A System for Efficient Cleaning and Transformation of Geospatial Data Attributes

Yao-Yi Chiang
University of Southern California
Spatial Sciences Institute
Los Angeles, CA 90089, USA
yaoyic@usc.edu

Bo Wu
Akshay Anand
Ketan Akade
University of Southern California
Department of Computer Science
Los Angeles, CA 90089, USA
bowu@isi.edu, [akshayan, akade]@usc.edu

Craig A. Knoblock
University of Southern California
Department of Computer Science and Information Sciences Institute
Los Angeles, CA 90089, USA
knoblock@isi.edu

ABSTRACT

A significant challenge in handling geographic datasets is that the datasets can come from heterogeneous sources with various data qualities and formats. Before these datasets can be used in a Geographic Information System (GIS) for spatial analysis or to create maps, a typical task is to clean the attribute data and transform the data into a uniform format. However, conventional GIS products focus on manipulating the spatial component of geographic features and only offer basic tools for editing the attribute data (e.g., one row at a time). This limits the capability for handling large datasets in a GIS since manually editing and transforming attribute data between different formats is not practical for thousands of geographic features. In this demo, we present ArcKarma, which is built on our previous work on data transformation, to efficiently clean and transform data attributes in a GIS. ArcKarma generates transformation programs from a few user-provided examples and applies these programs to transform individual attribute columns into the desired formats. We show that ArcKarma produces accurate results and eliminates the need for laborious manual data cleaning and scripting tasks.

Categories and Subject Descriptors

H.4.m [**Information Systems Applications**]: Miscellaneous—*Geographic Information Systems*; H.2.8 [**Database Management**]: Database Applications—*Spatial Databases and Geographic Information Systems*

General Terms

Algorithms, Design, Human Factors

SIGSPATIAL '14, Nov 04-07 2014, Dallas/Fort Worth, TX, USA
Copyright 2014 ACM 978-1-4503-3131-9/14/11...$15.00
http://dx.doi.org/10.1145/2666310.2666373.

Keywords

Geographic information system, data cleaning, data transformation

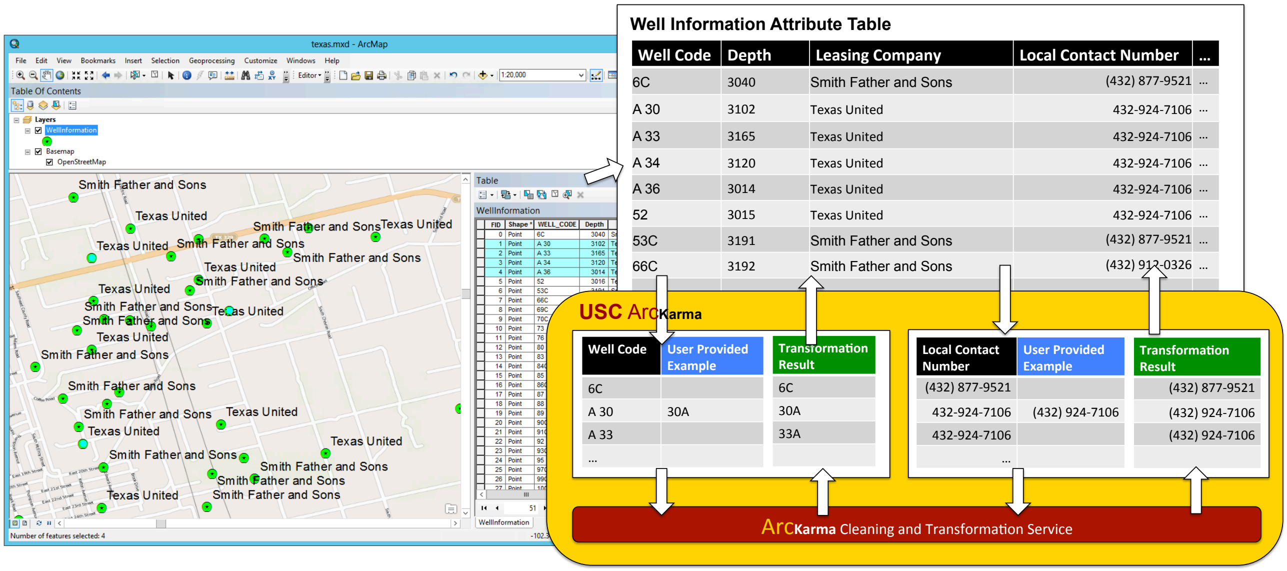
1 INTRODUCTION

A key problem facing geographic information system (GIS) users is that the geographic datasets used in a project very often come from multiple sources, and the attribute data of these datasets can be presented in a variety of data formats (e.g., telephone numbers can be stored as "213-740-2311" or "(213) 740-2311"). In addition, even individual geographic features within one data source can have different formats for an attribute because the datasets were collected at different time periods or a consistent attribute format was not enforced during data entry. A GIS supports a wide range of operations to integrate the spatial component of geographic datasets (e.g., data overlay and spatial join). However, linking and joining geographic features using attribute data generally relies on manually preparing data or requires task-dependent programming work to first ensure the feature attributes are in a uniform format. These data preparation steps can be laborious and require scripting or programming skills.

This paper presents ArcKarma, an interactive data transformation tool, which is integrated with a conventional GIS product (Esri ArcMap). ArcKarma employs a training-by-example strategy to generate transformation programs and automatically transforms attribute data to a uniform format. ArcKarma does not require any in scripting or programming. In the remainder of this paper, Section 2 presents the ArcKarma user interface and workflow, Section 3 describes the ArcKarma data cleaning and transformation algorithm, Section 4 presents related systems, and Section 5 discusses the future work.

2 WORKFLOW AND USER INTERFACE

Figure 1 shows the ArcKarma architecture and a typical use case. The "Well Information" layer in this ArcMap project contains an attribute table of oil well information for maintaining the records of wells leased from a petroleum production company. This list is an integration of well data


texas.mxd - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
Table Of Contents
Layers
WellInformation
Basemap
OpenStreetMap
Table
WellInformation
Number of features selected: 4
USC ArcKarma
ArcKarma Cleaning and Transformation Service

Well Information Attribute Table

Well Code	Depth	Leasing Company	Local Contact Number	...
6C	3040	Smith Father and Sons	(432) 877-9521	...
A 30	3102	Texas United	432-924-7106	...
A 33	3165	Texas United	432-924-7106	...
A 34	3120	Texas United	432-924-7106	...
A 36	3014	Texas United	432-924-7106	...
52	3015	Texas United	432-924-7106	...
53C	3191	Smith Father and Sons	(432) 877-9521	...
66C	3192	Smith Father and Sons	(432) 912-0326	...

Well Code	User Provided Example	Transformation Result
6C		6C
A 30	30A	30A
A 33		33A
...		

Local Contact Number	User Provided Example	Transformation Result
(432) 877-9521		(432) 877-9521
432-924-7106	(432) 924-7106	(432) 924-7106
432-924-7106		(432) 924-7106
...		

Figure 1: ArcKarma workflow

from multiple sources and contains data created at different time periods. The spatial component of the "Well Information" layer contains the well locations (in this case, given as point data). The attribute table stores auxiliary information about individual wells. Before the year 2003, the petroleum company used one alphabetic character, one space, and one numeric string to identify the well types (e.g., "A 30"), and the current system uses a new naming convention of one or more numeric characters followed by an alphabetic character, (e.g., "6C"). As the information about the wells leased from the company Texas United existed before the system changed, these well codes follow the old naming convention. In addition, the previous system user interface did not enforce a standard format during data entry and hence the contact numbers could be in various formats and do not follow the current standard.

To normalize the data format for both the "Well Code" and "Local Contact Number" attributes, the user enters example values of the desired format (the blue columns in Figure 1). ArcKarma sends the original data (the black columns in Figure 1) and the example values to the cleaning and transformation service to transform individual attribute values. Finally, the transformation results (the green columns in Figure 1) are pushed back to the "Well Information" attribute table in ArcMap.

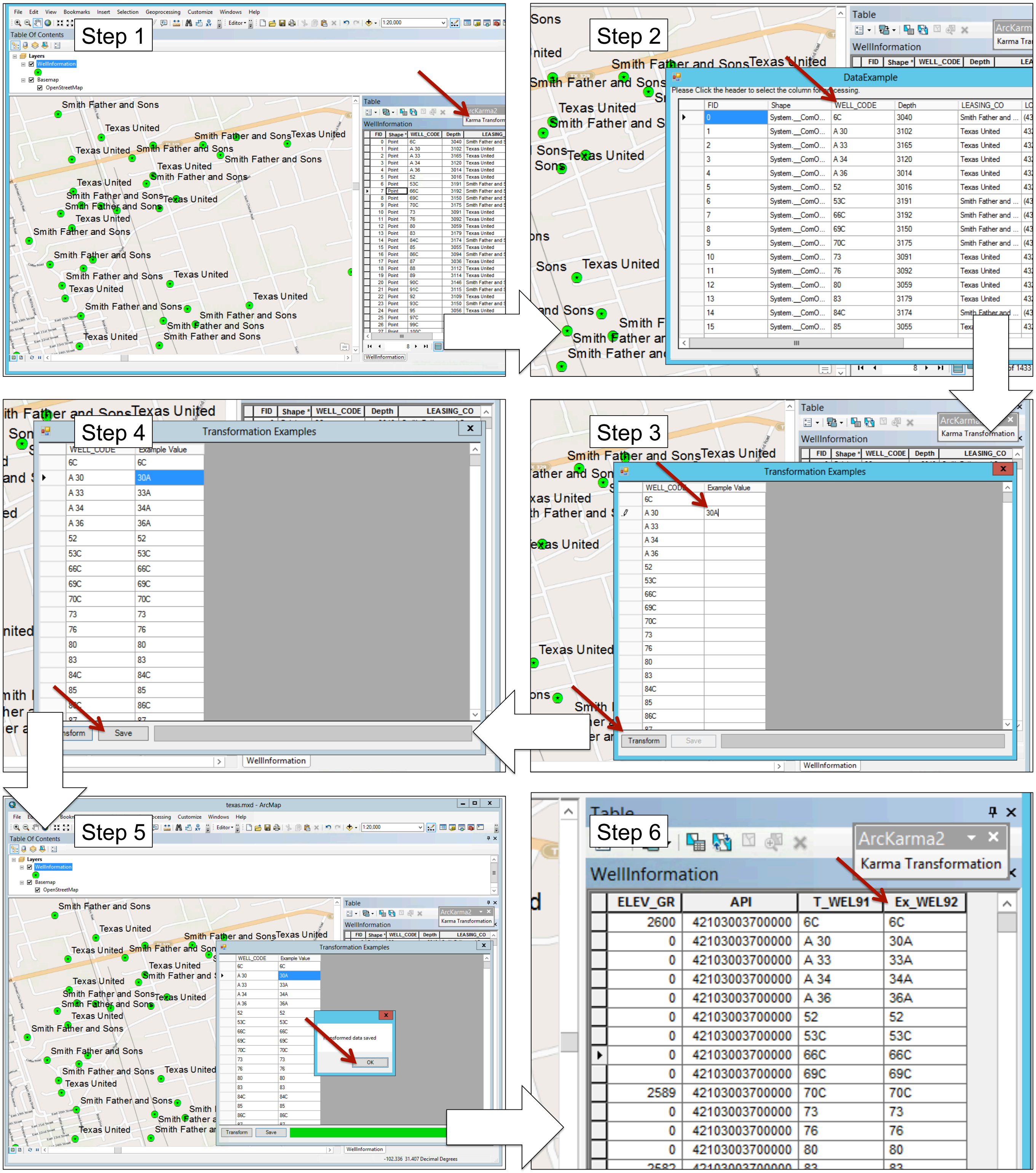
Figure 2 shows the ArcKarma user interface. Within the ArcMap environment, the user clicks the "Karma Transformation" button and a table shows up. The user can select the desired attribute for transformation by clicking on the column header (steps 1 and 2). In this example, the user chooses to transform the "WELL_CODE" attribute and provides an example value of "30A" for the original value "A 30" (step 3). After the user finishes providing the transformation example(s), the user clicks the "Transform" button to generate the transformation results. The user can then click the "Save" button to send the results back to the attribute table in ArcMap (steps 4, 5, and 6). In addition to immediately saving the transformation results, the user can edit, discard, or add more examples if the results are not satisfactory. During the entire process, the user stays in ArcMap and does not need to manually import/export datasets from/to different software platforms.

3 DATA CLEANING AND TRANSFORMATION SERVICE

The ArcKarma data cleaning and transformation service (DCTS) is an implementation of our previous work in [6] that synthesizes transformation programs using examples. This previous work is based on the programming-by-example approach developed by Gulwani [2], which defines a string transformation language that supports a restricted, but expressive form of regular expressions including conditionals and loops. In this section, we first present a sample transformation program and then describe how DCTS generates the program.

Figure 3 shows a sample transformation program generated by DCTS using an example with the original value as "A 30" and the target value as "30A". For a raw data input (e.g., "A 30"), the program first uses a classifier to identify the format of the input string and then selects a transformation branch to transform the input string (i.e., *Switch(OriginalFormat)*). Each transformation branch (e.g., *Case "Format 1":*) contains the code to convert the input string from a certain format to the target format. The transformation code is a concatenation of several substring expressions (e.g., *RawData.SubString(P1, P2)*). A substring expression can either be (1) a constant string or (2) an extraction rule describing how to extract specific substrings from the raw data.

An extraction rule has two position statements (e.g., *P1* and *P2*) that identify the start and end positions of a substring. The positions can be stated using (1) an absolute position or (2) a relative position, which are represented as (*LCXT*, *RCXT*, *OCC*). For a given position, the *LCXT* describes the context to the left of the position, the *RCXT* describes the context to the right of the position, and the *OCC* is the ordinal number of the occurrence. For example, the position of "A" in "A 30" can be expressed as an absolute position *0* or a relative position (*START*, *UpperCaseWord*, *1*). Here "*LCXT* = *START*" means that the the



Step 2: DataExample

Please Click the header to select the column for processing.

FID	Shape	WELL_CODE	Depth	LEASING_CO
0	System.__ComO...	6C	3040	Smith Father and ...
1	System.__ComO...	A 30	3102	Texas United
2	System.__ComO...	A 33	3165	Texas United
3	System.__ComO...	A 34	3120	Texas United
4	System.__ComO...	A 36	3014	Texas United
5	System.__ComO...	52	3016	Texas United
6	System.__ComO...	53C	3191	Smith Father and ...
7	System.__ComO...	66C	3192	Smith Father and ...
8	System.__ComO...	69C	3150	Smith Father and ...
9	System.__ComO...	70C	3175	Smith Father and ...
10	System.__ComO...	73	3091	Texas United
11	System.__ComO...	76	3092	Texas United
12	System.__ComO...	80	3059	Texas United
13	System.__ComO...	83	3179	Texas United
14	System.__ComO...	84C	3174	Smith Father and ...
15	System.__ComO...	85	3055	Texa

Step 4: Transformation Examples

WELL_CODE	Example Value
6C	6C
A 30	30A
A 33	33A
A 34	34A
A 36	36A
52	52
53C	53C
66C	66C
69C	69C
70C	70C
73	73
76	76
80	80
83	83
84C	84C
85	85
86C	86C

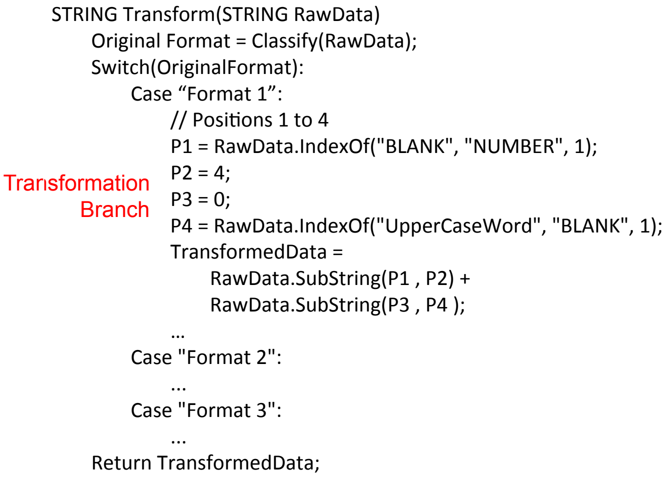
Step 6: WellInformation

ELEV_GR	API	T_WEL91	Ex_WEL92
2600	42103003700000	6C	6C
0	42103003700000	A 30	30A
0	42103003700000	A 33	33A
0	42103003700000	A 34	34A
0	42103003700000	A 36	36A
0	42103003700000	52	52
0	42103003700000	53C	53C
0	42103003700000	66C	66C
0	42103003700000	69C	69C
2589	42103003700000	70C	70C
0	42103003700000	73	73
0	42103003700000	76	76
0	42103003700000	80	80
2582	42103003700000	83	83

Figure 2: ArcKarma user interface. Steps 1 and 2: Selecting the attribute column for transformation. Step 3: Providing a transformation example. Steps 4, 5, and 6: Reviewing the transformation result and saving the result back to the attribute table in ArcMap.

left position of "A" is the beginning of the raw data. "*RCXT = UpperCaseWord*" means that the right position of "A" is an uppercase letter. "*OCC = 1*" means that the first occurrence of a position that meets the conditions "*LCXT = START*" and "*RCXT = UpperCaseWord*".

There are other token types that can be used: *END* represents the end of the raw data. *LowerCaseWord* represents a continuing sequence of lowercase letters. *NUMBER* represents a continuing sequence of digits. *BLANK* represents a space. Using the list of position expressions *P1*, *P2*, *P3*, and *P4*, the transformation program extracts specific substrings and concatenates them to form the output (*TransformedData*).


Transformation
Branch

```
STRING Transform(STRING RawData)
    Original Format = Classify(RawData);
    Switch(OriginalFormat):
        Case "Format 1":
            // Positions 1 to 4
            P1 = RawData.IndexOf("BLANK", "NUMBER", 1);
            P2 = 4;
            P3 = 0;
            P4 = RawData.IndexOf("UpperCaseWord", "BLANK", 1);
            TransformedData =
                RawData.SubString(P1 , P2) +
                RawData.SubString(P3 , P4 );
            ...
        Case "Format 2":
            ...
        Case "Format 3":
            ...
    Return TransformedData;
```

Figure 3: Program generated by the Data transformation Module

To learn a transformation program, DCTS first separates the target values into string segments. For example, the raw data "30A" is separated into two segments "30" and "A" as shown in Figure 4. DCTS then tries to independently generate these segments by either extracting a substring from the input or inserting a new string. For instance, the segment "30" in the target string can be extracted from the original input. With these alignments, DCTS then identifies the locations from which the substrings are extracted, and provides a variety of ways specifying these locations such as using absolute or relative positions.

If there are multiple examples, DCTS uses an efficient algorithm to construct a version space for each example [4]. A version space contains a set of transformation programs that are generated with an example. DCT finds the version space that is consistent with every example to consolidate multiple version spaces into one. However, if there are examples that cannot be covered by a single version space, DCTS partitions the examples and generates a version space for each partition individually. Each partition corresponds to an input format. A classifier can be used to distinguish multiple different formats. Finally, to efficiently generate a transformation program from a version space, DCTS defines a partial order over the version space. This partial order helps to generate the simplest program first (Occam's principle).

4 RELATED WORK

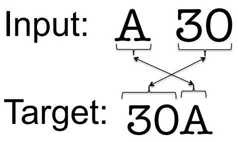


Figure 4: Learning the transformation programs

OpenRefine[1] and Potter's wheel [5] allow the user to specify string edit operations. OpenRefine is a data cleaning tool that supports a regular expression style of string transformation and data layout transformation. Potter's Wheel has predefined transformation operations and users gradually build transformation programs by adding or undoing transformation operations using an interactive user interface. Many programming-by-demonstration approaches can learn edit operations by asking the user to demonstrate the editing process [1]. Lau [4] presents a system that learns from a user's edit operations to generate a sequence of text editing programs. Data Wrangler [3] is an interactive tool that uses the transformation operations defined in Potter's Wheel to create data transformation programs. In addition to supporting string level transformation, Data Wrangler also supports data layout transformation including column split, column merge, fold, and unfold. In contrast, ArcKarma only requires the user to enter a few examples of the desired data value without asking the user to demonstrate the format conversion steps.

5 DISCUSSION AND FUTURE WORK

We plan to integrate two additional capabilities from [6] to enhance the usability of ArcKarma: (1) recommending rows to provide examples and (2) highlighting the corresponding substrings between the raw and transformed values, which helps the user to visualize the applied transformation.

6 ACKNOWLEDGMENTS

This work is supported in part by Chevron U.S.A. Inc. under the joint project Center for Interactive Smart Oilfield Technologies (CiSoft), at the University of Southern California.

References

[1] A. Cypher, D. C. Halbert, D. Kurlander, H. Lieberman, D. Maulsby, B. A. Myers, and A. Turransky, editors. *Watch what I do: programming by demonstration*. MIT Press, 1993.

[2] S. Gulwani. Automating string processing in spreadsheets using input-output examples. In *POPL*, pages 317–330, 2011.

[3] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer. Wrangler: interactive visual specification of data transformation scripts. In *CHI*, pages 3363–3372, 2011.

[4] T. Lau, S. A. Wolfman, P. Domingos, and D. S. Weld. Programming by demonstration using version space algebra. *Mach. Learn.*, pages 111–156, 2003.

[5] V. Raman and J. M. Hellerstein. Potter's wheel: An interactive data cleaning system. In *VLDB*, pages 381–390, 2001.

[6] B. Wu, P. Szekely, and C. A. Knoblock. Minimizing user effort in transforming data by example. In *IUI*, pages 317–322, 2014.

[1] http://openrefine.org