Multiple Match

Brian Kundinger

April 3, 2024

Abstract

Abstract

1 INTRODUCTION

2 MULTIPLE MATCH

We can allow each record in B to match to multiple records in A through a Dirichlet process prior. Define a vector of probabilities $\boldsymbol{\pi} = (\pi_0, \ldots)$ where π_k is the probability that some record in B has exactly k matches in A. We can model each π_k as a product of conditional probabilities: let τ_k be the probability that some record in B has at least k matches, given that it has at least $k-1$ matches. This gives us the stick breaking representation

$$\pi_k = (1 - \tau_{k+1}) \prod_{c=1}^{k} \tau_c, \tag{1}$$

where τ_k are independent random variables from a $\text{Beta}(1, \alpha_\tau)$ distribution.

2.1 MCMC

We provide updated notation to allow us to describe one record in B having multiple matches in A. Let Z_j be a vector containing the indices for all of the records in A that are a match with record B_j. Let $|Z_j| = \sum_{k=1}^{\infty} Z_j^k > 0$ denote the number of links associated with record B_j. Through MCMC, we are able to sequentially sample the components Z_j^k of the matching vector Z_j. In this context, we propose the sequence of priors

$$p(Z_j^k|\tau_k) = \begin{cases} \frac{\tau_k}{n_j^k}, & z_j^k \in N_j^k, \\ 1 - \tau_k, & z_j = 0; \end{cases} \tag{2}$$

$$\tau^k \sim \text{Beta}(\alpha_{\tau^k}, \beta_{\tau^k}) \tag{3}$$

where N_j^k is the set of records in A that are available to be matched with B_j, and $n_j^k = |N_j^k| = n_1 - (k-1)$ is the number of such records.

This sequence of priors leads to the sequence of posteriors

$$p(Z_j^k|\tau, m, u, \gamma) = \begin{cases} w_{ij}, & z_j^k \in N_j^k, \\ n_j^k \frac{1-\tau^k}{\tau^k}, & z_j = 0; \end{cases} \tag{4}$$

$$\tau^k \sim \text{Beta}(\alpha_{\tau^k} + n_k(\boldsymbol{Z}), \beta_{\tau^k} + n_{k-1}(\boldsymbol{Z}) - n_k(\boldsymbol{Z})) \tag{5}$$

where $n_k(\boldsymbol{Z}) = \sum_{j=1}^{n_B} I(|Z_j| \geq k)$ is the number of records in B that have at least k matches in A. Note that $n_0(\boldsymbol{Z}) = n_B$, and that for each k, we can view $n_{k-1}\boldsymbol{Z}$ as a number of trials, and $n_k\boldsymbol{Z}$ as a number of successes.

This specification induces an extension of `fabl` with an iterative matching phase. In each iteration of the Gibbs sampler, we sample an initial set of links using τ_1. For each record in B that was found to have a link, we remove the linked record in A from consideration, and then sample another potential link with τ_2. We continue, using τ_k in the k^{th} matching step, until no new links are found, at which we point the matching phase terminates. The $\boldsymbol{\tau}, \boldsymbol{m}$, and $\boldsymbol{u}$ parameters are estimated based on all of the links identified. Crucially, there is no need to specify a maximum number of links per record, as this estimated through the model.

2.2 Variational Inference

There are several reasons why this prior would not be practical through variational inference.

- Through MCMC, we can sequentially sample all plausible matches for record B_j. In variational inference however, we need to estimate the match probability for the joint distribution of ever possible combination of matches.
- Without hashing, the number of possible combinations is $\binom{n_1}{k}$. With hashing, this number is $P^k/k!$ (I think), and would still require some operations at the scale of $\binom{n_1}{k}$ when conducting the actual hashing.
- However, this approach might work if using aggressive indexing/filtering.

Despite this drawback, I think multiple match might still be useful. In cases where we know at least one file is duplicate free, we can safely use `vabl`. However, if we want to be cautious about the potential for duplicates, the multiple match prior under `fabl` is a good option.

2.3 Comparison to Base Fellegi-Sunter

- At the point at which we allow for duplication within and across datasets, one might ask why we don't just use the independent record pair assumption of base Fellegi

Sunter

- We have found that standard FS (and its implementation in fastlink) tends to over-match. It generally attains comparable precision to BRL/fabl once you use the Jaro post-processing algorithm to obtain a one-to-one matching
- In this setting, we do not want a one-to-one matching, so we need to take the raw output of FS
- Conceptually, one could modify the Jaro algorithm to allow for one-to-K matchings. However, this would take some work, and it would have the downside of having to prespecify K.

2.4 Another thought about Fellegi Sunter (not directly related)

It seems to me that base Fellegi Sunter is bound to have a higher false positive rate as the size of the problem grows.

- Assuming there are at maximum K records in A that match with record $j \in B$. For one to one matching, this means $K = 1$.
- There are at least $n_A - K$ nonmatching record pairs for each $j \in B$
- If we're making $n_A n_B$ independent classification decisions it just seems intuitive that we would get more false positives as $(n_A - K)n_B$ grows.
- It also seems intuitive that we would tend to see less false positives when we can only make at most n_B classification decisions.

A possible approach: In FS a record pair gets classified as a match if it has an appropriately high w_{ij}. But in fabl, you need to have the high w_{ij}, and no other $w_{i'j}$ can be greater. This might be tangible place to start?

Does that make sense? Does it seem possible to prove rigorously? Have you thought about this before?

If we could do this, it would give a nice theoretical justification for using the fabl framework over base FS (or practically, using vabl over fastLink).

2.5 Computational Considerations

We can set the maximum number of linkages per record, K, or let it be estimated by the model. Setting it ahead of time gives a minor (I would say, negligible) computational advantage over leaving it unrestricted.

Let $L_k^{(s)}$ be the number of records in B with at least k links in A during iteration s of the Gibbs sampler. Note that $L_0 = n_B$ and $L_1 = n_{12}(Z^{(s)})$ when conducting single matching. The computational complexity of multiple match using MCMC is $\mathcal{O}\left(P\sum_{k=0}^{K} L_k\right)$.

When X_1 is free of duplicates, the multiple match algorithm makes a second attempt at matching, which has complexity (PL_1). This is a minor addition of computation time, but gives the added security of identifying duplicates. It may be a nice option just to use when you're not sure!

3 SIMULATIONS

- Show accuracy under various settings of maximum cluster size. Recall will fall very quickly for fabl. Multilink may be very dependent on correctly specifying K.

4 CASE STUDIES

- Provide an example of record pairs that fabl cannot identify as matches because of duplication. I know there are cases in NCVR.

5 APPENDIX