Data Cleaning using a Matching Dependency Technique

by

Shashank Jain
B. Tech. , Gautam Buddh Technical University, 2013

A Project Submitted in Partial Fulfillment
of the Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

Data Cleaning using a Matching Dependency Technique

by

Shashank Jain
B.Tech., Gautam Buddh Technical University, 2013

Supervisory Committee

Dr. Yvonne Coady, Supervisor
(Department of Computer Science)

Dr. Sudhakar Ganti, Departmental Member
(Department of Computer Science)

Supervisory Committee

Dr. Yvonne Coady, Supervisor
(Department of Computer Science)

Dr. Sudhakar Ganti, Departmental Member
(Department of Computer Science)

ABSTRACT

In today's digital society, people are often required to enter their home or office addresses on forms available online. It is not uncommon for people to introduce some minor mistakes, such as misspelled addresses, or incorrect postal codes/zip codes. Such mistakes made by the user can be quite problematic when automated systems must process their request. For example, if a person orders something online providing the incorrect postal code in the entered address, this mistake could lead to delay in the delivery of the item or even worse, the item may remain undelivered. To avoid such situations, these systems often use a machine learning technique called 'Matching Dependency' which has been proven helpful in making recommendations for the correction of any incorrect value in the input address. This technique uses a binary search algorithm to reduce the number of cycles the process has to go through to make recommendations. Our exploration of one possible implementation of this algorithm uses our own synthesized sample data sets instead of real user input with the external data. External data has been used as the

authenticated data source to verify the user input data. We compare our synthesized user input data with the external data that is considered to be completely trust worthy. The system then makes possible recommendations based on the correctness of the user input.

The evaluation was mainly done on two different sizes of data sets, 1000 and 15000. The results had zero false negatives, few false positives, and mostly relevant recommendations.

Contents

List of Figures

List of Tables

Acknowledgments

I am highly obliged to:

Dr. Yvonne Coady, for giving me the opportunity to work under her guidance which helped me a lot through out the process of implementation of the master's project. I would also like to show my gratitude to her for her suggestions and for all the patience and encouragement she has shown to me. It has been a wonderful experience working with her.

Peter Smith, for showing me the right path whenever I needed the support to get on the right track. I am thankful to him for all the help and ideas he provided during the implementation of project.

My parents and family, who supported and encouraged me throughout the process.

Dedication

I dedicate this to my parents, my family and my supervisor Dr. Yvonne Coady, without whom this project would not have been possible for me.

Chapter 1

Introduction

1.1 Motivation and Problem Statement

Humans have witnessed a substantial decrease in mundane physical efforts with the help of computers. In other words, we are becoming more dependent on machines day by day. In the previous era, in school, government sector, private sector or hospital, all the data/records used to be physical or on papers/files. It would be completely natural to make mistakes while having this data secured on file or paper.

In today's world, all the hand written work and physical files either have turned or are being turned into soft copies everyday. It would not be a surprise to expect mistakes in the results after getting the data fed into a system by humans. Spelling mistakes, wrong value selection and other reasons could lead to such errors. The most common entered values by any human nowadays are their names, phone numbers, and addresses while signing up on any website, where a person's name cannot be standardized. The same name can be spelled in several ways as per the sound. For example, "Begbie Street" is a common street name in British Columbia, which may be spelled as "Begbee Street" or "Bagbee Street" because it would sound similar to the original street.

This report illustrates how we can avoid mistakes in the addresses of USA/Canada entered by any user on a system. We have tried to achieve accuracy in getting the address corrected in case of any mistake, which includes street name, city, province, country, and postal code.

To handle such errors, we have used a recently proposed machine learning technique 'Matching Dependency Technique' [3] which is a part of machine learning. As per the name, this technique uses a reliable external source of data which helps in verifying and validating the user input data. If the entered address happens to have any incorrect values, then this algorithm tries to make recommendations to the user with the set of possible correct addresses.

1.2 My Contribution

I started my research by going through the papers of machine learning and techniques to explore data cleaning [3] [9]. The Matching Dependency technique seemed very promising for data cleaning process, as the results were more accurate and efficient. The other techniques were able to fix partial errors. I chose the USA/Canada addresses which any user living in either country would be entering online.

I started the implementation using Python language by having sample data sets of external source of data and user input data. Here, external source data is the authorized and reliable data which I used to verify and validate the user input data. As per the name, user input data is the data entered by any user.

In this project, I took street name, city, province, country and postal code into consideration for the data cleaning process. For instance, if the user happens to make a mistake in the

postal code while entering the address, then this algorithm is likely to detect it by validating it using the external source of data and recommend all possible postal codes that satisfy the other address values i.e., street name, city, province and country.

1.3 Report Structure

This section provides the information about what each Chapter contains:

Chapter 2 provides the explanation about the technique implemented.

Chapter 3 explains the implementation strategy and the data used.

Chapter 4 shows our evaluation with the several data sets and their results.

Chapter 5 contains the possible future work for the implemented technique, and concludes the purpose and implementation of the project.

Chapter 2

Background

This chapter discusses the background of the machine learning technique, Matching Dependency which has been implemented for cleaning up the data sets that might contain errors in addresses.

2.1 Matching Dependency Technique

As per the name, the Matching Dependency technique requires a source of data which can be used as an authenticated source. This authenticated data helps in verifying the correctness of the input by matching them together for the cleaning process.

To understand it better, lets take an example. We have an external (authenticated) data source of addresses for Chicago, Illinois as shown in Figure 2.1 below.

Ext_Address	Ext_City	Ext_State	Ext_Zip
3465 S Morgan ST	Chicago	IL	60608
1208 N Wells ST	Chicago	IL	60610
259 E Erie ST	Chicago	IL	60611
2806 W Cermak Rd	Chicago	IL	60623

Figure 2.1: External information, address listings in Chicago

The user input data sets are in Figure 2.2:

Business Name	Street address	City	Province	Zip
John Veliotis Sr.	3465 S Morgan ST	Chicago	IL	60608
John Veliotis Sr.	3465 S Morgan ST	Chicago	IL	**60609**
John Veliotis Sr.	3465 S Morgan ST	Chicago	IL	**60609**
Johnnyo's	3465 S Morgan ST	**Cicago**	IL	60608

Figure 2.2: User input data sets

The user data sets in Figure 2.2 have bad input data which can be seen clearly by comparing these data sets with the external source of data in Figure 2.1. The zip code in row 2 and 3 seem to be incorrect, the business name and city in row 4 also seem to be incorrect.

We have certain dependencies which would be helpful in the clean up process of the user data using the external data. These dependencies are shown in Figure 2.3:

$$\begin{aligned}
&\text{m1: } \text{Zip} = \text{Ext_Zip} \rightarrow \text{City} = \text{Ext_City} \\
&\text{m2: } \text{Zip} = \text{Ext_Zip} \rightarrow \text{State} = \text{Ext_State} \\
&\text{m3: } \text{City} = \text{Ext_City} \wedge \text{State} = \text{Ext_State} \wedge \\
&\qquad \wedge \text{Address} = \text{Ext_Address} \rightarrow \text{Zip} = \text{Ext_Zip}
\end{aligned}$$

Figure 2.3: Matching dependencies

Here matching dependencies are the possible signals which help in identifying and correcting the bad input data by comparing them with the external source of data.

Using the matching dependencies and the external data, the results in Figure 2.4 have been achieved:

Business Name	***Street address***	***City***	***Province***	***Zip***
John Veliotis Sr.	3465 S Morgan ST	Chicago	IL	60608
John Veliotis Sr.	3465 S Morgan ST	Chicago	IL	**60608**
John Veliotis Sr.	3465 S Morgan ST	Chicago	IL	**60608**
Johnnyo's	3465 S Morgan ST	**Chicago**	IL	60608

Figure 2.4: Repair using matching dependency technique

The repaired output data sets we get with the help of the external data and their possible matching dependencies, have a correct city in row 4 and a correct zip code in row 2 and 3. The business name in row 4 could not be repaired as the external data did not have any information about the business name column.

The pseudo code for the implemented algorithm is given below:

```
Set testData = list()  #Create list of list with the test data
Set trainData = list() #Create list of list with the training data
Set indexValue = list()
testDataStreet = sorted list of street names with their index from trainData
testDataCity = sorted list of city names with their index from trainData
testDataProvince = sorted list of province names with their index from trainData
testDataZipCode = sorted list of zip codes with their index from trainData
for each data from the testData do
        streetIndexResults = binarySearch(testDataStreet, first element of data)
        Set indexValue = list()
        cityIndexResults = binarySearch(testDataCity, first element of data)
        Set indexValue = list()
        provinceIndexResults = binarySearch(testDataProvince, first element of data)
        Set indexValue = list()
        zipCodeIndexResults = binarySearch(testDataZipCode, first element of data)
        Set indexValue = list()
        commonIndex = set(streetIndexResults) & set(cityIndexResults) &
        set(provinceIndexResults) & set(zipCodeIndexResults)

        Set commonIndicesCount = dict()
        if not commonIndex then
                allResultsList = list(streetIndexResults, cityIndexResults,
provinceIndexResults, zipCodeIndexResults)
                for each attrResultList from the allResultsList do
                        for each index from the attrResultList do
                                if index in keys of commonIndicesCount then
                                        Increment commonIndicesCount[index] by 1
                                else then
                                        Set commonIndicesCount[index] = 1
                                end if
                        end for
                end for
                Set mostFrequentIndices = maximum of values of commonIndicesCount
                recommendations = get the most frequent indices from
mostFrequentIndices
                for each recommendation from the recommendations do
                        print trainData[recommendation]
                end for
        end if
end for
```

The implemented algorithm has followed the steps given below:

- The algorithm takes the user input and authenticated data in the two-dimensional list format, where every list (inside the list) has the address.

- Another two-dimensional list is created for individual address item (for Street Name/City/Province/Postal Code) separately from the authenticated data, that contains the sorted address item along with the original indexing.
- The user input data runs in a loop to get a single address at a time to verify its authenticity. Within the loop, a new list of original indices from external data for an individual input address item is created. This list is retrieved from binary search function, which takes input address item and a two-dimensional list of external address item along with their indices.
- The binary search function keeps searching for the sent input address item in a list of external address item and records all the indices of external address item into a list when the match is found.
- After getting all the separate lists of original indices for input address items, a check of common indices for all lists is done. If a common index is found among the lists, then it verifies the correctness of input address. Otherwise, the input address has errors in it.
- To recommend the correct address for a negative case, all the separate lists of original indices are put as an item into a new list. A new dictionary is created now to have all the indices as keys and their counts as values from this two-dimensional list.
- Using max function, the highest value from the dictionary is retrieved and saved in a new variable. Then a loop on the keys of dictionary runs following an if condition, where the value is retrieved from dictionary using its key. If the retrieved value is equal to max function value, then the key is stored in a list for recommendation.
- Finally, a loop runs on the list of recommendation to extract out the original indices of authenticated data to make recommendations for the specific input data. Those results are printed as recommendations.

Chapter 3

Design and Implementation

This Chapter discusses the design of the prototype, how it was implemented, the logic used for evaluation and how the desired results have been achieved to recommend the possible correct addresses.

3.1 Technology Used

The general problem was to clean user input data with the help of authenticated data. There are a number of machine learning techniques which would be helpful for cleaning up the data [9]. In this project, matching dependency technique was chosen [3].

This method is dependent on the external source of authenticated data which could be of any data type, for example –

- dictionaries
- lists
- two dimensional lists

Python and repl.it [4] (an online interpreter) were used to implement this technique.

Figure 3.1: repl.it online interpreter homepage

3.2 Data

The user input data used in this project, which contains the lists of addresses, is manually created. Even though this data may not exist in real, the format used is the same as any existing address.

In our sample data, we have considered:

- Street name
- City
- Province
- Country
- Postal Code/Zip Code

This data typically is used for USA and Canada, as every other country could have its own address format. USA/Canada addresses are pretty much similar; the only difference is in the formatting of postal code or zip code.

For Canadian postal code, it's a six-character alphanumeric string (example: ANA NAN, where A represents an alphabetic character and N represents a numeric character).

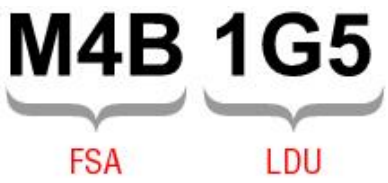


Figure 3.2: Canadian Postal Code structure

FSA stands for Forward Sortation Area [1] which is a first half portion of the postal code. It represents a specific area within a major geographic province or region. The first character of FSA identifies one of the 18 major provinces, districts or geographic areas as shown in Figure given below:

Figure 3.3: First segment of the Postal Code

The second character of FSA identifies either an urban Postal Code which is a numeral from 1 to 9 (ex. V8N), or a rural Postal Code which is numeral 0 (ex. A0A). The third character of FSA segment (E2J) in conjunction with the first two characters, identifies the exact area in the town or city or other geographic region.

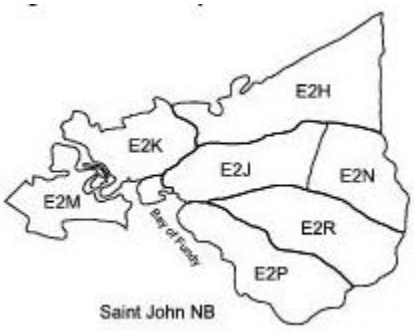


Figure 3.4: FSA example

LDU stands for Local Delivery Unit, which is the combination of last three postal code characters, identifies the address more accurately within the range of given FSA or Forward Sortation Area. In urban areas, the last three postal code characters may indicate a single

building, a specific city block, or a large-volume mail receiver. In rural area, the last three postal code characters along with the forward sortation area, identify a specific rural community.

For American zip code, it's a five-digit numeric value (example: NNNNN, where N represents a numeric character). The USA is divided into geographical areas and the first digit of zip code identifies one of these area:

- 0 = Connecticut, Massachusetts, Maine, New Hampshire, New Jersey, Puerto Rico, Rhode Island, Vermont, Virgin Islands, Army Post Office Europe, Fleet Post Office Europe
- 1 = Delaware, New York, Pennsylvania
- 2 = District of Columbia, Maryland, North Carolina, South Carolina, Virginia, West Virginia
- 3 = Alabama, Florida, Georgia, Mississippi, Tennessee, Army Post Office Americas, Fleet Post Office Americas
- 4 = Indiana, Kentucky, Michigan, Ohio
- 5 = Iowa, Minnesota, Montana, North Dakota, South Dakota, Wisconsin
- 6 = Illinois, Kansas, Missouri, Nebraska
- 7 = Arkansas, Louisiana, Oklahoma, Texas
- 8 = Arizona, Colorado, Idaho, New Mexico, Nevada, Utah, Wyoming
- 9 = Alaska, American Samoa, California, Guam, Hawaii, Marshall Islands, Federated States of Micronesia, Northern Mariana Islands, Oregon, Palau, Washington, Army Post Office Pacific, Fleet Post Office Pacific

Figure 3.5: USA Zip Code first digit details [2]

The next two digits in the zip code identify the region in that geographical area. The first three digits together show the Sectional center facility [12]. The fourth and fifth digits identify the city or a village/town.

The data sets which were used as unclean user input data and external source of data in this project are given below:

```
# Address - Street Name, City, Province, Postal Code

my_data = [["Chicago Street","Toronto","ON","Canada","852963"],
           ["van StreeT","Vancouver", "BC","Canada","123456"],
           ["Mckenzie Avenue","Victoria","BC","Canada","133456"],
           ["Mckenzie Avenue","Victoria","BC","Canada","133457"],
           ["Vernon StreET","Calgary","CB", "Canada","789426"],
           ["Man Street", "Hampton", "NB", "Canada", "556631"],
           ["Freeport Street", "Salina", "KS", "USA", "27899"],
           ["Ocen Drive", "Detin", "FL", "USA", "32550"]]
```

Figure 3.6: User input data with uncleaned data sets

```
ext_data = [["Mckenzie Avenue","Victoria","BC","Canada","133450"],
            ["Mckenzie Avenue","Victoria","BC","Canada","133451"],
            ["Stone Avenue","Winnipeg","MB","Canada","133452"],
            ["Van Street","Vancouver", "BC","Canada","123456"],
            ["Chicago Street","Toronto","ON","Canada","852963"],
            ["Begbie Street","Victoria","BC", "Canada","789456"],
            ["Shelbourne Street","Vancouver","BC", "Canada","789458"],
            ["Young Street","Halifax","NS", "Canada","789459"],
            ["Maple Street","Montreal","QC", "Canada","789416"],
            ["Vernon Street","Calgary","AB", "Canada","789426"],
            ["Main Street", "Hampton", "NB", "Canada", "556631"],
            ["Brown Street","Groton Avenue", "SD","USA","57701"],
            ["Kingston Avenue", "Ney York", "NY", "USA", "10021"],
            ["Palm Way","Austin", "TX","USA","78758"],
            ["Holiday Drive", "Moorhead", "MN", "USA", "56560"],
            ["Freeport Street", "Salina", "KS", "USA", "27896"],
            ["Skyland Park", "Topeka", "KS", "USA", "10508"],
            ["Vine Street", "Sebring", "FL", "USA", "33870"],
            ["Ocean Drive", "Destin", "FL", "USA", "32550"],
            ["Lincoln Road", "Miami", "FL", "USA", "33139"]]
```

Figure 3.7: External data which has authenticated data sets of addresses

After running the algorithm on the above data sets, we get following results:

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
>

------------------------------------------------------------------------



------------------------------------------------------------------------


Final result:
Recommendation for  ['Mckenzie Avenue', 'Victoria', 'BC', 'Canada', '133456']
['Mckenzie Avenue', 'Victoria', 'BC', 'Canada', '133450']
['Mckenzie Avenue', 'Victoria', 'BC', 'Canada', '133451']

------------------------------------------------------------------------


Final result:
Recommendation for  ['Mckenzie Avenue', 'Victoria', 'BC', 'Canada', '133457']
['Mckenzie Avenue', 'Victoria', 'BC', 'Canada', '133450']
['Mckenzie Avenue', 'Victoria', 'BC', 'Canada', '133451']

------------------------------------------------------------------------


Final result:
Recommendation for  ['Vernon StreET', 'Calgary', 'CB', 'Canada', '789426']
['Vernon Street', 'Calgary', 'AB', 'Canada', '789426']

------------------------------------------------------------------------
```

Figure 3.8: Generated output (part 1) on running the algorithm

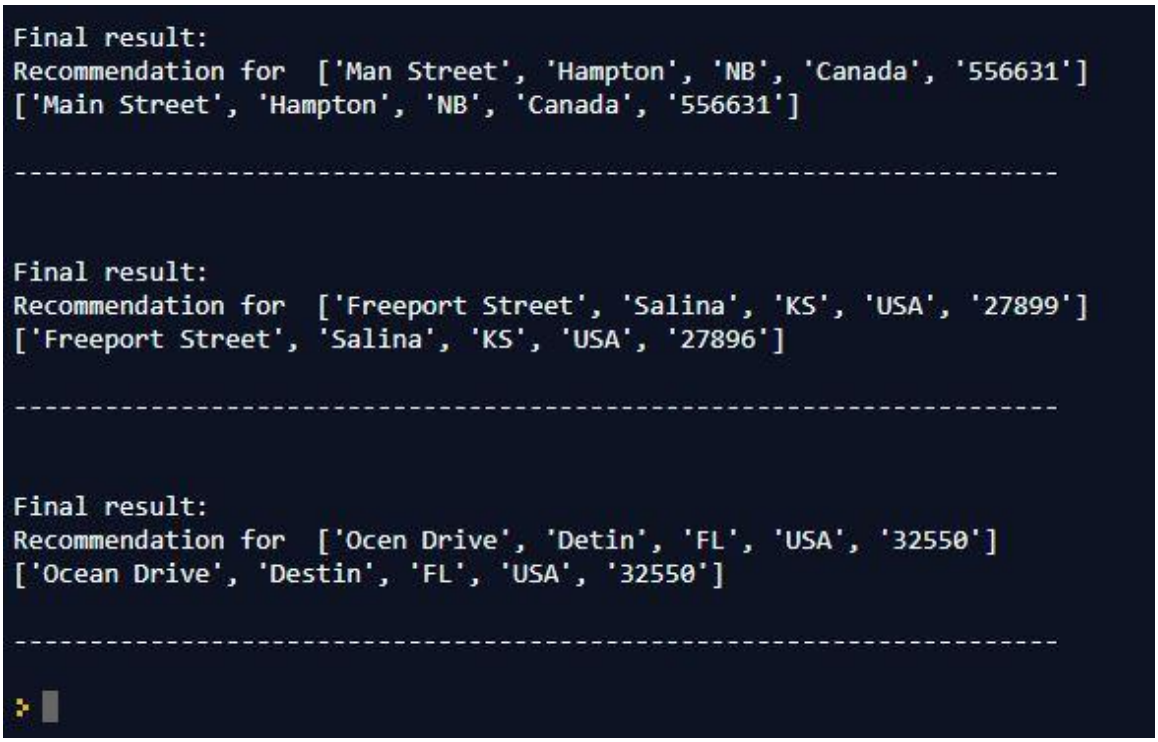


Figure 3.9: Generated output (part 2) on running the algorithm

Since it is a small set of results, it can be seen manually that the results have two false positives and zero false negatives. The recommendations which have false positives and false negatives, also contain the relevant result.

The analysis results are given in Table 3.1:

Table 3.1: Evaluation analysis data of 20 records

External data size	20
Test data size	8
No. of errors in test data	6
Type of Errors	Misspelled street name/city/province/country/ postal code
No. of recommendations	6
No. of false positives	2
No. of false negatives	0
Time taken	nearly 0.12 second

False positive is the count of the input records which have more than one recommendations. False negative is the count of the input records which are valid but the authenticated data does not have information about those records.

Chapter 4

Evaluation

This chapter provides a brief evaluation of the Matching Dependency Technique with 968 and 14911 records of USA/Canada addresses, which were generated online.

4.1 Test Data Generation

The first task for the evaluation was to have some large amount of data to test the implemented technique. This data needs to be in the following format for the USA/Canada: [Street Name, City, Province, Country, Postal Code]

The online generated data was a complete address, which had the whole street address instead of street name (shown in Figure 4.1). In this case it is important to note that French is spoken and practiced in Quebec, Canada. The city names of Quebec are also in French. Since I have only handled English language values in this project, I removed all the Quebec records from the test data programmatically. After applying that filter, I got 968 addresses from 1000 in a list of JSON format.

The next task was to fix the street address issue, in other words, I needed to convert the street address into street name.

```
[
{"Street Address":"Ap #416-9498 Arcu. Avenue","City":"Midway","Province":"BC","Country":"Canada","Postal Code":"V3R 0P8"},
{"Street Address":"Ap #261-8822 Magnis Ave","City":"Watson Lake","Province":"YT","Country":"Canada","Postal Code":"Y4V 3B2"},
{"Street Address":"754-5072 Nisl Street","City":"Baie-D'Urfé","Province":"QC","Country":"Canada","Postal Code":"J0Z 9A9"},
{"Street Address":"699 Fermentum Av.","City":"Shipshaw","Province":"QC","Country":"Canada","Postal Code":"J6G 2J2"},
{"Street Address":"912-3645 Morbi Rd.","City":"Sunset Point","Province":"AB","Country":"Canada","Postal Code":"T3K 4P4"},
{"Street Address":"P.O. Box 415, 2880 Egestas, Street","City":"Provo","Province":"UT","Country":"United States","Postal Code":"79930"},
{"Street Address":"P.O. Box 106, 2531 Pellentesque St.","City":"Overland Park","Province":"KS","Country":"United States","Postal Code":"91606"},
{"Street Address":"727-1020 Convallis Rd.","City":"Gillette","Province":"WY","Country":"United States","Postal Code":"99754"},
{"Street Address":"965-4802 Iaculis St.","City":"Lincoln","Province":"NE","Country":"United States","Postal Code":"98882"},
{"Street Address":"Ap #920-6022 Sed Road","City":"Lafayette","Province":"LA","Country":"United States","Postal Code":"67282"}
]
```


Figure 4.1: Sample of online generated data [5]

I downloaded a randomly generated list of 400 street names for North America. The sample list is shown in Figure 4.2:

```
Bridge Street
Cambridge Drive
Grove Avenue
Canterbury Drive
Railroad Avenue
Lantern Lane
East Avenue
Victoria Court
Homestead Drive
Augusta Drive
Park Street
Evergreen Drive
Pin Oak Drive
Lexington Court
Tanglewood Drive
Race Street
Forest Avenue
```

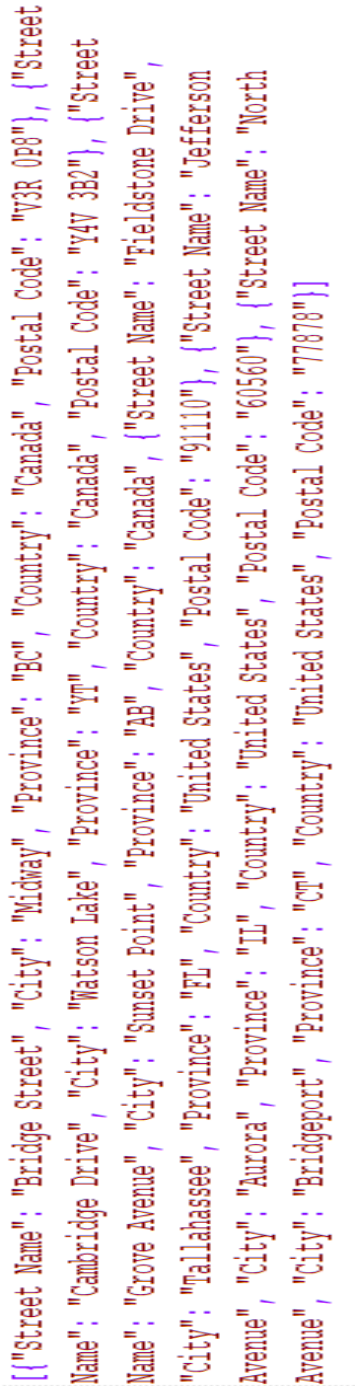
Figure 4.2: Sample of online generated street names [6] [7] [8]

I converted the above text data into a list of data programmatically. The sample list is shown in Figure 4.3:

```
['Bridge Street', 'Cambridge Drive', 'Grove Avenue', 'Canterbury Drive',
'Railroad Avenue', 'Lantern Lane', 'East Avenue', 'Victoria Court',
'Homestead Drive', 'Augusta Drive', 'Park Street', 'Evergreen Drive',
'Pin Oak Drive', 'Lexington Court', 'Tanglewood Drive', 'Race Street']
```

Figure 4.3: Sample list of online generated street names

The address data set had the whole street address, which I updated programmatically with the list of street names shown in the above figures. This left 968 USA/Canada addresses and 400 street names. For every 400 address records, the list of street names was used to replace the street addresses consecutively. The sample of address records with updated street names is shown in Figure 4.4:



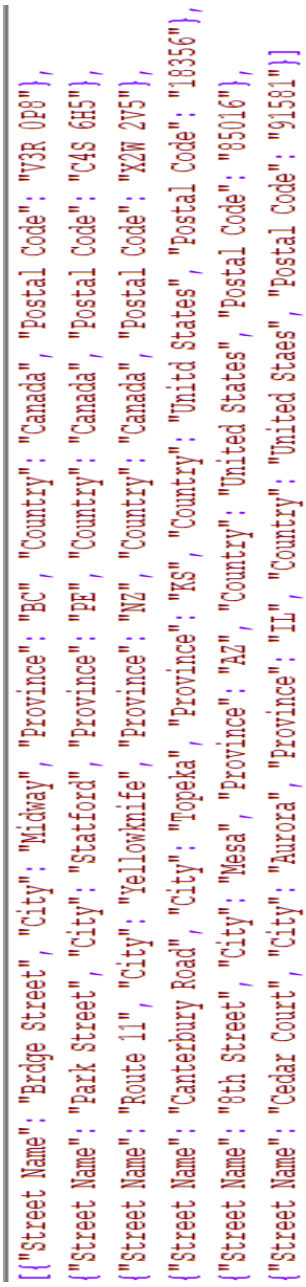
```
[{"Street Name": "Bridge Street", "City": "Midway", "Province": "BC", "Country": "Canada", "Postal Code": "V3R 0P8"}, {"Street
Name": "Cambridge Drive", "City": "Watson Lake", "Province": "YT", "Country": "Canada", "Postal Code": "Y4V 3B2"}, {"Street
Name": "Grove Avenue", "City": "Sunset Point", "Province": "AB", "Country": "Canada"}, {"Street Name": "Fieldstone Drive",
"City": "Tallahassee", "Province": "FL", "Country": "United States", "Postal Code": "91110"}, {"Street Name": "Jefferson
Avenue", "City": "Aurora", "Province": "IL", "Country": "United States", "Postal Code": "60560"}, {"Street Name": "North
Avenue", "City": "Bridgeport", "Province": "CT", "Country": "United States", "Postal Code": "77878"}]
```


Figure 4.4: Sample authenticated data JSON for evaluation

The data shown in Figure 4.4 was considered as the authenticated data for evaluation purposes. For the user input data which may contain misspelled/wrong values in the set of

addresses, I copied every 10th dataset programmatically from the authenticated data and stored it separately in another JSON file.

Since the user input data is retrieved from the authenticated data, I manually made mistakes randomly in user input data JSON. The sample of user input data with incorrect values is shown below in Figure 4.5:



[{"Street Name": "Bridge Street", "City": "Midway", "Province": "BC", "Country": "Canada", "Postal Code": "V3R 0P8"},
{"Street Name": "Park Street", "City": "Stafford", "Province": "PE", "Country": "Canada", "Postal Code": "C4S 6H5"},
{"Street Name": "Route 11", "City": "Yellowknife", "Province": "NZ", "Country": "Canada", "Postal Code": "X2W 2V5"},
{"Street Name": "Canterbury Road", "City": "Topeka", "Province": "KS", "Country": "Unitd States", "Postal Code": "18356"},
{"Street Name": "8th Street", "City": "Mesa", "Province": "AZ", "Country": "United States", "Postal Code": "85016"},
{"Street Name": "Cedar Court", "City": "Aurora", "Province": "IL", "Country": "United Staes", "Postal Code": "91581"}]

Figure 4.5: Sample test data JSON for evaluation

4.2 Evaluation

The test data JSON has a combination of misspelled/wrong street name, city, country, province and postal code.

After running the implemented technique against the JSON files of test data and authenticated data, we get the results shown in Figure 4.6, Figure 4.7, Figure 4.8:

```
C:\Users\shash\Dropbox\Shashank M.Sc. Project Report\data set>python eval_final_code.py

Final result:
Recommendation for  ['Brdge Street', 'Midway', 'BC', 'Canada', 'V3R 0P8']
['Bridge Street', 'Midway', 'BC', 'Canada', 'V3R 0P8']

Final result:
Recommendation for  ['Park Street', 'Statford', 'PE', 'Canada', 'C4S 6H5']
['Park Street', 'Stratford', 'PE', 'Canada', 'C4S 6H5']

Final result:
Recommendation for  ['Route 11', 'Yellowknife', 'NZ', 'Canada', 'X2W 2V5']
['Route 11', 'Yellowknife', 'NT', 'Canada', 'X2W 2V5']

Final result:
Recommendation for  ['Atlantic Avenue', 'Calder', 'SK', 'Canda', 'S6A 3V3']
['Atlantic Avenue', 'Calder', 'SK', 'Canada', 'S6A 3V3']

Final result:
Recommendation for  ['Parker Street', 'Ramara', 'ON', 'Canada', 'K4K 4Z5']
['Parker Street', 'Ramara', 'ON', 'Canada', 'K4K 4Y3']

Final result:
Recommendation for  ['Highlad Drive', 'Chalottetown', 'PE', 'Canada', 'C4V 7K6']
['Highland Drive', 'Charlottetown', 'PE', 'Canada', 'C4V 7K6']

Final result:
Recommendation for  ['Linden Street', 'Picou', 'NR', 'Canada', 'B6N 8E1']
['Linden Street', 'Pictou', 'NS', 'Canada', 'B6N 8E1']

Final result:
Recommendation for  ['Route 202', 'Edmundston', 'NH', 'Caada', 'E7G 1L0']
['Route 202', 'Edmundston', 'NB', 'Canada', 'E7G 1L0']

Final result:
Recommendation for  ['Bridle Court', 'Weyburn', 'SK', 'Cnada', 'S2Z 6T7']
['Bridle Court', 'Weyburn', 'SK', 'Canada', 'S1P 6T7']

Final result:
Recommendation for  ['Bridle Lan', 'rgyle', 'DS', 'Caada', 'B8N 3S4']
['Bridle Lane', 'Argyle', 'NS', 'Canada', 'B8N 3S4']

Final result:
Recommendation for  ['Route 100', 'Deppe', 'NB', 'Cnada', 'E9K 2R9']
['Route 100', 'Dieppe', 'NB', 'Canada', 'E9K 2R9']

Final result:
Recommendation for  ['Wst Street', 'Port Alice', 'KC', 'Canada', 'V5M 0J9']
['West Street', 'Port Alice', 'BC', 'Canada', 'V5M 0J9']
```

Figure 4.6: Evaluation result part 1

```
Final result:
Recommendation for  ['Frankin Street', 'Guysboroh', 'NS', 'Camada', 'B0Y 6E2']
['Franklin Street', 'Guysborough', 'NS', 'Canada', 'B0Y 6E2']

Final result:
Recommendation for  ['Ivy Court', 'Kent', 'BC', 'Canada', 'V0Q 7P1']
['Ivy Court', 'Kent', 'BC', 'Canada', 'V0S 3P3']

Final result:
Recommendation for  ['Canterury Road', 'Yellowknife', 'NV', 'Can@da', 'X5G 5C0']
['Canterbury Road', 'Yellowknife', 'NT', 'Canada', 'X5G 5C0']

Final result:
Recommendation for  ['8th Street', 'Nanaimo', 'AC', 'Canada', 'M0Z 9Q0']
['8th Street', 'Nanaimo', 'BC', 'Canada', 'V9Y 5Y0']

Final result:
Recommendation for  ['Cedar Cort', 'Port Alice', 'BC', 'Canada', 'V2Y 8P9']
['Cedar Court', 'Port Alice', 'BC', 'Canada', 'V2Y 8P9']

Final result:
Recommendation for  ['Bayerry Drive', 'Yoqkton', 'ZK', 'Canad', 'S2R 4Y4']
['Bayberry Drive', 'Yorkton', 'SK', 'Canada', 'S2R 4Y4']

Final result:
Recommendation for  ['9t Street', 'Iqalit', 'HU', 'Canad', 'X0L 5E2']
['9th Street', 'Iqaluit', 'NU', 'Canada', 'X0L 5E2']

Final result:
Recommendation for  ['Route 2', 'Charlottetown', 'RK', 'Canada', 'C3P 0Z6']
['Route 2', 'Charlottetown', 'PE', 'Canada', 'C3P 7T6']

Final result:
Recommendation for  ['5th Steet North', 'Whit Rock', 'CB', 'Canfda', 'V3S 8B1']
['5th Street North', 'White Rock', 'BC', 'Canada', 'V3S 8B1']

Final result:
Recommendation for  ['Sunset Avenue', 'Bathurst', 'NB', 'Canda', 'E2K 7C7']
['Sunset Avenue', 'Bathurst', 'NB', 'Canada', 'E2J 7C7']

Final result:
Recommendation for  ['Route 3', 'Edmundsto', 'NB', 'Canada', 'E2W 9G2']
['Route 32', 'Edmundston', 'NB', 'Canada', 'E2W 9G2']

Final result:
Recommendation for  ['2nd Street Nort', '100 Mileouse', 'BC', 'Canada', 'V0J 2E5']
['2nd Street North', '100 Mile House', 'BC', 'Canada', 'V0J 2E5']

Final result:
Recommendation for  ['Heather Court', 'Monctn', 'NB', 'Canada', 'E0Z 6W1']
['Heather Court', 'Moncton', 'NB', 'Canada', 'E0Z 6W1']

Final result:
Recommendation for  ['Somerset Drive', 'Baddeck', 'NS', 'Canaa', 'B1P 2B2']
['Somerset Drive', 'Baddeck', 'NS', 'Canada', 'B1P 2B2']
```

Figure 4.7: Evaluation result part 2

```
Final result:
Recommendation for  ['Magnolia Drive', 'Wolfville', 'NS', 'Canaa', 'B8Q 3G7']
['Magnolia Drive', 'Wolfville', 'NS', 'Canada', 'B8W 3G7']

Final result:
Recommendation for  ['3rd Street Eat', 'Whitewaer Region Township', 'ON', 'Canada', 'P9W 6N5']
['3rd Street East', 'Whitewater Region Township', 'ON', 'Canada', 'P9W 6N5']

Final result:
Recommendation for  ['Atlantic Aenue', 'Bridgeort', 'CT', 'United States', '69327']
['Atlantic Avenue', 'Bridgeport', 'CT', 'United States', '69327']

Final result:
Recommendation for  ['Parker Street', 'Butt', 'NT', 'Unitd States', '53657']
['Parker Street', 'Butte', 'MT', 'United States', '53657']

Final result:
Recommendation for  ['Highland Drive', 'Gaithersburg', 'MD', 'United States', '39005']
['Highland Drive', 'Gaithersburg', 'MD', 'United States', '39050']

Final result:
Recommendation for  ['Linden Sreet', 'San Antonio', 'TZ', 'United States', '27472']
['Linden Street', 'San Antonio', 'TX', 'United States', '27472']

Final result:
Recommendation for  ['Route 201', 'Tulsa', 'OK', 'United States', '47331']
['Route 202', 'Tulsa', 'OK', 'United States', '47335']
['13th Street', 'Tulsa', 'OK', 'United States', '48840']

Final result:
Recommendation for  ['Bridle Court', 'Tucson', 'ZA', 'United State', '85105']
['Bridle Court', 'Tucson', 'AZ', 'United States', '85195']

Final result:
Recommendation for  ['Bidle Lane', 'Tama', 'LF', 'Unted States', '91541']
['Bridle Lane', 'Tampa', 'FL', 'United States', '91541']

Final result:
Recommendation for  ['Route 100', 'Chandler', 'AZ', 'United State', '85827']
['Route 100', 'Chandler', 'AZ', 'United States', '85826']

Final result:
Recommendation for  ['West Steet', 'Great Falls', 'MT', 'United States', '49465']
['Grant Avenue', 'Great Falls', 'MT', 'United States', '79174']
['West Street', 'Great Falls', 'MT', 'United States', '49469']
['Garfield Avenue', 'Great Falls', 'MT', 'United States', '14096']

Final result:
Recommendation for  ['Ivy Cout', 'Frankfort', 'KY', 'United States', '19992']
['Ivy Court', 'Frankfort', 'KY', 'United States', '19992']

Final result:
Recommendation for  ['Canterbury Road', 'Topeka', 'KS', 'Unitd States', '18356']
['Canterbury Road', 'Topeka', 'KS', 'United States', '18356']

Final result:
Recommendation for  ['8th Street', 'Mesa', 'AZ', 'United States', '85016']
['8th Street', 'Mesa', 'AZ', 'United States', '85256']
```

Figure 4.8: Evaluation result part 3

The actual output is much longer than the results shown in the above figures. The analysis of the above test is given in Table 4.1:

Table 4.1: Evaluation analysis data of 968 records

External data size	968
Test data size	97
No. of errors in test data	41
Type of Errors	Misspelled street name/city/province/country/ postal code
No. of recommendations	41
No. of false positives	2
No. of false negatives	0
Time taken	nearly 0.45 second

4.3 Large Data Evaluation

Initially I created the data set of 1000 records to do the evaluation to test the implemented technique. Another milestone was to test it with a data set larger than the initial evaluation. This time I created 15000 address records for USA/Canada (7500 each) in the same way. Afterwards I took out 1500 records from large data JSON (every 10^{th} record) programmatically and saved it in another JSON file separately, which will be used as user input data.

Currently the user input data has clean records. So I inserted incorrect values into user input data programmatically in the following way:

- Every 5^{th} street name is set to “bad_street_name”
- Every 10^{th} city name is set to “bad_city_name”
- Every 20^{th} province name is set to “bad_province_name”
- Every 40^{th} country name is set to “bad_country_name”
- Every 12^{th} postal code is set to “bad_zipCode”, where the record divisible by 10 was ignored.

If all the values of an address are wrong in the user input data, then the algorithm will fail because of its design. We have used max() in Python, which will throw ValueError if an empty sequence is provided to it. So, while creating user input data, I made sure that no address record is completely wrong.

The evaluation results with the large data set are shown in Figure 4.9, Figure 4.10, Figure 4.11:

```
C:\Users\shash\Dropbox\Shashank M.Sc. Project Report\data set\Big Data>python eval_final_code.py

Final result:
Recommendation for  ['bad_street_name', 'bad_city_name', 'bad_province_name', 'bad_country_name', 'E5G 8Y9']
['Academy Close', 'Fredericton', 'NB', 'Canada', 'E5G 8Y9']

Final result:
Recommendation for  ['bad_street_name', 'Charlottetown', 'PE', 'Canada', 'C3A 9L0']
['Blackwood Street', 'Charlottetown', 'PE', 'Canada', 'C3A 9L0']

Final result:
Recommendation for  ['bad_street_name', 'bad_city_name', 'NT', 'Canada', 'X5T 3Z0']
['Clawthorpe Avenue', 'Hay River', 'NT', 'Canada', 'X5T 3Z0']

Final result:
Recommendation for  ['Cross Street', 'Iqaluit', 'NU', 'Canada', 'bad_zipCode']
['Cross Street', 'Iqaluit', 'NU', 'Canada', 'X9X 8L4']

Final result:
Recommendation for  ['bad_street_name', 'Watson Lake', 'YT', 'Canada', 'Y7N 0M9']
['Edward Street', 'Watson Lake', 'YT', 'Canada', 'Y7N 0M9']

Final result:
Recommendation for  ['bad_street_name', 'bad_city_name', 'bad_province_name', 'Canada', 'C1V 0X0']
['Hamley Street', 'Charlottetown', 'PE', 'Canada', 'C1V 0X0']

Final result:
Recommendation for  ['Joseph Street', 'Watson Lake', 'YT', 'Canada', 'bad_zipCode']
['Joseph Street', 'Watson Lake', 'YT', 'Canada', 'Y7S 0B1']

Final result:
Recommendation for  ['bad_street_name', 'Whitehorse', 'YT', 'Canada', 'Y3V 3T0']
['Ladysmith Street', 'Whitehorse', 'YT', 'Canada', 'Y3V 3T0']

Final result:
Recommendation for  ['bad_street_name', 'bad_city_name', 'NB', 'Canada', 'E5X 2T4']
['Milne Street', 'Dieppe', 'NB', 'Canada', 'E5X 2T4']

Final result:
Recommendation for  ['bad_street_name', 'Calder', 'SK', 'Canada', 'S1C 9Y8']
['Penwell Street', 'Calder', 'SK', 'Canada', 'S1C 9Y8']

Final result:
Recommendation for  ['Princess Avenue', 'Fogo', 'NL', 'Canada', 'bad_zipCode']
['Princess Avenue', 'Fogo', 'NL', 'Canada', 'A0S 6E9']

Final result:
Recommendation for  ['bad_street_name', 'bad_city_name', 'bad_province_name', 'bad_country_name', 'X4W 7R2']
['Russell Street', 'Gjoa Haven', 'NU', 'Canada', 'X4W 7R2']

Final result:
Recommendation for  ['bad_street_name', 'Winnipeg', 'MB', 'Canada', 'R5Y 8N1']
['Sylvia Street', 'Winnipeg', 'MB', 'Canada', 'R5Y 8N1']

Final result:
Recommendation for  ['Waterfront Crescent', 'Saint John', 'NB', 'Canada', 'bad_zipCode']
['Waterfront Crescent', 'Saint John', 'NB', 'Canada', 'E3P 0V5']
```

Figure 4.9: Large-data evaluation result part 1

```
Final result:
Recommendation for  ['bad_street_name', 'bad_city_name', 'MB', 'Canada', 'R9Z 9K7']
['Oak Street', 'Daly', 'MB', 'Canada', 'R9Z 9K7']

Final result:
Recommendation for  ['bad_street_name', 'Iqaluit', 'NU', 'Canada', 'X1V 1M5']
['Spruce Street', 'Iqaluit', 'NU', 'Canada', 'X1V 1M5']

Final result:
Recommendation for  ['bad_street_name', 'bad_city_name', 'bad_province_name', 'Canada', 'S7Y 4P7']
['Clark Street', 'Macklin', 'SK', 'Canada', 'S7Y 4P7']

Final result:
Recommendation for  ['bad_street_name', 'Deline', 'NT', 'Canada', 'X5L 2E5']
['Cedar Street', 'Deline', 'NT', 'Canada', 'X5L 2E5']

Final result:
Recommendation for  ['Wall Street', 'Langenburg', 'SK', 'Canada', 'S0K 8C9']
['Wall Street', 'Langenburg', 'SK', 'Canada', 'S0K 8C9']

Final result:
Recommendation for  ['bad_street_name', 'bad_city_name', 'NT', 'Canada', 'X8N 3T9']
['Madison Court', 'Wekweti', 'NT', 'Canada', 'X8N 3T9']

Final result:
Recommendation for  ['Essex Court', 'Drumheller', 'AB', 'Canada', 'bad_zipCode']
['Essex Court', 'Drumheller', 'AB', 'Canada', 'T5K 6L0']

Final result:
Recommendation for  ['bad_street_name', 'Fort Providence', 'NT', 'Canada', 'X0T 2X7']
['Elmwood Avenue', 'Fort Providence', 'NT', 'Canada', 'X0T 2X7']

Final result:
Recommendation for  ['bad_street_name', 'bad_city_name', 'bad_province_name', 'bad_country_name', 'E6B 3L8']
['Hickory Street', 'Moncton', 'NB', 'Canada', 'E6B 3L8']

Final result:
Recommendation for  ['Orchard Street', 'Charlottetown', 'PE', 'Canada', 'bad_zipCode']
['Orchard Street', 'Charlottetown', 'PE', 'Canada', 'C9W 4N2']

Final result:
Recommendation for  ['bad_street_name', 'Cumberland', 'ON', 'Canada', 'M5T 2L4']
['Main Street', 'Cumberland', 'ON', 'Canada', 'M5T 2L4']

Final result:
Recommendation for  ['bad_street_name', 'bad_city_name', 'YT', 'Canada', 'Y5Z 0N5']
['10th Street', 'Whitehorse', 'YT', 'Canada', 'Y5Z 0N5']

Final result:
Recommendation for  ['bad_street_name', 'Stratford', 'PE', 'Canada', 'C6G 7N9']
['3430 Kingston Road', 'Stratford', 'PE', 'Canada', 'C6G 7N9']

Final result:
Recommendation for  ['4 Winds Drive', 'Sylvan Lake', 'AB', 'Canada', 'bad_zipCode']
['4 Winds Drive', 'Sylvan Lake', 'AB', 'Canada', 'T1B 2L9']

Final result:
Recommendation for  ['bad_street_name', 'bad_city_name', 'bad_province_name', 'Canada', 'P7Y 0Z5']
['Academy Road', 'Ottawa', 'ON', 'Canada', 'P7Y 0Z5']
```

Figure 4.10: Large-data evaluation result part 2

```
Final result:
Recommendation for  ['bad_street_name', 'bad_city_name', 'bad_province_name', 'bad_country_name', 'S0Z 9B8']
['Amoro Drive', 'Calder', 'SK', 'Canada', 'S0Z 9B8']

Final result:
Recommendation for  ['bad_street_name', 'Arviat', 'NU', 'Canada', 'X0H 3E2']
['Antler Street', 'Arviat', 'NU', 'Canada', 'X0H 3E2']

Final result:
Recommendation for  ['bad_street_name', 'bad_city_name', 'NU', 'Canada', 'X2C 3Y8']
['Ardmore Road', 'Iqaluit', 'NU', 'Canada', 'X2C 3Y8']

Final result:
Recommendation for  ['Arleta Avenue', 'Scarborough', 'ON', 'Canada', 'bad_zipCode']
['Arleta Avenue', 'Scarborough', 'ON', 'Canada', 'M4S 8J0']

Final result:
Recommendation for  ['bad_street_name', "Spaniard's Bay", 'NL', 'Canada', 'A3N 2V1']
['Arundel Avenue', "Spaniard's Bay", 'NL', 'Canada', 'A3N 2V1']

Final result:
Recommendation for  ['bad_street_name', 'bad_city_name', 'bad_province_name', 'Canada', 'A5L 7C4']
['Atkins Avenue', 'Fortune', 'NL', 'Canada', 'A5L 7C4']

Final result:
Recommendation for  ['Aveline Crescent', 'Harbour Grace', 'NL', 'Canada', 'bad_zipCode']
['Aveline Crescent', 'Harbour Grace', 'NL', 'Canada', 'A5P 0Z0']

Final result:
Recommendation for  ['bad_street_name', 'Legal', 'AB', 'Canada', 'T4S 2R9']
['Avis Crescent', 'Legal', 'AB', 'Canada', 'T4S 2R9']

Final result:
Recommendation for  ['bad_street_name', 'bad_city_name', 'NB', 'Canada', 'E5H 1R2']
['Baker Avenue', 'Dieppe', 'NB', 'Canada', 'E5H 1R2']

Final result:
Recommendation for  ['bad_street_name', 'Cambridge Bay', 'NU', 'Canada', 'X4E 2W2']
['Banstock Drive', 'Cambridge Bay', 'NU', 'Canada', 'X4E 2W2']

Final result:
Recommendation for  ['Barclay Road', 'Argyle', 'NS', 'Canada', 'bad_zipCode']
['Barclay Road', 'Argyle', 'NS', 'Canada', 'B8M 7R1']

Final result:
Recommendation for  ['bad_street_name', 'bad_city_name', 'bad_province_name', 'bad_country_name', 'B9S 9G2']
['Bartor Road', 'Municipal District', 'NS', 'Canada', 'B9S 9G2']

Final result:
Recommendation for  ['bad_street_name', 'Moncton', 'NB', 'Canada', 'E5R 6L6']
['Bayview Ridge Crescent', 'Moncton', 'NB', 'Canada', 'E5R 6L6']

Final result:
Recommendation for  ['Beaufield Avenue', 'Bonavista', 'NL', 'Canada', 'bad_zipCode']
['Beaufield Avenue', 'Bonavista', 'NL', 'Canada', 'A7R 1Y6']

Final result:
Recommendation for  ['bad_street_name', 'bad_city_name', 'ON', 'Canada', 'M4W 4G9']
['Bedle Avenue', 'Kawartha Lakes', 'ON', 'Canada', 'M4W 4G9']
```

Figure 4.11: Large-data evaluation result part 3

The above results are the starting piece of actual result, which is huge in size. The analysis of the test with 15000 records is given in Table 4.2:

Table 4.2: Evaluation analysis data of 14911 records

External data size	14911
Test data size	1492
No. of errors in test data	399
Type of Errors	Misspelled street name/city/province/country/ postal code
No. of recommendations	400
No. of false positives	11
No. of false negatives	0
Time taken	nearly 15 seconds

We did analysis with many other data sets of different sizes, from 2000 and 14000. The recorded time for all the evaluations is shown in Figure 4.12:

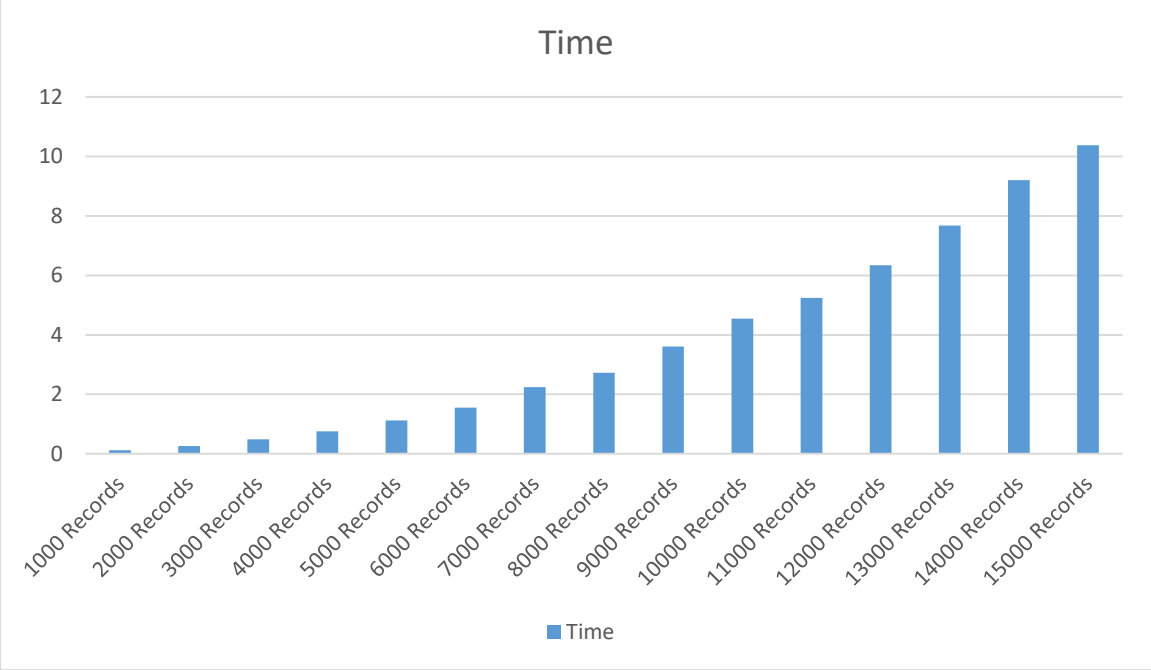


Figure 4.12: Time analysis for different sizes of data sets

Table 4.3: The data of evaluation analysis for different sizes of data sets

External data size	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000	11000	12000	13000	14000	14911
Test data size	100	200	300	400	500	600	700	800	900	1000	1100	1200	1300	1400	1492
No. of errors in test data	27	53	80	107	133	160	187	213	240	267	293	320	347	373	399
No. of recommendations	27	53	80	107	133	160	187	213	240	267	293	320	347	373	399
No. of false positives	0	0	1	2	2	2	3	3	1	4	6	5	4	6	11
No. of false negatives	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Time taken (in seconds)	0.12	0.26	0.49	0.75	1.12	1.55	2.24	2.72	3.61	4.54	5.24	6.34	7.68	9.21	10.38

The errors for the evaluation Table 4.3 are misspelled street name/city/province/country/ postal code.

Another evaluation was done, where the user has entered valid data but the record is missing in the authenticated data. The output is shown in Figure 4.13

```
Final result:
Recommendation for  ['Bramber Road', 'East Gwillimbury', 'ON', 'Canada', 'L7K 1Y1']
['Lakewood Avenue', 'Kapuskasing', 'ON', 'Canada', 'K7J 1M1']
['Batterswood Drive', 'Port Hope', 'ON', 'Canada', 'N6W 8W9']
['Acton Avenue', 'Osgoode', 'ON', 'Canada', 'P3E 4C0']
['Burridge Road', 'Welland', 'ON', 'Canada', 'K1K 3E9']
['Lane N College E Lansdowne', 'Kearny', 'ON', 'Canada', 'K6G 4A4']
['Lane E Indian Road Crescent S Annette', 'Kapuskasing', 'ON', 'Canada', 'N7P 5M7']
['Lane E Spadina N Lonsdale', 'Gloucester', 'ON', 'Canada', 'K8C 3C0']
['Emmett Avenue', 'Newmarket', 'ON', 'Canada', 'N1T 6C1']
['Lane 1 W Sherbourne S Richmond', 'Northumberland', 'ON', 'Canada', 'P3B 0P0']
['Elm Grove Avenue', 'Aurora', 'ON', 'Canada', 'L6C 1X9']
['6 Point Road', 'Whitby', 'ON', 'Canada', 'P7C 8H0']
['Green Belt Drive', 'Cornwall', 'ON', 'Canada', 'P5X 8A4']
['Grovedale Avenue', 'Newbury', 'ON', 'Canada', 'K6R 0Y4']
['Lane N Dundas W Tiverton', 'Goderich', 'ON', 'Canada', 'N2V 3L9']
['Dundurn Crescent', 'Cumberland', 'ON', 'Canada', 'L5P 0A3']
['Lane 3 N College W Margueretta', 'Owen Sound', 'ON', 'Canada', 'K6M 4X8']
['Coral Gable Drive', 'Scarborough', 'ON', 'Canada', 'N9J 3N8']
['Pendergast Street', 'Aurora', 'ON', 'Canada', 'M6Y 6Z8']
['Ffolkes Court', 'Orilla', 'ON', 'Canada', 'M5K 6R7']
['Central Park Roadway', 'Kearny', 'ON', 'Canada', 'P0P 2T0']
['Lamay Crescent', 'Cobourg', 'ON', 'Canada', 'P1S 3E3']
['Gemini Road', 'Bath', 'ON', 'Canada', 'P4P 8X2']
['Lane N Dupont E Perth', 'Markham', 'ON', 'Canada', 'N0K 6S1']
['Easthampton Drive', 'Midlands', 'ON', 'Canada', 'M5H 8K0']
['Drummond Street', 'Hearst', 'ON', 'Canada', 'M9E 4E1']
['Lane N Bloor W Manning', 'Blind River', 'ON', 'Canada', 'L1J 6X9']
['James Gray Drive', 'Vaughan', 'ON', 'Canada', 'M4T 3A6']
['Azzarello Lane', 'Bath', 'ON', 'Canada', 'L1S 3B3']
['Lane E Wychwood S Louise', 'Norfolk County', 'ON', 'Canada', 'K9C 8J1']
['High Street', 'Newmarket', 'ON', 'Canada', 'N4N 6N1']
['Athlone Road', 'Ajax', 'ON', 'Canada', 'N7K 0Y0']
['Lane N Eglinton W Latimer', 'Ramara', 'ON', 'Canada', 'P9R 8R9']
['Carnarvon Street', 'Russell', 'ON', 'Canada', 'N5E 5T6']
['Chudleigh Road', 'Owen Sound', 'ON', 'Canada', 'N2N 3H4']
['Falwyn Avenue', 'Barrie', 'ON', 'Canada', 'L0X 3V0']
['Braga Gardens', 'Greater Sudbury', 'ON', 'Canada', 'P1L 6M7']
['Castlebar Road', 'Norfolk County', 'ON', 'Canada', 'M8M 3L0']
['Crimson Millway', 'Burlington', 'ON', 'Canada', 'N6G 0R5']
['Connaught Road', 'Renfrew', 'ON', 'Canada', 'K4T 9C7']
['Clueson Park', 'Whitchurch-Stouffville', 'ON', 'Canada', 'L1S 5P7']
['Dentonia Park Avenue', 'Windsor', 'ON', 'Canada', 'N2Z 6S2']
['Hilda Avenue', 'Lakeshore', 'ON', 'Canada', 'L7G 1Y5']
['Carnahan Terrace', 'Quinte West', 'ON', 'Canada', 'K6B 7T1']
['Lane N Bloor W Westmoreland', 'Cornwall', 'ON', 'Canada', 'K4C 3G3']
['Pennsylvania Avenue', 'Port Hope', 'ON', 'Canada', 'M0J 8V5']
['Hanley Street', 'Newmarket', 'ON', 'Canada', 'P3M 7T2']
['Chillery Avenue', 'Ajax', 'ON', 'Canada', 'M6L 1J0']
['Ladyshot Crescent', 'Kingston', 'ON', 'Canada', 'N2X 1R6']
['Adeline Court', 'Cumberland', 'ON', 'Canada', 'N5N 0N6']
['Kingsview Boulevard', 'Vaughan', 'ON', 'Canada', 'L1S 6J1']
['Anastacia Court', 'Township of Minden Hills', 'ON', 'Canada', 'L3J 0L6']
```

Figure 4.13: Evaluation result when valid address is missing in external data

To continue the evaluation with the above result, when I saved the entered address in the authenticated data, the output did not recommend anything to the entered address.

```
Final result:
Recommendation for  ['bad_street_name', 'bad_city_name', 'bad_province_name', 'bad_country_name', '92078']
['Stibbard Avenue', 'San Diego', 'CA', 'United States', '92078']

Final result:
Recommendation for  ['bad_street_name', 'Iowa City', 'IA', 'United States', '57840']
['Coldharbour Road', 'Iowa City', 'IA', 'United States', '57840']

Final result:
Recommendation for  ['bad_street_name', 'bad_city_name', 'NE', 'United States', '49802']
['Somerset Drive', 'Grand Island', 'NE', 'United States', '49802']

Final result:
Recommendation for  ['Academy Street', 'Flin Flon', 'MB', 'Canada', 'bad_zipCode']
['Academy Street', 'Flin Flon', 'MB', 'Canada', 'R0W 4R4']

Final result:
Recommendation for  ['bad_street_name', 'Fort Simpson', 'NT', 'Canada', 'X2E 6R2']
['Flatwoods Drive', 'Fort Simpson', 'NT', 'Canada', 'X2E 6R2']

Final result:
Recommendation for  ['bad_street_name', 'bad_city_name', 'bad_province_name', 'Canada', 'T2R 9R3']
['Earl Grey Road', 'Westlock', 'AB', 'Canada', 'T2R 9R3']

Final result:
Recommendation for  ['Evermede Drive', 'Assiniboia', 'SK', 'Canada', 'bad_zipCode']
['Evermede Drive', 'Assiniboia', 'SK', 'Canada', 'S3T 8R1']

Final result:
Recommendation for  ['bad_street_name', 'Kearny', 'ON', 'Canada', 'P0P 2T0']
['Central Park Roadway', 'Kearny', 'ON', 'Canada', 'P0P 2T0']

Final result:
Recommendation for  ['bad_street_name', 'bad_city_name', 'CT', 'United States', '34202']
['Lane W Shaw S Halton', 'Hartford', 'CT', 'United States', '34202']

Final result:
Recommendation for  ['bad_street_name', 'Flin Flon', 'MB', 'Canada', 'R6S 6Z4']
['Edgar Woods Road', 'Flin Flon', 'MB', 'Canada', 'R6S 6Z4']

Final result:
Recommendation for  ['Green Belt Drive', 'Cornwall', 'ON', 'Canada', 'bad_zipCode']
['Green Belt Drive', 'Cornwall', 'ON', 'Canada', 'P5X 8A4']

Final result:
Recommendation for  ['bad_street_name', 'bad_city_name', 'bad_province_name', 'bad_country_name', 'Y6W 7S7']
['Colwick Drive', 'Whitehorse', 'YT', 'Canada', 'Y6W 7S7']

Final result:
Recommendation for  ['bad_street_name', 'Nanaimo', 'BC', 'Canada', 'V8C 2R8']
['Hickory Tree Road', 'Nanaimo', 'BC', 'Canada', 'V8C 2R8']
```

Figure 4.14: Evaluation result after saving the valid address in authenticated data

All the evaluations were done on a personal laptop with the following configurations:

- System Model: HP Pavilion g6 Notebook PC
- Processor: Intel Core i3-2350M CPU @ 2.30GHz
- Installed Physical Memory (RAM): 6GB

The operating system is Microsoft Widows 10 Pro 10.0.17134 and the Python version is 3.6.1.

Chapter 5

Conclusions

This chapter shows the possible future work, which can improve this algorithm. It also concludes the work and my contributions.

We can conclude the following points from the performance of Matching Dependency technique:

- It can be one of the useful techniques in the clean up of any kind of data set which has bad input data.
- It is able to handle the large data sets as the binary search has been used, which improves the search process efficiency.
- The data set clean up process became easier as the implemented technique finds out the unclean data sets and then makes corresponding possible recommendations.

5.1 Contributions

The following summarize the contributions of this project:

- Research was done with the papers on machine learning and techniques to explore data cleaning [3] [9]. The Matching Dependency technique seemed promising for data cleaning process.
- I implemented the technique using Python language and self-created data sets, user input data and authenticated data.

- For evaluation process, I generated large data sets of different sizes, ranging from 1000 to 15000.
- To have deep analysis I evaluated time, size of user-input/authenticated data sets, number of errors in user-input data, false positives, false negatives and type of errors, for all different sizes of data sets.

5.2 Future Work

The matching dependency technique that has been implemented in this project, may be improved as follows:

- Street number can be added in the data sets, which would get more efficient results.
- Another search algorithm such as Hash Tables, could be used instead of binary search to make this implemented technique work more efficiently [10].
- A more developed machine learning technique [11] could be used. For example, if the street name is “Albion Road” and the user has entered the street name as “Albion Rd”, then the advanced technique should find it as good input data and use it as valid data to verify other entries for the same address.
- The addresses in French (from Quebec, Canada) could also be handled.

All wrong values in an address could get relevant recommendations.

Bibliography

1. Structure of Canadian Postal Code. Visited on 24 October, 2018. Retrieved from: https://www.canadapost.ca/tools/pg/manual/PGaddress-e.asp?ecid=murl10006450#1449273

2. Structure of USA Postal Code. Visited on 25 October, 2018. Retrieved from: http://www.zippostalcodes.com/postcodes/us/us-zip-codes-format/

3. Theodoros Rekatsinas, Xu Chu and Christopher Ré, “HoloClean: Holistic Data repairs with Probabilistic Inference”, Proceedings of the VLDB Endowment, Volume 10, No. 11, pp. 1190-1191, August 2017

4. Repl online interpreter. Visited on 10 August, 2018. Retrieved from: https://repl.it/

5. Generate data for evaluation. Visited on 5 November, 2018. Retrieved from: https://www.generatedata.com/

6. Generate street names for evaluation. Visited on 6 November, 2018. Retrieved from: https://www.randomlists.com/random-street-names

7. Generate street names for evaluation. Visited on 8 November, 2018. Retrieved from: https://geographic.org/streetview/canada/bc/victoria.html

8. Generate street names for evaluation. Visited on 8 November, 2018. Retrieved from: https://geographic.org/streetview/canada/on/city_of_toronto.html

9. Xu Chu, Ihab F. Ilyas, Sanjay Krishnan and Jiannan Wang, “Data Cleaning: Overview and Emerging Challenges”, In Proceedings of the 2016 ACM SIGMOD Conference on Management of Data, San Francisco, USA, pp. 1-3

10. Binary Search Algorithm. Visited on 25 November, 2018. Retrieved from: https://en.wikipedia.org/wiki/Binary_search_algorithm

11. Spell Checker with TensorFlow. Visited on 25 November, 2018. Retrieved from: https://towardsdatascience.com/creating-a-spell-checker-with-tensorflow-d35b23939f60

12. Sectional center facility. Visited on 25 October, 2018. Retrieved from: https://en.wikipedia.org/wiki/Sectional_center_facility

Appendix

Binary Search pseudo code

```
function binarySearch(sortedList, item)
        Set first = 0
        Set last = length(sortedList) - 1
        Set found = false
        while first <= last AND not found do
                Set midpoint = (first + last)//2
                Set userInput = item.lower()
                Set trainItem = second element of midpoint of sortedList
                if trainItem = userInput then
                        indexValue.append(first element of midpoint of sortedList)
                        delete midpoint of sortedList
                        return binarySearch(sortedList, item)
                else then
                        if userInput < trainItem then
                                last = midpoint - 1
                        else then
                                first = midpoint + 1
                        end if
                end if
        end while
        return indexValue
  end function
```