Social networks and immigrant ships. A Machine learning approach for data cleaning and record linkage: Predetermined Rules and Hidden Markov Models

Diego Battiston*

2016

Abstract

This document explains the methods and procedures used to clean and standardise the data from passenger list used in the paper "Networks at work: Immigrant ships and labour markets during the age of mass migration". It also discusses the methods used for linking records across datasets. The linking methodology was designed to match records across large datasets where due to the number of observations, pair comparisons become unfeasible even after applying standard blocking strategies. (forthcoming).

JEL *classification*: D02, D23, L29.

1 Introduction

Steps in data standarization.

*Department of Economics and CEP, London School of Economics, Houghton Street London WC2A 2AE, United Kingdom. Email: d.e.battiston@lse.ac.uk, Tel.:

2 Hidden Markov Chains

When input data is not in a clean format, standardization process is carried on using Hidden Markov Models (HMM). These type of supervised machine learning methods usually perform better when the data field containing the person identification (given names, surnames, etc.) is embedded in a long text string, potentially mixed with noisy information or unnecessary details. Assume for instance that we want to extract information of given name and surname from the following string chain STR1 = 'mr nikola tesla engineer austria waesland 28y'. Ideally we would like to extract the name (nikola) and surname(tesla) for linking purposes. In order to introduce the HMM approach it is necessary to define two key concepts:

- Observation status: This is a classification tag assigned by a series of rules and human supervised inputs. Formally, a string S is an ordered chain of words $(s_1, s_2, .., s_P)$. An observation status is defined as $x_i \in X$ and a rule is a relationship $f : \times_j s_j \longrightarrow X$ for $s_j \in S' \subseteq S$ In our example, the observation status will assign a tag to each word in the string. So for example, suppose we define the set of possible status as {*suffix, male_name, female_name, surname, city, country, occupation, age, shipname*}. Each element correspond to a possible classification, so for example "nikola" can be easily assigned to the category *male_name*. We can use a set of rules to determine that STR1 can be segmented as [*(suffix), (male_name), (surname), (occupation), (country), (shipname), (age)*]. In this case we assign "engineer" to the category *occupation*, "mr" to the category *suffix*, etc. But alternatively STR1 could be segmented as [*(suffix), (male_name), (surname), (occupation), (shipname), (shipname), (age))*]. In this case we assign "austria" to the category *shipname* (as there are ships with this name in the dataset). Usually, each string can be classified into many different ways.

- State: State is a hidden characteristic of word or group of words. Formally, a state is defined as $\omega_j \in \Omega$ The ultimate goal of the HMM is to allocate the most likely chain of states that could have originated the string. We need to assume that the set of hidden states is predetermined and that observation status are generated by hidden states with certain probability distribution. In other words, $\sum_j H(x_i|\omega_j) = 1$

$\forall x_i \in X$, where $H(x_i|\omega_j)$ is the probability that observation status x_i is generated by hidden state ω_j. For instance, we can assume that the hidden states that generate our strings are given by {*TITLE, GIVEN NAME, MIDDLE NAME, SURNAME, OCCUPATION, OTHER INFO, AGE*}.

Note that obtaining the required information or standardizing it is a straightforward task for a human user. However, when dealing with millions of records, we need to emulate the human learning behaviour process.Assuming that there exists m observation status and n hidden states, it is useful to define the following matrices:

Transition matrix:

$$\Gamma = \begin{pmatrix} P(\omega_1|\omega_1) & P(\omega_2|\omega_1) & ... & P(\omega_n|\omega_1) \\ P(\omega_1|\omega_2) & P(\omega_2|\omega_2) & ... & P(\omega_n|\omega_2) \\ \vdots & \vdots & \vdots & \vdots \\ P(\omega_1|\omega_n) & P(\omega_2|\omega_n) & ... & P(\omega_n|\omega_n) \end{pmatrix}$$

An element $P(\omega_j|\omega_k)$ is the transition probability from state $omega_k$ to state $omega_j$. Note that the order of states correspond with the order of observation status in the string, so for example $P(SURNAME|GIVENNAME)$ is the probability that the position of the surname in the string is right after the given name. We assume that these probabilities are independent from previous states (i.e. probability of surname being written right after the given name does not depend on what was written before the given name). This assumption can be relaxed but given the structure of the data, this assumption is convenient to keep the algorithms tractable. It is also convenient to define "virtual states" $\omega_1 = START$ and $\omega_n = END$. The $START$ state is defined such that $P(\omega_1|\omega_k) = 0$ for $k = 1, ..., n$ and the END state is defined such that $P(\omega_k|\omega_n) = 0$ for $k = 1, ..., n$.

Emission matrix:

$$\Lambda = \begin{pmatrix} H(x_1|\omega_1) & H(x_1|\omega_2) & ... & H(x_1|\omega_n) \\ H(x_2|\omega_1) & H(x_2|\omega_2) & ... & H(x_2|\omega_n) \\ \vdots & \vdots & \vdots & \vdots \\ H(x_m|\omega_1) & H(x_m|\omega_2) & ... & H(x_m|\omega_n) \end{pmatrix}$$

We assume that the virtual states do not generate any type of strings, therefore $H(x_i|\omega_1) = 0$ and $H(x_i|\omega_n) = 0$ for $i = 1, .., m$

A *tag sequence* is a representation of a string in terms of observation status components, in other words, $Y = (y_1, y_2, ..., y_p)$ with $p \leq P$ and $y_r \in X$ for $r = 1, ..., p$. A *chain* $C_t(Y) = [c_1(y_1), c_2(y_2), ..., c_p(y_p)]$ with $c_r(y_r) \in \Omega$ is an ordered vector of hidden states that potentially generated the *tag sequence* representation of the string.
We can use the two matrices defined above to define the unconditional probability [1] of a given *chain* as:

$$Prob(C_t(Y)|\Lambda, \Gamma) = \prod_{r=2,..,p} P(c_r(y_r)|c_{r-1}(y_{r-1}))H(y_r|c_r(y_r))$$

Assume that a string S has Q possible different tag sequence representations Y_q (where the tag sequences are defined as possible by the assumed rules). The HMM tries to solve the problem of finding the chain of hidden states that conditional on a tag sequence is the most likely for a given string [2]:

$$Max_{(Y_q, C_t(Y_q))} Prob(C_t(Y)|\Lambda, \Gamma)$$

In the simplest case where a string has only one possible representation, the HMM problem is simplified to find the *chain* that has the highest unconditional probability of having generated the string S

Example: explain here one example with the string we started with

Unfortunately the maximization problem is computationally hard to solve as given a tag representation of length p, there are n^p potential chains to contrast. If we are dealing with millions of records this operation must be repeated for each individual record. Nevertheless, the maximization problem can be solved by using the *Viterbi algorithm* which is explained below:

[1] Note that this probability is unconditional in the sense of accounting for both transition and the tag representation probability, however it is conditional on the observed string realization

[2] This is usually called the "decoding" problem of HMM

Viterbi Algorithm:

The algorithm approach is similar to a dynamic programming problem. For a given tag sequence Y_q, we start by defining $v_t(j)$ as the transition probability between the most likely chain of hidden states that could have generated the tag sequence $(y_1, y_2, ..., y_{t-1})$ and the hidden state ω_j. In other words, we assume that we already know the best chain of hidden states that could have generated the tag sequence up to the $t-1$ element. Formally, it is defined as[3]:

$$v_t(j) = Max_{c_1,...,c_{t-1}} Prob(c_1, ..., c_{t-1}, c_t = \omega_j | \Lambda, \Gamma)$$

The algorithm computes these probabilities recursively as:

$$v_t(j) = Max_{z=1}^{n} v_{t-1}(z) P(\omega_z|\omega_1) H(y_z|\omega_j) \text{ , for } t = 1, ..., p$$

The algorithm thus end up with the calculation of v_p which implicitly track the most likely hidden state chain $C_t^*(Y_q)$. In order to deal with multiple tag sequence representations, we only need to compare the v_p originated from different sequences Y_q and pick the one with highest value. The algorithmic approach reduce considerably the number of calculations performed to solve the decoding problem. A pseudo-code implementation of the algorithm can be found in Jurafsky and Martin (2014).

3 Matching across large datasets

INCOMPLETE - The methodology discussed in this section extends the standard record linking methodology (Fellegi-Sunter)to cases where data size is large enough to result in an infeasible number of pair comparisons.

[3]We simplify the notation using c_r and $c_r(y_r)$ as equivalent