Record Linkage with Nonparametric EM Algorithm

Yibin Xiong, advised by Prof. Jacob Bien

December 16, 2022

1 Introduction

In this project, our main task is to determine the identities of authors with similar names. We have obtained two datasets, arXiv and Semantic Scholars (S2), that provide abundant citation information for authors in the statistics community. When we cross-reference these two datasets for attendees at the Joint Statistical Meetings (JSM) 2022, the name entries in arXiv/S2 and their counterpart listed on the JSM program are often NOT identical. In this case, we need to figure out whether two records with simialr names correspond to the same person indeed. This matching process is known as *record linkage*. In addition, to make matching simpler and enhance data quality, we also need to eliminate the duplicated records *within* arXiv/S2 dataset. The particular case of record linkage when the two datasets to merge are the same is known as *deduplication*.

2 Method

2.1 Fellegi and Sunter Framework

Fellegi and Sunter [3] formalized a model for record linkage. Given any two datasets with common features (e.g. name, sex, address, etc.), we decide whether a pair of records coming from different datasets represent the same identity (*link* or *matched*), do NOT represent the same identity (*non-link* or *unmatched*), or undecided (*possible link*). An optimal decision rule minimizes the probability of indecision while controlling the probabilities of making false links and non-links. I summarize the optimal linkage rule proposed in [3] as follows.

Algorithm 1 Fellegi and Sunter

▷ Given a pair of records, produce the comparison vector $\gamma = (\gamma_1, \ldots, \gamma_K)$ on based on k features common to both datasets.
▷ Estimate $m(\gamma) := \mathbb{P}(\gamma|M)$ and $u(\gamma) := \mathbb{P}(\gamma|U)$ (known as m and u probabilities).
▷ Consider the weight $W(\gamma) := \log(\frac{m(\gamma)}{u(\gamma)}) = \log m(\gamma) - \log u(\gamma)$.
▷ Given error levels μ, λ, find the thresholds T_μ, T_λ and the decision rule is
- Matched if $W(\gamma) > T_\mu$
- Unmatched if $W(\gamma) < T_\lambda$
- Undecided if $T_\lambda \leq W(\gamma) \leq T_\mu$

The entries of the comparison vector γ are either binary if we directly compare two values or continuous if we use some similarity metric (e.g. Jaro-Winkler for strings). An assumption the authors made is that the features are independent conditionally on the ground-truth matching status [3]. I refer readers to [2] as a typical example for practical details about γ. Regarding the second step, if we have labels of the ground-truth matching status, we can fit supervised learning models for $\hat{m}(\gamma) = f_\theta(\gamma), \hat{u}(\gamma) = g_\theta(\gamma)$. However, in our situation the labels are unknown, which prompts us to use unsupervised learning models.

2.2 EM Algorithm

In unsupervised setting, essentially we want to separate pairs of records into two clusters, one of which corresponds to matches and the other corresponds to non-matches. Let a binary random variable M denote the ground-truth matching status, which is unobserved. Our model is that

$$M \sim \text{Bernoulli}(p)$$
$$\gamma_k \mid M = m \sim f(\gamma_k; m) \text{ for } m = 0, 1$$

where $f(\gamma_k; m)$ is the density of the conditional distribution $\gamma_k \mid M = m$. We often specify a parametric model, such as Gaussian or categorical distribution, for this conditional density. However, since our data seems not to follow those common distributions (See Figure 1), I used kernel density estimation to fit $f(\gamma_k; m)$ *non-parametrically*

$$f(\gamma_k; m) = \frac{1}{n\, h_{m,k}} \sum_{i=1}^{n} K\left(\frac{\gamma_k - \gamma_{i,k}}{h_{m,k}}\right)$$

where n is the sample size, $\gamma_{i,k}$ is the k-th feature of the i-th pair of records, $K(\cdot)$ is the kernel function, and $h_{m,k}$ is the bandwidth parameter for the m-th cluster and k-th feature. Typically, we use the Gaussian kernel $K(x) = \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{x^2}{2}\right\}$. The log-likelihood of the data is

$$\log p(\Gamma) = \sum_{i=1}^{n} \log\left(\sum_{m=0}^{1} \prod_{k=1}^{K} p^m (1-p)^{1-m} \cdot f(\gamma_{i,k}; m)\right)$$

and maximizing it involves the expectation-maximization (EM) algorithm. In practice, I employed the R package `mixtools` [1] to run this non-parametric EM algorithm.

It worth pointing out that our method is different from the Fellegi and Sunter framework. Rather than having three kinds of decisions (matched, unmatched, undecided), we only keep the matched and unmatched cases since we don't want more manual inspection. Our decision rule is not based on the m, u probabilities but the posterior probabilities $\mathbb{P}(M|\gamma)$, which is estimated in the E-step of EM. A pair is predicted to be a match if $\mathbb{P}(M = 1|\gamma) \geq 0.5$. Our method is simpler despite the loss of guarantees on errors for matching and non-matching decisions.

3 Implementation

Our raw data are adjacency matrices that indicate which author writes which paper and which paper cites which paper. Simple matrix multiplications give us a collaboration matrix and a citation matrix of dimension $N_{author} \times N_{author}$. I will describe the procedures for matching S2 authors to JSM authors and the other tasks are similar. In the first round of matching, we identify all author names in S2 that are *compatiable* with their counterparts in JSM program, which means the last names must be identical and the first names must also be identical unless one of them is the initial or nickname of the other (e.g. "R." and "Robert", "Abby" and "Abigail"). There are some JSM authors mapped to *multiple* S2 authors, which defines the pairs we need to scrutinize.

Then we build the features on which we want to compare the pair of authors. Unfortunately, since our datasets do not contain much information about authors except their names and publications. I can only produce two features: name similarity and word similarity. To compute the former, we generate a set of all possible name variants (e.g. F. Middle Last) for each name and count how many variants are in common. `Inf` is used for exact matches. The latter is defined as an inner product of author-word vectors for the two authors.

With a dataframe of the similarity scores, I first eliminate entries with extreme values (`name_sim == Inf` or `word_sim > 40`) by directly marking these pairs as matches. Then I input the matrix to `mixtools::npEM`, train the model with 500 iterations, and obtain the posterior probabilities to make the decisions.

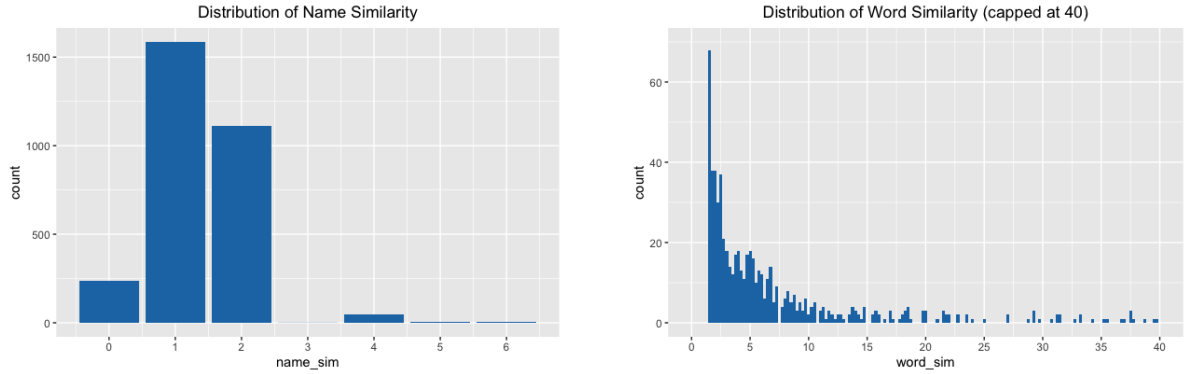

Figure 1: Distributions of Similarity Scores

4 Results

	idx_jsm	idx_s2	jsm_name	s2_name	name_sim	word_sim	match_prob	match
1	1	137200	(Tony) Jianguo J Sun	Jianguo Sun	2.00	5.27	1.00	TRUE
2	1	245695	(Tony) Jianguo J Sun	Jianguo Sun	2.00	0.00	0.03	FALSE
3	16	14262	Abby Sloan	A P Sloan	1.00	0.00	0.00	FALSE
4	16	15908	Abby Sloan	Abigail Grace Sloan	0.00	8.10	1.00	TRUE
5	29	51214	Abhishek Kumar Dubey	Abhishek Kumar Dubey	Inf	0.70	1.00	TRUE
6	29	191664	Abhishek Kumar Dubey	Abhishek Dubey	2.00	0.36	0.15	FALSE
7	44	237528	Adam J Sullivan	Adam John Sullivan	4.00	0.00	0.12	FALSE
8	44	248374	Adam J Sullivan	Adam J. Sullivan	4.00	0.00	0.12	FALSE
9	46	261914	Adam Kaplan	Adam Kaplan	Inf	0.00	1.00	TRUE
10	46	317035	Adam Kaplan	Adam Ryan Kaplan	2.00	6.07	0.99	TRUE
11	47	98381	Adam Lee	A. T. Lee	1.00	0.00	0.00	FALSE
12	47	194208	Adam Lee	A H P Lee	1.00	0.00	0.00	FALSE
13	47	273086	Adam Lee	A. Y. Lee	1.00	0.00	0.00	FALSE
14	49	216446	Adam Michael Edwards	Adam Edwards	2.00	0.00	0.03	FALSE
15	49	277760	Adam Michael Edwards	Adam Michael Edwards	Inf	0.00	1.00	TRUE
16	84	84618	Alan Chiang	Alan Y. Chiang	2.00	1.71	0.76	TRUE
17	84	155218	Alan Chiang	Alan Chiang	Inf	1.03	1.00	TRUE
18	84	195460	Alan Chiang	Alan King Ip Chiang	2.00	0.00	0.03	FALSE
19	87	56029	Alan Hubbard	Alan E. Hubbard	2.00	0.00	0.03	FALSE
20	87	134762	Alan Hubbard	Alan H. Hubbard	2.00	0.00	0.03	FALSE
21	87	223807	Alan Hubbard	Alan Edward Hubbard	2.00	0.00	0.03	FALSE
22	108	9703	Alex Franks	Alexander Franks	0.00	0.00	0.03	FALSE
23	108	20832	Alex Franks	Alexander M. Franks	0.00	1.10	0.66	TRUE
24	120	35517	Alexander Bauer	Alexander Bauer	Inf	0.00	1.00	TRUE
25	120	45377	Alexander Bauer	Alexander Bauer	Inf	4.85	1.00	TRUE

Table 1: A Sample Result

Results Here I present the first 25 pairs to compare as a sample result. By inspection, the algorithm correctly predicts the matching status for the majority of the pairs. In particular, checking the predictions for the pairs that are hard to distinguish, i.e. those with lower similarities, I discovered that the algorithm relies on word_sim much more than name_sim. For instance, Pair 23 (Alex Franks V.S. Alexander M. Franks) has name_sim = 0 and word_sim = 1.10. This is because the author prefers to use a nickname (Alex) on daily basis but a formal name (Alexander) for publications. The algorithm correctly predicts it as a match. Similar examples are Pair 4 and 6. On the other hand, Pair 7 and 8 both have name_sim = 4 and word_sim = 0 and they are predicted as unmatched though the name pairs are very similar. The contrast with Pair 23 shows that word_sim is more important in the model. This makes sense because the training data contains

only the pairs with *compatible* names, which are already very similar. However, the prediction for Pair 7 and 8 are wrong because we merely have 2 features. For the model, it is difficult to tell whether $(4, 0)$ is more likely than $(0, 1.10)$ to be a match without seeing and reasoning with the strings of names as humans did.

Comparison Previously, we resolved multiple matches by computing word_sim and handpicked a threshold of 0.095 so that any pair with word_sim above this threshold were matched. If some JSM authors still corresponded to multiple S2 authors, we averages the citation and collaboration records and match JSM author to the "average" S2 author. The difficulty is that selecting the threshold requires many inspections and the heuristic search can be unreliable. With EM algorithm, we no long need to choose this threshold and the predictions are theoretically grounded by maximum likelihood.

Future work One direction of future work is to find and produce more features. It would be very helpful if we had affiliation information for most authors in S2 and arXiv datasets because JSM program contains affiliations and it is very discrimative. However, the affiliation data in S2 is so sparse that most authors have an empty entry. In addition, affiliation is often in the PDF of an arXiv preprint but not explicitly listed on the webpage. Downloading and parsing information from the PDF files are much more arduous and time-consuming. Another thing I could elaborate on is how to produce the word vectors for all authors, which we used to calculate word_sim. Now we build them via TF-IDF, but S2 has a dataset called `embedding`, which contains embeddings for all paper abstracts produced by a deep learning model. An average of abstract embeddings over an author's all papers could be a better word vector.

References

[1] T. Benaglia, D. Chauveau, D. R. Hunter, and D. S. Young. mixtools: an r package for analyzing mixture models. *Journal of statistical software*, 32:1–29, 2010.

[2] T. Enamorado, B. Fifield, and K. Imai. Using a probabilistic model to assist merging of large-scale administrative records. *American Political Science Review*, 113(2):353–371, 2019.

[3] I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.