those components. The first two are needed to deliver a minimum viable deliverable.