Traces Through Time

Prosopography in practice across Big Data

Record Linker Documentation

Universiteit Leiden

Authors :

- Kleanthi Georgala
- Benjamin van der Burgh
- Cláudio Rebelo de Sá
- Arno Knobbe

Document version : 2.0
Application version : 2.0
Date : 27/02/2015

Contents

Introduction

The Record Linker (RL) is being developed in the Universiteit Leiden (Netherlands) by Kleanthi Georgala, Benjamin van der Burgh and Cláudio Rebelo de Sá, under the guidance and supervision of Dr. Arno Knobbe, as part of the Traces Through Time project. *RL* is a probabilistic text mining tool for entity resolution, implemented in Java. The goal of the linker is to assign a score to each pair of occurrences that expresses the confidence that they refer to the same real-world person. By thresholding these scores, the occurrences can be linked and their information can be aggregated, constructing rich profiles in the process.

The objective of this document is to acquaint a first-time user of *RL* with some of the most important feature of our linker. It includes a brief technical description of our system and a review of the files included in this package and their usage. Additionally, we introduce a section in which we specify the system requirements and the type of input that is suitable for our system. Finally, we describe how to compile and run the application and we provide a small output example and explanation of the linkage results.

Technical Overview

Within the historic documents, there are *occurrences* of names, be it first or last names, and other words that provide information about an individual. These occurrences can range from being very ambiguous (e.g. Mr. Johnson) to more or less uniquely identifying (e.g. King Richard I). In collaboration with the *Natural Language Processing* (NLP) team from *Brighton University* and *The National Archives* (TNA), we have specified a fixed set of attributes. Some of these attributes can be missing within an occurrence, if a particular dataset does not provide information about them. For example, it is very common among the datasets of the Medieval era to use family relations to uniquely identify a person, whereas in datasets from the Early Modern era, people were mostly described using surnames. Further details on the accepted list of attributes and their definition is given in the Files Description and Input Requirements sections.

The task of the record linkage system is to deduce which of these occurrences refer to the same person. Given a set of occurrences and a set of statistics, *RL* computes a *confidence score* based on the probabilities associated with each of the specific values within an occurrence. The probabilities are associated with statistics, that can be either obtained from an already disambiguated dataset of the same time period (e.g. index) or can be computed using the ambiguous references from the text itself.

Each occurrence o_i is defined as a tuple $o_i = (v_i^1, v_i^2, \ldots, v_i^n)$, with v_i^k being the value of the *k-th* attribute of the *i-th* occurrence and n the number of accepted

attributes found within a reference. The basis of the probabilistic record linker is to compute how likely it is that one finds a certain combination of properties, such as forename, surname, role and title, in a population of individuals. In order to do so, *RL* compares the values of the corresponding attributes between two occurrences, maps the value pairs to their associated probabilities and aggregates the partially computed estimations into one final value, $p()$. Then, a *confidence score* is computed as the reciprocal of the estimated number of people (N) with such properties at the time:

$$conf(o_i, o_j) = \frac{1}{N \cdot p(o_i, o_j)} \tag{1}$$

When enough contextual information is supplied within an occurrence and/or the values are rare, the result will be a relatively high confidence score, indicating that we are quite confident that the occurrences refer to the same person. If the information supplied is less in quantity or if the occurrence includes less informative attribute values, the less confident the system will be, resulting in a lower confidence score. Note that values below 1.0 can occur, although they would strictly indicate that less than one person is expected to have the specified properties. In other words, it is best not to interpret the confidence scores in an absolute sense, but rather as a *ranking* of candidate matches: higher confidence indicates a more likely match between occurrences.

In the presence of errors and variations in names, it is apparent that the suggested procedure relies on both the similarity measure applied in order to estimate $p()$ and the quality of the statistics.

Files Description

In this section, we will give an analytic description of the folders and the files included in this package (TTT). *Before running the application, we suggest that you check if all the following files and folders are included within the TTT directory. We would also advise you not to move the contents of each sub-directory in different locations.*

- *record_linkage* folder : *record_linkage* is the application folder. It consists the following sub-folders and files:
  - *batch* includes two files, *compile_jar_TTT.sh* and *compile_jar_TTT.bat*, that are used in order to compile the application, or MacOS/Linux and Windows systems, respectively.
  - *bin* consists of the *MANIFEST.MF* file and the binary files of our application (after compiling).
  - *lib* includes the external libraries used by our linker.

  - *src* consists of the source code files and folders of *RL* and the sub-folder *resources* that includes the *configuration.properties* file.
  - *Run.sh* and *Run.bat* are the Startup Scripts to run the application for MacOS/Linux and Windows systems, respectively. Their contents can be viewed by opening the files with a text editor. Both files include commands in order to run the application (*ttt.jar*).
  - *ttt.jar* is the executable file of our application.

- *data* folder : It includes of one sub-folder (*statepapers*), that is an example input obtained from the State Papers, Domestic dataset of the Early Modern period. *statepapers* includes three folders:
  - *occurrences* : It includes two sample data files in .json format, that can be given as input to the linker. It consists of a set of 1000 occurrences found in the State Papers, Domestic dataset. More details on the preferable input format will be given in section Input Requirements.
  - *statistics* : It includes a set of files with .tsv format. Each file includes statistics for a particular attribute, obtained from the partially deduplicated index of the State Papers, Domestic dataset. The files are named after the attribute they refer to.
  - *output* : a folder for storing the linkage results.

- *documentation* : It includes the documentation file of our application.

Configuration

As described in the previous section, our package includes a *configuration.properties* file, in which the user can specify the parameters that are needed in order to run our application and obtain linkage results. The configuration file can be found in the *record_linkage/src/resources* directory and we strongly advise you *not* to change the name and the location of the file.

More analytically, the configuration file includes the following parametric attribute:

- **root** specifies the absolute path of the data directory, that includes the occurrences file, statistics and output file. Since the data directory path is going to be used often within the configuration file, therefore, for simplicity reasons, we suggest to assign its value to an attribute.
- **occurrences.basedir** specifies the absolute path of the occurrences' file directory.
- **occurrences.format** specifies the format of the occurrences file. **Accepted values : json**.

- **occurrences.file** specifies absolute path of the occurrences file.
- **statistics.basedir** specifies the absolute path of the statistics files directory.
- **statistics.format** specifies the format of the statistics files.
- **linker.type** specifies the type of the linker. **Accepted values : naive**
- **attributes.list** specifies the attributes that our linker takes into consideration in order to compare a pair of occurrences. **Accepted values : forename, surname, article, provenance, role, title, and childOf**.
- **distance.metric** specifies the similarity function used to compare the corresponding attributes of two occurrences, included in the **attributes.list**. Additionally, we give the user the possibility of choosing a different similarity function for each attribute. This can be specified using the *attribute_name***.distance.metric** attribute. **Accepted values : absolute, soundex, levenshtein, dameraulevenshtein, jaro, qgrams**.
- **distance.threshold** specifies the threshold used by the **distance.metric** to determine if two values of an attribute can be considered similar or not. If the user has chosen to specify a similarity metric for a particular attribute, he must indicate the threshold for the preferred metric using the *attribute_name*.distance.metric attribute. Accepted threshold values for each proposed similarity function:
  - **absolute and soundex** consider two values similar if the actual values or their homophones are the same. Therefore, assign 0.
  - **levenshtein** distance assume that two values are similar if the number of single-character edits (insertions, deletions or substitutions) required to transform the first value to the second value is less than a threshold. **dameraulevenshtein** is a modification of the levenshtein, that additionally defines as a possible single-character edit the transposition of two adjacent characters. Therefore the accepted values can be any integer value, with the smallest being 1.
  - **jaro** considers two value as similar if the weighted combination of the matching characters and transpositions is greater than a threshold. The accepted values range from $[0, 1]$. Please use decimal number for this threshold.
  - **qgrams** considers two values to be similar if the normalized sum of the equal tri-gram between the vectors of the two values is greater than a threshold. The accepted values range from $[0, 1]$. Please use decimal number for this threshold.
- *attribute_name***.statistics.file** specifies the absolute path for the statistics of the *attribute_name*. The accepted values of *attribute_name* are described previously.

- **filters.list** specifies the name of the filter(s) used by the application. A filter provides the opportunity to the user to exclude less informative occurrences from the linking process. **Accepted values : basic and/or quality and/or quantity**. More analytically :
  - **Basic Filter** : By choosing this filter, the application excludes occurrences that do not have the *forename* and *surname* attributes filled.
  - **Quantity Filter** : By choosing this filter, the application excludes occurrences, whose total amout of filled attributes is less than the number specified in **quantity.filter.threshold**. The **quantity.filter.threshold** must be between 1 and the size of the **attributes.list**.
  - **Quality Filter** : By choosing this filter, the application excludes occurrences that do not have values in the set of attributes specified in the **quality.filter.attributes**. The **quality.filter.attributes** must be a subset of the **attributes.list**.
- **blocker.type** specifies the name of the blocking technique that our application uses in order to dividing occurrences into blocks. It allows to link occurrences that only belong to the same block. Blocking techniques minimise the computational time needed in order to produce linking results. The main idea is that occurrences of different blocks are highly unlikely to refer to the same entity. **Accepted values : firstletter, nullblocker**. More analytically:
  - **FirstLetter Blocker** : By choosing this blocking method, the occurrences are divided into blocks based on the first letter of their *surname* attribute. There is a special block for occurrences with a missing *surname* value. Each block is processed separately. Once the linking of each block has finished, the applications links the occurrences with no *surname* to the rest of the set.
  - **Null Blocker** : By choosing this blocking method, the occurrences are placed within one block (default).
- **output.basedir** specifies the absolute path of the output file directory.
- **output.file** specifies absolute path of the output file.
- **output.format** specifies the format of the output file. **Accepted values : json**.
- **confidence.threshold** specifies the lowest accepted confidence score between two occurrences that is needed in order for the pair to be present in the **output.file**.
- **results.maxsize** specifies the amount of occurrences that can be stored in the **output.file**.

Requirements

System Requirements

RL requires a working installation of Java Runtime Environment (Java JRE)[1]. We recommend that the Java package installed in your computer should be version 7 or above. Additionally, you will need increasing amounts of memory when larger datasets are used. Basic experimentation of a few thousand occurrences can be performed on a computer with 4 GB of memory.

Input Requirements

As described in section Files Description, we have include an example data that can be used by our linker (*data* folder). Within the *occurrence* folder, we have included a sample input in JSON format, with the following pattern :

{"sentenceId": "2", "surname": "", "fileId": "Edward VI - Volume 1", "title": "", "gender": "", "childOf": "", "provenance": "", "adjectivetitle": "", "role": "Earl of Hertford", "adjectivesurname": "", "forename": "", "article": "", "origOccurrence": "Earl of Hertford", "grandchildOf": "", "appearanceDate": "1547", "id": "1", "sectionId": "Edward VI - Volume 1##January 1547##Jan. 29.Hertford.##1", "adjectiveforename": ""}

Each occurrence is transformed into a Python dictionary, whose keys are a set of predefined attributes and its values are the actual values obtained from the occurrence. The attributes can have nominal (string) values, placed within quotes (""). Each occurrence is placed within brackets ({}) and it is separated from other occurrences in the file using a comma (,). The set of occurrences is placed within ([]).

When creating your own input data for *RL*, please make sure that you have converted the parsed occurrences following the above pattern, using only the following attributes :

- **origOccurrence** : the exact occurrence found in text before processing.
- **title** : everything that can be found before a forename or a surname such as *sir/lord/king/etc.*
- **forename** : the first name of the occurrence.

[1]The Java Runtime Environment can downloaded for free from www.java.com/download. It is developed and maintained by Oracle Corporation, which is in no way associated with the Traces Through Time project.

- **article** : a particle, placed between the first and the last name of an occurrence, such as *Jon "the/de/la/le/etc" Black*.
- **surname** : the last name name of the occurrence.
- **role** : *Bishop of London/Archdeacon of Oxford/etc*, everything that can be found after a forename or a surname.
- **provenance** : such as *of "London/Oxford/etc"*. We do not store "of".
- **childOf** : it indicates a father/mother relation with the current occurrence. It references to the occurrence that is the parent as this occurrence by including its **id** attribute.
- **gender** : male, female or unknown.
- **appearanceDate** : it refers to the year of the section that the occurrence has been mentioned in. Accepted format : "1978-02-31" or "1978-02" or "1978".
- **id** : unique id for each occurrence.
- **sentenceId** : it refers to the unique id of the sentence the occurrences was mentioned in.
- **sectioneId** : it refers to the unique id of the section the occurrences was mentioned in.
- **fileId** : it refers to the unique id of the file the occurrences was mentioned in.

If an attribute value is missing from an occurrence, the user can either exclude completely the corresponding key or insert an empty string ("").

Finally, the attributes

- **grandchildOf** : same as the textbfchildOf attribute, for grandfather/mother.
- **adjectiveforename** : such as *Paul's* obtained from the occurrence *Paul's servant*, where servant is a title and Paul's is an adjective forename.
- **adjectivesurname** : such as *Smith's* in *Smith's sailor*, same as above.
- **adjectivetitle**: such as *Queen's guard*, same as above.

can be absent from the input data since they are currently excluded from the linking process and from the output file.

Compile

If the user would like to compile our source code files, we have created a script within the `record_linkage/batch` folder (*compile_jar_TTT.sh*) in order to create a new `ttt.jar` file. The content of the *compile_jar_TTT.sh* can be viewed by using a text editor, but we strongly advise you to not change any of the commands, apart from the *PROJECT* variable that indicates the location of the application folder. The *compile_jar_TTT.sh* can be executed using a Terminal window from MacOS or Linux operating systems. In your terminal window, type `cd TTT/record_linkage/batch` and press `Enter`. Then, type `./compile_jar_TTT.sh` and press `Enter`. The new executable file `ttt.jar` will be placed within the `record_linkage` folder. Windows users are also able to compile the source code by following the above sequence of commands, replacing the *compile_jar_TTT.sh* with the *compile_jar_TTT.bat* file.

Run

Our application has been already compiled for you, therefore we have created two Startup Scripts (*Run.sh* and *Run.bat*) in order to run our linker. Using a text editor, you can observe the content of the two scripts. Both scripts include the same two arguments : *–conf* and a path, that describes the location of the *configuration.properties* file. Please make sure that the path argument includes a valid location of the *configuration.properties* file, since this file is a very important component of the running process.

Initially, please make sure you have downloaded and installed the Java version recommended in the System Requirements section. Firstly, place the *TTT* folder within a preferred directory, such as *Documents*.

- For Windows users : Make sure that Windows can find the Java interpreter. In order to do that, open a Terminal windows by selecting `Start`, the type *cmd* and press `Enter`. Then, type `java -version`. You should see something similar to the information printed below:

  > java version "1.6.0_27"
  > Java(TM) 2 Runtime Environment, Standard Edition (build 1.6.0\27-b07)
  > Java HotSpot(TM) Client VM (build 1.6.0\27-b13, mixed mode, sharing)

  Once you have identified the java version that it is installed in you system, the next step is to find the installation directory within your system. By navigating through your system files, go to `C:\Program Files\Java` where you will find the folder `jdk1.6.0_27`, which must include the

same numbers as the first line of the `java -version` output. Then, select `Start` → `Computer` → `System Properties` → `Advanced system settings` → `Environment Variables` → `System variables` → `PATH` and prepend `C:\Program Files\Java\jdk1.6.0_27\bin;` to the beginning of the PATH variable.

Finally, at the same `cmd` prompt, type `cd Documents/TTT/record_linkage/`, press `Enter`, type `Run.bat`, press `Enter` and enjoy.

- For MacOS/Linux users : Make sure you have installed the recommended Java version, mentioned in the System Requirements section. Then, open a Terminal window, type `cd Documents/TTT/record_linkage/`, type `./Run.sh`, press `Enter` and enjoy.

Results

After running our application, the linkage results are stored in the folder *output.basedir*, under the name *output.file*, that are both specified in the configuration file. For each occurrence that was matched with at least one occurrence with a confidence score equal or greater than the *confidence.threshold*, our application stores the resulting matching pairs and their scores, using the schema specified by TNA. You can observe a small example of the output in the *sampe_output.json*. This example indicates that the occurrence *Andrew Duddeley* was matched with an occurrences such with id *23*, with a confidence score of 97.56171. Also, the output provides information about the the occurrence (*HasForeName* and *HasFamilyName*), that was found in text but it also includes metadata such as *UploadDate* and sentence, section and file identifier.

Comments

- In case the *Run.sh* is unable to run in MacOS/Linux, please check the scripts permission rights and make sure that the `Owner` group has executable rights for these scripts. Otherwise, please type `chmod u+rwx Run.sh`.
- In case *Run.sh* and/or *Run.bat* are unable to allocate the `java` command in your system, please do the following:
  - For MacOS/Linux : instead of typing *./Run.sh*, type `which java` and press `Enter`. Copy and paste the result of the previous command, press `Space`, type `-jar ttt.jar -conf` *`configuration.properties_location`*, and press `Enter`. Alternately, just type `/usr/bin/java -jar ttt.jar -conf` *`configuration.properties_location`*.

– For Windows : If you have the latest version of Java installed, in the Terminal windows type `%PATH%` and press `Enter`. Check if the output of the command includes `C:\ProgramData\Oracle\Java\javapath`. In case it is not included, add to the `PATH` variable by selecting `Start` → `Computer` → `System Properties` → `Advanced system settings` → `Environment Variables` → `System variables` → `PATH`. Then, at the same Terminal window, type `Run.bat`, press `Enter` and enjoy.

In case the previous attempts fail to run the application, please check that you have Java installed in your system. In case you do not, download and install the Java version suggested in the System Requirements section.

- In case of an "IllegalArgumentException" error, please follow the instructions displayed in the message.
- If the user would like to create/use his own set of occurrences and corresponding statistics files, we suggest to create a new folder within the *data* directory, naming it after the original dataset, from which he extracted the occurrences. Additionally, please make sure that you have specified the right directory paths of your input in the configuration file.
- Please make sure that for each of the attributes within the *attributes.list*, you have included an appropriate value to the *attribute_name.statistics.file* field of the configuration.file.
- The *gender* attribute in the output representation of an occurrence has the value "unknown" by default.

Contact

For further questions and comments, please contact Kleanthi Georgala using kleanthie.georgala@gmail.com or k.georgala@liacs.leidenuniv.nl.

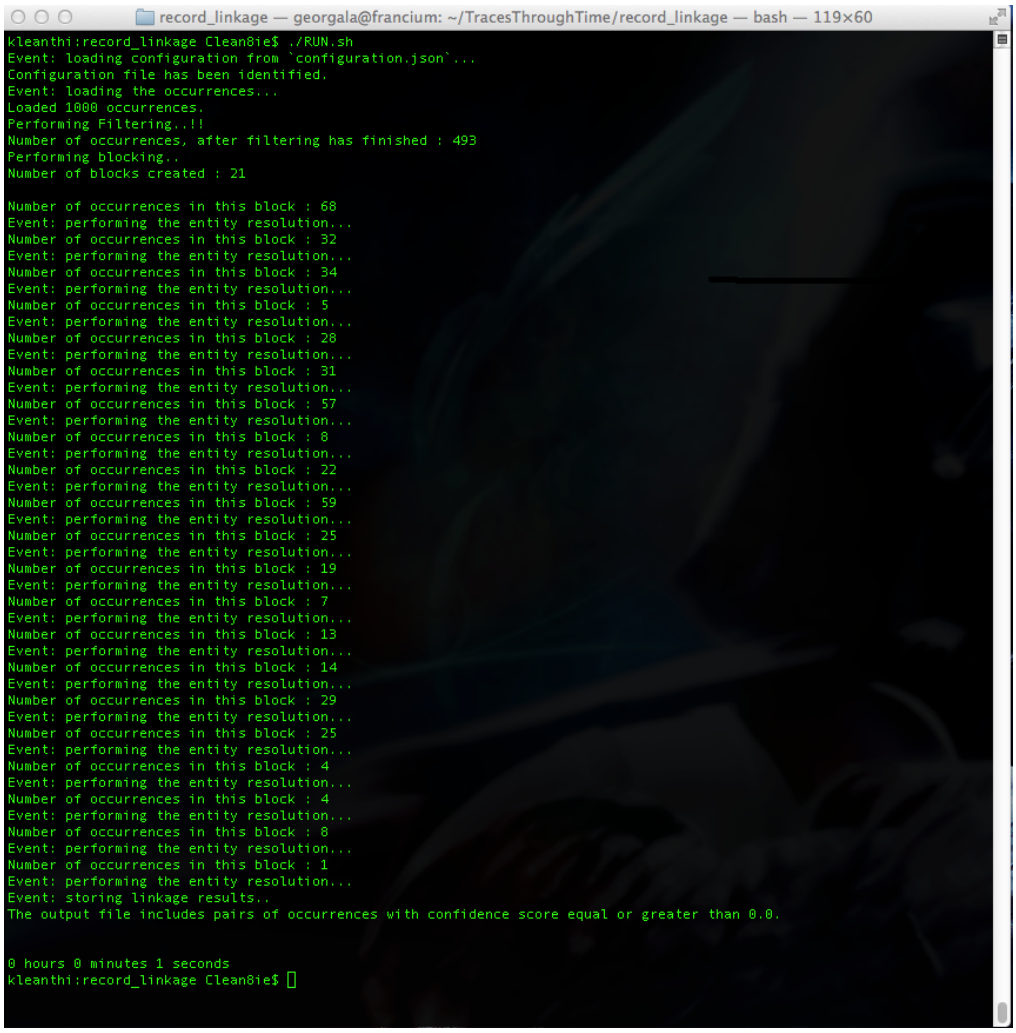

Figure 1: Application Screen