Matching and Mapping for the TSDB (KairosDB)

SEED project

November 30, 2015

Contents

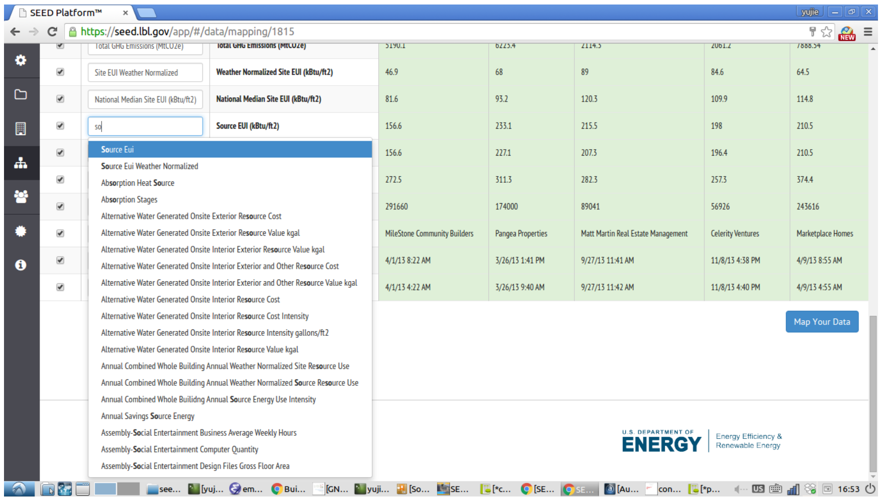
1 Introduction

The document records the process of implementing the mapping and matching operations for TSDB (KairosDB.

2 SEED Mapping and Matching

2.1 Mapping

Mapping rename the column/field names of the imported data set to terms in Building Energy Data Exchange Specification (BEDES) [5]. In the process, the program search through the terms in BEDES and returns a suggested field name for each of the imported field in the dataset. user can 1) choose which field they want to retain or ignore 2) modify the suggested mapping and input the BEDES term. During the input process, there is a list of 20 strings in the drop-down menu under the input bar each string contains the current input as a sub string (Figure 1).


SEED Platform™
https://seed.lbl.gov/app/#/data/mapping/1815
yujie
Source Eui
Source Eui Weather Normalized
Absorption Heat Source
Absorption Stages
Alternative Water Generated Onsite Exterior Resource Cost
Alternative Water Generated Onsite Exterior Resource Value kgal
Alternative Water Generated Onsite Interior Exterior Resource Value kgal
Alternative Water Generated Onsite Interior Exterior and Other Resource Cost
Alternative Water Generated Onsite Interior Exterior and Other Resource Value kgal
Alternative Water Generated Onsite Interior Resource Cost
Alternative Water Generated Onsite Interior Resource Cost Intensity
Alternative Water Generated Onsite Interior Resource Intensity gallons/ft2
Alternative Water Generated Onsite Interior Resource Value kgal
Annual Combined Whole Building Annual Weather Normalized Site Resource Use
Annual Combined Whole Building Annual Weather Normalized Source Resource Use
Annual Combined Whole Builidng Annual Source Energy Use Intensity
Annual Savings Source Energy
Assembly-Social Entertainment Business Average Weekly Hours
Assembly-Social Entertainment Computer Quantity
Assembly-Social Entertainment Design Files Gross Floor Area
Map Your Data
U.S. DEPARTMENT OF ENERGY
Energy Efficiency & Renewable Energy

✓	Total GHG Emissions (MtCO2e)	Total GHG Emissions (MtCO2e)	5190.1	6223.4	2114.5	2061.2	7888.54
✓	Site EUI Weather Normalized	Weather Normalized Site EUI (kBtu/ft2)	46.9	68	89	84.6	64.5
✓	National Median Site EUI (kBtu/ft2)	National Median Site EUI (kBtu/ft2)	81.6	93.2	120.3	109.9	114.8
✓	so	Source EUI (kBtu/ft2)	156.6	233.1	215.5	198	210.5
✓			156.6	227.1	207.3	196.4	210.5
✓			272.5	311.3	282.3	257.3	374.4
✓			291660	174000	89041	56926	243616
✓			MileStone Community Builders	Pangea Properties	Matt Martin Real Estate Management	Celerity Ventures	Marketplace Homes
✓			4/1/13 8:22 AM	3/26/13 1:41 PM	9/27/13 11:41 AM	11/8/13 4:38 PM	4/9/13 8:55 AM
✓			4/1/13 4:22 AM	3/26/13 9:40 AM	9/27/13 11:42 AM	11/8/13 4:40 PM	4/9/13 4:55 AM

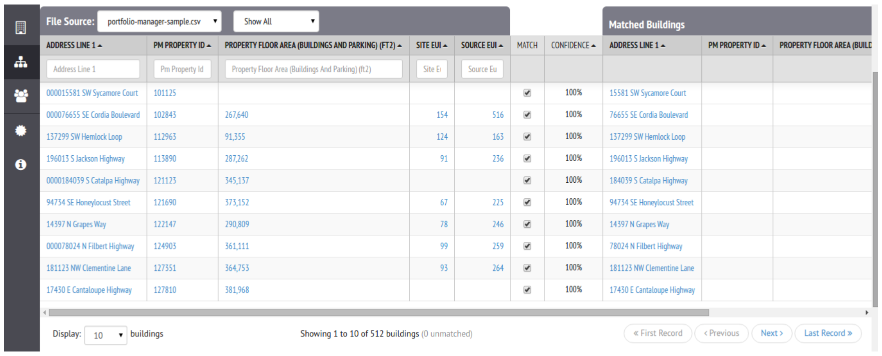
Figure 1: Drop-down list for input BEDES term
[3]

2.2 Matching

In the matching process, the fields in the two input tables, the building list and the PM data, are combined. The combining process utilizes some common fields from the two tables.

2.2.1 Automatic

The process uses fuzzy string searching [2,6] to auto-match the records in the two tables and returns a confidence score for the matching. In Figure 2 we can see the leading zeros does not affect the matching result.



ADDRESS LINE 1	PM PROPERTY ID	PROPERTY FLOOR AREA (BUILDINGS AND PARKING) (FT2)	SITE EUI	SOURCE EUI	MATCH	CONFIDENCE	ADDRESS LINE 1	PM PROPERTY ID	PROPERTY FLOOR AREA (BUILD
000015581 SW Sycamore Court	101125				✓	100%	15581 SW Sycamore Court		
000076655 SE Cordia Boulevard	102843	267,640	154	516	✓	100%	76655 SE Cordia Boulevard		
137299 SW Hemlock Loop	112963	91,355	124	163	✓	100%	137299 SW Hemlock Loop		
196013 S Jackson Highway	113890	287,262	91	236	✓	100%	196013 S Jackson Highway		
0000184039 S Catalpa Highway	121123	345,137			✓	100%	184039 S Catalpa Highway		
94734 SE Honeylocust Street	121690	373,152	67	225	✓	100%	94734 SE Honeylocust Street		
14397 N Grapes Way	122147	290,809	78	246	✓	100%	14397 N Grapes Way		
000078024 N Filbert Highway	124903	361,111	99	259	✓	100%	78024 N Filbert Highway		
181123 NW Clementine Lane	127351	364,753	93	264	✓	100%	181123 NW Clementine Lane		
17430 E Cantaloupe Highway	127810	381,968			✓	100%	17430 E Cantaloupe Highway		

Display: 10 buildings   Showing 1 to 10 of 512 buildings (0 unmatched)   « First Record ‹ Previous Next › Last Record »

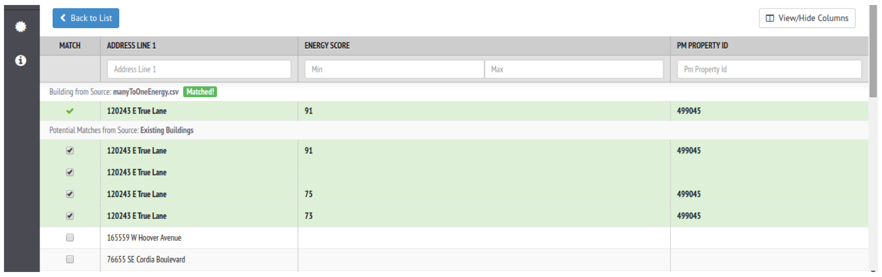
Figure 2: Matching result with confidence score

[3]

The guideline for the process is "to improve results in matching buildings across different data files, map as many of the following four (4) fields as possible: Tax Lot ID, PM Property ID, Custom ID, Address Line 1" [3].

2.2.2 Manual

The matching can be manually corrected by clicking on the value of the shared field in the source table and one can choose **one or more** records from the drop-down list that matches the record in the source table (Figure 3).

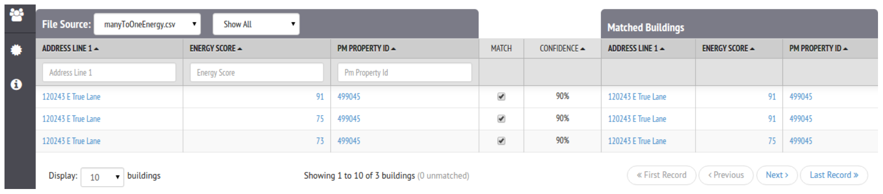


MATCH	ADDRESS LINE 1	ENERGY SCORE	PM PROPERTY ID
Building from Source: manyToOneEnergy.csv Matched!			
✓	120243 E True Lane	91	499045
Potential Matches from Source: Existing Buildings			
✓	120243 E True Lane	91	499045
✓	120243 E True Lane		
✓	120243 E True Lane	75	499045
✓	120243 E True Lane	73	499045
☐	165559 W Hoover Avenue		
☐	76655 SE Cordia Boulevard		

Figure 3: Manually correct matching result by selecting one or more potential record

[3]

In the matching process, one table is the source and the other is the target. For each record/row in the target table, if there exists a unique record in the source table that matches this record, a match will be successful, but the score of confidence will not be 100% if there are multiple records in the target that matches the source.



File Source: manyToOneEnergy.csv | Show All

Matched Buildings

ADDRESS LINE 1	ENERGY SCORE	PM PROPERTY ID	MATCH	CONFIDENCE	ADDRESS LINE 1	ENERGY SCORE	PM PROPERTY ID
Address Line 1	Energy Score	Pm Property Id					
120243 E True Lane	91	499045	✓	90%	120243 E True Lane	91	499045
120243 E True Lane	75	499045	✓	90%	120243 E True Lane	91	499045
120243 E True Lane	73	499045	✓	90%	120243 E True Lane	75	499045

Display: 10 buildings   Showing 1 to 10 of 3 buildings (0 unmatched)

Figure 4: Three records in the target table (PM table) matches one record in the source table (Building table)

[3]

3 Implementation strategy of mapping and matching: approximate string matching

The approximate string matching aims at finding strings that *approximately* match some pattern. The matching is normally evaluated by some edit distance, which is the minimum number of primitive operations (e.g. insertion, deletion and substitution) needed to convert the approximate match to an exact match [6]. There are several versions of the set of primitive operations. One common definition is the Levenshtein distance, which include single-character operations as insertion, deletion and substitution.

There is a package in Python called FuzzyWuzzy [4] that evaluates the difference between strings with Levenshtein distance. The package is built upon the Python package difflib (which has a class called "SequenceMatcher" that compares two sequences (str, unicode, list, tuple, bytearray, buffer, xrange) as long as they are hashable (those that can become a dictionary key). Immutable types (number, string, tuples) are all hashable in Python). There are some explanation of the FuzzyWuzzy package here. The key functions include [1]:

```
from fuzzywuzzy import fuzz
# simple ratio: pure edit distance
# similar to difflib.SequenceMatcher
>>> fuzz.ratio("this is a test", "this is a test!")
    96

# partial ratio: when s1 and s2 have very different lengths
# WOLG, s1 < s2,
# partial_ratio(s1, s2) returns fuzz.ratio(s1, s2') where s2' is a
# sub string of s2 and len(s1) == len(s2')
```

```
>>> fuzz.partial_ratio("this is a test", "this is a test!")
    100

# token sort ratio:
# to deal with the word re-order of strings
break strings to tokens, sort tokens and then re-assemble them to strings before calculating tl
fuzz.token_sort_ratio("fuzzy wuzzy was a bear", "wuzzy fuzzy was a bear")

# token set ratio:
# another way to deal with the word re-order of strings
# do not use it if duplicate words are important patterns
# let the input strings be s0 and s1
# t0 = intersection(sorted(s0), sorted(s1))
# s0' = sorted(s0) \ t0
# s1' = sorted(s1) \ t1
# t1 = intersection(t0, s0')
# t2 = intersection(t0, s1')
# return max(fuzz.ratio(t0, t1), fuzz.ratio(t1, t2), fuzz.ratio(t0, t2))
>>> fuzz.token_sort_ratio("fuzzy was a bear", "fuzzy fuzzy was a bear")
    84
>>> fuzz.token_set_ratio("fuzzy was a bear", "fuzzy fuzzy was a bear")
    100

# extracting a list of tuples of (str, score) where str is in choices
# and score is the matching score between query and choice)
# scorer: the ratio calculation method, can also be user defined
# limit: length of the returned list
>>> extract(query, choices, processor=None, scorer=None, limit=5)

>>> choices = ["Atlanta Falcons", "New York Jets", "New York Giants", "Dallas Cowboys"]
>>> process.extract("new york jets", choices, limit=2)
    [('New York Jets', 100), ('New York Giants', 78)]
```

However, if the address line is selected as the field for matching calculation, a **substitution of common abbreviations** should be performed before the string searching process (Figure 2 in this paper).

Now I wrote a wrapper of matching function with FuzzyWuzzy package, next step is to know which field to match (address or not) find a set of testing source and target strings to see if the matching function works.

References

[1] JeffPaine et al. seatgeek/fuzzywuzzy. web, November 2015. https://github.com/seatgeek/fuzzywuzzy.

[2] Lawrence Berkeley National Laboratory. Seed 1.1 tutorial. web, November 2015. https://windows.lbl.gov/projects/SEED/IntroTutorial/SEED%201.2%20Overview.htm.

[3] Lawrence Berkeley National Laboratory. Seed platform. web, November 2015. https://seed.lbl.gov/app/#/data/mapping/1815.

[4] Python Software Foundation. fuzzywuzzy 0.8.0. web, November 2015. https://pypi.python.org/pypi/fuzzywuzzy.

[5] U.S. Department of Energy. Bedes. web, November 2015. https://bedes.lbl.gov/.

[6] Wikipedia. Approximate string matching. web, November 2015. https://en.wikipedia.org/wiki/Approximate_string_matching.