SESUG Paper 214-2019

When fuzzy matching doesn't work: using the CONTAINS and JOIN functions through SAS EG to find foreign words in text strings

Arthur Laciak, U.S. Consumer Product Safety Commission

ABSTRACT

I have found that fuzzy matching functions in SAS, such as SOUNDEX and SPEDIS, are not effective in matching foreign words, especially Chinese words. When transliterated into the Latin alphabet, too many syllables are similarly spelled that fuzzy matching is ineffective, even with strict parameters. Many Chinese words are unintentionally matched, resulting in more complications.

This paper explores a work-around approach, using the CONTAINS and JOIN functions in SAS EG through PROC SQL to search for keywords from a reference table in text strings of a given data set. In this procedure, each text string is compared against keywords and matched keywords are displayed in a new column, resulting in clean data for further analysis. The example given in this paper matches Chinese addresses written in a text string to Chinese cities and provinces.

INTRODUCTION

Fuzzy matching functions are critical tools when trying to match unstructured text data, especially given misspellings due to human error. However, through my experience, I found that fuzzy matching is ineffective with Chinese. Although Chinese characters are transliterated into the Latin alphabet, fuzzy matching results in too many false matches. In particular, Soundex was developed to match similar-sounding English surnames, so the function does not factor Chinese phonetics (Sloan and Laffler, 2018).

Take for example the city Nanjing. Its SOUNDEX code is N5252; the same for Nanchang and Nanchong. Consider also the city Quanzhou. When using COMPLEV, it has a Levenshtein Edit Distance of less than three with Guangzhou, Taizhou, Suzhou, and many other cities.[1] Using SPEDIS and COMPGED has similar results.

In this paper, I present an alternative approach that first involves creating a reference table and then using the CONTAINS and JOIN functions in PROC SQL to search for the Chinese names in text strings of a given data set. I will also show how to use SAS EG to query results. Although my example uses Chinese city names, this approach can be applied to different languages.[2]

CREATING A REFERENCE TABLE

The SAS code, which is presented in the next section, uses the JOIN function to merge a table containing text strings (in this case, Chinese addresses) with a reference table containing a list of keywords (in this case, Chinese cities). The end result is a dataset with added columns of matched keywords. However, the first step is to create a reference table and to understand postal address hierarchy in China.

China has three levels of cities – in ascending order, county-level, prefectural-level, and provincial-level. Additionally, there are two administrative regions, Hong Kong and Macau.[3] When creating my reference table in Excel, I created a separate table for county-level and prefectural-level cities, even though prefecture-level cities are made up of multiple county-level cities. This is because I found, through experience, that an address listing a county-level city may not always have a prefecture-level city listed. I

---

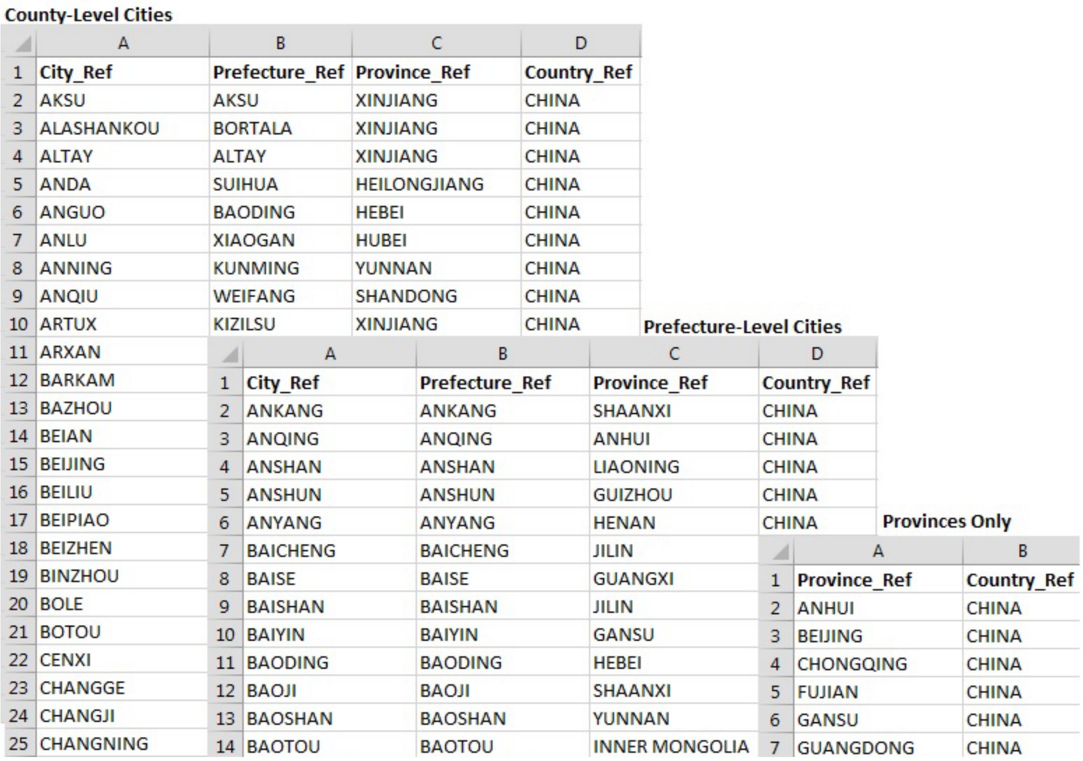
[1] The Levenshtein Edit Distance is a count of "the number of single character deletions, insertions, or substitutions required to transform one string into [another]." (Moler, 2017).

[2] The sample Chinese addresses used in this paper were downloaded from the Violations page of the U.S. Consumer Product Safety Commission's public website, found at: https://cpsc.gov/Recalls/violations. The addresses are of firms that were found to be in violation of a mandatory consumer product standard.

[3] A list of Chinese cities can be easily found on Wikipedia at: https://en.wikipedia.org/wiki/List_of_cities_in_China.

also created an additional table containing only the provinces. Since I will run my SAS code with each of the three reference tables, I will reduce the number of non-matches. In the end, if I am unable to match an address to a city, I will at least attempt to match it to a province, which will still allow for some form of analysis afterwards.

For both of the city reference tables, I created four columns: City_Ref, Prefecture_Ref, Province_Ref, and Country_Ref, as shown in Display 1 below. For the prefecture-level cities reference table, I repeated their name in both of the first two columns for the ease of appending tables later. I did the same for provincial-level cities (not shown in code), such as Beijing, where I repeated the name across the first three columns. Additionally, due to the special nature of Hong Kong, I listed it as a province and country and its subdivisions, such as Kowloon, as a city and prefecture.



County-Level Cities

	A	B	C	D
1	City_Ref	Prefecture_Ref	Province_Ref	Country_Ref
2	AKSU	AKSU	XINJIANG	CHINA
3	ALASHANKOU	BORTALA	XINJIANG	CHINA
4	ALTAY	ALTAY	XINJIANG	CHINA
5	ANDA	SUIHUA	HEILONGJIANG	CHINA
6	ANGUO	BAODING	HEBEI	CHINA
7	ANLU	XIAOGAN	HUBEI	CHINA
8	ANNING	KUNMING	YUNNAN	CHINA
9	ANQIU	WEIFANG	SHANDONG	CHINA
10	ARTUX	KIZILSU	XINJIANG	CHINA
11	ARXAN			
12	BARKAM			
13	BAZHOU			
14	BEIAN			
15	BEIJING			
16	BEILIU			
17	BEIPIAO			
18	BEIZHEN			
19	BINZHOU			
20	BOLE			
21	BOTOU			
22	CENXI			
23	CHANGGE			
24	CHANGJI			
25	CHANGNING			

Prefecture-Level Cities

	A	B	C	D
1	City_Ref	Prefecture_Ref	Province_Ref	Country_Ref
2	ANKANG	ANKANG	SHAANXI	CHINA
3	ANQING	ANQING	ANHUI	CHINA
4	ANSHAN	ANSHAN	LIAONING	CHINA
5	ANSHUN	ANSHUN	GUIZHOU	CHINA
6	ANYANG	ANYANG	HENAN	CHINA
7	BAICHENG	BAICHENG	JILIN	
8	BAISE	BAISE	GUANGXI	
9	BAISHAN	BAISHAN	JILIN	
10	BAIYIN	BAIYIN	GANSU	
11	BAODING	BAODING	HEBEI	
12	BAOJI	BAOJI	SHAANXI	
13	BAOSHAN	BAOSHAN	YUNNAN	
14	BAOTOU	BAOTOU	INNER MONGOLIA	

Provinces Only

	A	B
1	Province_Ref	Country_Ref
2	ANHUI	CHINA
3	BEIJING	CHINA
4	CHONGQING	CHINA
5	FUJIAN	CHINA
6	GANSU	CHINA
7	GUANGDONG	CHINA

Display 1. Reference Tables

SAS CODE

The SQL procedure for merging the tables is rather short, as shown below. In this example, I am merging my dataset, WORK.ADDRESS_LIST, with one reference table, WORK.COUNTY_CITIES, to create WORK.MATCHED_COUNTY_CITIES.

```
PROC SQL;
     CREATE TABLE WORK.MATCHED_COUNTY_CITIES AS
     SELECT t1.FIRM_NAME,
          t1.FIRM_ADDRESS,
          t2.City_Ref,
          t2.Prefecture_Ref,
          t2.Province_Ref,
          t2.Country_Ref,
          (COUNT(*)) AS Count
          FROM WORK.ADDRESS_LIST t1
     LEFT JOIN WORK.COUNTY_CITIES t2 ON
```

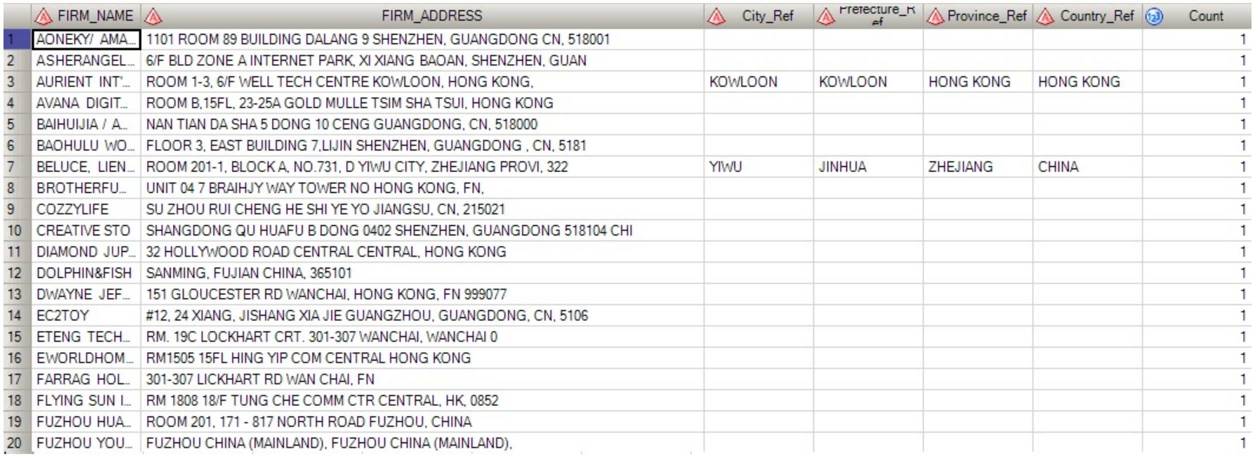
```
          (compress(tranwrd(t1.FIRM_ADDRESS,",","  "),,"P") CONTAINS
          cat(' ',trim(t2.City_Ref),' '))
     GROUP BY t1.FIRM_NAME
     ORDER BY t1.FIRM_NAME;
QUIT;
```

The critical element of this PROC SQL code is the LEFT JOIN statement. In this step, the reference table is being joined to the address list table, using the CONTAINS function. This function searches the WORK.ADDRESS_LIST data table line-by-line for each city in the reference table. Whenever there is a positive match, then the city, prefecture, province, and country reference variables are joined to that address. Acknowledging that there may be multiple matches per address, the Count variable is added. Along with the GROUP BY statement, the variable Count lists the number of times the FIRM_NAME variable appears. When the count is greater than 1, then that means there were multiple matches for a single address. Those results can then be filtered out.

The TRANWRD, COMPRESS, TRIM, and CAT functions are used to clean the data before joining the tables and are optional:

- TRANWRD replaces all commas with spaces. This is useful in cases when a comma was used as a delimiter instead of a space.
- COMPRESS with the “P” option removes all punctuation, because some city and province names have apostrophes.
- TRIM removes any leading or trailing blanks. This is useful to ensure that there are no hidden blanks in the reference tables.
- CAT adds a leading and trailing blank. This seems contradictory with the TRIM function, but if the reference data is not perfectly clean, then there can be many non-matches due to a hidden blank. These two functions together ensure that only the reference city (or province) is matched in its entirety to a city name in the address and not to a portion of a name (*i.e.*, prevents a city such as Jian from being matched with Zhejiang).

The output results in seven columns, with the last one being the Count column, as shown in the display below.



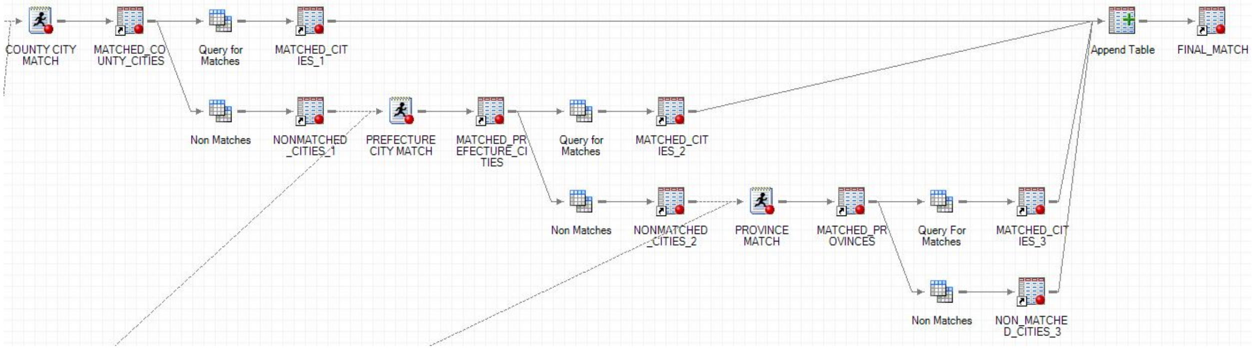
	FIRM_NAME	FIRM_ADDRESS	City_Ref	Prefecture_Ref	Province_Ref	Country_Ref	Count
1	AONEKY/ AMA...	1101 ROOM 89 BUILDING DALANG 9 SHENZHEN, GUANGDONG CN, 518001					1
2	ASHERANGEL...	6/F BLD ZONE A INTERNET PARK, XI XIANG BAOAN, SHENZHEN, GUAN					1
3	AURIENT INT'...	ROOM 1-3, 6/F WELL TECH CENTRE KOWLOON, HONG KONG,	KOWLOON	KOWLOON	HONG KONG	HONG KONG	1
4	AVANA DIGIT...	ROOM B,15FL, 23-25A GOLD MULLE TSIM SHA TSUI, HONG KONG					1
5	BAIHUIJIA / A...	NAN TIAN DA SHA 5 DONG 10 CENG GUANGDONG, CN, 518000					1
6	BAOHULU WO...	FLOOR 3, EAST BUILDING 7,LIJIN SHENZHEN, GUANGDONG , CN, 5181					1
7	BELUCE, LIEN...	ROOM 201-1, BLOCK A, NO.731, D YIWU CITY, ZHEJIANG PROVI, 322	YIWU	JINHUA	ZHEJIANG	CHINA	1
8	BROTHERFU...	UNIT 04 7 BRAIHJY WAY TOWER NO HONG KONG, FN,					1
9	COZZYLIFE	SU ZHOU RUI CHENG HE SHI YE YO JIANGSU, CN, 215021					1
10	CREATIVE STO	SHANGDONG QU HUAFU B DONG 0402 SHENZHEN, GUANGDONG 518104 CHI					1
11	DIAMOND JUP...	32 HOLLYWOOD ROAD CENTRAL CENTRAL, HONG KONG					1
12	DOLPHIN&FISH	SANMING, FUJIAN CHINA, 365101					1
13	DWAYNE JEF...	151 GLOUCESTER RD WANCHAI, HONG KONG, FN 999077					1
14	EC2TOY	#12, 24 XIANG, JISHANG XIA JIE GUANGZHOU, GUANGDONG, CN, 5106					1
15	ETENG TECH...	RM. 19C LOCKHART CRT. 301-307 WANCHAI, WANCHAI 0					1
16	EWORLDHOM...	RM1505 15FL HING YIP COM CENTRAL HONG KONG					1
17	FARRAG HOL...	301-307 LICKHART RD WAN CHAI, FN					1
18	FLYING SUN I...	RM 1808 18/F TUNG CHE COMM CTR CENTRAL, HK, 0852					1
19	FUZHOU HUA...	ROOM 201, 171 - 817 NORTH ROAD FUZHOU, CHINA					1
20	FUZHOU YOU...	FUZHOU CHINA (MAINLAND), FUZHOU CHINA (MAINLAND),					1

Display 2. Output of PROC SQL Code

USING SAS ENTERPRISE GUIDE TO QUERY RESULTS

To examine the output at each step, I use SAS Enterprise Guide to query the results. In the previous section, I only matched the address list with the county level cities, so I would have to repeat the same PROC SQL code with the prefecture level cities and province, if no cities are matched. Therefore, between each step, I need to query the output and remove any matched cities.

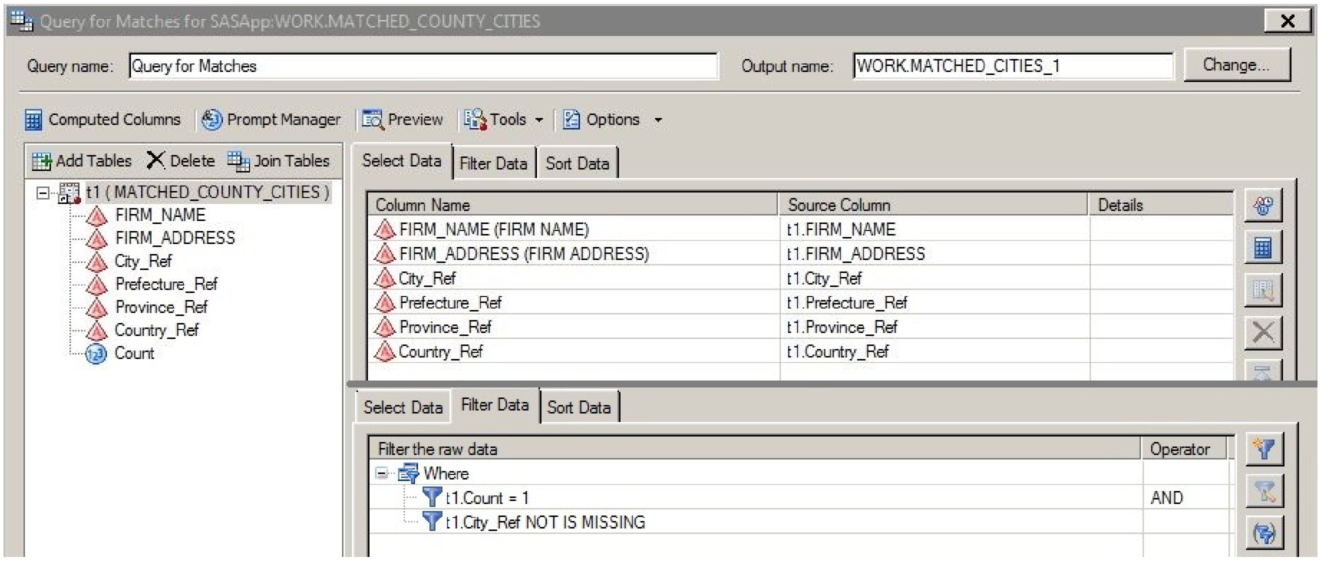
The figure below displays the process flow taken in SAS EG to achieve my final output of matched Chinese cities. Since I run my PROC SQL sequence three times (for county level cities, prefecture level cities, and provinces), I query the results for matches and non-matches three times. I only apply the PROC SQL sequence a second or third time to the non-matched output of the previous query. Once I exhaust all reference tables, I append all of the matched outputs into one dataset. (Note – you can link tables to any SAS program, by right clicking the desired table and selecting "Link 'table name' to…." This will ensure that the project runs in order.)



Display 3. SAS EG Process Flow

QUERYING FOR MATCHES AND NON-MATCHES

After each PROC SQL sequence, the results are queried for matches and non-matches, using the COUNT variable. Each query for matched addresses is identical. Using Query Builder, I filter the PROC SQL output, where *t1.Count* = 1 AND *t1.City_Ref* NOT IS MISSING. The first filter identifies the addresses that have at most one matched city; and the second filter identifies the addresses that actually have a matched city. The AND statement ensures that both criteria are met. These steps are repeated for prefecture-level city and province matches. The output of each query will display the original firm name and address and the city, prefecture, province, and country reference.



Query name: Query for Matches

Output name: WORK.MATCHED_CITIES_1

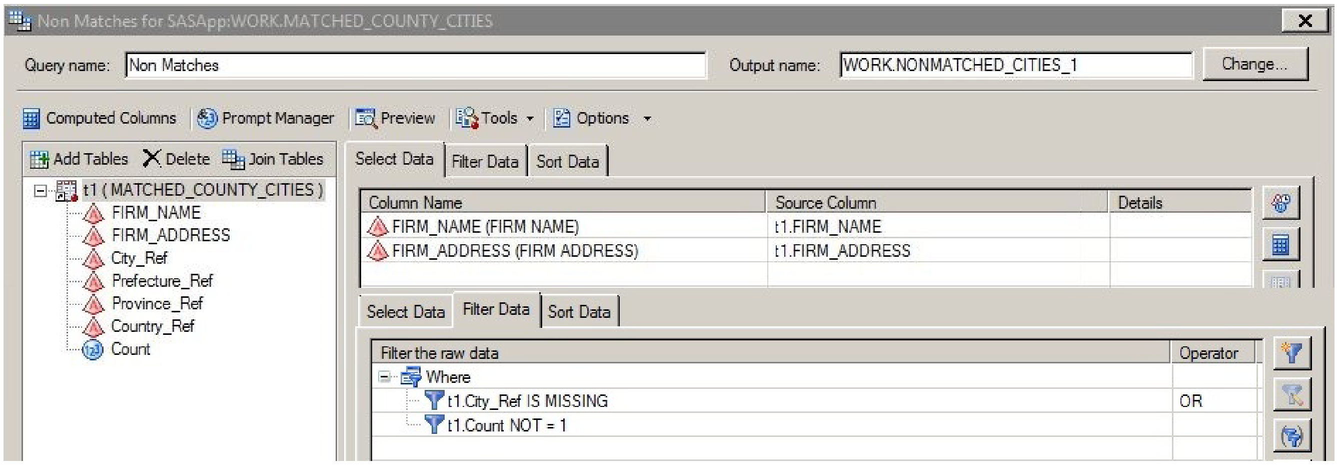
t1 (MATCHED_COUNTY_CITIES)
- FIRM_NAME
- FIRM_ADDRESS
- City_Ref
- Prefecture_Ref
- Province_Ref
- Country_Ref
- Count

Column Name	Source Column	Details
FIRM_NAME (FIRM NAME)	t1.FIRM_NAME	
FIRM_ADDRESS (FIRM ADDRESS)	t1.FIRM_ADDRESS	
City_Ref	t1.City_Ref	
Prefecture_Ref	t1.Prefecture_Ref	
Province_Ref	t1.Province_Ref	
Country_Ref	t1.Country_Ref	

Filter the raw data	Operator
Where	
t1.Count = 1	AND
t1.City_Ref NOT IS MISSING	

Display 4. Query for Matches

The query for non-matched addresses is very similar. Instead, the query filters for *t1.City_Ref* IS MISSING OR *t1.Count* NOT = 1. The first filter identifies any address that does not have a matched city and the second filter identifies any address that has multiple matches. This way, any addresses with multiple matches can be removed from the matched city output, which will eliminate any duplication in the final dataset. The OR statement only requires that one criterion is met. Lastly, only the original firm name and firm address needs to be selected for this query, so the original data can be reused in the next iteration of the PROC SQL sequence. It is important to check “Select distinct rows only,” so no duplicative rows are in the output.



Query name: Non Matches

Output name: WORK.NONMATCHED_CITIES_1

t1 (MATCHED_COUNTY_CITIES)
- FIRM_NAME
- FIRM_ADDRESS
- City_Ref
- Prefecture_Ref
- Province_Ref
- Country_Ref
- Count

Column Name	Source Column	Details
FIRM_NAME (FIRM NAME)	t1.FIRM_NAME	
FIRM_ADDRESS (FIRM ADDRESS)	t1.FIRM_ADDRESS	

Filter the raw data	Operator
Where	
t1.City_Ref IS MISSING	OR
t1.Count NOT = 1	

Display 5. Query for Non-Matches

The final step is to append all of the outputs together. Using the Append Table task, select the three matched addresses output tables from each query and the non-matched addresses output table from the final query to create one final output table. Now the final dataset contains the original address list with reference cities and provinces attached. The unstructured text data is now matched with structure text data that can be used for future analysis.

CONCLUSION

When it comes to matching Chinese words in SAS, fuzzy matching functions, such as SOUNDEX and COMPLEV, are ineffective. The PROC SQL code and SAS EG procedure presented in this paper is a work-around approach that can be used for other languages as well. In fact, it can also be used to search for a list of keywords in English in a text string given a reference table. The end result is a dataset of usable data for analysis.

REFERENCES

Moler, Cleve. August 14, 2017. “Levenshtein Edit Distance Between Strings.” Accessed September 5, 2019. https://blogs.mathworks.com/cleve/2017/08/14/levenshtein-edit-distance-between-strings/.

Sloan, Stephen and Kirk Paul Lafler. 2018. “Fuzzy Matching Programming Techniques Using SAS® Software.” *Proceedings of the Southeast SAS User Group 2018 Conference*, Paper 143-2018.

U.S. Consumer Product Safety Commission. “Violations.” Accessed July 17, 2019. Available at https://cpsc.gov/Recalls/violations.

Wikipedia. “List of cities in China.” Accessed July 17, 2019. Available at https://en.wikipedia.org/wiki/List_of_cities_in_China.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Arthur Laciak
U.S. Consumer Product Safety Commission
alaciak@cpsc.gov