# A Machine Learning Approach to Census Record Linking[*]

James J. Feigenbaum[†]

March 28, 2016

**Abstract**

Thanks to the availability of new historical census sources and advances in record linking technology, economic historians are becoming big data genealogists. Linking individuals over time and between databases has opened up new avenues for research into intergenerational mobility, the long run effects of early life conditions, assimilation, discrimination, and the returns to education. To take advantage of these new research opportunities, scholars need to be able to accurately and efficiently match historical records and produce an unbiased dataset of links for analysis. I detail a standard and transparent census matching technique for constructing linked samples that can be replicated across a variety of cases. The procedure applies insights from machine learning classification and text comparison to record linkage of historical data. My method teaches an algorithm to replicate how a well trained and consistent researcher would create a linked sample across sources. I begin by extracting a subset of possible matches for each record, and then use training data to tune a matching algorithm that attempts to minimize both false positives and false negatives, taking into account the inherent noise in historical records. To make the procedure precise, I trace its application to an example from my own work, linking children from the 1915 Iowa State Census to their adult-selves in the 1940 Federal Census. In addition, I provide guidance on a number of practical questions, including how large the training data needs to be relative to the sample.

[*] I thank Christoph Hafemeister, Jamie Lee, Christopher Muller, Martin Rotemberg, and Nicolas Ziebarth for detailed feedback and thoughts on this project, as well as seminar participants at the NBER DAE Summer Institute and the Berkeley Demography Conference on Census Linking. Viroopa Volla, Andrew Creamer, Justin Meretab, and especially Alex Velez-Green all provided excellent research assistance on the data collection used in this project. This research has been supported by the NSF-IGERT Multidisciplinary Program in Inequality & Social Policy at Harvard University (Grant No. 0333403).

[†] PhD Candidate, Department of Economics, Harvard University, email: jfeigenb@fas.harvard.edu, site: scholar.harvard.edu/jfeigenbaum

# 1 Introduction

Thanks to the availability of new historical census sources and advances in record linking technology, economic historians are becoming big data genealogists. Linking individuals over time and between databases has opened up new avenues for research into intergenerational mobility, assimilation, discrimination, and the returns to education. To take advantage of these new research opportunities, scholars need to be able to accurately and efficiently match historical records and produce an unbiased dataset of links for downstream analysis.

The problems in record linkage facing economic historians are distinct from those faced by users of modern datasets. Where uniquely identifying variables such as social security numbers are available, it is mostly a question of getting access to restricted use files.[1] But such variables are rarely found in historical data. Instead, economic historians have access to other variables that can be combined to try and uniquely identify individuals, such as first and last names, year of birth, state of birth, and parents' place of birth. Unfortunately, historical data are also not as clean as modern data, and these variables may be mismeasured, including transcription errors, spelling mistakes, name changes, or name shortening.

In this paper, I detail a transparent census matching technique for constructing linked samples that can be replicated across a variety of cases. The procedure applies insights from machine learning classification and text comparison to the well known problem of record linkage, but with a focus on the sorts of costs and benefits of working with historical data. I use a supervised learning procedure, teaching an algorithm to discriminate between correct and incorrect matches based on training data generated by the researcher. The method begins by cross matching two census-like datasets and extracting a wide subset of possible matches for each record. I then build a training dataset on a small share of these possible links and use the training data to tune a matching algorithm. The algorithm attempts to minimize both false positives and false negatives, taking into account the inherent noise in historical

[1] For example, Chetty et al. (2014) use tax payer IDs to link tax payer records over time and social security numbers to match dependents to heads of household across and within samples.

records. To make the procedure precise, I trace its application to an example from my own work, linking children from the 1915 Iowa State Census to their adult-selves in the 1940 Federal Census. Using my linking procedure, I am able to match nearly 60% of the sons in my data ahead to 1940.[2] The procedure follows many of the central ideas outlined by Goeken et al. (2011) regarding the Minnesota Population Center linkage project that ultimately built the IPUMS linked samples, but with an extension of the method to other linking procedures and more detail on the utilized record comparison characteristics included. The procedure I outline in this paper can be fully and transparently implemented with software commonly used by empirical social scientists—either Stata or R, for example.

In addition, I provide guidance on three practical questions for social scientists undertaking historical record linkage. First, I show how many records need to be manually coded as matches and non-matches by the researcher before tuning the match algorithm. Fortunately, the procedure is quite accurate even with a relatively small training data set. Second, I run a horse race between potential classification models—including probit and logit models, random forests, and support vector machines—and show that the probit model, familiar to all social scientists has the best cross-validation, test set performance. Third, I document that the matching algorithm avoids making one particularly bad type of false match—linking people who have died between census waves—using the records for Major League Baseball players active in the 1903 season.[3]

The linking of historical records by scholars is not new. Thernstrom (1964, 1973) matched generations in Boston and Newburyport, MA to study intergenerational mobility. The Minnesota Population Center, taking advantage of the 1880 complete count census, provides linked data from each census between 1850 and 1920 to 1880.[4] Joseph Ferrie linked records between the 1850 and 1860 Federal Censuses,

[2] I will also show that these matches are not just made but made accurately and that the algorithm is able to replicate the careful manual matching work done by a trained RA.

[3] Thanks to both the prominence of the people in the dataset and the interest among baseball fans, the Lahman database (Lahman 2016) presents a unique source that includes both the biographical information necessary to link to the census, as well as detailed information on date of death.

[4] See `https://usa.ipums.org/usa/linkeddatasamples.shtml`

exploring various dimensions of economic mobility (Ferrie 1996). However, no standard method for linking these records has emerged. Each scholar alters the process slightly based on the data at hand and the tools available.

In April 2012, the US Census Bureau released the full, un-anonymized 1940 Federal Census, opening many new possibilities for historical research. By law, the complete census, which includes the names of all respondents, must be sealed for 72 years after its completion.[5] The 1940 census was the first nationwide survey to include questions on educational attainment and annual income.[6] The data in the 1940 census has been used by researchers in the past to measure the returns to education, to quantify racial and gender discrimination, and to answer many other research questions. These analyses have all been possible through the use of an anonymized 1% 1940 Census sample collected by IPUMS (Ruggles et al. 2010).

What specifically changed for researchers in April 2012? The full census allows for the matching of individuals from other datasets—other federal and state censuses, enlistment records, legal records, etc—by name to the 1940 Census. With such matched records, it is possible to conduct research that follows individuals over time or across generations— intergenerational mobility, stability of income over time, long run effects of exposure in childhood to pandemics, etc. But how can a researcher merge a list of names from one set of records into another, such as the 1940 census?

Any matching procedure should aspire to three important criteria: it should be efficient, accurate, and unbiased. I define these terms in the record linkage context:

- Efficient: A high share of the records to be searched for are found and matched. The match rate will naturally vary across applications and source or target databases, but generally, a procedure that requires thousands of records to match only a handful would be quite inefficient and not very useful for econometric analysis. An efficient match process will have a low share of type I

[5] The 72 year seal is driven by privacy concerns. When the law was first passed, life expectancies were such that few census subjects would be alive 72 years later. That is less true today, but the privacy law remains in effect. The 1950 census will be unsealed in April 2022.

[6] In 1915, the Iowa State Census compiled similar data; see Goldin and Katz (2000).

errors. In the machine learning context, one measure of efficiency is the true positive rate or TPR. This records the ratio of true positives with the total number of positive: $TPR = \frac{TP}{TP+FN}$.

- Accurate: A high share of the records matched are true matches and not false positives. Ideally, this rate would be close to 100%, but naturally the higher the bar for declaring two records matched, the less efficient it will be. An accurate match process will have a low share of type II errors. In machine learning, accuracy could be measured with the positive predictive value or PPV. This measures the ratio of the true positives to all of the records identified as matches by the algorithm: $PPV = \frac{TP}{TP+FP}$.

- Unbiased: A match procedure will generate a dataset for downstream analysis. To what extent is this final dataset representative of the records that the researcher attempted to link in the first place? Improvements in either efficiency and accuracy will necessarily decrease the bias in the resulting dataset. But non-random variation in either error rate will generate bias. One manifestation of bias would be an unrepresentative linked sample. Using spouse names to create links, for example, would increase the match rate among married people and over-represent them in final analysis; similarly matching on county or state of residence would bias against including interstate migrants in the sample (Goeken et al. 2011).

Manual matching is one option for record linkage. Scholars can hire research assistants to search for each name in one dataset in the 1940 census, either via the index file or through a commercial provider like Ancestry.com. With dutiful RAs, this process could be quite efficient with assistants tracking down as many links as possible. And with skilled RAs, the process could be highly accurate. However, this method is costly and time-consuming.[7] Perhaps more importantly, it is inconsistent,

[7] I have found RAs can search for approximately one record per minute on Ancestry.com. With a match rate of 50%, generating a database of 1000 matched records will cost $\$2000/60 \times .5 \times w$, where w is the RA's wage (or double that for double entry). Search time costs will certainly vary by researcher and project.

certainly across projects but perhaps within a project as well. Different RAs will use different internal heuristics or decision rules in matching or not-matching close hits. While a clear set of rules can reduce such problems, a complete decision tree is impractical if not impossible.

Researchers would be better off using a formalized matching algorithm that made consistent choices between potential matches in all scenarios. Goeken et al. (2011) describe the method used by the Minnesota Population Center for IPUMS.[8] The process makes use of the Freely Extensible Biomedical Record Linkage (FEBRL) software from Peter Christen and Tim Churches.[9] This method also relies on highly trained researchers to identify and approve of huge numbers of links between different censuses. The matches are made initially by FEBRL, comparing records by first name, last name, year of birth, state of birth, race,and gender. A subset of these possible matches are then completed by researchers, identifying true and false links on a training data set. Then, IPUMS uses a standard machine learning technique, Support Vector Machines (SVMs), trained on name and age similarity scores, to classify matched records as links or non-links.

The method I propose in this paper deviates from the IPUMS strategy in three key ways. First, the tools used in my method are all available in Stata and R and should be more familiar to most economists and other social scientists than either FEBRL or SVMs. This should make implementation of the procedure in different datasets easier. Second, unlike the FEBRL system used to collect possible matches, my method was designed for historical work, using datasets with transcription errors and other historical noise. Third, the method can be easily modified for use with different datasets that contain alternative potential matching variables.

[8] Mill (2012) proposes another alternative record linking procedure, one that is a fully automated learning algorithm that does not require even a training dataset. Using an EM maximization process, the algorithm attempts to split the data into a set of matched and a set of unmatched records. One key strength of this method is that it does not require any training data and thus suffers from no human-induced bias in determining which records are or are not matches. However, the Bayesian learning process employed is constrained by independence between parameters.

[9] `http://sourceforge.net/projects/febrl/`

# 2 Procedure

To fix ideas, I will describe the process used to locate matches between a smaller list of records and a full list of records in a 100% census. For example, in Feigenbaum (2014), I match a list of boys from the Iowa 1915 State Census into the 1940 Federal census. Other examples that could potentially use this method include Aizer et al. (2014) which linked the male children of recipients of a Progressive era welfare program for poor families into the 1940 Census, WWII enlistment records, and death records or Collins and Wanamaker (2014) which linked black and white census respondents from the IPUMS sample of the 1910 census to the 1930 census.

The size of the first set of records can vary from project to project. The list of sons from Iowa 1915 was approximately 5500 observations. In other cases, it will be much larger: Collins and Wanamaker (2014) started with nearly thirty thousand observations of men from southern states in 1910.

### Extracting Possible Matches

Call the first set of records $X1$. What variables does it need to contain to ensure a good (and unbiased match)? To begin with, first and last names for each record. Without names matching would be unlikely to work with any method. In addition, an age or year of birth,[10] gender, and a state of birth will be improve match quality. Mother and father state of birth have proven to be valuable matching variables in past work. However, in the 1940 complete count census, these questions were not asked of all respondents.[11]

I prefer not to use reported race in my matching algorithm. As documented by Mill and Stein (2012), individuals change their reported race in the census with some

[10] Of course, assuming that the year the records were compiled is known, year of birth and age are substitutes.

[11] Only those respondents entered on the 14th or 29th line of each enumeration page were asked additional questions. This is roughly 5% of the population. Mother and father place of birth were among these supplementary questions. See `http://www.archives.gov/research/census/1940/general-info.html#questions`. For this subsample, parent's place of birth, if it is available in the original list of records, can be used in testing the accuracy of the match algorithm and to compare between different possible match algorithms. But it would significantly reduce the sample size to require these variables in the record linking.

frequency. To the extent that changing a respondent's reported race is endogenous to whatever process the researcher ultimately hopes to study, then conditioning matches on race will likely bias the final sample. However, it may be useful to use race in both datasets as an outcome of interest in assessing match quality of the final data.[12]

Call the second set of records to be matched into $X2$. The records in $X2$ should include first name, last name, and any of the other variables to be used for matching that were also included in $X1$, often gender, state of birth and year of birth (or age).[13] 1940 complete count census available from Minnesota Population Center and IPUMS (via NBER) has these variables and more, but other censuses do as well (complete counts of 1880 and 1850 are both available via IPUMS, for example). To this point, I have listed gender as one of the variables used in the census record linking. However, given the frequency with which women change their surnames in marriage, the task of linking women across censuses is much more difficult than linking men. While this will mean that there are important questions that I cannot answer, I focus here on linking only men. Thus, as a preliminary step, imagine limiting both $X1$ and $X2$ to only male observations.

The first step is to extract records in $X2$ that could be matched with records in $X1$. What defines which records could be matched? In an extreme case, the entire Cartesian product of $X1$ and $X2$ has $X1 \times X2$ observations. The matching algorithm could compare each of these potential matches, but even with a smaller original list (for example, the sons in Iowa in 1915), matching it with any complete census index would result in a huge database, full of potential matches that are clearly not matches.[14]

To limit the comparison to links that have some non-trivial likelihood of being matches, I first extract the records in $X2$ with the attributes sufficiently similar to the attributes of the record in $X1$. The attributes I focus on are birth year, state of

[12] Mill (2012) uses race as well as county of residence between multiple census waves in such a way.

[13] And a unique identifier that will be used to merge our final data back into $X2$ to collect the variables of interest in any analysis.

[14] More the 700 billion for the 1915 sons and the full 1940 census.

birth, and names. Similarity of birth year is measured by the absolute difference in years. An indicator variable for matching or non-matching state of birth is clear to define as well. How does one measure string similarity for first and last names? I follow IPUMS, Goeken et al. (2011), and Mill (2012) in relying on the Jaro-Winkler string distance as a measure of string dissimilarity. Though other string distance measures, notably Editex and syllable-comparison, show better results in identifying common English homonyms, the differences between these measures on American names are less striking. Moreover, algorithms to calculate Jaro-Winkler distances are available for most statistical software packages;[15] an Editex module only exists for Python. For more details on the Jaro-Winkler distance and its properties, see Winkler (1994). Traditionally, strings that match will have a Jaro-Winkler distance of 1 and strings that are not similar at all will have a distance of 0. However, I use $1 - JW$ here, so that increases in distance correspond with words that are less similar, as one would expect for a distance measure.[16]

Returning to $X1$ and $X2$, I limit the set of possible links between the two datasets to be those records with the same reported state of birth, a year of birth distance less than 3 years, a first name Jaro-Winkler distance of less than .2, and a last name Jaro-Winkler distance of less than .2.

Why are these good starting point conditions? The state blocking condition is the same used by Mill (2012). IPUMS does not use a state blocking condition explicitly, however, of the 98,317 links between the 1880 census other decennial censuses, only 21 list different states of birth between matches, 20 of which were in the link between 1880 and 1850. The state blocking requirement reduces the complexity of the matching problem and the computing power required to solve it: string distance measures and matrix Cartesian products are both computationally expensive. For a given individual, state of birth should only change between censuses due to random enumerator error.

The birth year limits have been used in past matching work: The IPUMS project

[15] For stata, see my own package: `https://github.com/jamesfeigenbaum/jarowinkler-ado`. For R, see the RecordLinkage package.

[16] Mill (2012) makes the same translation of Jaro-Winkler string distance.

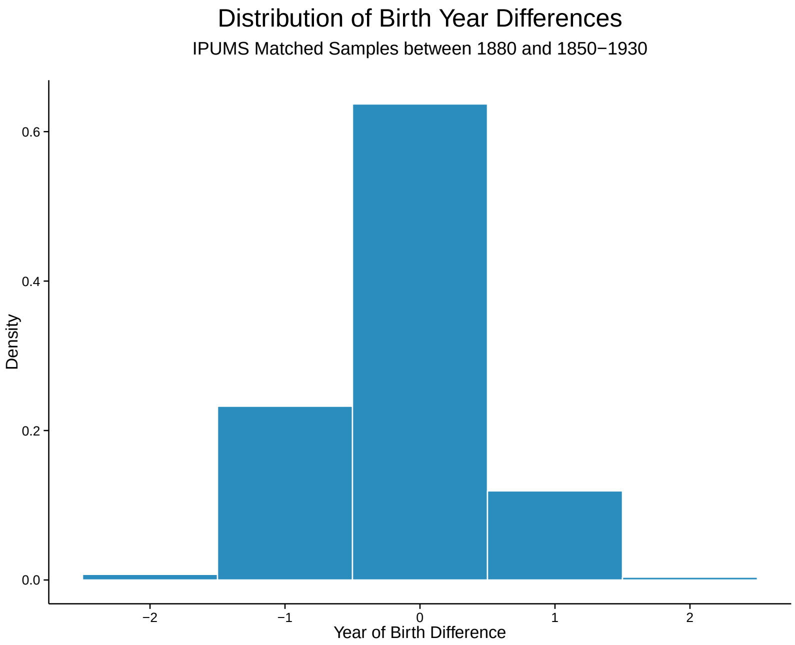


Figure 1: Distribution of Birth Year Differences. Differences are defined as the reported year of birth in the less recent census less the reported year of birth in the more recent census. For a record linked between 1880 and 1900 with a birth year of 1850 in 1880 and 1851 in 1900, the difference would be -1. All year differences are between -2 and 2.

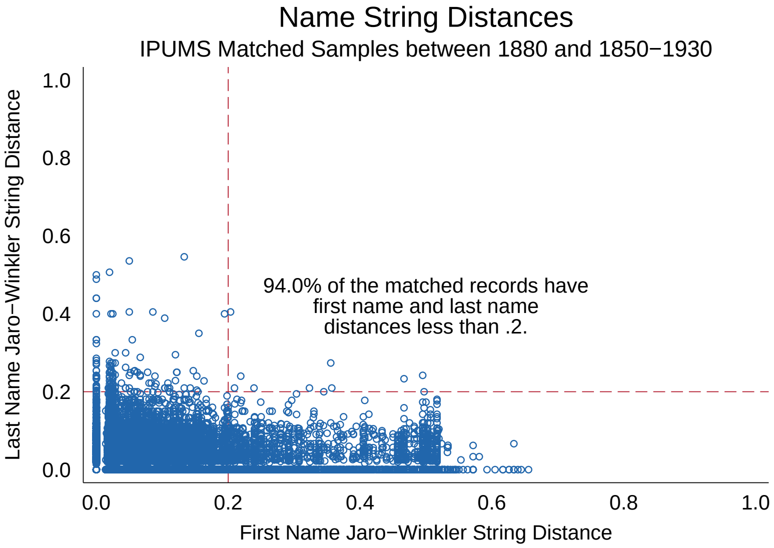


Figure 2: Distribution of First and Last Name String Distances, calculated with the Jaro-Winkler string distance metric. 94% of the records linked between the IPUMS samples have first and last name distances of less than 0.2.

merging various historical census samples into the 1880 complete count census used a year of birth distance of 7 (Goeken et al. 2011). However, in the final matched samples produced by IPUMS, no matches were made between records with birth years differing by more than 3 years. I follow these results and use 3 years as the distance bound here. As I show in Figure 1, based on data from the IPUMS matching procedure between 1880 and other censuses, the vast majority of matches are either 0 or 1 years apart, in either direction.[17]

The name distances may be more controversial. The stricter the bounds, the fewer potential matches will emerge and the quicker the entire matching process will be. The looser the bounds, however, the less likely it is that good matches will be lost in this early stage. Figure 2, presents a scatter plot of first and last name Jaro-Winkler distances for all the matched records between the 1880 census and other IPUMS samples. Only 6% of the matched sample falls outside the .2 distance

[17] Why consider any potential matches that do not share the exact birth year? Aside from simple errors of mis-transcription, there may also be “translation” errors. In many waves, the census asked respondents not their year of birth but their age. With census enumeration taking place over time (and the final census day varying slightly throughout the 19th and 20th century), the exact same respondent enumerated ten years apart may have a different constructed year of birth.

Table 1: IPUMS Linked Samples: Matched Record Name Similarity

| | Share of Links within Bounds | | |
|---|---|---|---|
| Linked Years | First Name Distance $\leq .2$ | Last Name Distance $\leq .2$ | Both Conditions |
| 1850-1880 | 94.07 | 99.92 | 93.99 |
| 1860-1880 | 92.86 | 99.81 | 92.68 |
| 1870-1880 | 94.44 | 99.84 | 94.30 |
| 1880-1900 | 94.50 | 99.79 | 94.31 |
| 1880-1910 | 95.01 | 99.69 | 94.72 |
| 1880-1920 | 95.47 | 99.94 | 95.41 |
| 1880-1930 | 96.23 | 99.96 | 96.19 |

Share of records matched between the 1880 complete count census and various IPUMS census samples by the Minnesota Population Center. These samples were built not using Jaro-Winkler distances but by trained research assistants comparing names (and other census data) manually. However, no more than 8% of the matched sample in any given year falls outside the .2 distance bounds for both first and last name Jaro-Winkler distances. Most linked records outside the bounds are farther apart in Jaro-Winkler distance in first names. Manual inspection reveals that most of these links match full names to initials or vice versa.

*Sources:* IPUMS Linked Representative Samples, 1850-1930: `https://usa.ipums.org/usa/linked_data_samples.shtml`

bounds for both first and last name distances. Most of these records outside the bounds appear to be farther apart in Jaro-Winkler distance in first names. Manual inspection reveals that most of these links match full names to initials or vice versa.[18] In Table 1, I show that for each IPUMS linked sample between 1880 and another census year, between 92.16% and 95.65% of the records are within these bounds.

### Two-Step Match Scores

Call the set of all possible matched data $XX$. Again, these are the records from our original data, $X1$, matched with records from the index file, $X2$, with matching state of birth, year of birth difference less than 3, and first and last name Jaro-Winkler distances less than .2. With the set of possible matches between each record of $X1$ and $X2$, we now how to decide which of these links (if any) are "good" links, in the sense that it is very likely that the record in $X1$ and $X2$ refer to the same individual.

[18] For data with a large number of first names recorded as initials, a looser distance bound may be useful.

To do so, we need to compile training data: a set of records for which the researcher (or research team) have determined that a record is either a match or not a match. I discuss in more detail in a later section how to generate the training data and how large it needs to be. For now, consider it simply as a subset of $XX$ called $XX_T$ , where each record is either described as a match or a non-match.[19] One key feature is that for every unique record in $X1$, there can be at most one match in the $XX_T$ data.[20] However, this one match is not a requirement. If there are no good matches for a given unique record in $X1$, then all the links will be marked as non-match. However, it need not be the case that each record in $X2$ have a match. Obviously in the case where $X1$ is a short list of people—like sons between 3 and 17 in Iowa in 1915—not every record in the entire 1940 census will be available.

I use the term training data because we will train our match algorithm using it as an input. However, there is one key way in which the training data in this context differs from training data more generally. In traditional machine learning applications, training data is a set of true or correctly classified observations. In the case of historical census matching, such "truth" does not exist. Instead these are the matches the researcher would be confident labeling as links or not. These decisions are clearly not infallible, but any algorithmic approach can (and should) approximate the best efforts of a trained researcher assessing matches.

Each record, $xx \in XX$ matches a record from $X1$ to a record in $X2$. In addition to first and last names and year of birth from both original datasets, I create a number of variables to describe the features of the potential match. In principle, this set of variables could differ across matching procedures with each new dataset. However, I found a relatively consistent set of variables to be necessary and sufficient to produce accurate matches. I list these variables, along with a brief description in Table 2. Many of these variables are link specific, such as the Jaro-Winkler distance in first name ($fdist$) or the absolute birth year distance in years ($ydist$). Others are constant for all records containing a given original record, $x1 \in X1$. For example,

[19] $XX_T$ should be stratified on unique values in $X1$ such that if one possible match from $X1$ to $X2$ is in $XX_T$ then all possible matches are in the training data.

[20] Christen (2012) defines this as one-to-one matching.

*hits* is the number of potential matches for a given record in $X1$ that we find in the entire $X2$ file. In addition to Jaro-Winkler string distances, which I use to compute both $fdist$ and $ldist$, I also use the Soundex system to generate another pair of important matching variables. These variables indicate whether or not the soundex scores for the first (or last) names match between the two possible records. Soundex groups (and other phonetic groupings such as NYSIIS and phonex) have been used in the past by economic historians and others performing record linkage. The scores attempt to encode the sound of a word, relying on the first letter and sound grouping for further letters.[21]

How were the variables in Table 2 chosen? For model selection, one could imagine using $k$-fold cross validation or another formal model selection process to pick variables from the space of all variables that might describe the similarity of two given records. These techniques are common to machine learning procedures, but less frequently used in the social sciences. While a more formal method will identify the variables that are most important to the classification procedure, such efforts may be unnecessary. Instead, I rely on my experience of constructing matches between several difference census datasets to specify the variables listed in Table 2. In the next section, when I walk through the example of matching sons from Iowa 1915 into the 1940 census, I show that the set of variables in Table 2 produces a classification method that is both accurate and efficient. More generally, there may be other variables that could improve the matching fit for other datasets and the procedure could be easily modified to include them.[22]

Once the variables describing each potential link have been constructed, we are ready to estimate the relative importance of these variables in determining links

[21] See the National Archives description of Soundex (`http://www.archives.gov/research/census/soundex.html`) for a more detailed description of the Soundex method. I use Soundex because it is commonly used in census linking, but also because there are prebuilt soundex functions in Stata and in the RecordLinkage package in R.

[22] Such variables may include alternative versions of string distance or phonetic classification of names. If linking records based on non-American data, language specific phonetic codes may be valuable. If the matching is into the 1930 Census, for example, mother and father state of birth might be a useful variable to include to improve match accuracy. In my experience, syllable-count comparisons, string length comparisons, and higher orders of the string distances above are not useful features to include but this may vary between datasets.

Table 2: Census Matching Variables

| Variable Names | Variable Description |
|---|---|
| | Record Variables |
| fname1 | First name of record in $X1$, including middle initial if available |
| fname2 | First name of record in $X2$, including middle initial if available |
| lname1 | Last name of record in $X1$ |
| lname2 | Last name of record in $X2$ |
| yob1 | Year of birth in $X1$ |
| yob2 | Year of birth in $X2$ |
| | Constructed Variables |
| fdist | Jaro-Winkler string distance between first names |
| ldist | Jaro-Winkler string distance between last names |
| ydist | Absolute value difference between year of birth in $X1$ and $X2$ |
| hits | Number of records in $X2$ matched for given $X1$ observation |
| hits2 | $hits^2$ |
| exact | Indicator if both $lname1 \equiv lname2$ and $fname1 \equiv fname2$ |
| exact.all | Indicator if $lname1 \equiv lname2$, $fname1 \equiv fname2$, and $yob1 \equiv yob2$ |
| f.start | Indicator if the first letter of the first names match |
| l.start | Indicator if the first letter of the last names match |
| f.end | Indicator if the last letter of the first names match |
| l.end | Indicator if the last letter of the last names match |
| mimatch | Indicator if the middle initials match |
| exact.mult | Indicator if more than one hit in $X2$ for a given record in $X1$ matches first and last names exactly |
| exact.all.mult | Indicator if more than one hit in $X2$ for a given record in $X1$ matches first name, last name, and year of birth exactly |
| fsoundex | Indicator if the soundex codes of the first names match |
| lsoundex | Indicator if the soundex codes of the last names match |

If there are any records in $X1$ with multiple exact matches in $X2$—that is exactly the same first name string, last name string, and year of birth—then we will be unable to pick between these possible matches. All possible matches are equally as likely to be the true match. Instead, we score any record links in $XX$ with failure if $exactmult1$ is not 0. Thus, the variable $exactmult1$ is not used directly in the prediction algorithm.

in the training data, $XX_T$ , and to assign links outside of our training data. The generation of links is a two step procedure. First, I note one special variable in Table 2, *exact.all.mult*, which indicates whether there are multiple hits in $X2$ which are exact matches on first name, last name, state of birth, and year of birth for a given record $x1 \in X1$. While somewhat unlikely, there are of course certain common names for whom this will occur. In most populous states in most year, for example, there are multiple men named John Smith. When *exact.all.mult* is 1, it should be clear that we cannot decide which of these exact matches is correct and we mark all of these potential matches are non-matches.

After removing the set of records with multiple exact matches, the second step is to run a probit model on the training data, $XX_T$. Using the probit, I calculate the probability a given record is a match.[23] Unlike other—more common—classification problems, there is a special feature of the census linking problem. Namely, if one record from $X1$ is coded as matched to a record in $X2$, then we do not want that record from $X1$ to be coded as matched to any other records in $X2$.[24] There are some variables generated and used in the matching procedure that account for this feature, including the total number of potential hits for a given $X1$ observation. However, the probit regression does not directly account for this fact. Instead, we use a second step to ensure records are not double-matched.

In the second step, we take the generated probit scores and define matches only as those records $xx \in XX$ that meet the three following criteria. First, the score for $xx$ is the highest score for that record in $X1$.[25] Second, the score of the match is sufficiently large (more on the parameter in the next section). Third, we require that the score of the second-best link is sufficiently small, relative to the top score, that we are confident that the best link is a match. Choosing these hyper-parameters or meta-parameters has an important role in the matching procedure and is discussed

[23] In section 5, I justify the use of a probit model rather than another machine learning classifier. The probits are both familiar to social scientists performing record linkage and do extremely well in out of sample prediction, minimizing both false positives and false negatives.

[24] The same is true in the reverse; two different records from $X1$ should not link to the same record in $X2$.

[25] That is, if "James Feigenbaum" is a record in $X1$, then the match in $XX$ could only be a match according to the final algorithm if it is the best link for $x1$.

in the next section.

This second stage is important in another way. For many common names in $X1$, there might be a variety of possible links in $X2$ that are all slightly different. For example, we might match Jonny to John and Jon, with exactly matched last names and years of birth. Alternatively, there may be many John Smiths matched to different years of birth. While the algorithm might prefer the match to one, if this preference is not sufficiently strong either absolutely or relatively, we should not be comfortable declaring a match. Rejecting some of these close matches will necessarily increase the rate of false negatives, but it will also decrease the false positive rate and replicate the manual matching procedure which should attempt to limit the close judgment calls. While false negatives will lower our final samples, the biases driven by false positives are likely to be quite problematic.

After running the second stage, we can limit $XX$ to the records that are coded as matches and proceed with analysis.[26] The match rate—the share of records in $X1$ which we are able to link to a record in $X2$—will vary across datasets. It also depends on the hyper-parameters, as shown in a different matching context by Mill (2012). In practice, I have found that the match rates from the procedure outlined here will be as good if not better than other methods traditionally used in economic history.

## 3 An Example of The Record Linking Procedure: Iowa 1915 to 1940

In Feigenbaum (2014), I measure intergenerational mobility of income, linking fathers from the 1915 Iowa State Census to their sons in the 1940 Federal Census. I utilize the Iowa State Census sample digitized by Claudia Goldin and Lawrence Katz for their work on the historical returns to education (Goldin and Katz 2000,

[26] In the interest of speeding up the matching process and conserving memory, $XX$ likely does not include any variables that we might want to use in analysis, like income, occupation, education, etc. The true final stage is simply locating the chosen records in $X1$ and $X2$ and bringing together all the variables of interest.

2008). To construct my sample for census matching, I limit the Goldin-Katz sample to families with boys aged between 3 and 17 in 1915. These sons will be between 28 and 42 when I observe them again in 1940. Of course, I restrict my analysis to sons in 1915, because name changes make it impossible to locate most daughters in the 1940 Census. This leaves me with a sample of 7,580 boys, each of whom is a son in 1915 Iowa. For each son, I know his first and last name, age in 1915, and state of birth—exactly the variables I need to match into the 1940 census.

As described above, I start by extracting, for each son, the set of matches in the 1940 census with a year of birth distance of less than 3, a first name string distance of less than .2, a last name string distance of less than .2, and a matching state of birth. This returns a dataset, $XX$, of 79,047 records. Only 6,889 of the 7,580 boys are in this sample, suggesting that there are nearly 700 sons for whom no possible matches can be found in the 1940 census.[27] On average, each of these 6,889 sons is matched to 11.4 records in 1940. However, the distribution of potential matches is quite skewed. 1,005 of the sons return only one possible match and the median is 6 matches; 37 sons return more than 100 matches.[28]

With this dataset, I then construct an indicator variable for matched and non-matched records. For the purposes of demonstrating the procedure in this paper, I do this for the entire sample. For every son in the Iowa 1915 sample, I indicate which link, if any, in 1940 is the correct match. The rest of the links are indicated as incorrect non-matches. Based on both my own speed and the speed several research assistants with experience assessing linked records, in an hour, 500 records in $X1$ could be assessed. Of course, constructing all of these links for every observation in the full sample obviates one reason for an algorithmic approach; the algorithm allows us to match large datasets without the time or monetary costs of manually matching. But it is necessary to have "correct" classifications for all observations

[27] These sons could no longer be living by 1940 or they may have moved out of the country or, perhaps more likely, the measurement error in recording their names, ages, or state of birth in either 1915 or 1940 was sufficiently severe that they cannot be matched at all.

[28] John Barile, born in Pennsylvania in 1909 returns 364 possible matches. John is a very common name, Barile is very close in string distance to a large number of names (Bradley, Bailey, Barnes, Barrett, and Riley), and Pennsylvania had a large state population in 1909.

to properly build the matching model in this paper and assess its accuracy.

I then randomly partition the data into two equal parts, sampling at the $x1$ record level so as to keep all possible matches for a given son in Iowa in the same data partition. One partition will be the training data, where I will fit my probit model. The other partition will be held back to test the accuracy of my algorithm out of sample.[29]

There are many possible variables that could describe how likely a given matched pair of records are to be a true match. I focus first on the set of variables listed in Table 2. As described previously, I run a probit regression on the dummy variable indicating whether or not the record was declared a match or not-match in the human review process on these variables. I rely on the `caret` package in R, a standard package used in machine learning to train my probit model. The model is trained with bootstrap sampling from the training data, taking 25 samples to estimate the final probit coefficients.[30]

The results of this regression are presented in Table 3. I show both the probit that I use and an alternative logit model. The parameters presented are the direct model coefficients, not marginal effects as is common in a probit model.

Using the coefficients estimated in the probit in Table 3, I calculate the predicted score. These scores run from 0 to 1; higher values suggest a stronger likelihood of the records being true matches.[31] Following the method described earlier, I define matches only as those records that meet the three following criteria. First, for a given son in the Iowa 1915 sample, the score is the highest score of all matches. Second, the score of the match is sufficiently large. Third, that the score of the second-best link for the Iowa 1915 son is sufficiently small, relative to the top score, that we are confident that the best link is a match.

Clearly, at this point we cannot proceed unless we define the two meta-parameters referenced above: the absolute threshold for declaring a match and the relative

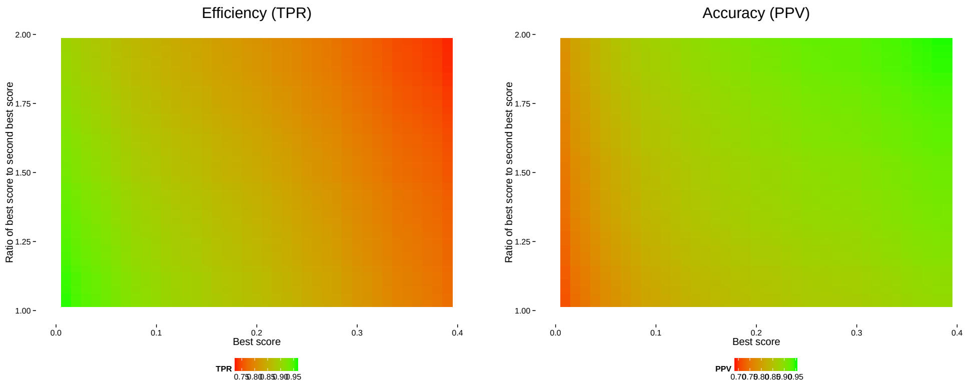
[29] In section 4, I show that in fact, training on 50% of the data is overkill and that the method yields accurate predicted matches using a much smaller share of data for calibration.

[30] The model parameters are largely the same when using k-fold cross validation to choose parameters or simply running one probit regression on the full sample of training data.

[31] As these scores are based on predictions from a probit model, the scores are simply $\Phi(X'\hat{\beta})$.

Table 3: Iowa 1915 to 1940 Census Linking Model

| | Probit | Logit |
|---|---|---|
| First and Last name match | $0.632^{***}$ | $1.129^{***}$ |
| | (0.086) | (0.168) |
| First name distance, Jaro-Winkler | $-6.071^{***}$ | $-11.543^{***}$ |
| | (0.525) | (0.994) |
| Last name distance, Jaro-Winkler | $-10.285^{***}$ | $-19.145^{***}$ |
| | (0.487) | (0.954) |
| Absolute Value Difference in Year of Birth is 1 | $-0.708^{***}$ | $-1.308^{***}$ |
| | (0.044) | (0.083) |
| Absolute Value Difference in Year of Birth is 2 | $-1.562^{***}$ | $-2.893^{***}$ |
| | (0.065) | (0.126) |
| Absolute Value Difference in Year of Birth is 3 | $-2.316^{***}$ | $-4.370^{***}$ |
| | (0.102) | (0.208) |
| First name Soundex match | $0.153^{***}$ | $0.294^{***}$ |
| | (0.054) | (0.100) |
| Last name Soundex match | $0.698^{***}$ | $1.341^{***}$ |
| | (0.069) | (0.135) |
| Hits | $-0.064^{***}$ | $-0.123^{***}$ |
| | (0.002) | (0.005) |
| Hits-squared | $0.0003^{***}$ | $0.001^{***}$ |
| | (0.00002) | (0.00004) |
| More than one match for first and last name | $-1.690^{***}$ | $-3.217^{***}$ |
| | (0.093) | (0.183) |
| First letter of first name matches | $0.871^{***}$ | $1.593^{***}$ |
| | (0.130) | (0.245) |
| First letter of last name matches | $0.886^{***}$ | $2.003^{***}$ |
| | (0.148) | (0.356) |
| Last letter of first name matches | $0.147^{***}$ | $0.312^{***}$ |
| | (0.053) | (0.101) |
| Last letter of last name matches | $0.649^{***}$ | $1.239^{***}$ |
| | (0.070) | (0.139) |
| Middle Initial matches, if there is a middle initial | $0.537^{***}$ | $0.908^{***}$ |
| | (0.097) | (0.186) |
| Constant | $-1.479^{***}$ | $-3.087^{***}$ |
| | (0.225) | (0.480) |
| Observations | 38,091 | 38,091 |
| Log Likelihood | −2,440.877 | −2,444.649 |
| Akaike Inf. Crit. | 4,915.753 | 4,923.298 |



(a) TPR Values (Share of Type I Errors) (b) PPV Values (Share of Type II Errors)

Figure 3: Match Algorithm Quality and Hyper-Parameter Values

threshold for declaring a match better than the next best alternative. How to pick these meta-parameters? I attempt to minimize false positives and false negatives. To make this concrete, I use two standard machine learning assessment measures, the true positive rate (TPR) and the positive prediction rate (PPR). The TPR is the ratio of true positive to total positive matches in our training data. The PPR is the ratio of the true positives to the total matches made by the algorithm.

Call the first meta-parameter $b_1$ and the second $b_2$. $b_1$ will take a value between 0 and 1, matching the range of our scores. In theory, $b_2$ should range from 1 to $\infty$ (it is the ratio of two scores between 0 and 1). I search over the grid of possible values of $b_1$ and $b_2$ for the maximums of TPR and PPV, calculated from the same training data we used to generate the probit coefficients previously. Figure 3 presents a graphical representation of these grid searches. As we change parameters to increase PPV, TPR tends to fall. This makes sense: the more restrictive we are on matching matches, the fewer false matches we will find, but fewer true matches will be found as well. One solution is to maximize some utility function with two arguments, PPV and TPR. The simplest function would be a weighted sum of the two values. Choosing weights is subjective and may vary between projects, depending on the costs of including false positives relative to the benefits of finding more true positives. In Table 4, I show the optimal hyper-parameters under various weighting schemes. In the analysis that follows, I use a weight of 1 and thus select $b_1 = 0.14$ and $b_2 =$

Table 4: Table Summary

| | Hyper-Parameters | | Algorithm Quality | |
|---|---|---|---|---|
| Relative Weight on PPV | b1 | b2 | PPV | TPR |
| 0.500 | 0.050 | 1.125 | 0.784 | 0.931 |
| 0.750 | 0.090 | 1.300 | 0.831 | 0.897 |
| 1 | 0.140 | 1.375 | 0.858 | 0.875 |
| 1.250 | 0.200 | 1.975 | 0.912 | 0.815 |
| 1.500 | 0.200 | 1.975 | 0.912 | 0.815 |

The utility function in PPV and TPR is defined as TPR + $\gamma$PPV where $\gamma$ is the relative weight on PPV in column 1. The algorithm quality metrics, PPV and TPR, are calculated with respect to the training partition of the data.

1.375.

With these hyper-parameters selected, I can return to my matched dataset, identify the links (and the non-links) and proceed to my other analysis. We can see from Table 4 that the PPV is 85.8% and the TPR is 87.5%. But these are the in-sample results and reflect only how well my model is tuned to the data I built it with. How does the algorithm do out of sample, predicting matches and non-matches on the data that I held back from the training? To test this, I create predictions on the test set, using the probit coefficients listed in Table 3 and the hyperparameters $b_1 = 0.14$ and $b_2 = 1.375$. Table 5 presents a confusion matrix, listing the counts of true and false positives and the true and false negatives, as well as the PPV and TPRs from applying the algorithm to the test data. This out-of-sample prediction is a much more difficult test for the record linking algorithm than the in-sample predictions made in Table 4. However, the algorithm does extremely well. The TPR is 86.4%, less than one percentage point worse on the test data relative to the training data. The PPV is 87.3% which is, in fact, slightly higher than in the training data. The interpretation is that of the matches identified by the algorithm, 87.3% were coded by RAs as a match and that of the matches coded by RAs, 86.4% were identified by the matching algorithm.

Rather than using the TPR and PPV, economic historians undertaking matching procedures typically judge a link between two censuses on its matching rate.

Table 5: Confusion Matrix: Out of Sample Predictions

| | True Status | |
|---|---|---|
| Algorithm Prediction | Not Matched | Matched |
| Not Matched | 35,608 | 295 |
| Matched | 272 | 1,869 |

The True Positive Rate is the number of true positives over the total number of matches, which is the sum of the true positives and the false negatives. This rate indicates the efficiency of the algorithm: how many of the matches in the full data were identified by the algorithm. The out-of-sample TPR is 86.4%.

The Positive Predictive Value is the number of true positives over the total number of positives, which is the sum of the true positives and the false positives. This rate indicates the accuracy of the algorithm: how many of the matches made by the algorithm are in fact matches in the true data. The out-of-sample PPV is 87.3%.

Matching rates vary in the literature between different samples and procedures, but are often between 20% and 50%. Of the original 7,580 boys in my Iowa sample, I match 4,349 uniquely and accurately with my matching algorithm, for a match rate of 57.37%. While the algorithm is far from perfect—neither the PPV nor TPR hits 100%, nor does the match rate—the improvements are large. For samples that begin relatively small, an increase in the match rate would be even more important to the potential analysis.

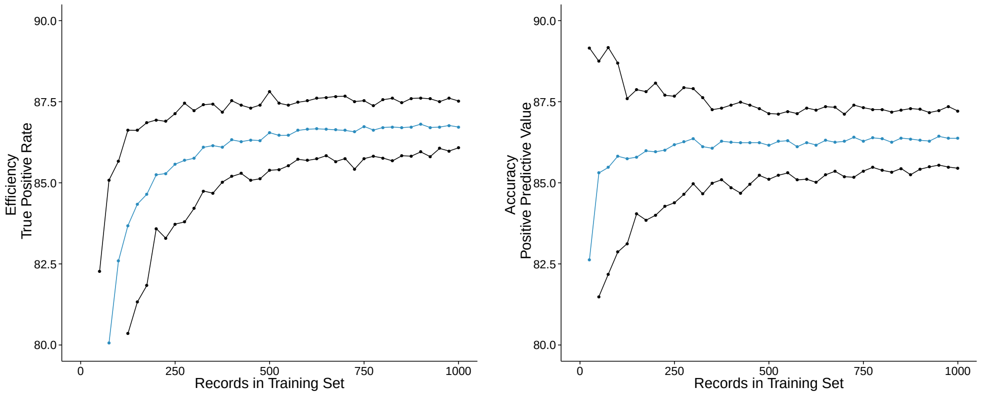
## 4 How Much Training Data is Enough?

In Table 3, I estimated a set of coefficient values that do well in locating matches between the Iowa 1915 State Census and the 1940 Federal Census. The coefficients were estimated using only half of the full dataset, the training data. In Table 5, I evaluated the predictions on the other half of the data—data that had not been used to tune the parameter. This suggested that the record linking procedure and coefficients were very able to identify matches out of sample. In future matching projects, for example between the 1910 and 1930 censuses, one could simply calculate the variables described in Table 2 and apply the coefficients to an entirely new sample. Given the good out-of-sample performance of the algorithm, this is likely to produce reasonably good matches. However, one might also create training data based on a subsample of the *new* data in question and follow the steps outlined above to generate a new project specific matching algorithm and hyper-parameters. Naturally, there are benefits and costs to either choice. Using the matching algorithm presented here will save time and research funds because no manual training data will be necessary. The major cost is common to any use of out-of-sample predictions: less accuracy. This cost is likely to be especially high when matching between datasets that are very different than the two I built the algorithm with in the first place. In this case, out of sample has two very different meanings. The algorithm was tested on data held back from its training, but it was not tested on data that was of the same type as the training data but built from a wholly different set of censuses.

If a researcher is going to use training data specific to the datasets to be used, one important practical question is how big should the training data be? How many records need to be manually adjudicated? The traditional empirical adage that more data is usually better applies here, but I suggest a more exact answer. To do this, I randomly subset the data used in the previous section, train a match algorithm, and test how much the algorithm changes and, more importantly, how much the measures of accuracy and efficiency change as the subset of data used for training grows.

The procedure is a straightforward cross-validation exercise. I start by drawing a $\pi$ random sample of my data, blocking at the unique $X1$ observation level.[32] This draw will be the training data for a given iteration. I then train the match algorithm on my training data and apply the algorithm to the test data—here the test data is the $N - \pi$ of the data not sampled. Normally, it would be impossible to measure the error rates on the non-training sample. However, because in reality I have matches for the entire sample, I can compare the predictions based on the $\pi$ sample with the test data and record a measure of efficiency (TPR) and accuracy (PPV). I draw 100 random samples for each $\pi$-sized block of my data from $\pi = 25$ to $\pi = 1000$,

[32] That is, I include all possible links for a given record in the $X1$ data to the $X2$ data.



(a) True Positive Rate

(b) Positive Prediction Value

Figure 4: Algorithm Efficiency and Accuracy increases as the size of the training data set increases, but most gains are realized with no more than 500 records of the data used in training.

by 25. I graph the resulting TPR and PPV in Figure 4, along with 95% confidence intervals.

The speed of convergence to the full-sample results is quite rapid. With around 500 records trained—on average, less than 10% of the original $X1$ data—the algorithm identifies links with a TPR of nearly 87% and a PPV of 86%. These TPR and PPV results are approximately as large as the corresponding rates when the algorithm had 50% or more of full data to train with. This implies that, in fact, the answer to "how much training data is required?" is quite small relative to the overall size of the project.

How long does it take to create training data based on 1400 or so observations? Based on records from my Iowa matching project, RAs can assess the links for about 500 $X1$ records per hour. Thus, in less than 3 hours of work (6 with double entry), training data can be constructed which—once used in the matching procedure outlined above—generate a final matched sample with a nearly 90% efficiency and match accuracy.

# 5 Machine Learning Method Selection

I use a probit model to assign scores to each potential match. However, there are many different classification techniques in the machine learning toolbox. Why probit? For one, probit regression is a well-understood concept among economic historians and other social scientists who might want to create census links. Beyond familiarity, I have found that the probit model, while perhaps simple when compared to newer methods in machine learning, performs extremely well out of sample. In this section, I compare the cross validation matching results from the probit model to two other common machine learning classifiers—random forests and support vector machines—as well as methods commonly used by economic historians. As I will show, my probit-based method makes an ideal choice to use in record linking procedures.

I use cross validation to assess the various record linkage procedures on my Iowa 1915 sons data. Because I have fully hand-linked this sample ahead to the 1940 census, I am able to apply the predictions from each model to the held-out testing data and assess the accuracy of each, based on both the PPV and the TPR. I begin by splitting the set of potential matches in the 1940 census for the sons of the Iowa 1915 census into a training and testing set. Each set contains 50% of the sons from 1915. I sample at the son level, rather than the potential match level, because matches are only identified relative to other potential matches for a given son. I use the same set of independent variables or features in each model, those listed in Table 2.[33]

I find the probit classifier to be the best regression method for record linkage. In addition to probit, I also assess a logit and OLS classifier. All three of these models are built in the same way. In the first stage, I regress an indicator variable for match or not match on the linkage features. In the second stage, I declare matches based on the absolute and relative scores from the models. As I document in the first three rows of Table 6, both the probit and logit models have strong

[33] While feature discovery is relatively standard in the machine learning literature, it is much less frequently used in the social sciences. I follow this convention and specify the features in advance.

Table 6: Comparing Matching Algorithms

| | Training Data | | Testing Data | |
|---|---|---|---|---|
| Model | Efficiency (TPR) | Accuracy (PPV) | Efficiency (TPR) | Accuracy (PPV) |
| My Method | 0.881 | 0.854 | 0.879 | 0.858 |
| Logit | 0.874 | 0.861 | 0.873 | 0.862 |
| OLS | 0.884 | 0.792 | 0.856 | 0.792 |
| Random Forest | 0.973 | 0.947 | 0.756 | 0.868 |
| SVM | 0.851 | 0.904 | 0.827 | 0.891 |
| Exact | 0.192 | 0.830 | 0.195 | 0.858 |
| Exact (yob ± 3) | 0.292 | 0.861 | 0.295 | 0.883 |
| Soundex | 0.631 | 0.782 | 0.601 | 0.766 |

All models fit on the same training data with the same set of features described in Table 2. Training data is a 50% random sample of the full Iowa sons linked data from 1915 to 1940. Hyper-parameters are optimized using an equal weighting scheme on both the TPR and PPV in the training data.

accuracy and efficiency on the testing data.[34] The OLS model, not surprisingly given the dichotomous nature of the outcome, is slightly worse on both efficiency and accuracy.

Many machine learning classification or discrimination tasks are solved not with regression models but with more flexible, non-parametric solutions. I assess both random forests and support vector machines (SVMs) in the third and fourth rows of Table 6. While both do quite well on the training data, the accuracy and efficiency of both are no better than my main probit method on the testing data which the models were not trained on.

Random forest classifiers may be best thought of as an advancement of simple decision trees. In the record linkage case, a decision tree could be used to split records into matches and non-matches. At the first node, the tree might split at some critical value of first name string distance and at the second on a critical value of last name distance. Later nodes could split based on matching first or last name soundex results, whether or not middle initials match, etc. In Figure 5, I show one possible decision tree with 2 levels. At the first node, the records are split on

[34] The training results for the probit model replicate the results from Table 4. For each model in Table 6, I have chosen the hyper-parameters to maximize the sum of the TPR and the PPV.

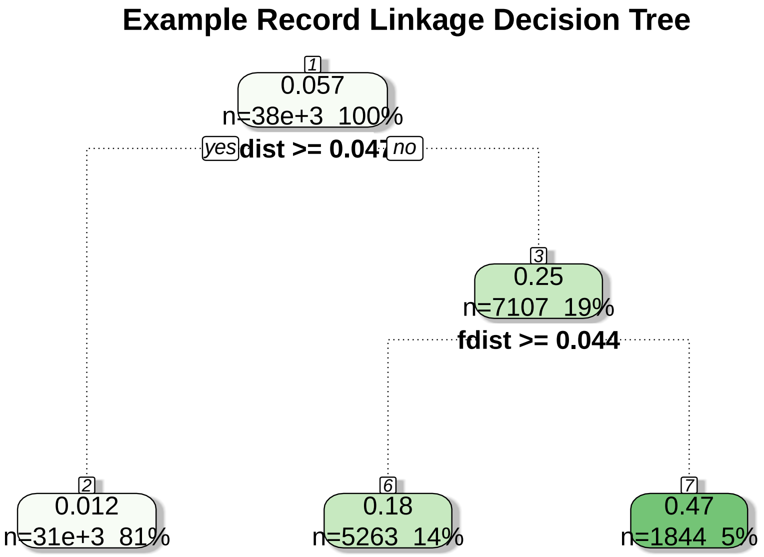


Figure 5: Example Decision Tree based on first and last name Jaro-Winkler String Distance

the Jaro-Winkler string distance in last name. Records with a last name distance of more than 0.047 are sent in one direction (and ultimately classified as non-matches), while records with relatively close last names ($\leq 0.047$) go to a second node. At this second node, records are classified based on the Jaro-Winkler string distance in first name. Records with small first name distances are coded as matches and the rest are coded as non-matches. For such a painfully simple model, the tree does reasonably well. Of the 1844 records in the matched node, 47% are matches; the non-match nodes are reasonably accurate as well.[35] However, this was only one of the many trees that could be grown to classify the census data. Further, this tree has only two levels of decisions—it might be possible to extract more precision by using additional variables.

Random forests consist of the averaging of many different decision trees grown to much greater depth than the tree in Figure 5. The randomness is introduced in two ways. First, the trees are trained on bootstrapped samples of the training

[35] Obviously, I have yet to utilize the special one-to-one matching feature of my data and these results are not directly comparable to those presented previously.

data. This induces natural variation in both the optimal splitting variables and in the critical values to split at. Second, the features available to split at each level are a random subset of all features in the data.[36] Within this sample of the training data and with these feature constraints at each node, the optimal decision tree is tuned. By averaging over each tree's prediction, the random forest algorithm can generate a classification or a score. In my case, this is a score from 0 to 1 and can be interpreted as the likelihood that a given record is in fact a match. As with the probit model, I then compare the scores for a record in $X1$ across all possible matches and identify as matched only those records that are the highest score at the $X1$ level, sufficiently large, and sufficiently larger than the next best record.[37]

Support vector machines (SVMs) search for the optimal separating hyperplane between two groups. Each feature in my dataset—the first and last name string distances, the year of birth difference, the soundex score indicator, etc—is a dimension. Again, the SVM returns a score for each possible match of a given record in $X1$ and I use the same second step to identify the matches that are sufficiently strong and relatively strong.

In the fourth and fifth rows of Table 6, I give the PPV and TPR in the training data and the testing data for the Random Forest classifier and the SVM. The probit beats both the random forest classifier and the SVM on the test data. First, it is quite clear that the random forest out-performs the probit on the training data. The random forest scores a TPR of 97%, suggesting that the model locates nearly all of the matches identified by a research assistant. The PPV is similarly high at nearly 95%—very few of the records suggested by the algorithm were incorrect. The SVM is comparable to the probit model on the training data, with less efficiency but more accuracy.

However, the results on the testing data are much less promising for the random

[36] Where $p$ is the number of features or independent variables, usually only $\sqrt{p}$ or $p/3$ features are considered at each node. This avoids high correlation across trees, which might arise if certain features are very powerful predictors.

[37] As before, the metaparameters used to distinguish how large of a score is needed, both absolutely and relatively, to be a match are generated by maximizing the sum of the TPR and the PPV in the training data. The metaparameters for the random forest are different from those of the probit or logit models.

forest method. Again, the testing data is the half of the full sample that was held back during training of the models and is an accurate assessment of the various model's out of sample properties. The random forest model has a TPR on the test data of only 75.6%. Thus, many of the actual matches coded manually are not identified by the algorithm. The PPV for the random forest model is similar the the PPV for the probit and logit models. Thus, the random forest is no more or less accurate than the probit model, but it is much less efficient, tagging many true matches as non-matched.

Finally, I compare my machine learning approach to a few standards from the economic history literature.[38] In all cases, I require state of birth to match exactly. The exact model requires matches to agree exactly on first name, last name, and year of birth. Not surprisingly, these method is much less efficient than the other models presented. Good matches that may be off because of a small transcription error will not be matched by the exact method. In addition, I find the accuracy of the exact procedure to be no better than my main method. I realize the exact match criteria slightly in the seventh row of Table 6: records can still only be declared matches if the first and last names match exactly, but I now allow a three year window for year of birth, accounting for small errors in age in the census. While this flexibility improves the efficiency somewhat, it is still far below the machine learning methods, suggesting that many matches that would be made by hand are not included.

How does my machine learning approach compare to a linking procedure based on phonetic codes, as is used by Long and Ferrie (2007), Collins and Wanamaker (2015), and others? In the final row of Table 6, I apply a soundex model to my Iowa data and report the results. The method requires that candidate links agree on both soundex of first names and soundex of last names. Within that set, if one record matches on year of birth exactly, it will be declared a match. If no such record exists, the year of birth error grows to ±1, then ±2, and then ±3, as is standard. While the soundex model outperforms exact matching, it is significantly worse on both

[38] Because none of these models are trained, the efficiency and accuracy metrics are roughly the same on the training data and the testing data.

efficiency and accuracy than my machine learning method. The efficiency loss is straightforward: any names with transcription errors in the first few characters will have different soundex codes.[39] The accuracy loss is likely driven by the same sorts of problems: if the correct match drops out because of early string transcription errors, another match may be just good enough to take its place.

The machine learning approach outperforms the standard record linkage in the economic history literature. Ultimately, this is not surprising: the machine learning approach embeds the other methods, but also increases the degrees of freedom. That is, if the name strings and years of birth match exactly or if the soundex scores match, that will increase the match score. But, rather than putting all the decision weight on just a few features, the machine learning approach allows the training data (and implicitly the researcher) to determine the weights over a large feature set.

Based on the results presented in Table 6 and the simplicity of implementing a probit classifier[40]—I suggest using the probit model when implementing the record linkage procedure outlined in this paper.

## 6 Does the Algorithm See Dead People?

Does my record linking procedure making incorrect links? One particularly troublesome error would be linking the record of someone who has died in between the collection of the two datasets. However, it is not usually possible to test for these sorts of bad matches; of course, the 1900 census does not include information on when (in the relative future) respondents will die. Instead, I turn to another source of records, the Lahman baseball encyclopedia, that includes both name, state of birth, and year of birth information, as well as externally validated date of death data. When linking the universe of Major League Baseball players ahead from 1903 to 1940, I find a "ghost matching" rate more than 45 percentage points lower than

[39] For example, I have seen "Feigenbaum" mistranscribed in the 1910 census as "Teigenbaum" likely because the cursive F and T are so similar looking.

[40] Easily done in Stata, R, etc. Random forest packages exist for these statistical programs and for others but are not often used by economists.

the live matching rate in the same sample, suggesting that my algorithm is usually able to avoid erroneous matches.

Constructed by journalist and sabermetrician Sean Lahman, the Lahman (2016) database records statistics for baseball players from 1871 to 2015. In addition to batting average, on-base percentage, and ERA, the database also includes biographical information for the more than 18 thousand players in Major League Baseball history. In particular, I draw on information about players' names, dates of birth, place of birth, and date of death that enable me to link players to the census.

I link the 381 active, American-born players in the 1903 season to the 1940 census. Of the 381 players, only 220 were still alive in 1940, according to the Lahman database. I am able to find 52% of these players in the 1940 census, using my algorithm.[41] The match rate is far smaller for the 161 players no longer alive in 1940: the algorithm has a ghost matching rate of only 6%. While this small handfull of matches are incorrect, the low number—both absolutely and relative to the live players—underscores the quality of the algorithm to make links between datasets, even when evaluated on a feature or metric not available during training.

## 7 Conclusion

In this paper, I detail a transparent census matching technique for constructing linked samples that can be replicated across a variety of cases. The procedure applies insights from machine learning classification and text comparison to the well known problem of record linkage, but with a focus on the sorts of costs and benefits of working with historical data. Using tools readily available to economists and other social scientists in Stata or R, the method can automate the linking of records with minimal manual matching work and high levels of efficiency and match accuracy.

[41] Because the number of players overall is so small, I do not train a new record linking algorithm. Instead, I use the algorithm trained with the Iowa 1915 sons, described above, and the weights shown in Table 3.

# References

Aizer, Anna, S Eli, Joseph P Ferrie, and Adriana Lleras-Muney. 2014. "The Long Term Impact of Cash Transfers to Poor Families."

Chetty, Raj, Nathaniel Hendren, Patrick Kline, Emmanuel Saez, and Nicholas Turner. 2014. "Is the United States Still a Land of Opportunity? Recent Trends in Intergenerational Mobility." *American Economic Review: Papers & Proceedings* 104:141–147.

Christen, P. 2012. *Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection*.

Collins, William J. and Marianne H. Wanamaker. 2014. "Selection and Economic Gains in the Great Migration of African Americans: New Evidence from Linked Census Data." *American Economic Journal: Applied Economics* 6:220–252.

Collins, William J. and Marianne H Wanamaker. 2015. "The Great Migration in Black and White: New Evidence on the Selection and Sorting of Southern Migrants." *The Journal of Economic History* 75:947–992.

Feigenbaum, James J. 2014. "A New Old Measure of Intergenerational Mobility: Iowa 1915 to 1940."

Ferrie, Joseph P. 1996. "A New Sample of Americans Linked from the 1850 Public Use Micro Sample of the Federal Census of Population to the 1860 Federal Census Manuscript." *Historical Methods* 29:141–156.

Goeken, Ron, Lap Huynh, Thomas Lenius, and Rebecca Vick. 2011. "New Methods of Census Record Linking." *Historical methods* 44:7–14.

Goldin, Claudia and Lawrence F. Katz. 2000. "Education and Income in the Early Twentieth Century: Evidence from the Prairies." *Journal of Economic History* 60:782–818.

Goldin, Claudia and Lawrence F. Katz. 2008. *The Race Between Education and Technology*. Cambridge, MA: The Belknap Press of Harvard University Press.

Lahman, Sean. 2016. "The Lahman Baseball Database."

Long, Jason and Joseph P Ferrie. 2007. "'Everything in Common ... [But] Language'?: Intergenerational Occupational Mobility in Britain and the U.S. Since 1850?"

Mill, Roy. 2012. "Assessing Individual-Level Record Linkage between Historical Datasets."

Mill, Roy and Luke C D Stein. 2012. "Race, Skin Color, and Economic Outcomes in Early Twentieth-Century America."

Ruggles, Steven, J. Trent Alexander, Katie Genadek, Ronald Goeken, Matthew B. Schroeder, and Matthew Sobek. 2010. *Integrated Public Use Microdata Series: Version 5.0 [Machine-readable database]*. Minneapolis, MN: Minnesota Population Center [producer and distributor].

Thernstrom, Stephan. 1964. *Poverty and progress; social mobility in a nineteenth century city*. Cambridge, MA: Harvard University Press.

Thernstrom, Stephan. 1973. *The other Bostonians : poverty and progress in the American metropolis, 1880-1970*. Cambridge, MA: Harvard University Press.

Winkler, WE. 1994. "Advanced methods for record linkage." .