Home
People
Research
Publications

**Current Projects**
- **Data Cleaning & Integration**
- **Data Science**
- **Magellan**
  - py_entitymatching
  - py_stringsimjoin
  - py_stringmatching
  - activelearn
  - py_labeler

**Useful Stuff**
Data Repository
How to Create Packages

**Past Projects**
Knowledge Bases/Graphs
Crowdsourcing
Schema/Ontology Matching
Silicon Valley Time Off

**Courses**
- **CS 638 DS**
  - Fall 2016
- **CS 838 DS**
  - Spring 2018
- **CS 564**

**Outreach & Funding**
Walmart Labs
NSF IIS-Medium 2016

**Miscellaneous**
Wisconsin DB Group

# CS 839 Spring 2018, Project Stage 3

In this project stage your team will perform entity matching (EM). Recall that in Project Stage 2 you have created two tables. In this project stage you will find entities across these two tables that match. Specifically, you will do the following.

Suppose the two tables that you have extracted are called A and B. These two tables should have the same schema, and each tuple in each table must describe a single entity (all of the same type). For example, if the entity type is  person, then each tuple describes a person, and a possible table schema can be A(name, city, state, zip, phone) (and the same schema for Table B).
**Each table must have at least 3000 tuples.**

Now you will use Magellan, an EM tool (developed here at UW) to match the two tables. Specifically, you should do the following:

- Download and install Magellan on your machine. See the Magellan's homepage (the Python package for Magellan is called py_entitymatching). There is a detailed instruction on that homepage on how to install Magellan.
- Read the how-to guide (also available on that homepage).
- Now read through the user manual of Magellan to get a feel for how to use it. There is a set of Jupyter notebooks that should give you an idea of how to use Magellan.
- Now use Magellan to match the two tables A and B. Pay particular attention to Section 3.8 (selecting and debugging a matcher: a general procedure) in the how-to guide, because that's what you will have to follow. Your goal should be to **create a matcher that achieves precision of at least 90% and recall as high as possible.**

**Some More Information**

Please note that Table A and B must be in the csv format with a mandatory header line representing the attribute names. The header line is of the form:

attribute_name1, attribute_name2, ….

Below the header line follow the tuples.  Each tuple is a comma-separated list of the attribute values.

Some guidelines to specify data in csv format are given below:

- Each line must contain the same number of comma-separated fields
- Fields containing comma must be double quoted
- Missing value is represented by an empty field value.

For Tables A and B, each tuple must have an ID attribute (this should be the first attribute of the table).

Below is a sample table in csv format:

```
ID,name,birth_year,hourly_wage,address,zipcode
a1,Kevin Smith,1989,30,"607 From St, San Francisco",94107
a2,Michael Franklin,1988,27.5,"1652 Stockton St, San Francisco",94122
a3,William Bridge,1986,32,"3131 Webster St, San Francisco",94107
a4,Binto George,1987,32.5,"423 Powell St, San Francisco",94122
a5,Alphonse Kemper,1984,,"1702 Post Street, San Francisco",94122
```

Note that for ID a5, the hourly_wage value is missing.

**What to Submit?**

Recall that in the project stage you have two tables A and B, each with at least 3000 tuples. You have done blocking on these tables to obtain a set of candidate tuple pairs C. Then you have taken a sample S from C and label it. Finally, you have used the labeled set to develop a matcher that achieves precision of at least 90% and recall as high as possible.

Submit the following on your team's project page, by the deadline:

- **Provide a link to a DATA directory** that stores both tables A and B. They should be stored in two files, each file storing a table in CSV format. We should be able to browse these files to examine the tables. There should be a README file in the same directory that explains the tables (e.g., the meaning of the attributes) and lists the number of tuples per table.

- In the same directory
    - provide a file that lists all tuple pairs that survive the blocking step.
    - provide a file that lists all tuple pairs in the sample you have taken, together with the labels, one label per each tuple pair. This file is referred to as file G in Sec 3.8 of the how-to guide.
    - provide two files that describe the sets I and J, respectively (see Sec 3.8 of the how-to guide for an explanation of these files).

- **Provide a link to a CODE directory** that stores all of your code (this directory must also be browsable). In this directory, also provide a Jupyter notebook that describes your blocking and matching process (see below for details).

- **Provide a link to a pdf document** that contains at least the following:
  - Describe the type of entity you want to match, briefly describe the two tables (e.g., where did you obtain these tables), list the number of tuples per table.
  - Describe the blocker that you use and list the number of tuple pairs in the candidate set obtained after the blocking step.
  - List the number of tuple pairs in the sample G that you have labeled.
  - For each of the six learning methods provided in Magellan (Decision Tree, Random Forest, SVM, Naive Bayes, Logistic Regression, Linear Regression), report the precision, recall, and F-1 that you obtain when you perform cross validation **for the first time** for these methods on I.
  - Report which learning based matcher you selected after that cross validation.
  - Report all debugging iterations and cross validation iterations that you performed. For each debugging iteration, report (a) what is the matcher that you are trying to debug, and its precision/recall/F-1, (b) what kind of problems you found, and what you did to fix them, (c) the final precision/recall/F-1 that you reached. For each cross validation iteration, report (a) what matchers were you trying to evaluate using the cross validation, and (b) precision/recall/F-1 of those.
  - Report the final best matcher that you selected, and its precision/recall/F-1.
  - **It is important to note that all precision/recall/F-1 numbers asked for in the aboves are supposed to be numbers obtained via CV on the set I. Do not yet use set J.**
  - Now report these numbers:
    - For each of the six learning methods, train the matcher based on that method on I, then **report its precision/recall/F-1 on J.**
    - For the final best matcher Y selected, train it on I, then **report its precision/recall/F-1 on J.**
  - Report approximate time estimates: (a) to do the blocking, (b) to label the data, (c) to find the best matcher.
  - Provide a discussion on why you didn't reach higher recall, and what you can do in the future to obtain higher recall.
  - **BONUS POINTS:** provide comments on what is good with Magellan and what is bad, that is, as users, what else would you like to see in Magellan. Are there any features/capabilities that you would really like to see being added? Any bugs? Depending on how detailed and helpful these comments are, you can get bonus point from 1-10 (which will help with the final grade, not just with the

project).

- **Jupyter notebook:** recall that in the CODE directory you should have a Jupyter notebook. This is because once you are done with your task, you often must communicate what you did and what you found to other fellow data scientists, the rest of the company, and the world. And usually a good way to do so is to produce a Jupyter notebook. This step asks you to practice this skill. Here's the instruction for creating the Jupyter notebook.

## Comments

You do not have permission to add comments.