Beyond Probabilistic Record Linkage: Using Neural Networks and Complex Features to Improve Genealogical Record Linkage

D. Randall Wilson

Abstract— **Probabilistic record linkage has been used for many years in a variety of industries, including medical, government, private sector and research groups. The formulas used for probabilistic record linkage have been recognized by some as being equivalent to the naïve Bayes classifier. While this method can produce useful results, it is not difficult to improve accuracy by using one of a host of other machine learning or neural network algorithms. Even a simple single-layer perceptron tends to outperform the naïve Bayes classifier—and thus traditional probabilistic record linkage methods—by a substantial margin. Furthermore, many record linkage system use simple field comparisons rather than more complex features, partially due to the limits of the probabilistic formulas they use. This paper presents an overview of probabilistic record linkage, shows how to cast it in machine learning terms, and then shows that it is equivalent to a naïve Bayes classifier. It then discusses how to use more complex features than simple field comparisons, and shows how probabilistic record linkage formulas can be modified to handle this. Finally, it demonstrates a huge improvement in accuracy through the use of neural networks and higher-level matching features, compared to traditional probabilistic record linkage on a large (80,000 pair) set of labeled pairs of genealogical records used by FamilySearch.org.**

I. Record Linkage Introduction

RECORD LINKAGE is a term first coined by H. L. Dunn [1] in the medical field. Record linkage is the process of identifying pairs of records that refer to the same thing. In most cases, each record refers to a person, and the challenge is to identify which records refer to the same real person. This challenge arises in many different areas, including companies trying to avoid sending mail to the same person multiple times; hospitals trying to track data on the same patient across several visits; and many other situations.

FamilySearch uses record linkage for *family history*, also called *genealogy*, in which people attempt to discover who their ancestors and other relatives are. It recently faced the task of having to find duplicate records in a billion-person database, and continues to work hard to examine incoming and existing records in its system to identify multiple records that refer to the same real person.

In each record linkage situation, the available records have certain *data fields* that contain information about each individual. In many applications, data fields may include such items as name, address, zip code, phone number, and so forth. In genealogy, data fields include names; dates and places for events such as birth, christening, marriage, death or burial; and names and events of relatives such as parents, spouses and children. It can also make use of information about what type of record it is and where it came from (for example, two individuals listed as a child in different original birth certificates are not usually the same person, although the parents in those birth certificates could be the same).

Data values often have *variation*, even among records that refer to the same real person. Names can have variations due to things like nicknames ("Bob" vs. "Robert"), married names vs. maiden names ("Elizabeth Turner" vs. "Elizabeth Smith"), spelling variations ("Elizabeth" vs. "Elisabeth"), initials ("John Henry" vs. "John H."), typographical errors, illegible handwriting, and so on.

Dates can vary due to formatting differences ("12 Jun 1850", "6/12/1850", "1850.12.6"), estimates ("1850", "about 1848"), typos ("1701" vs. "1710"), or even calendar changes (the beginning of the year moved from March to January in 1752, for example, and not everyone made the change at the same time). The same place is often spelled differently, abbreviated in various ways, or can be subject to boundary or name changes over time (after all, "Istanbul was Constantinople").

Often *normalization* is done to convert fields to lower case, remove or standardize punctuation, put dates and places in a consistent format, etc. It is even possible to use name and place catalogs to get standard values or ids that can be used for record comparison. For example, "Berkeley, CA" and "Berkeley, Alameda, Calif." could be looked up in a catalog to determine that both refer to "Berkeley, Alameda, California, United States." Such normalization (or *standardization*) helps avoid false disagreements in the data.

On the other hand, many different people can have the same name, and many different real people are born in the same place, or can have other fields that agree. So although fields tend to agree for *matching* pairs of records (i.e., records that refer to the same real person), it is also possible for them to disagree on matching pairs and it is certainly possible for some fields to agree on *differing* pairs of records (i.e., on records that do not refer to the same real person).

In order to separate the matching record pairs from the differing ones, it is important to make the best use of as much information as is reasonably possible.

Most record linkage systems have traditionally used simple field comparisons as the basis for classification. However, record linkage algorithms can take advantage of higher-level *features* that take into account not just whether

Manuscript received December 16, 2010.

D. Randall Wilson is with FamilySearch, 50 East North Temple, Salt Lake City, Utah. Tel. 801-240-9020; e-mail: *wilsonr@familysearch.org*.

978-1-4244-9636-5/11/$26.00 ©2011 IEEE

two values in a field agree, but *how well* they agree, or any other higher-level logic, such as "did one person die before the other one was born," or "do the two people have children born too close together or too far apart to be reasonable."

One challenge of record linkage, then, is to find a list of features (including simple normalized field comparisons) that can be useful in determining whether two records refer to the same person. Once the features have been found, the next step is to find the best way to use these features to give a score to a pair of records that reflects how likely it is that the two records represent the same person.

This paper reviews *probabilistic record linkage*, which is commonly used in the industry. It then shows that this method is equivalent to the *naïve Bayes classifier* in machine learning. The paper then discusses the use of more powerful features than simple field comparisons, and concludes with empirical results showing a dramatic reduction in error rates by using complex features coupled with neural network training, when compared with traditional probabilistic record linkage.

I. PROBABILISTIC RECORD LINKAGE

The most common algorithm for record linkage has traditionally been the statistical *probabilistic record linkage* formulas, as set forth by Howard Newcombe et al. [2, 3], and formalized by Fillegi and Sunter [4].

Let M be a set of *matched* record pairs (that both represent the same real person), and U be a set of *differing* (also called *unmatched*) record pairs (that represent two different people). If there are n data fields, then two *field agreement probabilities*, called the m-probability and the u-probability, can be defined for each data field i, with $i = 1..n$, as follows.

m_i = P(field i agrees on a matched pair) = $a_{m,i} / c_{m,i}$
u_i = P(field i agrees on a differing pair) = $a_{u,i} / c_{u,i}$

Similarly, the *field disagreement probabilities* can be defined as

m'_i = P(field i *disagrees* on a *matched* pair) = $d_{m,i} / c_{m,i}$
u'_i = P(field i *disagrees* on a *differing* pair) = $d_{u,i} / c_{u,i}$

where in both cases,

$a_{m,i}$ = number of matched pairs that agree on field i
$d_{m,i}$ = number of matched pairs that disagree on field i
$c_{m,i} = a_{m,i} + d_{m,i}$
$a_{u,i}$ = number of differing pairs that agree on field i
$d_{u,i}$ = number of differing pairs that disagree on field i
$c_{u,i} = a_{u,i} + d_{u,i}$

In much of the record linkage literature, it is assumed that $m'_i = (1 - m_i)$ and $u'_i = (1 - u_i)$. However, note that $c_{m,i}$ and $c_{u,i}$ could be less than $|M|$ or $|U|$, respectively, because often records are missing data for a given field, in which case the field neither agrees nor disagrees. In practice, therefore, we leave pairs of records out of the probability calculations for a field when either or both of the records are missing data for that field.

To compute a *score* for a given pair of records, a weight is added for each field. If the two records agree on the field, an *agreement weight* is added. If they disagree, a *disagreement weight* is added. If one or both records have no data for the field, then neither weight is added.

These weights can be computed as follows. The weight for agreement on field i is

$$w_{a,i} = \ln(m_i / u_i) = \ln(m_i) - \ln(u_i)$$

and the weight for disagreement on field i is

$$w_{d,i} = \ln(m'_i / u'_i) = \ln(m'_i) - \ln(u'_i)$$

Finally, a decision is made by comparing the "score" (i.e., the summed weights) to a *decision threshold,* θ.

II. MEASURING ACCURACY

The decision threshold θ is chosen to give the least objectionable trade-off between *recall* and *precision* on some labeled test data. *Recall* is the percent of known matched pairs that get a score above θ. *Precision* is the percent of pairs with a score above θ that are matched pairs. Put another way, the classifier is calling pairs a *match* if they get a score above θ. So recall is the percent of real matches that the classifier calls a match, and precision tells what percent of pairs that the classifier *calls* a match really *are* a match. Thus, the *false negative* (or "missed match") rate is 100% - recall, and the *false positive* (or "bad match") rate is 100% - precision.

It is important to understand this well, so Table 1 illustrates an example. Given 1000 labeled pairs of records, let us assume 400 are matching pairs and 600 are differing pairs. Assume also that 300 of the 400 matching pairs get a score above some θ. This means that the *recall* is 300 / 400 or 75%. Assume also that 50 of the differing pairs also get a score above θ. Then there are 300 matching and 50 differing records with a score above θ, meaning that the precision is 300 / (50 + 300) = 300 / 350 = 85.7%.

TABLE I
EXAMPLE OF PRECISION AND RECALL

	Matching Pairs	Differing Pairs	Total with score > θ
Score > θ	**300**	50	**350**
Score < θ	100	550	650
Total	**400**	600	1000

Recall = 300/(300+100) = 300/400 = 75%.
Precision = 300/(300+50) = 300/350 = 85.7%.

By choosing different values of θ, it is possible to increase the recall at the expense of precision, or vice-versa. The only way to increase *both*, however, is to improve the classifier. Some ways this can be done include (a) better normalization of data; (b) using different or more complex

features; or (c) generating the weights and/or scores using a more powerful algorithm.

III. Equivalence of Probabilistic Record Linkage and the Naïve Bayes Classifier

One of the often-admitted weaknesses of probabilistic record linkage is that it depends upon the assumption that each of its fields is independent of the others. This is clearly not the case, but results have been reasonable enough that the formulas have still been useful in practice.

It has been noted by some (*e.g.*, [5]) that the probabilistic record linkage formulas are equivalent to the *naïve Bayes classifier* [6, 7], which depends upon the same independence assumptions. To see that this is so, first consider the formulas used by a naïve Bayes classifier. This classifier attempts to calculate the probability of each class C—which, in our case, include the two classes *match (M)* and *differ (D)*—given the feature values $x_1,\ldots,x_n$, such as field agreement for each of the n fields. Using Bayes' theorem, this can be written as:

$$P(C \mid x_1,\ldots,x_n) = P(C)P(x_1,\ldots,x_n|C)/P(x_1,\ldots,x_n)$$

Using the independence assumption, this results in

$$P(C \mid x_1,\ldots,x_n) = P(C)P(x_1|C)\ldots P(x_n|C)/P(x_1,\ldots,x_n)$$

To decide whether a pair is a match or not, the classifier determines which probability is greater, i.e., it assumes the pair is a *match* (M) rather than a *differ* (D) if

$$P(M \mid x_1,\ldots,x_n) > P(D \mid x_1,\ldots,x_n)$$

$$\Rightarrow \frac{P(M)\cdot \prod_{i=1}^{n} P(x_i \mid M)}{P(\mathbf{x})} > \frac{P(D)\cdot \prod_{i=1}^{n} P(x_i \mid D)}{P(\mathbf{x})}$$

$$\Rightarrow \ln\left(P(M)\cdot \prod_{i=1}^{n} P(x_i \mid M)\right) > \ln\left(P(D)\cdot \prod_{i=1}^{n} P(x_i \mid D)\right)$$

$$\Rightarrow \ln(P(M)) + \sum_{i=1}^{n} \ln(P(x_i \mid M)) > \ln(P(D)) + \sum_{i=1}^{n} \ln(P(x_i \mid D))$$

$$\Rightarrow \sum_{i=1}^{n} (\ln(P(x_i \mid M)) - \ln(P(x_i \mid D))) > \theta_0$$

$$\Rightarrow \sum_{i=1}^{n} \begin{cases} \ln(m_i) - \ln(u_i), \text{ if } x_i \text{ agrees} \\ \ln(m_i') - \ln(u_i'), \text{ if } x_i \text{ disagrees} \end{cases} > \theta_0$$

$$\Rightarrow \sum_{i=1}^{n} \begin{cases} w_{a,i}, \text{ if } x_i \text{ agrees} \\ w_{d,i}, \text{ if } x_i \text{ disagrees} \end{cases} > \theta_0$$

where $\theta_0 = \ln(P(D)) - \ln(P(M))$, and is the constant, "default" decision threshold. In practice, any value of θ could be chosen that gives the best trade-off of recall and precision. Note that the final formula is the same decision rule and set of weights used by probabilistic record linkage.

IV. Using features instead of fields

Probabilistic record linkage traditionally uses simple field agreement or disagreement for its calculations. However, it is possible to use more complicated features to improve accuracy. To see how this can be done, we first recast probabilistic record linkage into machine learning terms by having two mutually exclusive binary *features*, f_{2i-1} and f_{2i}, for each original *field*, i. For example, assume we have two fields with weights as follows.

Field 1: given name
  agreement weight = $w_{a,1}$; disagreement weight = $w_{d,1}$
Field 2: surname
  agreement weight = $w_{a,2}$; disagreement weight = $w_{d,2}$

We can map these two *fields* to four *features* with the weights that apply when the feature "fires":

Feature f_1: given name agrees; weight = $w_{a,1}$
Feature f_2: given name disagrees; weight = $w_{d,1}$
Feature f_3: surname agrees; weight = $w_{a,2}$
Feature f_4: surname agrees; weight = $w_{d,2}$

Since a particular field cannot both agree and disagree, only one or the other feature can fire for each field. If either record in a pair is missing data for that field, then neither will fire.

Moving one step further, it is possible now to introduce more complex features than simple field comparisons. For example, in comparing birth dates, we have analyzed dates among matching pairs of records and among differing pairs of records, and have found several levels of agreement. An exact day, month and year agreement is best. But matching records often have a complete (day/month/year) dates that differ by up to a couple of weeks in genealogical data (due to calendar changes, the lag between birth and recording, etc.). Many genealogical dates are year-only dates, and these can be off by several years on matching records, as they often represent rough estimates by genealogists or automated systems. On the other hand, complete dates that are off by more than a couple of weeks are rarely found among matching records, so these are rated as a complete disagreement.

Similarly, name comparisons can benefit from having several levels of agreement ranging from multiple identical name pieces (e.g., "John Henry" for both), to similar but not identical names ("John Henry" / "Jonathan H.") to outright conflicts ("John Henry" / "John William" or "David H." / "Jason Evan").

By having several levels of matching for names, dates, places and other fields, it is possible to have different weights for those cases, which allows the classifier to discriminate more accurately.

Once we are free from the restriction of using simple field comparisons, it also becomes possible to use more complex features that can help distinguish between matching and differing pairs of records. For example, although we may originally use the birth date and the death date as two

fields for comparison, we can also create another feature that looks to see if one of the people died before the other was born. This is especially helpful when one record does not have a birth date and the other one does not have a death date.

By taking combinations of values and other higher-level features into account, we can provide the classifier with more domain knowledge so that it has more powerful pieces of information to use in scoring pairs of records.

Complex features might look something like this

Feature f_1: given name agrees very well
Feature f_2: given name agrees well
Feature f_3: given name partially agrees
Feature f_4: given name conflicts,

and so on.

Each of these features either fires or does not for each pair, and certain features can be clustered into groups that are known to be mutually exclusive such as the four features above. Genealogical matching features are described in detail in [8].

V. Multi-valued Probabilistic Record Linkage

With a slight modification, the probabilistic record linkage formulas can be altered to support arbitrary features instead of just field agreement and disagreement.

Instead of using an "agreement weight" and a "disagreement weight" for each "field", each *feature* gets a single weight that is added when that feature *fires*. Equivalently, each feature can be viewed as having a weight that is always multiplied by the input, which will be either 1 or 0.

The formula for each feature weight is computed as

$$w_i = \ln(m_i / u_i) = \ln(m_i) - \ln(u_i)$$

and m_i and u_i are altered just slightly to be defined as

$$m_i = \text{P(feature } i \text{ fires on a matched pair)} = n_{m,i} / |M|$$
$$u_i = \text{P(feature } i \text{ fires on a differing pair)} = n_{u,i} / |U|$$

where $n_{m,i}$ and $n_{u,i}$ are the number of times that the feature i fires on matched and differing ("unmatched") records, respectively.

When there are groups of features that are mutually exclusive and missing values are common, then m_i and u_i can use the number of matched pairs (and differing pairs, respectively) for which *any feature in the group* fired, instead of using $|M|$ (or $|U|$, respectively).

VI. Neural Networks for More Accuracy

Using a naïve Bayes classifier can produce useful results, but there are many other neural network and machine learning algorithms available, including a single-layer perceptron [9], multilayer backpropagation network [10], support vector machines [11], rule-based systems [12], instance-based systems [13, 14], radial basis functions [15], and so on. Often other algorithms can achieve significantly higher accuracy than the naïve Bayes classifier, and thus, higher accuracy than standard probabilistic record linkage.

Unfortunately, there has not be as much overlap between the record linkage and machine learning communities as there perhaps should have been, so often probabilistic record linkage is used by default even when more accurate alternatives are available. One purpose of this paper is to help rectify this disconnect.

A single-layer perceptron has a simple learning rule, which is given in the pseudo-code below.

```
Single Layer Perceptron(Training set T) {
   initialize w[1..n+1] to random values between -0.01 and +0.01
   let lr = 0.01     // small, constant learning rate
   for each iteration (up to 1000) {
      shuffle instances
      tss = 0     // total sum-squared error
      for each instance I in T (with inputs x[1..n+1] and target y) {
         x[n+1] = 1     // set bias input to 1 always
         sum = 0
         for i=1..n+1
            sum += x[i] * w[i]
         activation = 1/(1 + exp(-sum))     // sigmoid function
         error = y – activation    // = target - output
         for i = 1..n+1
            w[i] = w[i] + lr * error * x[i]
         tss += error * error
      }
      stop if tss has not changed by more than 0.0001
   }
}
```

The algorithm iterates through a *training set, T*, consisting of *instances*, each of which has an *input vector, **x***, and a binary *target value*, *y*. One *weight* is trained for each input, plus one more weight for a *bias* (which can be trained by having an extra input whose value is always 1), and these weights are initialized with small random values. One *iteration* (or *epoch*) consists of running through all of the training instances in random order. Usually several (e.g., a few hundred) iterations are run, or training is stopped when the error rate on the training set (or, ideally, on a hold-out set of additional data) stops improving. For each instance, the activation is calculated by multiplying the inputs by the current weights and running the sum through a nonlinear activation function such as a sigmoid. Then the error is computed and weights are updated as shown.

In our application, the input vector ***x*** consists of binary attributes, one per feature value. At most one feature value will be a "1" for each high-level feature, since each feature's set of values (or "levels of agreement") are mutually exclusive, as explained in earlier sections. The output value *y* is also a binary value, where 0 means "not a match" and 1 means a "match".

One advantage that both probabilistic record linkage and single-layer perceptrons have over some other algorithms is that they derive a single weight for each feature, which makes it possible to have some understanding of the "meaning" of the weights. For example, if there is a large negative weight on the "given name conflicts" feature,

and a small positive weight on the "given name weakly agrees" feature, those weights can be examined by hand to see if they make sense. Furthermore, when the classifier makes a mistake, the features and weights can be examined to gain insight into the best place to do further feature engineering in order to improve accuracy.

There are limits to how far one can trust their interpretation of these weights, however. In one case, for example, we found a negative weight associated with agreement on the father's name. This was counterintuitive, but seemed to be explained by the nature of our data, *i.e.*, the records that had agreement in the father's given name tended to also have quite a bit of other data about the persons that could be used; and yet it was quite common to have siblings being compared, who of course have the same father information, and a lot of other similar information. So the neural network apparently found that penalizing for father name agreement allowed better accuracy on the common sibling case without hurting accuracy on real matches much.

Having one weight per feature value does mean that they are both constrained by having a linear decision surface, unlike many other algorithms, but although we have found a perceptron to yield much better results than probabilistic record linkage formulas, we have not found multilayer perceptrons or most other algorithms to improve accuracy much beyond that, if any, in our genealogical data. This may be due in part to the complex features that already take advantage of domain knowledge and "flatten" the decision space somewhat.

Perhaps the largest advantage that even a single-layer perceptron has over probabilistic record linkage (and, equivalently, the naïve Bayes classifier) is that it does not rely upon the independence assumption. If some of the inputs are correlated, the training algorithm will tend to adjust weights to account for this. By observing the effect of weights on accuracy and adjusting weights accordingly, neural networks can avoid assigning too much weight to features that have correlation with other features. This happens naturally as part of the training process.

Neural networks typically initialize their training with small, random weights. It would be possible to use probabilistic record linkage formulas to calculate weights, and use *those* as the initial weights for a single-layer neural network. What would happen at that point is that the algorithm would take those weighs as a starting point, and then begin adjusting them based on the errors that are seen in the training data. In other words, the probabilistic formulas can produce weights that are perhaps a nice start, but they fail to adjust the weights to overcome errors caused by the independence assumption, whereas the neural network training algorithm continues to adjust until the error rate stabilizes.

VII. Empirical Results

Experiments were run using a set of 80,000 pairs of genealogical records that were hand-labeled by genealogical experts as a "matching" pair (i.e., both person records appear to represent the same real person) or a "differing" pair (where the two person records appear to represent two different real persons). 48,000 of the records were used for training and 32,000 were used for testing.

Two versions of the data were generated. The first was an original feature set with 159 binary features (in 32 "feature groups"). A simplified feature set was also generated, in which features were limited to just 16 fields, each of which was found to "agree" or "disagree", as is done in traditional probabilistic record linkage. (A field was said to "agree" if it exactly or closely agreed, and to "disagree" only if it conflicted. Very weak agreement was treated the same as a missing value in the simplified feature set).

The probabilistic record linkage formulas were implemented as outlined in this paper, *i.e.*, the original formulas from Section I were used for the simplified feature set to calculate agreement and disagreement weights, and the multi-value formulas from Section V were used on the full feature set to support the more complicated features. (Incidentally, the multi-value formulas produced identical results to those produced by the original formulas on the simplified feature set, as expected.)

A single-layer perceptron with initial weights in the range of –0.01 to +0.01 and a learning rate of 0.01 was used to train on both versions of the data as well. 1000 iterations were run, though it appeared to be well trained after about 250 iterations. Training time was less than a second for PRL and a few seconds for the neural network.

Both algorithms resulted in one weight per feature, and these weights were summed for all features that fired in order to classify pairs in the test set as *match* or *differ* using various thresholds. (In other words, although training is done differently, execution is done identically for both methods once the weights have been generated).

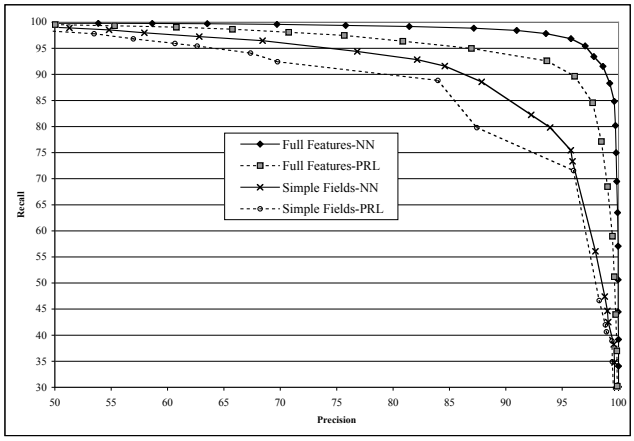

Fig. 1. Precision and recall of a single-layer perceptron neural network (NN) and a probabilistic record linkage (PRL) algorithm on genealogical records, using "simple" field agreement features or a full set of complex features.

Figure 1 shows the precision/recall results for the single-layer perceptron neural network (NN) and the probabilistic record linkage formulas (PRL), on both the simplified field-level comparisons ("Simple Fields") and on the full feature set ("Full Features"). The precision/recall

curves were generated by varying the threshold for each of the four runs.

As can be seen in Figure 1, the original PRL formulas on simple fields did have some success. However, using a single-layer perceptron on the exact same data produces better results. Switching from simple fields to the full features produces an even more dramatic improvement for PRL, but using a neural network on the full features improves accuracy even more.

TABLE II
RECALL AT LEVELS OF PRECISION ABOVE 90%

	Simple Fields		Full Features	
Precision	PRL	NN	PRL	NN
90	**77.3**	85.5	93.9	**98.6**
91	**76.4**	84.1	93.5	**98.5**
92	**75.4**	82.6	93.2	**98.2**
93	**74.5**	81.2	92.8	**98.0**
94	**73.5**	79.7	92.5	**97.7**
95	**72.5**	77.3	91.0	**97.2**
96	**71.6**	74.9	89.8	**96.7**
97	**60.7**	64.2	86.8	**95.5**
98	**49.8**	55.7	83.6	**92.9**
99	**40.6**	45.1	68.9	**90.7**
100	**5.9**	34.7	32.6	**81.6**

Table 2 shows the recall level at each precision level from 90 to 100. As can be seen from the numbers, using the full features was consistently better than using the simple ones, and using a neural network was consistently better than using the probabilistic record linkage weights. Note the dramatic jumps in accuracy by using both improvements: 77.3 to 98.6; 40.6 to 90.7; 5.9 to 81.6.

VIII. CONCLUSIONS

Early on in our work on genealogical record linkage, we compared PRL with various neural network and machine learning algorithms and found a great improvement by using even a single-layer neural network. As we repeated these experiments with more data (as presented in this paper), the results have continued to hold. Even a single-layer neural network has the huge advantage that it can adjust the weights to minimize error instead of just setting them once based on independent attribute statistics without any regard to whether they really produce good results.

In the machine learning and neural network literature, new algorithms often yield very small improvements over earlier ones. To see improvements this consistent and this dramatic indicates that those still using traditional record linkage need to take notice and use more powerful features and better classifiers to improve accuracy. For those already using neural networks, these results underscore the power of taking advantage of domain knowledge to improve features.

REFERENCES

[1] Dunn, H. L. "Record Linkage," *American Journal of Public Health*, vol. 36, 1412-1416, 1956.

[2] Newcombe, H. B., Kennedy, J. M., Axford, S.J., and James, A.P. "Automatic linkage of vital records." *Science*, vol. 130, 954-959, 1959.

[3] Newcombe, Howard B. *Handbook of Record Linkage*, Oxford University Press, New York, 1988.

[4] Fillegi, I. P. and Sunter, A. B. "A Theory for Record Linkage," *Journal of the American Statistical Association*, vol. 64, 1183-1210, 1969.

[5] Quass, Dallan, and Starkey, Paul. "Record Linkage for Genealogical Databases," *ACM SIGKDD '03 Workshop on Data Cleaning, Record Linkage, and Object Consolidation,* August 24-27, 2003, Washington, D.C.

[6] Langley, Pat, Wayne Iba, and Kevin Thompson. "An Analysis of Bayesian Classifiers," In *Proceedings of the 10th National Conference on Artificial Intelligence, (AAAI-92)*, AAAI Press/MIT Press, Cambridge, MA, pp. 223-228, 1992.

[7] Michie, D., D. Spiegelhalter, and C. Taylor. *Machine Learning, Neural and Statistical Classification*, Ellis Horwood, Hertfordshire, England. Book 19, 1994.

[8] Wilson, D. Randall. "Genealogical Record Linkage: Features for Automated Person Matching," *RootsTech 2011*, February 4, 2011, Salt Lake City, Utah.

[9] Rosenblatt, Frank. *Principles of Neurodynamics*, New York, Spartan Books, 1959.

[10] Rumelhart, D. E., and J. L. McClelland. *Parallel Distributed Processing*, MIT Press, 1986.

[11] Vapnik, V. *Statistical Learning Theory.* Wiley-Interscience, New York, 1998.

[12] Quinlan, J. R. *C4.5: Programs for Machine Learning*, San Mateo, CA: Morgan Kaufmann, 1993.

[13] Aha, David W., Dennis Kibler, Marc K. Albert. "Instance-Based Learning Algorithms," *Machine Learning*, vol. 6, pp. 37-66, 1991.

[14] Wilson, D. Randall and Martinez, Tony R. "An Integrated Instance-Based Learning Algorithm," *Computational Intelligence*, vol. 16, no. 1, pp. 1-28, 2000.

[15] Broomhead, D. S., and D. Lowe. "Multi-variable functional interpolation and adaptive networks." *Complex Systems*, vol. 2, pp. 321-355, 1988.