Building a Record Linkage System for Entities in the Investment Advising Industry

Alex Markules

Masters of Science - Data Science and Analytics

Internship Site: SalesPage Technologies, Kalamazoo, MI

Semester of Completion: Winter 2023

Table of Contents

Internship Objectives

My main goal during this internship experience was to solve a problem which can be stated as: “Given two sets of entities from different sources, find the pairs of entities from the two datasets that are the same”. This is generally thought of as a record linkage problem,[3] where the entities in the two datasets are “records” and the matched pairs of records are “links”. This can seem like a simple problem to solve when you start working with small datasets, only comparing a few records to find links, but it becomes quite computationally intensive as the datasets grow. Depending on the available data, it can also be difficult to determine whether two records are a match. In many cases, no deterministic solution exists, so a probabilistic approach needs to be taken.

Another important goal of my internship experience was to explore the different software tools available to assist this record linkage problem. The central tool that I used was a python library, appropriately named Record Linkage.[1] However, several other tools and libraries ended up being an important part of the solution I built. The problem space I worked in during this internship was to match entities – firms, offices, and individuals – from data on investment advisors. The data came from a variety of different “data packs”, which were usually spreadsheets from various companies in the industry. These spreadsheets were provided in a wide range of formats, so software tools for standardizing the data and measuring distances between values became an important part of the solution.

My third goal in this internship was to build a production ready solution that could be used in the real world. This included using software development best practices like automated testing, teaching users how to use and get the most out of the tool I developed, and making the solution configurable so it could be adjusted for varying needs.

Introduction

I completed this internship project while working as a Data Engineer on the Data Services Team at SalesPage Technologies. SalesPage is a small technology company based in Kalamazoo, Michigan. We offer a variety of technical solutions to the Asset Management industry, with our core product being a customizable CRM Solution, which offers a variety of features to help Asset Managers understand which investment advising companies and reps are buying and selling their products. The Data Services team at SalesPage is responsible for a set of tools which are used for loading and processing information from a variety of sources, standardizing the relevant data in a way that it can be viewed in the CRM tool by end users.

We work closely with the LumaSuite Operations team, which is responsible for using our tools to standardize data from various "data packs" provided by our customers. The data packs provided by our clients usually provide some portion of the data that exists in a client's golden copy, for some portion of the entities in that golden copy. These data packs also provide additional data that might not be in a client's golden copy yet, like recent trading data, and updated contact information. These additional data points are the main value add for matching a data pack to a client's golden copy. A big part of the LumaSuite Operations team's task involves resolving entities from these data packs to the same entities in a client's "golden copy" data, which has been validated and standardized as much as possible.

There are three main entity levels that both the golden copy and data pack information provide - firms, offices, and individuals. A firm is a company that offers investment advising services, an office is a branch location that one of those companies operates from, and an individual is a person who works at one of those companies. Each of these entity levels provides different types of information which can be used for record linkage, and each data pack provides a different

subset of that information, often in slightly different formats. These data packs are often provided to our clients on a monthly or quarterly basis, so record linkage needs to be performed on a recurring basis against updated data sets.

Description of Work

Data Formatting:

The entity records created from various data packs are largely standardized by an ETL tool which the Data Services team develops. This tool maps fields from the source files to standardized fields and loads them to the client's database. Oftentimes though, the data pack is missing some of the fields that appear in the standardized table. There are also some fields, like company name, address, and individual's name, which are formatted differently when they come from different sources. For example, a company's name might be listed as "A.C.M.E. Corp." in one data source, but "ACME Corporation" in another, an address might be "123 Main St NE" in one, but "123 Main Street" in another, and an individual might be listed as "Mr. Joseph Smith, CFA" in one, but "Joe Smith" in another. These differences might be easy for a human to resolve one by one, but when dealing with data sets that contain thousands or millions of records, it becomes an impossible task to accomplish manually.

The automated solution that I developed with other members of the Data Services team was to format these dynamic fields in as similar a way as possible: removing punctuation, standardizing capitalization, and expanding abbreviations as much as possible. When that approach fell short, we added the use of Natural Language Processing[6] tools to assist in the

standardization process. We used a slightly different NLP tool for each field, but the overall approach was essentially the same. Each NLP tool was trained to identify the different parts of the field it was intended to format, tagging them appropriately. For example, an address would be split into parts like address number, street name, address line 2, and postal code. This allowed us to filter down to the parts that most effectively identified an address. Similar approaches were employed for separating out the "company type" portion of company names (things like "LLC" and "Inc."), and prefixes and suffixes for people's names.

First Iteration of a Solution:

With formatted data in hand, I began developing a record linkage solution, starting with a single entity type and data pack. This approach allowed me to get familiar with the problem I was trying to solve, so that I could later generalize that solution. For this initial, exploratory solution, I used office data from a new data pack that a client hadn't done any manual matching on. This provided a large set of records that needed matching, and a particularly difficult version of the record linkage problem, since offices lack any unique industry identifiers, which could be used for exact matching.

Even with the formatting, it was apparent that some fields would be difficult to match exactly across data sets. In particular, slight variations in branch names and addresses prevented exact matching from working as expected. I solved this by implementing methods for measuring the distances between values in these fields. One distance measure my mentor suggested, which was particularly helpful, was the use of Levenshtein distance[2] to compare similarity between character strings. It works by computing the minimum number of characters to insert, delete, or exchange in a string to transform it into another one. This approach helped to identify office names

and addresses which might be spelled slightly differently, recognizing the similarities between values like “Applegate Road” and “Apple Gate Road”. It also provided a framework where exact matches were ranked higher than close matches, and the more two values differed, the lower they were ranked.

The problem with this more probabilistic solution is that I was trying to make comparisons across two data sets which each had hundreds of thousands of offices. Calculating that distance was a somewhat expensive procedure, and doing so for each of 100,000 offices in one data set, compared to each of 100,000 offices in another data set resulted in 10 billion comparisons to make. I needed a way to reduce the number of comparisons I needed to make to determine matches. The solution here was to use a python library called Record Linkage Toolkit.[1] This library provided a means for indexing two large sets of data. There were two types of indexes that I found helpful: blocking indexes and sorted neighborhood indexes. Blocking indexes were helpful for breaking the data into chunks, based on exactly matching values in a field; this allowed me to separate offices, only comparing the offices in each dataset that were in the same state, postal code, or firm. Sorted neighborhood indexes were useful for limiting comparisons to those offices that had similar, but not always exactly matching values in a field. It sorts values in alphabetical order, meaning that only the records with the smallest Levenshtein distances for fields like address and office name were compared. These approaches helped reduce the number of record comparisons from 10 billion to about 1 million, a massive performance improvement.

Throughout this initial round of experimenting with different record linkage techniques, I needed to try a wide variety of formulas for producing matches. I experimented using a Jupyter notebook, which allowed me to run individual code chunks repeatedly, making slight adjustments each time. I also needed a way to measure the success of each approach. For that, I took a small

subset of the data, in which I could link records by manual review, as well as by using the automated process I was developing. This allowed me to measure the accuracy of my results using precision and recall a technique I learned from my classes at GVSU. Precision gave me an estimate of how many records were linked inappropriately (also known as the false positive rate). Recall helped to estimate how many records were not linked, which should have been (false negative rate).[5]

Generalizing the Model:

After finding a solution that worked for offices from that one source, my next task was to come up with a more general model that could be used for different sources, which had different fields available. I needed solutions for firms and individuals as well. I found that it was most effective to use slightly different combinations of fields for different sources, so I computed a few different scores for each match, taking the most effective score in each scenario. In the context of firms and individuals, I also found that there were some deterministic matches that could be made, since these entities often had unique industry identifiers, like CRD numbers.[4] These approaches allowed me to produce accurate links between entities from a variety of sources, with a single linking process for each entity type.

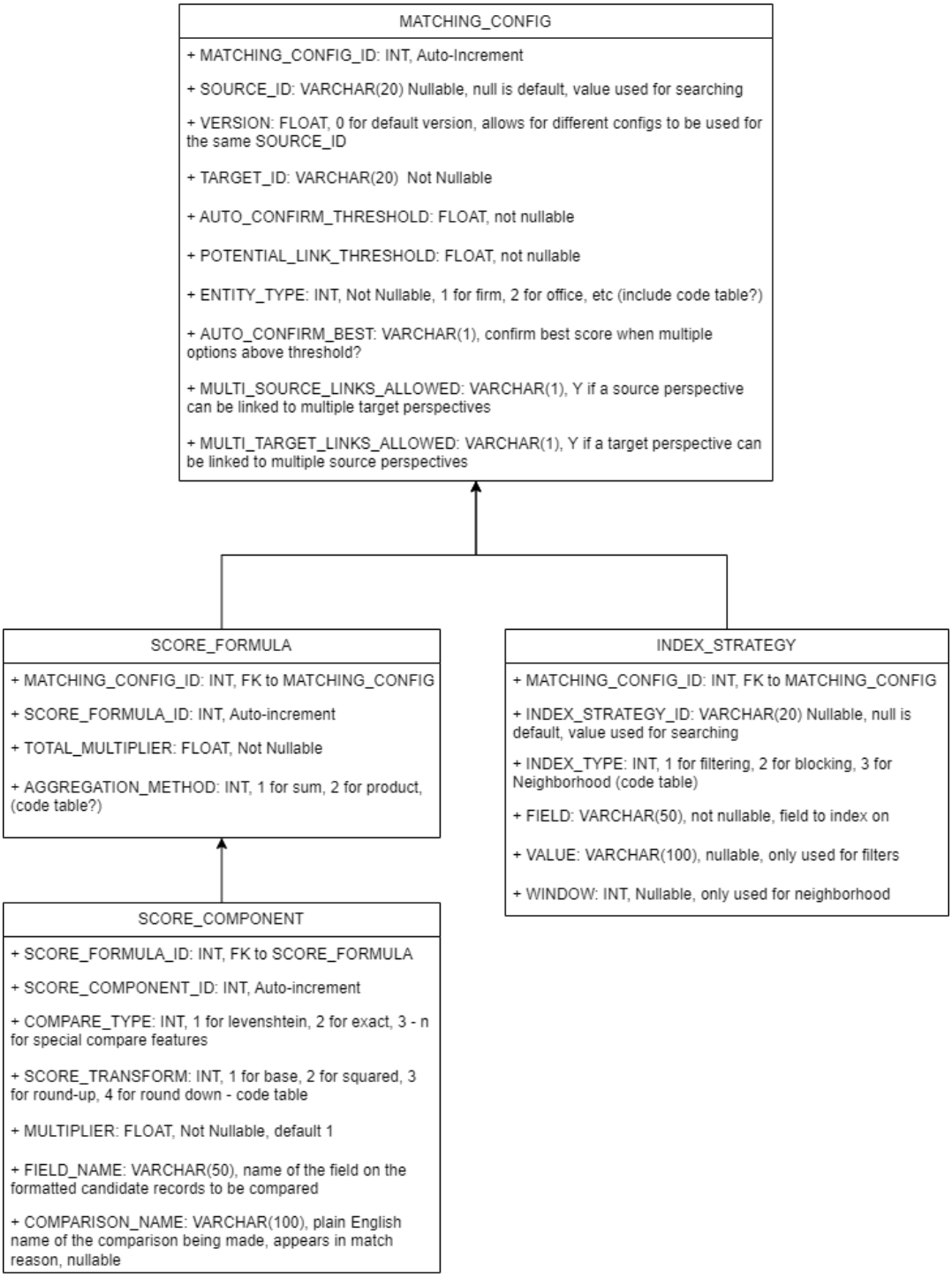
The addition of different sources and entity types also meant I had to adjust the indexing strategy used. Firms needed less indexing, since there were fewer of them, while a larger data set for individuals meant more indexing was needed. I again took an iterative approach, indexing on different fields until I found something that would work for each.

Producing a Production Ready Solution:

Now that I had a solution which could be used to link a wide variety of records, the next step was to turn it into something the LumaSuite Operations team could use in their day to day work. This meant packaging all the code I had written as a single batch application, which could be run on demand. Users would provide the entity type and data source they wanted to link records for, and the tool would be responsible for fetching the relevant records from a database, formatting them, and generating links. To help reduce the amount of manual review needed, the tool needed to distinguish between deterministic matches made by matching on a field or fields exactly, and probabilistic matches which would need manual review. It accomplished this by saving deterministic matches back to the database, and producing a report of the probabilistic potential matches that the user could review before uploading to the same table as the deterministic ones.

With such a large scope of responsibility, it was important to make sure that all the components of this tool were working as intended. To do this, I used software development practices such as unit tests and end to end tests. This helped me to quickly identify when a component wasn't working as it should yet, and to make sure that all components were working together to produce the desired results.

As the tool moved to a fast-moving production environment, where data sources can change, and new sources can be added from month to month, it became important for the tool to be configurable. There were times when the record linkage process wasn't working exactly as intended, usually because it was taking too long to find links, or not finding all the links it should for a new data source. My experiences prototyping the tool became extremely helpful here, giving me an idea of which parameters we might need to change on short notice. The following relational diagram shows the table structure I built to configure the tool as needed.



The configuration table structure outlines 4 main components. The top level component is MATCHING_CONFIG. One entry in this table will be selected each time the record linkage tool

runs. It identifies the type of entity being linked and the minimum score threshold for a link to be confirmed without review, among other options. The INDEX_STRATEGY table tells the tool how to perform indexing. It can specify fields for the blocking and sorted neighborhood indexes mentioned above. The SCORE_FORMULA table can have one or many records for a given MATCHING_CONFIG record. Each entry in this table represents a different combination of fields to link records using, and the maximum score that combination of fields can produce. The SCORE_COMPONENT table contains information about how to compare each of those fields, using exact comparison or Levenshtein distance among other options.

This structure allows us to make adjustments on the fly, when issues are noticed. It also means we can adjust the tool without having to make a code change directly. It is still important to test these changes before implementing them in a live production environment. We hope that in the future, users will be able to make and test these changes without the intervention of an engineer. This would provide a solution that domain experts can use without ever having to look at or understand the underlying code.

Internship Discussion

Were the objectives achieved?

Reviewing the objectives I set out to accomplish in this internship experience, I managed to make great progress on each of them. I got to explore the problem of record linkage, learning about tools like indexing and custom distance measures, which helped me develop a solution to that problem. I also got to explore some of the software packages that are available to help solve this problem, from basic python tools like pandas and numpy, to prototyping techniques like

jupyter notebooks, to libraries intended to solve the exact problem I was faced with like the record linkage toolkit. Finally, I got to work with the business stakeholders on the LumaSuite Operations team, developing a solution to a problem they had, and making iterative improvements as they used it.

What skills (scientific and professional) were learned during the internship?

I put many of the concepts I've learned in the Masters of Data Science and Analytics program to use during this project. Several classes in the program have focused on model validation, which was particularly helpful as I iterated towards a useful solution to this problem. Data cleaning, and indexing strategies were also major concepts in the DSA program that I put to use during the internship project. I also got to experiment with natural language processing tools, which weren't covered in the DSA program, but provided an interesting continued learning experience.

In terms of professional skills, this experience has provided me with a great opportunity to practice working with the people who use the tool. Explaining to them how the tool works, and working with them to make improvements based on their use was a great way to practice valuable skills. I also got the chance to put professional skills like test driven development and project planning to work in a real world environment. Lastly, the experience gave me a far better understanding of the asset management industry, learning about things like financial identifiers, trades and assets.

Did the PSM coursework properly prepare the student for the scientific content of the internship?

There were certainly aspects of the coursework for my degree which were helpful in this internship experience. As mentioned above, concepts such as data cleaning and model validation were particularly helpful. The coursework in the DSA program also helped improve my skills in reading software documentation, which was helpful as I explored new tools, concepts, and libraries to help solve the problems I was faced with.

Did the PSM coursework properly prepare the student for the professional content of the internship?

The PSM coursework reinforced good communication habits, which were helpful in communicating with all the people involved in this project. It also helped my professional writing skills, which were needed in order to plan and document the project.

What challenges did you experience during the internship? What could you have or did to overcome them?

The biggest problem I faced when starting this project was that the scope of the problem seemed huge. Through help from my mentor, [redacted] and the rest of the Data Services team, I was able to break it into small chunks and accomplish them one at a time, delegating to others and asking for help when needed. That allowed me to produce a working final result in a reasonable amount of time. Other problems I experienced were generally more technical in nature, such as how to improve the performance of the tool. In those instances, I drew from many different resources, ranging from reviewing notes from my coursework, to asking coworkers, to going out and reading papers and software documentation related to the issue I was experiencing.

What is your overall evaluation of the internship experience?

I had a really enjoyable experience during this internship. I got to work on a project that was somewhat related to my day to day work at SalesPage, but which involved building an entirely new solution to a problem I wasn't aware of previously. The record linkage problem I worked on is also one that has applications in many other contexts, which I might have an opportunity to work on in the future. [redacted] and the rest of the Data Services and LumaSuite teams were very helpful and provided knowledgeable suggestions when I needed it.

References

1. de Bruin, Jonathan. “Python Record Linkage Toolkit Documentation.” *Python Record Linkage Toolkit Documentation - Python Record Linkage Toolkit 0.15 Documentation*, https://recordlinkage.readthedocs.io/en/latest/index.html.
2. Gad, Ahmed Fawzy. “Measuring Text Similarity Using the Levenshtein Distance.” *Paperspace Blog*, Paperspace Blog, 9 Apr. 2021, https://blog.paperspace.com/measuring-text-similarity-using-levenshtein-distance/.
3. Judson, Dean H. “Record Linkage.” *Record Linkage - an Overview | ScienceDirect Topics*, 2005, https://www.sciencedirect.com/topics/computer-science/record-linkage.
4. Kagan, Julia. “What Is FINRA's Central Registration Depository (CRD)?” *Investopedia*, Investopedia, 7 Dec. 2022, https://www.investopedia.com/terms/c/crd.asp.
5. Shafi, Adam. “How to Learn the Definitions of Precision and Recall (for Good).” *Medium*, Towards Data Science, 19 Apr. 2022, https://towardsdatascience.com/precision-and-recall-88a3776c8007.
6. “What Is Natural Language Processing?” *IBM*, IBM, https://www.ibm.com/topics/natural-language-processing.