Probabilistic Matching at the Michigan Education Data Center

Introduction

Individual student records within educational datasets maintained by the Michigan Education Data Center (MEDC) are labelled with a unique identifier, generated by the State of Michigan, which allows researchers looking to perform analysis across multiple datasets to be fairly certain which records correspond to the same individual. However, this limits the scope of any analysis to these internal datasets and the variables they contain. In an instance where a researcher has access to an external dataset, they would be unable to investigate any relations between that dataset and MEDC's data.

In addition to educational research datasets, MEDC also maintains a dataset containing the personally identifiable information (PII) of a large proportion of Michigan's K-12 student, post-secondary student and K12 staff populations, including full names, dates of birth, racial/ethnic status, and addresses, each of which is associated with the state's unique identifier. With these datasets, MEDC has developed a probabilistic matching model that allows it to match MEDC data with external data in cases where the external dataset contains at least some personally identifiable information in common.

The objective of this document is to provide a mid- to high-level overview of the process by which MEDC performs a probabilistic match between any incoming dataset and the PII datasets maintained in house. We provide a broad overview of some of the major concepts relevant to record linkage, including data cleaning, blocking, performing field and record level comparisons, and evaluation metrics and techniques. For more information, please find a list of references that go into some of these topics at far greater detail at the end of this document.

"Truth” Data

One of the main difficulties with regard to record linkage is obtaining a reliable and accurate “truth” dataset. Without having some idea of which records ought to be linked to which, it is impossible to determine just how well any particular method is performing or to identify areas in which a particular approach is struggling. To that end, projects which are not already provided with a labeled subset of data will often opt for hand curated truth sets, in which a selection of likely record pairs is annotated by hand, indicating which pairs of records humans are likely to consider matching.[1] This is a time intensive approach to the problem, which is somewhat alleviated these days with projects like Amazon’s Mechanical Turk, but is one method of arriving at a reliable “truth” on which to base a project.

Our data, however, is of a sensitive nature which prevents us from looking outside of MEDC for any large scale labelling, and so we were required to look at other methods for arriving at a “truth”. We eventually settled on creating two datasets with which to measure our approach:

- A small subset of data (2,000 pairs), taken from the PII dataset and an in-house research project dataset were hand labeled by staff. Each pair was labelled as either match/no match by three annotators, with disagreeing annotation decided by majority (<1% of pairs).
- A synthetic dataset was created programmatically via a Python script. The script takes a dataset as input and randomly introduces errors into records. The rate at which errors appear and the proportions of errors are tunable via a collection of rules and probabilities; this allows us to quickly generate large simulated matches, labeled not only with the absolute truth, but also with an indication of what modifications were made to a given record.

Being driven by parameters, the synthetic data set is likely not entirely representative of the kind of datasets we might see for a match – we are assuring that certain types of errors will occur within a certain percentage of the records. By the nature of the project, we cannot know beforehand how common particular errors are before beginning a match, and at best, we can have little more than an expectation for how much intersection there might be between two sets of data. A synthetic set allows us to experiment with different settings. For example, we can contrast the performance of one model when 99% of the data points should have a true match as opposed to 10% and modify our approach to address both.

The hope behind the synthetic data is two-fold. First, we hoped that by testing on synthetic datasets, we can identify what kinds of perturbations are most effective at forcing an error for a given model and adjust our approach accordingly. Second, we wanted to ensure that our approaches were feasible at the extreme end of the scale of the data we expected to receive. Since we were unable to obtain enough curated labels to test at larger sizes, the synthetic sets were used to check the performance of our approach on very large matches.

Process

In order to perform a match, MEDC requires an incoming dataset to contain at the minimum a unique identifier per record, a last name field, and a date of birth field.[2] Any additional information which can be provided by a researcher regarding their dataset (e.g. matches should only occur in a given geographical area or age range) will also help reduce the incidence of false positives in matching results.

Datasets submitted to MEDC for matching are all subject to an identical cleaning process. This process is also applied to MEDC's internal PII dataset in order to both improve consistency between datasets and identify/remove erroneous data points.

1. Records are de-duplicated on the unique identifier.
2. String Fields (first/last name, street address, city, state, zip code):

    1. All strings are converted to uppercase and trimmed of leading and trailing whitespace.

    2. All characters not A-Z, 0-9, spaces, single apostrophes, or hyphens are removed.

    3. All strings which match any of a list of common missing value tokens (e.g. “UNK” or “NA”) are removed.
3. Date Fields (date of birth)

    1. The date format (e.g. D/M/Y, YMD, etc.) is inferred based on a random sample of 1000 records.

    2. The field is parsed according to the estimated format, with all invalid dates[3] removed.

4. Records missing either a last name or a date of birth are noted and removed from matching, as these fields are required for our process.

Some summary statistics about the incoming dataset are also generated at the same time that records are being cleaned:

- How many records will be matched?
- How many records are being matched against?
- For each matching field
    - What percent of values are unique?
    - What percent of values are missing?
- How many records appear to have
    - duplicates with the same ID?
    - unusual characters that would be removed by the validation process?

Blocking

Since the goal of matching is to find the matching record from one dataset in another, an intuitive approach might be to iterate over each incoming record and try to find the corresponding record in the other dataset. Although this method is reasonable in small datasets, it scales poorly as the size of each dataset increases:

Example 1: Two Small Datasets

- 50 records to be matched
- 100 records to match against
- Total comparisons: 5,000
- Evaluation time at 10,000 comparisons/sec: 0.5 seconds

Example 2: Two Large Datasets

- 1 million records to be matched
- 3 million records to match against
- Total comparisons: 3 trillion
- Evaluation time at 10,000 comparisons/sec: ~9 ½ years

Clearly, some form of efficiency is needed in order to reduce the computational time needed to perform a large match. A common solution is to implement various "blocking" strategies, in which records with similar characteristics are identified in both datasets such that we are maximally sure that, for a given record, its corresponding matching record from the opposite dataset is present in the same block. The matching process then only needs to operate within each block, rather than performing a pairwise set of comparisons between datasets.

Blocking is highly effective at reducing the number of comparisons made.

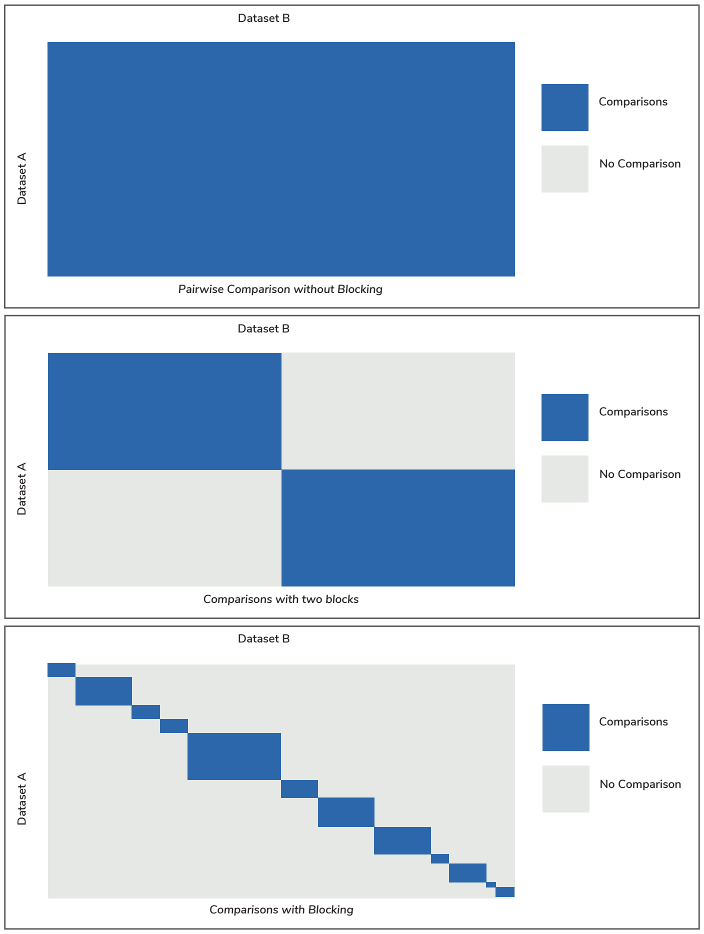


Figure 1 - Visualizing Various Blocking Methods

For example, a very simple blocking method – by gender – applied to the large dataset example from above (assuming an equal distribution of gender in each dataset) would now look like this:

Example 3: Two Large Datasets

- 1 million records to be matched
- 3 million records to match against
- Blocks:
  - Female Block: 500k records matched against 1.5 million records
  - Male Block: 500k records matched against 1.5 million records
- Total comparisons: 1.5 trillion
- Evaluation time at 10,000 comparisons/sec: ~ 4 ¾ years

This blocking strategy in the third example might be visualized as the middle image of Figure 1 – here we no longer have to match 50% of the records against each other, only the shaded area are included in the evaluation. More complex blocking methods might look like the bottom image, in which significantly more of the records are never directly compared. More restrictive and complex blocking mechanisms can be introduced for large datasets that further reduce the computational time until we reach fairly modest run times for even the largest datasets. The tradeoff with blocking, however, is that we enforce a certain unknown proportion of false negative matches. False negative matches are matches which a perfect record matching model would have made had it seen both records, but because those records were placed into separate blocks they were never directly compared (in Example 3, a record which should be matched, but that contained a different gender in each dataset). We can alleviate this drawback by including multiple blocking schemes so that two records that one scheme might place into separate blocks could still be compared if they fell into the same block in another scheme.

Blocking methods can be evaluated by three metrics:

- Reduction Ratio: How efficient is this blocking method at reducing the number of comparisons?
- Pair Completeness: How often are matching record pairs put into the same block?

- Pair Quality: What proportion of pairwise comparisons within a block will be performed on a matching record pair?

Blocking has the additional benefit of being an inherently parallelizable problem. Since no block ever relies on the result of any other block, we can easily scale our solution to the number of available processing cores – sending each block out for calculation and appending the result as it completes. This allows us to complete even very large record matches in a few hours.

MEDC's matching process contains two blocking schemes:

- Blocking by Exact Date of Birth
- Blocking by the Phonetic Encoding of Last Name
  - The Soundex[4] of the full last name is always taken as a block
    - If the last name contains any hyphens or spaces, each part is looked at in turn.[5]
    - If the name segment is longer than three characters[6], the record also belongs to the block designated by that Soundex

Figure 2 - Example Blocks Generated From Last Name / Date Of Birth

Last Name	***DOB***	***Block Memberships***
Smith-Jones	12/18/1994	12/18/1994, S532, S530, J520
Jones	12/18/1994	12/18/1994, J520
Richards	5/24/1993	5/26/1993, R263
Richard	5/29/1993	5/29/1993, R263

Figure 2 contains some examples of this blocking method applied to two pairs of invented records. In the first pair of rows, we can see that these two records will be compared twice – once in the "12/18/1994" block as well as once in the "J520" block. In the second pair, we have two records with slightly different birthdates, so they will end up in different birthday blocks. They will, however, be compared against one another in the "R263" block.

Based on internal experimentation on our “truth” datasets, these two methods were chosen for being both highly likely to contain true matches as well as vastly reducing the number of comparisons required of the matching model. Enforced false negatives will occur only when a true match contains both an incorrect birth date and a last name significantly different enough to be assigned a different Soundex.[7]

Matching

Within a probabilistic matching model, there are three hurdles that need to be overcome:

- How to handle the number of comparisons necessary to find all potential matches
- How to calculate a similarity “score” for a given pair of records
- How to calculate a similarity “score” for a field within a pair of records

The first hurdle is addressed with blocking – we will essentially turn the one giant matching problem into a series of much more reasonably sized matching problems, aware that we may be enforcing a certain number of missed matches in exchange for shorter calculation time. The second two hurdles are now much more surmountable: within a given pair of blocks. We now need to identify a way of calculating how similar each pair of records are, and in order to do this, we need to be able to calculate how similar two fields are between a pair of records.

Field Matching

For some very simple fields, like gender, we can use a simple Boolean match – that is, if the genders of a pair of records are the same, then they score a 1. If they are different, they score a 0. What we’re really interested in regarding our model, however, is how to incorporate a level of “fuzzy” matching that will allow us to pick up on minor spelling variations between words or names.

Our process uses a string metric called “Jaro-Winkler distance” (Winkler 1990) to quantify how similar two strings are on a scale of 0-1. The JaroWinkler algorithm takes into account the number of characters that match in each sequence, putting extra weight on a proportion of the first characters[8] and lessening the negative impact of transposed characters. For our purposes, string pairs are considered “matching” if their Jaro-Winkler distance is greater than 0.92, "similar" if their distance is between 0.92 and 0.88, and "no match" if below 0.88. As demonstrated by Figure 3, identical or nearly identical names are given a very high score, whereas dissimilar names fall below the match threshold.

Figure 3 - Jaro-Winkler Distance For Example Name Pairs

Name 1	Name 2	Jaro-Winkler Distance
Roberts	Roberts	1.000
Jacobson	Jacobosn	0.975
Smith	Smyth	0.893
Jones	Johnson	0.832
Richards	Christoph	0.593

We can also implement a level of fuzziness in numerical fields – with Year of Birth, for instance, we consider identical years "matching", while records within +/- one year "similar"[9].

Field Matching

The matching model MEDC uses is an unsupervised version of the Fellegi-Sunter probabilistic model (Fellegi and Sunter 1969) as implemented by the open source fastLink R library[10] and tailored to our particular use case. The Fellegi-Sunter model estimates the probability of a given match given a pattern of agreement and disagreement between the features of two records.

Traditionally, the model requires a certain amount of labelled data in the form of a "ground truth" and estimates two probabilities for each field:

> *m*: "How likely is it that in a given record pair these two fields will match when they are a true match?"
>
> *u*: "How likely is it that in a given record pair these two fields will match entirely by chance?"

As an example, let's take two datasets with two fields – gender and last name. Within our labelled data, we can see that when two records are labelled as a match, there is agreement within the gender field 90% of the time and within the last name field 95% of the time. We can also see that in pairs that are not labelled as matches, there is agreement by chance in the gender field 50% of the time, and in the last name field .01% of the time. These are our *m* and *u* values from which we can calculate a weight to give to the probability of either a match or a nonmatch.

	Pair is a Match	**Pair is Not a Match**
Gender Fields Match	m = 0.9	u = 0.5
Last Name Fields Match	m = 0.95	u = 0.01

What this tells the model is that a gender that matches is an indicator that this pair might match, but a nonmatching gender field should not make us at all certain that this is not a match, since that happens regularly. Identical last names, on the other hand, are a pretty strong indicator that the pair is a match, since matching pairs nearly always share a last name. It also is a strong indicator in favor of making non-matches, since it is fairly rare that two last names will agree by chance. Taken together, we would be more confident in making a match when both gender and last name agree than if either did not, and although we might still be willing to consider a match where the last name matched and the gender didn't, we would be far less confident in a pair where gender matched and last name didn't.

What we don't have in our use case, and what this model expects, is a set of labelled data. A researcher who requests a match is only providing a collection of PII, and we have no way of knowing in advance how much variation in spelling/birthdates/address exists and thus no way of directly assigning values to *m* and *u* for each field. We could obtain this with some human effort – with each incoming match, we might identify approximately a thousand most likely pairs and calculate the *m*/*u* parameters for the Fellegi-Sunter model based on those labels. That would require a significant expenditure of expensive human effort, so the fastLink package attempts to estimate these parameters with the Expectation Maximization (EM) algorithm in order to best approximate what *m* and *u* should be for each field given the records.

Broadly, the EM algorithm in this context looks at how often each combination of a given pattern of Match/Partial/Non-Match appears between all records and iteratively tries to estimate how to best assign *m*/*u* parameters for each field so that the model will properly discriminate between matches/non-matches.[11] In early experimentation, we determined that this methodology appeared to perform at least on par with other methods and in some cases, even outperformed models that required human input.

Post-Analysis

As part of the output of the matching model, each pair of predicted matches has associated with it a posterior probability – a value between 0 and 1 that indicates how "sure" the model is that the pair of records represents a true match. In cases where all fields are identical, we would expect this probability to be almost 1. Conversely, we would also expect there to be a subset of records in the incoming dataset that do not exist in our dataset, and we would expect these probabilities to be near 0, since no close match should exist. However there will also exist a certain number of records for which the probability falls somewhere between the extremes, where the model found a "maybe" match. While with small matches it might be possible to hand label all of these "maybe" matches, this becomes far too time expensive at larger scales, and so we have to identify a threshold at which we are willing to accept a match.

It may be tempting to set the threshold very high – 0.99 for instance – and only retain those matches about which the model was very certain in order to minimize the incidence of incorrect matches. This would likely result in discarding a large number of possibly good matches, but we would be highly confident in the matches we did have. On the other hand, we might be willing to set the threshold very low – say 0.40 – with the understanding that we are willing to take the risk of including a number of incorrect matches in exchange for more matches overall. In either case, we are not truly able to quantify the risk we would be taking.

Metrics

What we can do to quantify the risk is to select a random sample of matches that were returned by the model and label this subset in expectation that it will be representative enough of the overall match to make reasonable estimations of how many errors exist at a given threshold.[12] With our labeled subset of data, we can identify the different types of "good" and "bad" matches that exist in our output:

True Positives = The model predicted a match, and the human annotators agree

False Positives = The model predicted a match, and the human annotators disagree (Type 1 Error)

False Negatives = The model doesn't predict a match humans say should exist (Type II Error)

True Negatives = The model did not predict a match and neither did the human annotators

By counting the total number of each of these types of matches that we see in our labelled data, we can calculate two further statistics:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$ = *Of all of the matches predicted, what proportion were correct?*

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$ = *Of matches that should exist, how many were predicted?*

With these in hand, we can now look at our labeled data and see how the match would score at various thresholds. As we change the threshold, we are no longer claiming the model would have predicted that records below a given probability would be a match, so we are removing false positives at the cost of introducing false negatives. In general, we expect that at a threshold of 0, at which we accept every match regardless of how certain the model is, recall will be very high and precision is very low. As we increase the threshold, recall will began to drop, and precision will increase as we eliminate "maybe" matches, until at a point the only matches that remain are those of which the model was almost certain.

Figure 4 - Accuracy on a Hypothetical Match

		Actual		
		Match	No Match	Accuracy
Predicted	Match	35	10	0.999
	No Match	5	9940	

What we are still missing here is a metric that helps us to identify where we should be setting the threshold for a given match. Take for example a hypothetical match that attempted to match 50 records to 200 records. We would perform 10,000 total comparisons, of which, at best, 50 would be true positives and 9,950 true negatives. If we assume that some of the matches were incorrect, as in Figure 4, then accuracy (# correct out of total predictions) gives us an unclear view of how well the match performed because, as we are performing a pairwise comparison, True Negatives dominate any calculation. Matching more than 1/5 of all of the records incorrectly is not an ideal outcome, but the accuracy metric might suggest that we had actually performed well. What we need instead is a metric that will ignore the false negatives in our data:

$$F_1 \text{ score} = 2 \times \frac{\textit{Precision x Recall}}{\textit{Precision + Rec}} = \textit{Harmonized Mean of Precision and Recall}$$

F_1 score is a common metric applied in binary prediction tasks to evaluate the performance of a classifier. Here, we are also making a binary classification – is a record a match or not? We can see that in our hypothetical match from before, the F_1 score is much lower than the accuracy and better reflects the quality of the match. We still managed to correctly match 35 of our records, but the influence of the 15 incorrect matches is still visible in the score.[13]

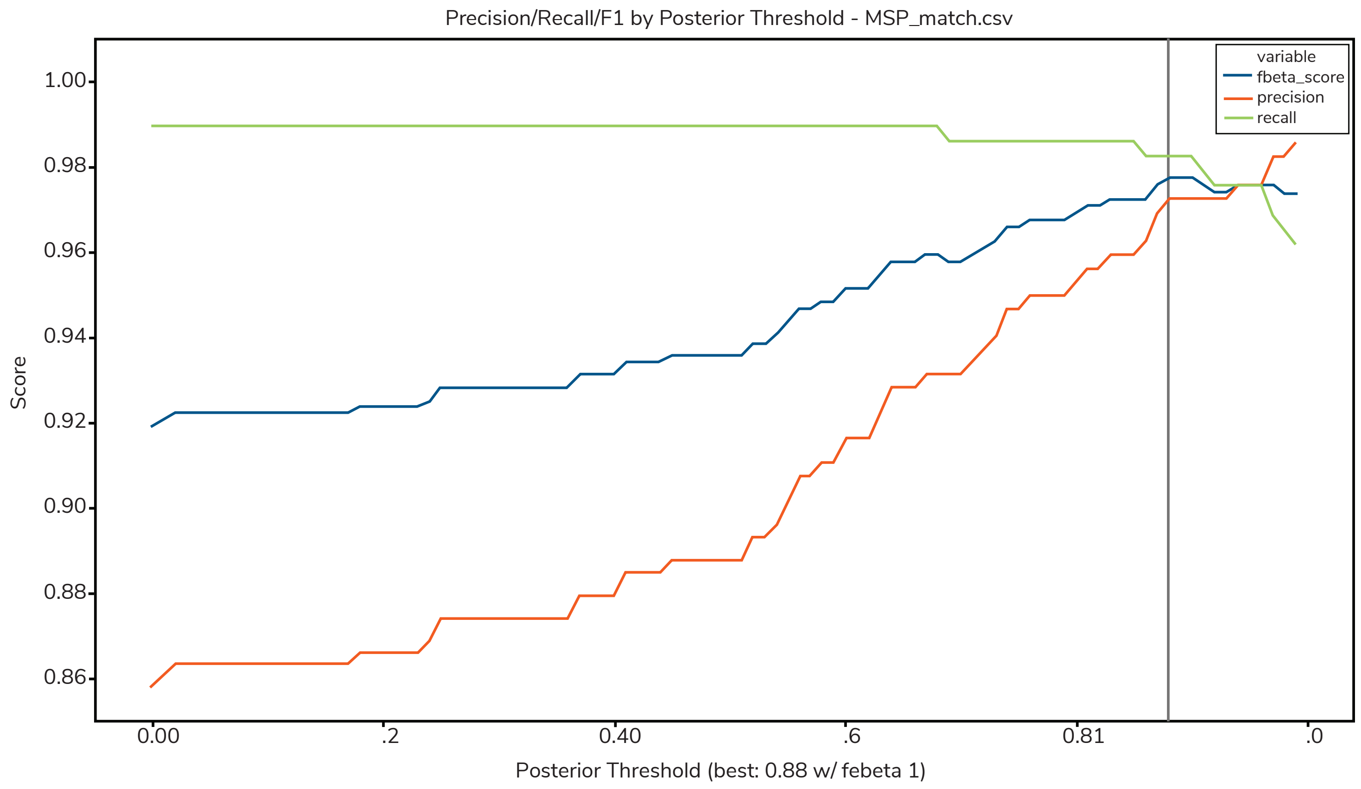
By using an F_1 score, we are providing equal weight to Precision and Recall, by which we seek to minimize the overall error rate of the output by locating the threshold at which F_1 is at its highest. Although this will result in a less conservative overall matching result, some literature suggests that being too restrictive within matching criteria is detrimental to the result of a study that incorporates the match (Tahmont, et al. 2019).[14]

Figure 5 - F_1 Score on a Hypothetical Match

		Actual			
		Match	No Match	Accuracy	F_1 Score
Predicted	Match	35	10	0.999	0.824
	No Match	5	9940		

Setting a Threshold

Figure 6 - Precision (Orange), F_1 Score (Blue), and Recall (Green) Of a Sample Match By Posterior Threshold



Applying this to a match, we can calculate these metrics at a variety of thresholds and identify the point at which the F_1 score was at its highest point. Figure 6 is a plot of these metrics from an early match performed by MEDC. We can see the precision of the match increasing as we raise the threshold while recall remains mostly constant. However once the threshold starts to exceed 0.70, we begin to eliminate from the match some record pairs that were labelled as “correct”. Once we hit a threshold value of 0.90, the decrease in recall is enough to start balancing out the increase in precision and the overall F_1 descends from this point on. For this particular match, a threshold of 0.88 was identified.

Matching

Upon completion of the entire matching process, we are able to provide the researcher with

- Summary Statistics collected before the match about the incoming dataset
- A csv with:
    - The unique Identifier provided by the researcher in their dataset
    - A unique Identifier from our dataset that was determined to be the best match
    - A value indicating how confident our model is that the records are a match
    - A label indicating anything unusual about the record pair (e.g. it failed validation, was manually labelled, etc.)
- A plot of the precision/recall/F_1 score by threshold indicating why that threshold was selected

Endnotes

1. It is also important to note that the "truth" data we deal with here is not an objective truth, as often exists in traditional machine learning. The labeled pairs represent a human's opinion on a pair of data points, and it is entirely possible that they are mistaken – two individuals with the same name born on the same day is rare, but not impossible.

2. Last name and date of birth represent the minimum amount of data with which a match can be performed; additional information will improve the quality of the match.

3. "Invalid" dates would be those dates which either do not exist in the calendar (e.g. February 30th or 14/06/2001), are far too old (likely placeholders, like 1/1/1900), or are in the future.

4. Soundex is a phonetic algorithm that attempts to encode a name such that homophones like "Smith" and "Smyth" are given the same code. The code consists of the first letter of the name followed by three digits which represent the consonant sounds after the first letter. An overview of the encoding rules is available at https://www.archives.gov/research/census/soundex.html.

5. Since Soundex appends zeros to a code if the name is short, we can often end up with compound last names receiving a different Soundex code than either last name on its own – e.g. "Smith" would be encoded as "S530", but "Smith-Jones" would be encoded as "S532", with the last "2" coming from the "J" in "Jones". By encoding each last name segment as well as the overall name, we can ensure that "Smith-Jones" will end up in a block with all of the other "Smith-Jones" records as well as "Smith" and "Jones" records.

6. A minimum length was chosen to avoid creating very large blocks for names which contain name affixes like "de la", "al" and "von".

7. We are currently using "Standard Blocking" in our approach, but there are many alternative methods – see (Baxter, Christen and Churches 2003) and (Steorts, et al. 2014) for a description of more advanced methods.

8. In data which is manually entered into an electronic system, entry errors are more likely to occur after the first few letters. This weighting will ensure that “Johnson” and “Jonson” – two different names - are not considered as similar as “Johnson” and “Johnosn” – the result of a typographical error.

9. Date fields offer an unusual challenge in that, though they could be considered Boolean matches, we might allow for specific rules that provide for partial matches where common clerical mistakes occur – for example, data entry off of a handwritten document mistaking a “4” for a “9”. This is not yet implemented in our model.

10. https://github.com/kosukeimai/fastLink

11. See (Abramitzky, Mill and Perez 2018) for an in depth explanation of the method.

12. Through experimentation, we have determined that at minimum, 200 labelled examples are needed to give a reasonable estimate.

13. An alternative metric might be F_{Beta}, which is an F_1 score in which the relative importance of Precision and Recall is determined by a value Beta – a more conservative metric might be to use a beta of 0.5, indicating that we consider Precision twice as important as recall.

14. For example, a match with 2% false positives and 2% false negatives has an overall lower error rate than if a more restrictive criteria were used, decreasing the false positive rate to 1% at the cost of 5% false negatives.

References

Abramitzky, Ran, Roy Mill, and Santiago Perez. 2018. "Linking Individuals Across Historical Sources: a Fully Automated Approach." National Bureau of Economic Research. http://www.nber.org/papers/w24324.

Baxter, Rohan, Peter Christen, and Tim Churches. 2003. "A Comparison of Fast Blocking Methods for Record Linkage." Proc. of ACM SIGKDD'03 Workshop on Data Cleaning, Record Linkage, and Object Consolidation.

Fellegi, Ivan P, and Alan B Sunter. 1969. "A Theory of Record Linkage." Journal of the American Statistical Association 1183-1210.

Steorts, Rebecca C, Samuel L Ventura, Mauricio Sadinle, and Stephen E Feinberg. 2014. "A Comparison of Blocking Methods for Record Linkage." CoRR abs/1407.3191. http://arxiv.org/abs/1407.3191.

Tahmont, Sarah, Zubin Jelveh, Aaron Chalfin, Shi Yan, and Benjamin Hansen. 2019. "Administrative Data Linking and Statistical Power Problems in Randomized Experiments." National Bureau of Economic Research. doi:10.3386/w25657.

Winkler, Willian E. 1990. "String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage." Proceedings of the Section on Survey Research Methods (American Statistical Association) 354-359.