Name matching

[WBAA Recruiting case]

Amsterdam, 2017

ING

Business context

ING would like empower its customers to stay one step ahead in life and in business.

One of the strategic focus points to do this is Advanced Analytics.

One of the key problems to solve is to link external data sources to our own ‘basis administration’;
We need to excel at this, to get the best understanding of our customers.
An example of this is linking Bloomberg data-streams to a client_id within ING systems.

The challenge in this is that the spelling/conventions of external sources might differ from our ‘basis administration’.
This is one of the tasks that we apply Data Science to.

We are curious to see how you are able to solve these types of challenges.

Our question to you

We would like you to build a prototype model that matches any external data-source (`S`) to our ground-truth administration (`G`)

- Base this matching on the name that appears in `S` only.

Your solution should provide a prediction for each entry in `S` whether:

- It is in `G` → provide the id in `G` that this entry is matched to;
- It is not in `G` → provide a `-1` for these

For example you will be given (here we use dictionaries, but will be plain text column separated format in practice, see next page):

- `G = {1: "ING Bank NV", 42: "Foo NV", 255: "Bar GmbH"}`
- `S = {1: "ING bannk N.V.", 2: "Bar Deutschland", 4: "Quz"}`

The ideal outcome of your model would be:

`{1: 1, 2: 255, 4: -1}`

We evaluate this model on the following cost function: matched correct = 0, matched incorrect to `G` = 5, matched incorrect as 'not `G`' = 1.

- Note that multiple entries in `S` can be linked correctly to a single entry in `G;`
- The cost of `{1: 1, 2: 255, 4: -1}` is `0`, as the outcome is 'perfect'.
- The cost of `{1: 1, 2: 255, 4: 1}` is `5`, as the entry with key `4` is incorrectly linked to `1` in `G`.
- The cost of `{1: -1, 2: 255, 4: -1}` is `1`, as the entry with key `1` is incorrectly labelled `-1` ('not in `G`').

Description of data provided

All data is provided as plain text column separated files:

- UTF-8 encoded, column separated by '|' (bar), one line per entry;
- the test set will have the same size as `STrain` provided to you now (hence, no memory issues during test).

We provide you up-front with three datasets: `G`, `STrain` and `sample_submission`

- `G` has as columns:
  - `company_id`: the id of the company in our ground-truth administration
  - `name`: the name of the company in our ground-truth administration
- `STrain` has as columns:
  - `train_index:` an index of the company in the external source dataset;
  - `name`: the name of the company as represented in the external source dataset;
  - `company_id`: the correct match of this entry to `G`. Is `-1` if correct label is 'not in `G`', otherwise corresponds to `G_id`
- `sample_submission` has as columns:
  - `test_index`: index of the company in an external source dataset (note: different index from `STrain`, full `STest` will only be provided during the interview);
  - `company_id`: the predicted match of this entry to `G`. Note that in this file, these are randomly generated predictions.

You need to design, code and train a model that predicts `company_id`, minimizing the cost function as specified on the previous page

Please make sure that your trained model:

- accepts as input a plain text file of the format `STest`, containing two columns `test_index` and `name;`
- runs from the command line, using as input the path to the file `STest`;
- prediction time should be 'near real time', i.e. about 1 minute for 10,000 entries (on a regular laptop);
- It should return a file with the above plain text format, including the columns `test_index` and `company_id` (an example submission is provided).

What does the process look like for this assignment?

Make sure you finish this assignment and share your solution directly with the interviewers:

- the assignment typically takes approximately 4 to 8 hours to complete;
- please send your code via email, at least 1 day before the interview, to allow enough time for review;
- provide a readme to give instructions on how to run your script.

You can use any *open-source tool* and *programming language* you like (Python, Scala, R, Java, C++, ...).
Please *do not* use commercial software like SAS, MatLab, Oracle etc.

You can use any library / package available, provided it is reproducible for the reviewers.
No need to program everything from scratch: for example, you are allowed to use Tensorflow, Postgres, Docker, etc...

During the interview:

- you will explain the logic and demonstrate the working of your model;
- both your approach and methodology, and the quality of your code will be reviewed and discussed;
- we will provide a separate *test-set* (`STest`) to evaluate the performance of your model.