Address Matching

Project 2 Milestone Report1

Overview

- Problem
- The data
- Data Wrangling and preparation
- Exploratory Data Analysis
- Machine Learning

Problem

- Background: All service installation records are stored and maintained in a file store with address info. However, no system or field linkage is built between the file store and ERP application. The address is entered manually each time the file added to the file store. The only way to find the records for an address in ERP is to manually search the file store using the address.

- Problem: Match two sets of addresses

The Data

Two sets of address data

- Set 1 is the list of addresses from a legacy system
- Set 2 is the list of addresses from address master table
- USPS Suffix data

Primary_Suffix	Comm_suffix	Std_suffix
alley	allee	aly
alley	alley	aly
alley	ally	aly
alley	aly	aly

```
Data columns (total 6 columns):
DATAID          530437 non-null object
M_TXT           530437 non-null object
M_DIV_CD        530437 non-null object
M_ADDR          526190 non-null object
M_CITY          530320 non-null object
M_ADDR_TYPE     530401 non-null object
dtypes: object(6)
```

```
Data columns (total 3 columns):
P_DIV_CD    505767 non-null object
P_ADDR      505761 non-null object
P_CITY      505767 non-null object
dtypes: object(3)
```

```
Data columns (total 3 columns):
Primary_Suffix    515 non-null object
Comm_suffix       515 non-null object
Std_suffix        515 non-null object
dtypes: object(3)
```

Data Wrangling

Perform text normalization includes:

- converting all letters to lower case
- Removing punctuations
- Removing non ascii chars
- Removing multiple spaces with a single space

```
def cleanse_str(string):
    string = ftfy.fix_text(string) # fix text encoding issues
    string = string.encode("ascii", errors="ignore").decode() #remove non ascii chars
    string = string.lower() # make lower case
    chars_to_remove = ["#", "@", ")","(",".","|","[","]","{","}","'"]
    rx = '[' + re.escape(''.join(chars_to_remove)) + ']'
    string = re.sub(rx, '', string) # remove the list of chars defined above
    string = string.replace('&', 'and')
    string = string.replace(',', ' ')
    string = string.replace('-', ' ')
    string = string.replace('+', ' and ')
    string = re.sub(' +',' ',string).strip() # get rid of multiple spaces and replace with a single space
    string = ' '+ string +' ' # pad names for ngrams...
    string = re.sub(r'[,-./]|\sBD',r'', string)

    return string
```

Fill in missing categorical data using default value

Feature engineering

- Computing and adding columns for data exploration and machine learning
- Prepare data sets including legacy data, master data and USPS suffix data

```
Data columns (total 16 columns):
DATAID          30000 non-null int64
M_TXT           30000 non-null object
M_DIV_CD        30000 non-null object
M_ADDR          30000 non-null object
M_CITY          30000 non-null object
M_ADDR_TYPE     30000 non-null object
P_ID            30000 non-null int64
P_ADDR          30000 non-null object
P_CITY          30000 non-null object
MATCHCODE       30000 non-null object
M_ADDR_c        30000 non-null object
P_ADDR_c        30000 non-null object
M_TXT_c         30000 non-null object
M_ADDR_s        30000 non-null object
P_ADDR_s        30000 non-null object
M_TXT_s         30000 non-null object
dtypes: int64(2), object(14)
```

Data Wrangling – Suffix consolidation

Analyze suffix variances and usages

- Split address into words
- Use Collection Counter to calculate the word occurrences
- Use USPS suffix data to identify suffix words
- Generate suffix variance matrix

```
# generate matrix for top 10 primary suffix which have multiple variances in legacy address data set
include = list(result1.groupby("Primary_Suffix").Word_Seg.count().sort_values(ascending=False).head(10).index)
df_m_suffix = result1.query('Primary_Suffix in @include').sort_values(['Primary_Suffix'], ascending=True)
df_m_suffix.groupby(['Primary_Suffix','Word_Seg']).size().unstack(fill_value=0)
```

Word_Seg	av	ave	aven	avenue	cent	center	centr	centre	cir	circl	...	ridge	st	sta	station	stn	str	street	ter	terr	terrace
Primary_Suffix																					
avenue	1	1	1	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
center	0	0	0	0	1	1	1	1	0	0	...	0	0	0	0	0	0	0	0	0	0
circle	0	0	0	0	0	0	0	0	1	1	...	0	0	0	0	0	0	0	0	0	0
drive	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
heights	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
highway	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
ridge	0	0	0	0	0	0	0	0	0	0	...	1	0	0	0	0	0	0	0	0	0
station	0	0	0	0	0	0	0	0	0	0	...	0	0	1	1	1	0	0	0	0	0
street	0	0	0	0	0	0	0	0	0	0	...	0	1	0	0	0	1	1	0	0	0
terrace	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	1	1

Prepare data sets

- Standardize suffix variances into USPS primary suffix

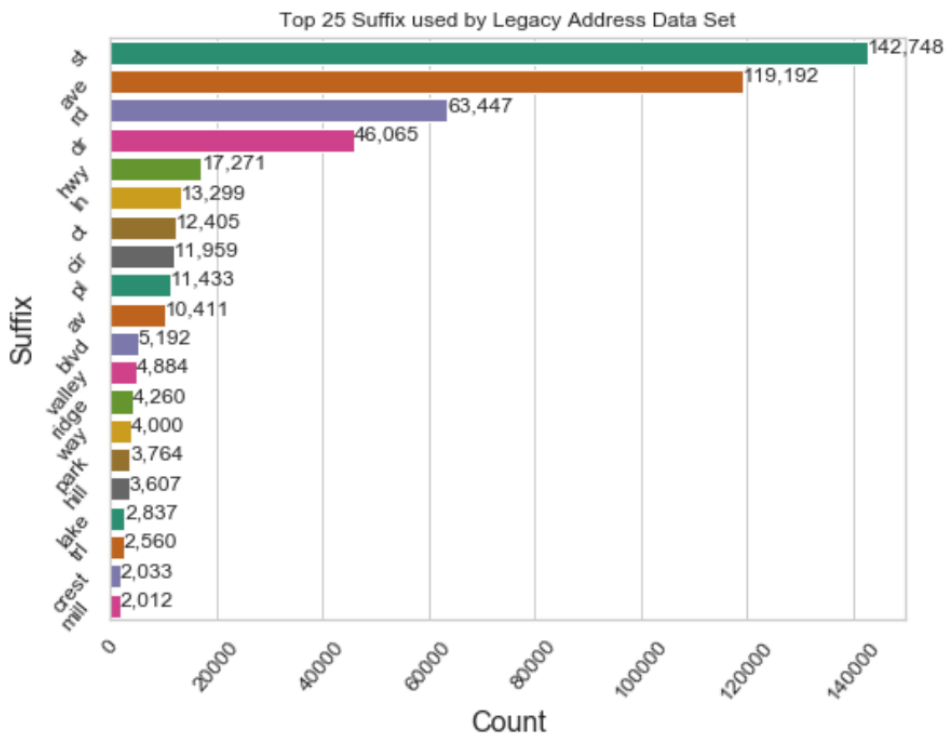
```
string = re.sub(' aly ', ' alley ', string)
string = re.sub(' annex ', ' anex ', string)
string = re.sub(' anx ', ' anex ', string)
string = re.sub(' ave ', ' avenue ', string)
string = re.sub(' av ', ' avenue ', string)
string = re.sub(' aven ', ' avenue ', string)
string = re.sub(' bch ', ' beach ', string)
string = re.sub(' bnd ', ' bend ', string)
string = re.sub(' blf ', ' bluff ', string)
string = re.sub(' btm ', ' bottom ', string)
string = re.sub(' blvd ', ' boulevard ', string)
```

Word_Seg	cnt
center	933
ctr	338
centre	14
cent	94
cntr	7
centr	1

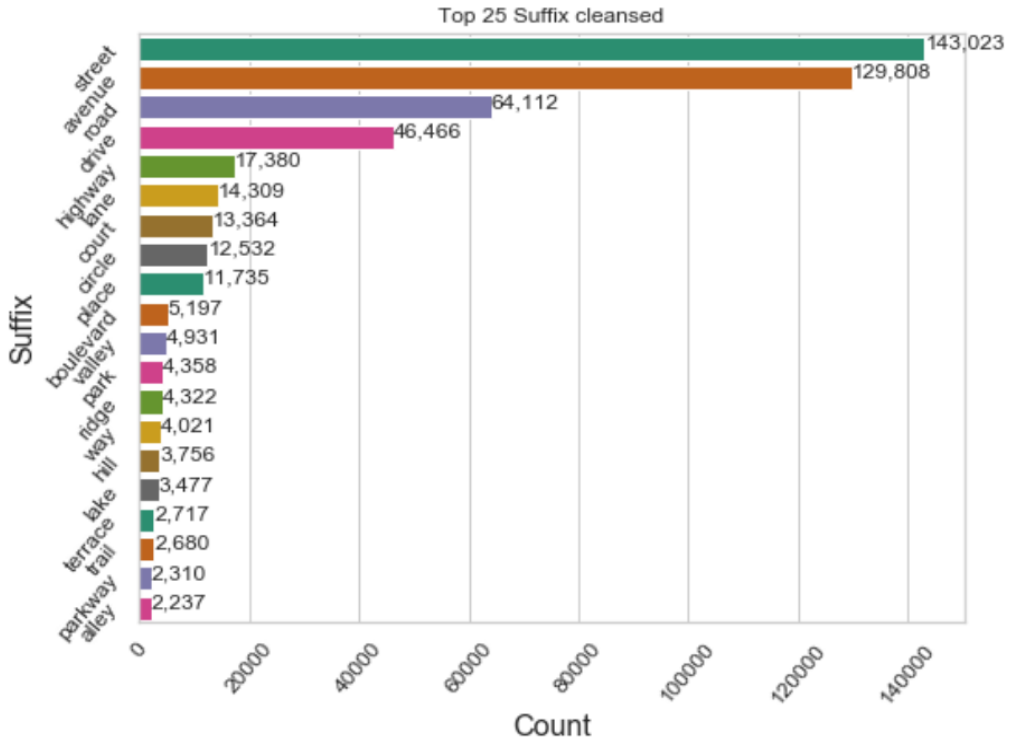
Word_Seg	cnt
center	1387

Exploratory Data Analysis – Suffix consolidation

Before suffix consolidation



After suffix consolidation



Exploratory Data Analysis – Fuzzywuzzy

Run Fuzzywuzzy functions against two data sets and compare the matching results

- Ratio (result: r1, r1s)
- partial_ratio (pr1, pr1s)
- token_sort_ratio (tsr1, tsr1s)
- token_set_ratio (tstr1, tstr1s)

Token_set_ratio returns the best matching ratio 93% (ratio > 90)

Run 1
Data set – cleansed with raw suffix

```
r1 match (%>90) =  0.8581
pr1 match (%>90) =  0.8755
tsr1 match (%>90) =  0.8383666666666667
tstr1 match (%>90) =  0.908
```

Run 2
Data set – cleansed with suffix consolidated

```
r1s match (%>90) =  0.8698333333333333
pr1s match (%>90) =  0.8921
tsr1s match (%>90) =  0.8601
tstr1s match (%>90) =  0.9305333333333333
```