Final Report

CIS*4910 – Record Linkage

Erik Zorn-Wallentin

Table of Contents

Table of contents for Final Report

i Introduction

I took on a project with two professors and I was directed to perform a record linkage on an 1871 Canadian census of all of Canada, and an industrial census that was also performed in Canada in the same year (databases provided by Dr. Inwood). The 1871 census contains over 3.6 million records, and the industrial census contains almost 50 thousand records. The goal of this semester was to link as many records in the industrial census as I could to the 1871 census.

Starting this project, I did not have any knowledge in the data mining field or on record linkage. This project was focused primarily on learning the concepts and applying the theories learned. I became interested in this project and data mining in general after talking with the course coordinator Luiza after she gave a small story about data mining. The story was about grocery stores that entice you to use their “point cards / discount cards” for cheaper groceries and in exchange, they collect data on your shopping habits. In this story, she described how they analyzed the shopping habits of customers and found out that customers that buy bread, will also usually buy milk. Since the grocery store has information about the types of products you will buy, they will place these products at the opposite ends of the grocery store to get you to walk across the entire store; this provides the customer an opportunity to view all the other products that are on sale and may entice them to make additional purchases. I was very interested in this story and wanted to know more, and this is how I began this record linkage project with Luiza.

ii Record Linkage

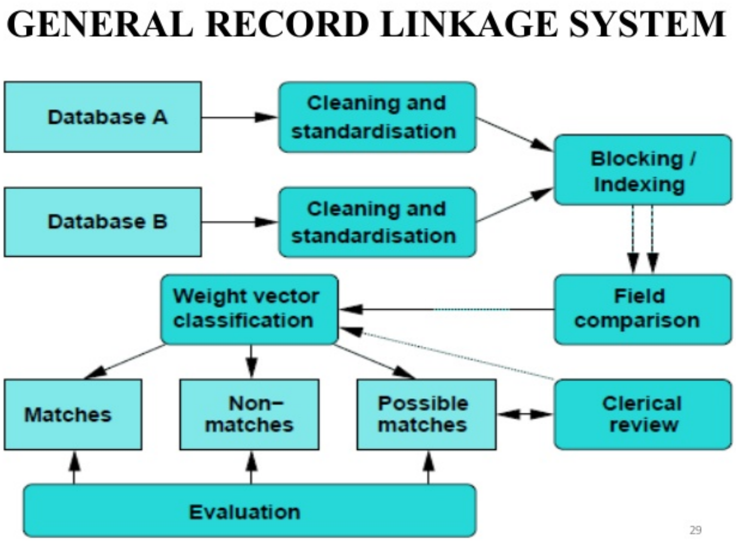


Figure 1 – Record Linkage Process [2]

I will go over a brief summary of record linkage, and Figure 1 above shows the general process of record linkage. Record linkage is the process of identifying and matching records from the same entity in one or more databases. This is also known as data matching, entity resolution, or object identification [1].

TABLE I. NAMING DIFFERENCES [3]

Record 1	Sam Smith	54 Bellamy St.
Record 2	Dr. Smith, Sam	54 Bellamy Street
Record 3	S. Smith	54 Bellamy Road

Table I shows an example of naming differences that could occur in the data. “Smith” is a common last name and “Sam” is a common first name. The way the data was entered/recorded could vary, despite them being information collected from the same person. The person may have been recorded as “Dr.” in one census and not the other, or there could be differences in the spelling of the name, where only the first letter of the first name is recorded. “Sam” could also be a male or female name. Moreover, the name

“Sam” could be represented in a variety of ways, such as “Samantha” or “Sammy”. Ultimately, all of these discrepancies need to be considered to properly find matches in the data. Record linkage methods are used to analyze these differences and perform algorithms to match the records. Table I showcases some of the challenges in performing record linkage, and these challenges are present in this project which I will describe in more detail later in this report.

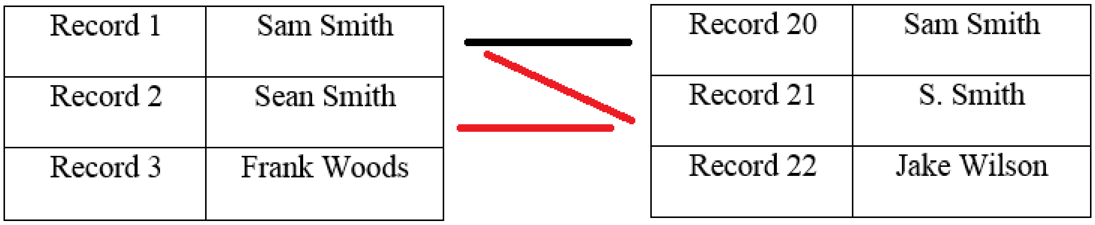
Record linkage is performed in several steps, as shown in Figure 1. The first step is cleaning and standardization of the databases. The data needs to be enriched and cleaned prior to being linked. For example, the naming differences described above could be standardized (first and last names follow the same format), or if there are any empty names in the database, they could be removed.

The next step is blocking/indexing, which is very important in this project – otherwise, I would not be able to handle the 3.6 million records in the 1871 census all at once without running into memory issues with the Python programming language. Since memory issues are a problem, there needs to be some form of blocking method invoked to handle large amounts of data. Blocking is a method for filtering the data into smaller groups that contain the same entity, also known as the blocking key. The blocking key in the data is the “district” field, where data is sorted based on a district number between 1 and 206. There is also a “sub district” within the district (Ex. “Ward One”) and within a sub district, there is a “ced”, which is an enumerator number of the sub district (Ex. A-1); these fields may be used for additional blocking of the data.

After blocking is complete, there is a field comparison performed on the records, to determine if there is a match, non-match, or a possible match. A comparison of records could be an exact string match on the full names from each database, and if both strings are a match, it is considered a true match. If both strings are not a match in the comparison, they will be considered a non-match in this case [3]. When there is one character difference in the spelling of the name, the result is a non-match. Different algorithms and techniques need to be used to determine what is a match, non-match, or a possible match on the data.

TABLE II. 1871 CANADIAN CENSUS

TABLE III. 1871 INDUSTRIAL CENSUS



Record 1	Sam Smith
Record 2	Sean Smith
Record 3	Frank Woods

Record 20	Sam Smith
Record 21	S. Smith
Record 22	Jake Wilson

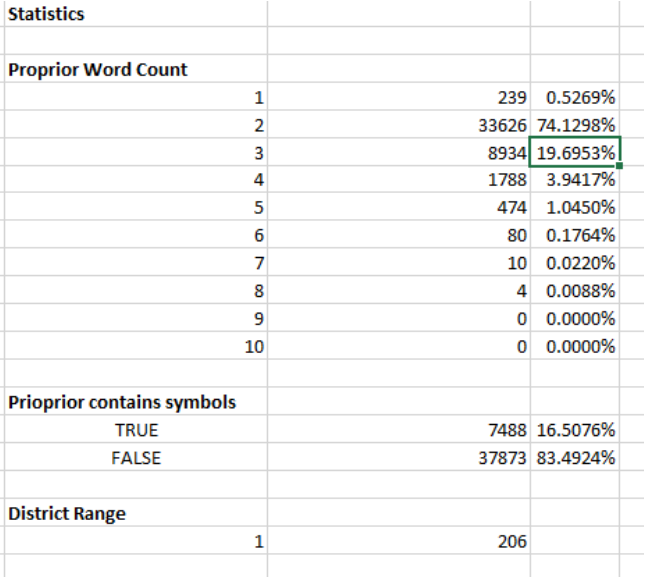
Using Table II and Table III as an example, the black line indicates a match between “Sam Smith” from Record 1 in one database and Record 20 in a different database. The red lines indicate “S. Smith” from Record 21 in Table II is a possible match to “Sam Smith” from Record 1, or “Sean Smith” from Record 2. More information is required in cases like these, such as age, sex, etc. to confirm whether it is a match or not. The last records in both the tables represent non-matches.

iii Data

This project contains two databases, with the goal of linking the records from one database to the other. To perform record linkage, you need to first find the common fields. The primary and most important common field in both databases is “full name”. Another common field is “sex”, which will help enrich the possible matches in the future based on full names. To filter the data, other common fields are “district”, “subdistrict” and “ced”, which may be used to block the data into groups and perform analysis on smaller sections of the data. The common fields of “full name” and “sex” were used to compare both the databases, because they were the only common fields between the databases that could be used to enrich the record linkage. Naturally you would want to compare “age” and “career” to confirm the matches, but these fields were not common in both databases and were unable to be used. There are also around 30+ fields in each of the databases that were either not beneficial to the record linkage, or not common in both the databases.

When Dr. Inwood provided the databases, he informed me that the 1871 census data was richer and a lot cleaner in comparison to the industrial census, and I can confirm this with examples provided later on of both databases. The 1871 census had no issues in the data and did not need to be altered at all to perform the record linkage successfully. On the other hand, the industrial census was split up into two databases that needed to be combined into one database as the fields were not organized in the same manner, there were errors in the data (Latin characters in street numbers ex. 1Õ2 Deer Cres should have been 102 Deer Cres.), and the recording of company names in the industrial census was different from the naming style used in the 1871 census.

Before performing record linkage on the databases, the professor requested me to view the data and perform an analysis to decide how to proceed with the record linkage process. Statistical analysis was targeted on the "proprior" ("proprietor" is the correct spelling) field in the industrial census and I obtained the following results below in Figure 2.



Statistics		
Proprior Word Count		
1	239	0.5269%
2	33626	74.1298%
3	8934	19.6953%
4	1788	3.9417%
5	474	1.0450%
6	80	0.1764%
7	10	0.0220%
8	4	0.0088%
9	0	0.0000%
10	0	0.0000%
Prioprior contains symbols		
TRUE	7488	16.5076%
FALSE	37873	83.4924%
District Range		
1	206	

Figure 2 – Proprior Statistical Analysis

The "proprior" field in the industrial census contains two words 74% of the time (ex. "Joseph Brown"). From the results above there are rare cases where a name can contain up to 8 words (ex. "Thompson Wellsley & Wood William"). The name field in the 1871 census was recorded as “First Name, Last Name”, which differs from the convention used in the industrial census (“Last Name, First Name”). Proper analysis is required to properly extract the names from the “proprior” field. Another problem with the industrial census “proprior” field is that the data may also contain symbols (ex. &, ?, /, etc.), which do not follow the format of the 1871 census.

Both databases have a common district range from 1-206, and since the 1871 census has over 3.6 million records, filtering will need to be performed to make the record linkage task more manageable in a timely manner. By choosing specific district ranges, it will allow for quicker implementation of the record linkage algorithms and easier debugging of the results.

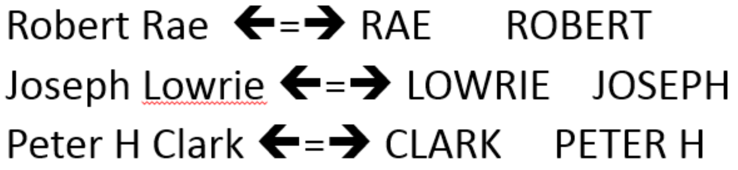
After completing several passes, we examined the non-matches to see if any were possible matches. Since a common field is the “sex” field in both databases, a frequency analysis was performed on this field to determine the categories of the data. There were 37764 “male” records in the database, 2719 “female” records, 14 “male + female” records, and 4864 records had an unknown identity and were unable to be resolved by the professors or I.

The data itself was formatted differently in both the databases for the “proprior” field. The 1871 census names are organized from first name to last name in lowercase, while the industrial census names are organized from last name to first name in uppercase. The industrial census data was cleaned by performing such tasks as removing the symbols, changing uppercase to lowercase, and changing the order of the data so that they could be matched properly with the 1871 census.

iv How I Solved the Problem and Results

After performing the analysis, I followed the recommendations by the professor to first filter the data by the “district”, “subdistrict” and “ced” fields. I performed several passes on the data, where each pass accomplishes a different type of analysis on the data.

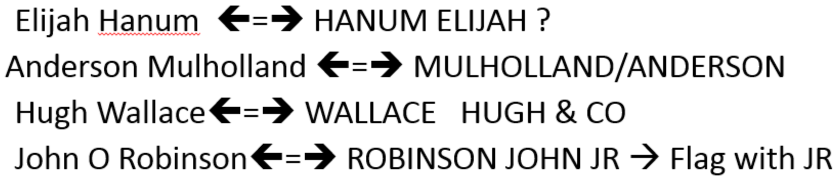
I followed the methods depicted in Figure 3 for a sample of unmodified data from both databases. The spaces were stripped first, and were then standardized so the fields in both databases followed the same format. The first pass on the data was performed by finding exact matches of strings in both the databases, following stripping and cleaning the data.



Robert Rae ←=→ RAE ROBERT
Joseph Lowrie ←=→ LOWRIE JOSEPH
Peter H Clark ←=→ CLARK PETER H

Figure 3 – First Pass Sample Matches

The first pass method matched 23142 records out of 45361 which is 51.01% of the records, and 22219 records remained non-matched. For the matches, there are 11113 one-to-one matches, and 54871 many-to-one and one-to-many matches.



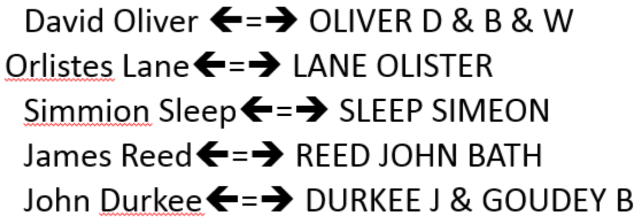
Elijah Hanum ←=→ HANUM ELIJAH ?
Anderson Mulholland ←=→ MULHOLLAND/ANDERSON
Hugh Wallace←=→ WALLACE HUGH & CO
John O Robinson←=→ ROBINSON JOHN JR → Flag with JR

Figure 4 – Second Pass Sample Matches

The second pass was designed to handle the symbols in the data and multiple names in the “proprior” field. An example of symbols being used in the names is shown in Figure 4 above. For the second pass, I would strip out symbols such as “/”, or “?”, etc. and search for a first and last name from the stripped results. This method matched 2341 records out of 22219, which is 10.53% of the records, while 19878 records remained non-matched. For the matches, there are 1651 one-to-one matches, and 3956 many-to-one and one-to-many matches. Every match in this second pass had a symbol in the name, so I also appended a record comment field that gave more information about the match. For example, if the match contained a “JR” in the full name, I would provide a comment to assist in finding this record in the future.

Once I completed the second pass, the professor requested a few features that should have been implemented from the beginning. The matched records should be organized into one-to-one matches, one-to-many matches, and many-to-one matches from both the databases. Doing this caused a halt in my progress as I needed to refactor most of my code in order to accomplish this task. Attempting to perform this task caused me difficulties as I could not implement it with the current fields provided. The professor recommended that I give unique IDs to each of the records in both the databases, which enabled me to complete the task. Once I implemented it, my code could not handle the large amount of matched fields from the one-to-many and many-to-one matches, and ran into memory issues when using Python. Upon further investigation, I discovered that the Openpyxl library used in Python can only write a limited number of records to an Excel sheet, which is beyond the scope of this project to fix. Consequently, I am only able to perform passes on the data in a range of 50 districts at a time, otherwise the program will crash because of the Openpyxl library limitations.

The final pass was performed on the remaining non-matches from the second pass. This was designed to obtain possible matches out of the data. The primary goal was to check the last name and the first character of the first name in both databases; if a match was found, it would be a possible match.



David Oliver ←=→ OLIVER D & B & W
Orlistes Lane←=→ LANE OLISTER
Simmion Sleep←=→ SLEEP SIMEON
James Reed←=→ REED JOHN BATH
John Durkee←=→ DURKEE J & GOUDEY B

Figure 5 – Possible Matches Samples

Figure 5 shows a sample of the results I achieved. Checking the last name and the first character of the first name was an effective way to obtain possible matches as it caught cases with misspellings in the name, cases where only one character was provided for the name, etc. To enrich the data, the “sex” field was also compared in each record to confirm whether gender was the same in both matched records. There are some problems with this algorithm as it it catches a lot of cases of siblings or family members that share the first character of the first name (ex. James Reed and John Reed are probably not the same person). Performing this method, I matched 5090 records out of 19878 which is 25.60% of the records, while 14788 records remained non-matched. For the matches, there were 1939 one-to-one matches, and for many-to-one and one-to-many, there were 16950 matches.

I described a few problems I ran into above, but a limitation of the project I have is matching records from different districts. I implemented everything based on looking at each district one by one, and if someone happened to move to a different district, I would not be able to link them. The data still needs to be filtered by districts or I will run into memory issues and will be unable to perform the record linkage.

When the results from the matches and possible matches were collected, the records from both databases were placed in one row: the 1871 census fields were placed first, and were separated by a single field from the industrial census fields. All results are provided with the submission of this report.

v Conclusion

This project was performed as an individual project over one semester and I had to learn the record linkage process and write code from scratch. Therefore, it took a while to get started and complete small tasks such as the first pass. The semester quickly came to an end, and I was only able to perform the first and second passes and obtain possible matches from the second pass. Even though I did not obtain as many matches as I would have liked, I still am satisfied with the results because this was primarily about learning record linkage and applying my knowledge to a real life project.

After observing the results and the lack of matches, I decided to examine the non-matches and realized there could potentially be hundreds of edge cases. I did not have enough time to cover most of these cases this semester. To match records from other districts I would need to refactor the code to handle the district limitation, so I that I may retrieve matches from people that moved districts.

For the future, I would make some improvements on the code. I already described that the code needed to be refactored, but the biggest issue was the Python Openpyxl library that could only handle so much data when writing results to the Excel sheet. Python list limitation was less of a problem compared to the Openpyxl library and could handle more data without issue, which made it difficult when writing to an Excel sheet in the end.

With more time, I would have performed more advanced techniques on the data, performed more analyses on the data to find edge cases in the records, refactored my code to better handle memory issues, and solved the current limitations of the project.

vi References

[1] *Febrl – A freely Available Record Linkage System with a Graphical User Interface*, Peter Christen, Department of Computer Science, The Australian National University.

[2] [Digital image]. (n.d.). Retrieved from https://image.slidesharecdn.com/pem-rls-130422205511-phpapp01/95/prescription-event-monitoring-record-linkage-systems-29-638.jpg?cb=1366664170

[3] Peter Christen: Session 1 – Record Linkage Workshop at the ADRC-Scotland, 13 July 2015 ADRNUK, Retrieved from https://www.youtube.com/watch?v=DyGonV7A_EY

[4] Peter Christen: Session 2 – Record Linkage Workshop at the ADRC-Scotland, 13 July 2015 ADRNUK, Retrieved from https://www.youtube.com/watch?v=dcNTvYDdun0

[5] Peter Christen: Session 3 – Record Linkage Workshop at the ADRC-Scotland, 13 July 2015 ADRNUK, Retrieved from https://www.youtube.com/watch?v=HAKW5tHVCmw

[6] Peter Christen: Session 4 – Record Linkage Workshop at the ADRC-Scotland, 13 July 2015 ADRNUK, Retrieved from https://www.youtube.com/watch?v=4Iv5axrAWqQ