Use of the Levenshtein Distance For Merging Longitudinal Student Records

Harold C. Doran
hdoran@air.org

Paul Van Wamelen
pvanwamelen@air.org

American Institutes for Research
Washington, DC

Working Draft

October 1, 2008

Abstract

The analysis of longitudinal data in education is becoming more prevalent given the nature of testing systems constructed for No Child Left Behind. While these longitudinal analyses are more common, constructing the longitudinal data files remains a significant challenge. Students tend to move into new schools and districts, and in many cases, the unique identifiers (ID) that should remain constant for the student actually changes. It often occurs that different students share the same ID. Merging records for different students with the same ID is clearly problematic. However, in the absence of methods for confirmation, it is not possible to know if the records for different students sharing the same ID are merged. In very small data sets, quality control can proceed through reviews of the data to ensure all merges were properly performed. However, in data sets with hundreds of thousands of cases or more, quality control via human review is impossible. While some informal protocols may be in place for quality control (e.g., check to see if grade increments by 1), the educational analysis literature lacks formal protocols for quality control. This paper presents an empirical quality control procedure that may be used to verify the integrity of the merges performed for longitudinal analysis.

Keywords: longitudinal analysis; Levenshtein algorithm; quality control; R program

1 Introduction

Under the No Child Left Behind Act (NCLB) students in grades 3 to 8 and high school are assessed at least once per year in reading and math. The scores from these yearly assessments are now becoming valuable pieces of information as they can be used in different forms of longitudinal statistical analyses, such as value-added models and other growth models. One particular challenge is that the yearly data files must be merged together such that linkages across time for students can be constructed in order to implement these analyses.

The typical approach is to use a unique student identifier and merge data file 1 with data file 2. However, there is often a lack of quality control implemented in order to ensure that records with the same student identifier are in fact records for the same student. Some *ad hoc* methods, such as checking to see if the grade level increments by one or possibly by examining gain scores for individuals, may be used. However, these methods lack formal criterion for determining if the merge was performed correctly.

As growth models, and the merges needed to implement them, become even more common, it is important to establish formal protocols for assessing the integrity of merged data sets. Consequently, the purpose of this paper is to present a general purpose algorithm that may be useful for guiding the quality control procedures needed to ensure that the correct student records are merged.

2 The Levenshtein Distance Metric

The Levenshtein Distance (LD) is a metric useful for determining the similarity of two character strings. The LD, or the edit distance, is defined as the minimum number of operations needed to transform `string 1` into `string 2` where an operation is either an insertion, deletion, or substitution of a character (Levenshtein, 1966). When the edit distance is 0, the two character strings are exactly the same. That is, no changes are needed to transform `string 1` into `string 2`. When the edit distance is non-zero, this is an indication of differences between strings with larger edit distances indicative of larger differences.

Because the LD provides an empirical basis for comparing the similarity of character strings, it is extremely valuable as a confirmatory tool in judging whether the merges from data set 1 and data set 2 properly join the records for the same student. The following sections provide substantive details and examples on the edit distance. We subsequently demonstrate how the LD, and its variants, can be used to verify the integrity of merged data sets.

2.1 Example of the Levenshtein Distance

Assume we have two character strings we wish to compare. The first string is `Bill Clinton` and the second string is `William Clinton`. The purpose of the LD procedure is to empirically determine how similar these two character strings are. In this example, the last name is exactly the same and no edits, insertions, or substitutions are necessary. The first name differs, however. Transforming `Bill` to `William` would require the following operations:

Operation 1: Substitute `W` for the `B`

Operation 2: Insert `i` after the `Will`

Operation 3: Insert `a` after the `Willi`

Operation 4: Insert `m` after the `Willia`

There are a total of four operations needed to transform `Bill` to `William`; hence the edit distance is 4. In order to facilitate the use of this procedure, the `stringMatch` function was developed in the `R` statistical computing environment (R Development Core Team, 2008). The `stringMatch` function is available in the **MiscPsycho** package (Doran, 2008).

The following examples illustrate how the function operates:

```
> stringMatch('William Clinton', 'Bill Clinton', normalize='no')
[1] 4
> stringMatch('Barack Obama', 'Barry Obama', normalize='no')
[1] 3
> stringMatch('John McCain', 'Jon McCain', normalize='no')
[1] 1
> stringMatch('John McCain', 'Barack Obama', normalize='no')
[1] 11
```

The two character strings with the smallest edit distance is the `John McCain` to `Jon McCain` comparison, an indication that there is a high degree of similarity between these strings. The largest edit distance is the `John McCain` to `Barack Obama` comparison, an indication that these two strings not similar.

2.2 The Normalized Edit Distance

The edit distance is useful, but normalizing the distance to fall within the interval [0,1] is preferred. This normalization is preferred as it is somewhat difficult to judge whether an LD of 4 is large or small. In our implementation, the Levenshtein normalized distance (LND) is transformed to fall in this interval as follows:

$$LND = 1 - \frac{LD}{max(s1, s2)} \tag{1}$$

where LD is the edit distance and $max(s1, s2)$ denotes that we divide by the length of the larger of the two character strings. This normalization yields a statistic where 1 indicates perfect agreement between the two strings and a 0 denotes imperfect agreement. The closer a value is to 1, the more certain we can be that the character strings are the same. The closer to 0, the less certain. For example:

```
> stringMatch('William Clinton', 'Bill Clinton', normalize='yes')
[1] 0.7333333
> stringMatch('Barack Obama', 'Barry Obama', normalize='yes')
[1] 0.75
> stringMatch('John McCain', 'Jon McCain', normalize='yes')
```

```
[1] 0.9090909
> stringMatch('John McCain', 'Barack Obama', normalize='yes')
[1] 0.08333333
```

Recall that the edit distance for transforming `William Clinton` into `Bill Clinton` is 4. The normalization in this case occurs as:

$$1 - \frac{4}{15} = .73 \tag{2}$$

We divide by 15 because the length of the character string `William Clinton` is 15, which is the larger of the two strings (the space between the first and last name is included in this example). As in Section 2.2, the `John McCain` to `Jon McCain` yields the value closest to 1, a strong indication of similarity. Additionally, the `John McCain` to `Barack Obama` comparison yields a very low value, a strong indication of dissimilarity.

2.3 Probability of the Normalized Edit Distance

In addition to the LND, it is useful to determine the chances of observing an LND value of x or larger in a population of names. In other words, the substantive question at hand is, "what are the chances that an LND of x would be observed if two random character strings were compared"?

Given a large data set with many names (such as a student test score database), the following procedure can be used to empirically obtain these probabilities:

1. Take a random sample without replacement of n names from the data.
2. Compare each of these n names to all N student names in the data to obtain the LND.
3. Count the number of times the LND of x is observed.
4. Divide x by the total number of comparisons made

To illustrate this process, assume we have the following data:

```
          fname1          fname2
1   Joseph McCall      Joe McCall
2      Paul Jones      Paul Jones
3   Larry Everett   Barry Everett
4    Sam Thompson Samuel Thompson
5    Sally Fields    Sally Fields
6     Doug Carter  Douglas Carter
7   Bill Friendly  William Friend
8     Tom Davison   Tommy Davison
9      Frank Mann   Franklin Mann
10     Mary Jones      Cary Jones
```

	Joseph McCall	Sally Fields	Frank Mann	Paul Jones	Larry Everett
Joe McCall	0.77	0.08	0.10	0.00	0.08
Paul Jones	0.08	0.33	0.20	1.00	0.23
Barry Everett	0.00	0.31	0.08	0.23	0.92
Samuel Thompson	0.13	0.27	0.20	0.40	0.13
Sally Fields	0.08	1.00	0.08	0.33	0.31
Douglas Carter	0.14	0.14	0.14	0.29	0.07
William Friend	0.07	0.43	0.14	0.21	0.14
Tommy Davison	0.08	0.15	0.23	0.15	0.15
Franklin Mann	0.08	0.08	0.77	0.23	0.00
Cary Jones	0.08	0.33	0.20	0.70	0.38

Table 1: Table of LND Statistics Under Multiple Comparisons

where `fname1` is the name in year 1 and `fname2` is the name in year 2. Assume we take a random sample without replacement of five names from `fname1`. This might result in `Joseph McCall, Sally Fields, Frank Mann, Paul Jones, Larry Everett`. Now, we compute the normalized edit distance for these sampled names against every name in `fname2`.

Table 1 shows how the names from the random sample are compared to every other name in the data yielding a normalized edit distance for each comparison. Retrieving the probability now only requires that we count the number of times the same normalized distance is observed divided by the total number of cells in the table. For instance, the value of 1 occurs twice, once in the `Sally Fields` comparison and again for the `Paul Jones` comparison. There are 50 total cells in the table. Hence, the probability of observing a 1 in these data is $2/50 = .04$.

To facilitate use of this procedure, a second `R` program also available in the **MiscPsycho** package called `stringProbs` is available to automate this process. In this function, the user simply provides a dataframe with the names to be compared. The argument $N = 5$ is used to determine how many names are randomly sampled for the comparison. The value of N can be equal to the total number of rows in the dataframe `dat`, but it cannot be larger.

Figure 1 plots the cumulative probability of occurance against the LND statistic for these data. This figure shows that the $P(LND = 1) \approx 0$ and $P(LND = 0) \approx 1$. For example, the probability of $P(LND = .2)$ or larger is approximately .4.

3 Application of the Levenshtein Normalized Distance for Merging Student Records

Many states maintain unique student identifier codes that should remain constant over time. If these IDs were perfectly reliable, then it would be possible to merge using these IDs alone. However, these student IDs can and do change, thus making it feasible that two different students can share the same ID over time. It would of course be improper (and would yield unreliable statistical estimates) if the records for these different students were merged.

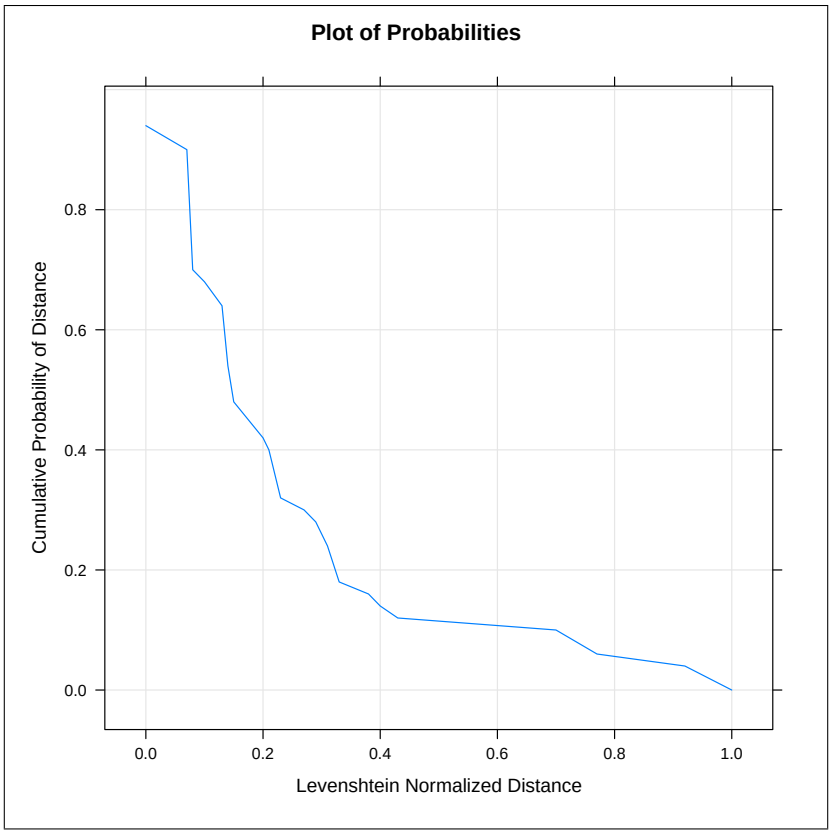

Figure 1: Sample Plot of Cumulative Probabilities

Therefore, the following strategy can be used to merge and validate that the correct student records were merged. Doing so would be performed as:

1. First, merge the year 1 data file with the year 2 data file using the available unique student identifiers.

2. Second, compute the LND for each record in the data using the student first and last names in both years.

3. Subset the data and keep only those records where the normalized edit distance obtained by comparing the first and last names in year 1 against the first and last names in year 2 is $\geq x$ where x is a value determined through examination of the data and the probability of occurance.

When this process is applied to student test score data, the LND is used to verify that the correct student records are merged. One particular challenge, discussed in the next section, is what value of x should be used as the cutpoint.

One particular issue that often arises in student data sets is that the student names are transposed from one year to the next. For example, a student's first name may be properly recorded in year 1 as `William Clinton` but in year two, the first and last names are switched as `Clinton Bill`. If the LND were obtained under this circumstance it would yield:

```
> stringMatch('William Clinton', 'Clinton Bill')
[1] 0.1333333
```

which is a very low value and may suggest this student should be dropped from the database even if the IDs were the same. One possible enhancement is to derive the LND under multiple permutations of the first and last name and use the maximum value of the LND under each permutation. The following three permutations may be useful:

Permutation 1: Compare year 1 (first, last) to year 2 (first, last)

Permutation 2: Compare year 1 (first, last) to year 2 (last first)

Permutation 3: Compare year 1 (last, first) to year 2 (first, last)

In this example, each of the three permutations would result in the following:

```
> stringMatch('William Clinton', 'Clinton Bill') # Permutation 1
[1] 0.1333333
> stringMatch('William Clinton', 'Bill Clinton') # Permutation 2
[1] 0.7333333
> stringMatch('Clinton William', 'Bill Clinton') # Permutation 3
[1] 0.1333333
```

The maximum LND of .73 would be used as the value for this student in the verification process.

4 Demonstration Using Data from Washington, DC

Two years of test score data were merged using the unique student identifier codes in year 1 and in year 2. Only students with first and last names in years 1 and 2 are retained in the data, resulting in a total of 22890 students.

As a first step, the first and last names in year 1 are concatenated to form the `string 1` and the first and last names in year 2 are concatenated to form `string 2`. The `stringMatch` function is applied to all students comparing `string 1` and `string 2` to compute the LND. For the current example, only **Permutation 1** is provided. Greater reliability would likely result if all three permutations were applied and the maximum LND from each were used.

Figure 2 is a set of conditional density plots showing the distribution of the LND statistic for all students in the data by grade level. The very large spike near 1 for each grade shows that the LND is very high for almost all cases in the data, suggesting that these data are very clean. That is, the very large LND for almost all students is indicative of the fact the the merge using student ID most likely merged the correct students together as verified using the `string 1` to `string 2` LND comparison.

As a second step, the `stringProbs` function is used to get an empirical estimate of the probability of observing an LND of x. In this application, a random sample of 10 names are drawn from `string 1`. These 10 names are compared to every `string 2` name in the data. The purpose of this step is to help determine what cutpoint might be used for subsetting

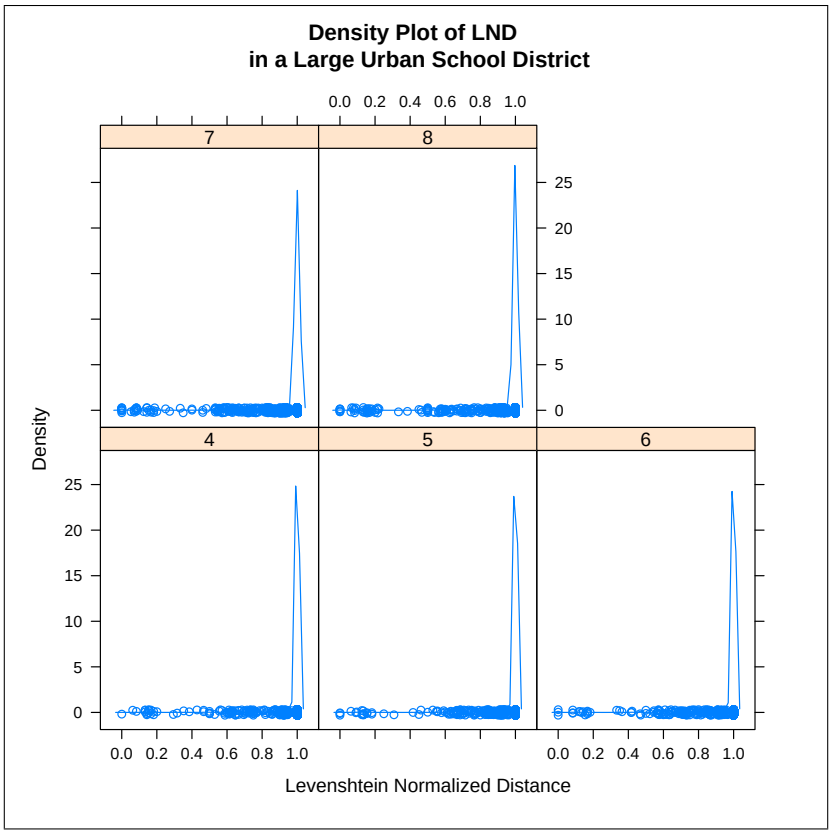

Figure 2: LND Density Plot

the data. That is, we may want to keep only those students whose IDs are the same and their LND statistic is $\geq x$.

Figure 3 plots the cumulative probability of occurance against the LND statistic. The results in this figure suggest that the probability of occurance begins to diverge from 0 around an LND of .4. This gives a good starting point for initial reviews of the data. In other words, we might begin our initial review of the records for individuals where $LND \leq .4$.

While the data used in this example cannot be publicly shared because of FERPA regulations [1], the matching of correct students is remarkably correct. In our review, students with LND statistics lower than .4 are always different students. Had these students been retained in the data without the LND verification, the wrong records would have been merged and subsequent statistical analyses would reflect this inaccuracy. This nicely conforms to the probabilities as displayed in Figure 3. Within the range of .4 to .7, there is quite a bit of ambiguity in the student names compared, therefore it is not always possible to discern if the names are the same. Within this range are many misspellings, reversals, and obvious non-matches. For all students with an LND above .7, our manual review indicates that these students are always correctly merged. Hence, LND = .7 would be a good cutpoint chosen for retaining students.

Our internal work has shown that with these data, if the LND cutscore were set at .7,

[1]FERPA is the Family Education Rights and Privacy Act and prevents any identifiable information from being released

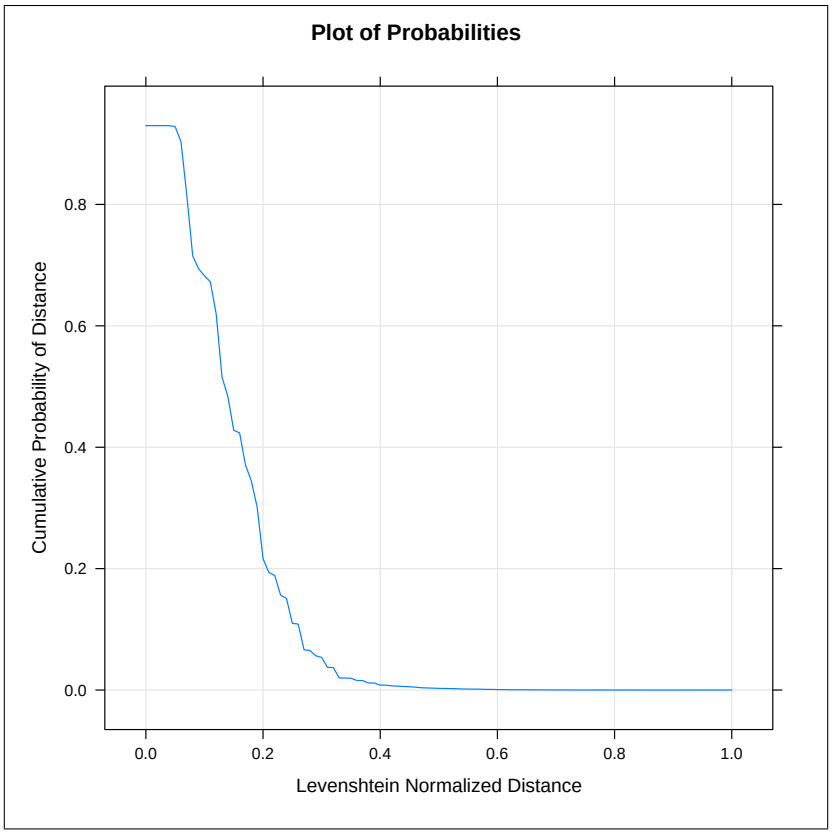

Figure 3: Plot of Cumulative Probabilities

only 200 students would be removed from the entire data set of over 22,000. None of these students are clustered in any given school and the characteristics of these students are varied. Consequently, removing these individuals does not create a missing not at random situation (MNAR) that is critical for these kinds of statistical analyses.

5 Generalizations of the LND

In the examples used in this paper, only first and last names were combined to compute the LND and the probability of its occurance, $P(LND)$. However, the method is clearly general and other demographic characteristics of students could be brought in for purposes of additional assurance.

For instance, assume that gender and date of birth are available in the year 1 and year 2 data files that are to be merged in addition to the first and last names and the unique student ID. These indicators can be used to further determine the chances of a correct merge. To illustrate, assume there are three students in the data.

The issue at hand is answering the question, "what is the probability of two random students having an exact match on all of these characteristics?". This probability can be used to determine a point of cutoff. For instance, the chances of an exact match on month in the date of birth is 1/12, the chances of an exact match on day in the date of birth is roughly 1/31, the chances of an exact match on year in the date of birth is $1/Q$ where Q are

Student	DOB1	DOB2	Gender1	Gender2	LND	P(LND)
1	10.20.2001	10.20.2001	F	F	.7	.001
2	12.15.2001	1.15.2001	F	M	.2	.2
3	4.17.2001	3.18.2002	M	F	.1	.4

Table 2: Sample Table of Demographic Characteristics

the number of years listed in the data file, and the chances of an exact match on gender is 1/2.

For the current illustration, assume a frequency count on year in the data shows there are only four years (2000, 2001, 2002, 2003), thus $Q = 4$. In the first case, Student 1 matches exactly on all of these demographic characteristics. So, we compute the probability of two random students having an exact match on all of these characteristics as $1/31 \times 1/12 \times 1/4 \times 1/2 \times .001 = 3.360e-07$. In the case of the Student 2, there is an exact match only on day and year in the data of birth. So, the probability is $1/12 \times 1/4 \times .2 = 0.0042$. The last student matches on none of the demographic characteristics and the probability of the LND is rather high. In this case, the chance of two random students matching on this single characteristic is .4.

In the case of Student 1, the probability is very low, therefore there is a good degree of assurance that the records for this student are properly merged. This is less true for Students 2 and 3. With large-scale data sets, this probability would be computed for all students in the data using any available demographic characteristics. Any student with a probability less than p may be retained as a proper merge and all others discarded as non-matches.

6 Summary

The LND can be a useful tool for those involved in the analysis of student achievement test scores where it becomes necessary to form a longitudinal data base by merging student records from one year to the next. This process acts only as a verification procedure to ensure that the correct student in year 1 is merged with the same student student in year 2. Some human intervention is involved as a review of the data is most likely required in order to determine the cutpoint that should be used to retain student records.

As in any analyses, it is of substantive interest to further review the dropped cases to assess whether those individuals cluster into specific common groups, such as schools or ethnicity, to evaluate whether the missingness created in the data may result in systematic biases in the statistical results.

References

Doran, H. C. (2008). *MiscPsycho: Miscellaneous psychometrics.* (R package version 1.2)

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, *10*, 707-710.

R Development Core Team. (2008). *R: A language and environment for statistical computing.* Vienna, Austria. (ISBN 3-900051-07-0)