Fast Record Linkage of Very Large Files in Support of Decennial and Administrative Records Projects

William E. Winkler, William E. Yancey, and Edward H. Porter,
{william.e.winkler, william.e.yancey, edward.h.porter }@census.gov

Abstract
There is an increased need to find duplicates in very large files. This paper details the current version of Bigmatch software (Yancey and Winkler 2004, 2009) that is sufficiently fast for processing 10^17 (=300 million x 300 million pairs) for the U.S. Decennial Census and even larger administrative-record situations with billions of records. The software, via a nontrivial application of a set of blocking strategies, is known to find more than 97.5% of true matches with very small error of less than 0.5% (Winkler 2004, 1995). It is 40-50 times as fast as recent parallel software (Kim and Lee 2007; Kawai, Garcia-Molina, Benjelloun, Menestrina, Whang and Gong 2006).

Keywords: entity resolution, approximate string search, classification

1. Introduction

Record linkage has traditionally been used for linking records from two files A and B or finding duplicate records in a single file C using *quasi-identifiers* such as name, address, date-of-birth, and other non-unique identifiers. If all quasi-identifiers had no missing data, had no typographical error, and in combination could be used for uniquely identifying, then it would be straightforward to match using sort-merge utilities. Real-world data, even when of relatively high quality, are characterized by errors (e.g., Herzog, Scheuren and Winkler 2007). For instance, in relatively high quality, 1990 Decennial Census data as much as 25% of first names, 10% of last names, and 5% of ages (30% in some hard-to-count areas) had typographical error (e.g., Smith versus Smoth, Robrt versus Bob, etc.) or missing values among 'true' matches that needed to be matched (Winkler 1990). Data that are obtained from scanning hand-written forms often have even higher typographical error rates. If efficient methods of accounting for minor typographical error were not available, then 30% or more of 'true' matches would be missed.

This paper provides background and some results from very large empirical experiments from applying BigMatch software. BigMatch is intended for matching very large lists having hundreds of millions of records with exceptional speed (300,000+ records per second on each cpu) while maintaining high accuracy (less than 0.5% false match rate and over 99% of all true matches brought together in the set of pairs agreeing on certain sets of blocking criteria. In the main Decennial Census application, BigMatch does detailed computation on 10^12 pairs among 10^17 pairs (300 million × 300 million) using 40 cpus of an SGI Linux machine in 15 hours. Each cpu processes 400,000 pairs per second during the 15 hours.

The outline of this paper is as follows. The second section provides background on the Fellegi-Sunter model of record linkage, some insight into matching parameters and the Jaro-Winkler string comparator (that have been widely adopted by computer scientists because of its computational speed), and the types of data structures and indexing that gives BigMatch its speed. The third section consists of matching applications from the Decennial Census and a high quality pair of voter registration databases (VRDs) from the States of Oregon and Washington that were used as test decks for advanced linkage methods in conjunction with a National Academies of Science study (National Research Council 2009). The final section is concluding remarks.

2. Background

In this paper, we apply BigMatch technology to the 2000 Decennial Census (as a test of very similar data and methods used in the 2010 Decennial Census) and to a pair of 2008-2009 voter registration databases from the states of Oregon and Washington. The former application will provide some insights into BigMatch technology that has been applied in a number of large administrative records projects with hundreds of millions of records. The latter project will give some insights into the extraordinary speed of the software in relatively small situations with files having millions of records. Before providing details of the applications, we provide some additional background on the Fellegi-Sunter model of record linkage and how it is applied in our software.

2.1. The Fellegi-Sunter model of record linkage

Fellegi and Sunter (1969) provided a formal mathematical model for ideas that had been introduced by Newcombe (1959, 1962). They provided many ways of estimating key parameters. The methods have been rediscovered in the computer science literature (Cooper and Maron 1978) but without proofs of optimality. To begin, notation is needed. Two files **A** and **B** are matched. The idea is to classify pairs in a product space **A** × **B** from two files A and B into M, the set of true matches, and U, the set of true nonmatches. Fellegi and Sunter, making rigorous concepts introduced by Newcombe (1959), considered ratios of probabilities of the form:

$$R = P(\gamma \in \Gamma \mid M) / P(\gamma \in \Gamma \mid U) \qquad (1)$$

where γ is an arbitrary agreement pattern in a comparison space Γ. For instance, Γ might consist of eight patterns representing simple agreement or not on the largest name component, street name, and street number. Alternatively, each $\gamma \in \Gamma$ might additionally account for the relative frequency with which specific values of name components such as "Smith", "Zabrinsky", "AAA", and "Capitol" occur. The ratio R or any monotonely increasing function of it such as the natural log is referred to as a *matching weight* (or score).

The decision rule is given by:

If $R > T_\mu$, then designate pair as a match.

If $T_\lambda \le R \le T_\mu$, then designate pair as a possible match and hold for clerical review. (2)

If $R < T_\lambda$, then designate pair as a nonmatch.

The cutoff thresholds T_μ and T_λ are determined by a priori error bounds on false matches and false nonmatches. Rule (2) agrees with intuition. If $\gamma \in \Gamma$ consists primarily of agreements, then it is intuitive that $\gamma \in \Gamma$ would be more likely to occur among matches than nonmatches and ratio (1) would be large. On the other hand, if $\gamma \in \Gamma$ consists primarily of disagreements, then ratio (1) would be small. Rule (2) partitions the set $\gamma \in \Gamma$ into three disjoint subregions. The region $T_\lambda \le R \le T_\mu$ is referred to as the no-decision region or clerical review region. In some situations, resources are available to review pairs clerically.

Figure 1 provides an illustration of the curves of log frequency versus log weight for matches and nonmatches, respectively. The two vertical lines represent the lower and upper cutoffs thresholds T_λ and T_μ, respectively. The x-axis is the log of the likelihood ratio R given by (1). The y-axis is the log of the frequency counts of the pairs associated with the given likelihood ratio. The plot uses pairs of records from a contiguous geographic region that was matched in

the1990 Decennial Census. The clerical review region between the two cutoffs primarily consists of pairs within the same household that are missing both first name and age.

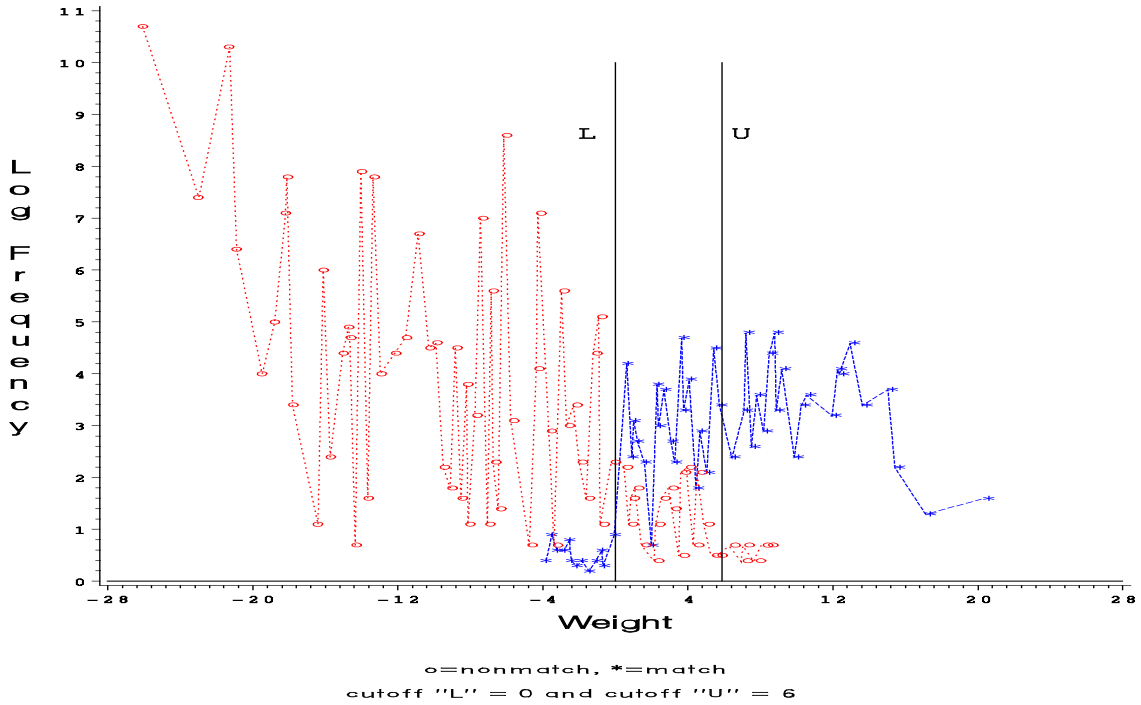

Table 1 provides examples of pairs of records that might be matched using name, address, and age. The pairs give the first indication that matching that might be straightforward for a suitably skilled person might not be easy with naïve rules based on (1) and (2), If the agreement pattern $\gamma \in \Gamma$ on the pairs is simple agree or disagree on name, address, and age, then we see none of the pairs would agree on any of the three fields. In most situations, a suitably skilled person would be able to recognize that the first two pairs may be the same but unlikely to put a suitable score (or matching weight) on the first two pairs. The third pair must be taken in context. If the first

Table 1. Elementary examples of matching pairs of records (dependent on context)

Name	Address	Age
John A Smith	16 Main Street	16
J H Smith	16 Main St	17
Javier Martinez	49 E Applecross Road	33
Haveir Marteenez	49 Aplecross Raod	36
Gillian Jones	645 Reading Aev	24
Jilliam Brown	123 Norcross Blvd	43

record in the pair were individuals in medical school at the University of Michigan 20 years ago and the second record is from a current list of physicians in Detroit, Michigan, then, after suitable follow-up, we might determine that the third pair is a match.

If we had computerized algorithms for separating the free-form name field into first name, middle initial, and last name and address in house number, street name, and other components, then we might have better patterns $\gamma \in \Gamma$ for applying (1) and (2). If we had suitable algorithms for comparing fields (e.g. Javier versus Haveir) having typographical error, then we might be to give partial agreement to minor typographical error rather than call a comparison a disagreement. Additionally, we might want standardization routines the replace commonly occurring words with a common spelling ('Raod' with 'Road' in pair two; 'Aev' with 'Ave' in pair three).

2.2. String comparators

In most matching situations, it is not possible to compare two strings exactly (character-by-character) because of typographical error. Dealing with typographical error via approximate string comparison has been a major research project in computer science (see e.g., Hall and Dowling 1980, Navarro 2001). In record linkage, we need to have a function that represents approximate agreement, with agreement being represented by 1 and degrees of partial agreement being represented by numbers between 0 and 1. We also need to adjust the likelihood ratios (1) according to the partial agreement values. Having such methods is crucial to matching. For instance, in a major census application for measuring undercount, more than 25% of matches would not have been found via exact character-by-character matching. Three geographic regions (St. Louis – urban, Columbia, MO – suburban, and Washington – suburban/rural) are considered in Table 2. The function Φ represents exact agreement when it takes value one and represents partial agreement when it takes values less than one. In the St Louis region, for instance, 25% of first names and 15% of last names did not agree character-by-character among pairs that are matches.

Jaro (1989) introduced a string comparator that accounts for insertions, deletions, and transpositions. The basic Jaro algorithm has three components: (1) compute the string lengths, (2) find the number of common characters in the two strings, and (3) find the number of transpositions. The definition of common is that the agreeing character must be within half the length of the shorter string. The definition of transposition is that the character from one string is out of order with the corresponding common character from the other string. The string comparator value (rescaled for consistency with the practice in computer science) is:

$$\Phi_j(s_1, s_2) = 1/3(N_C/len_{s1} + N_C/len_{s2} + 0.5N_t/N_C),$$

where s_1 and s_2 are the strings with lengths len_{s1} and len_{s2}, respectively, N_C is the number of common characters between strings s_1 and s_2 where the distance for common is half of the minimum length of s1 and s_2, and N_t is the number of transpositions. The number of transpositions N_t is computed somewhat differently from the obvious manner.

Using truth data sets, Winkler (1990) introduced methods for modeling how the different values of the string comparator affect the likelihood (1) in the Fellegi-Sunter decision rule. Winkler (1990) also showed how a variant of the Jaro string comparator Φ dramatically improves matching efficacy in comparison to situations when string comparators are not used. The Winkler variant employs some ideas of Pollock and Zamora (1984) in a large study for the Chemical Abstracts Service. They provided empirical evidence that quantified how the probability of keypunch errors increased as the character position in a string moved from the left to the right. The Winkler variant, referred to as the Jaro-Winkler string comparator, is widely used in computer science.

Work by Cohen et al. (2003a,b) provides empirical evidence that the new string comparators can perform favorably in comparison to Bigrams and Edit Distance. Edit Distance uses dynamic programming to determine the minimum number of insertions, deletions, and substitutions to get from one string to another. The Bigram metric counts the number of consecutive pairs of characters that agree between two strings. A generalization of bigrams is *q*-grams where *q* can be greater than 2.

Table 2. Proportional agreement
by string comparator values
among matches
Key fields by geography

	StL	Col	Wash
	First		
$\Phi = 1.0$	0.75	0.82	0.75
$\Phi \geq 0.6$	0.93	0.94	0.93
	Last		
$\Phi = 1.0$	0.85	0.88	0.86
$\Phi \geq 0.6$	0.95	0.96	0.96

Table 3 compares the values of the Jaro, Winkler, Bigram, and Edit-Distance values for selected first names and last names. Bigram and Edit Distance are normalized to be between 0 and 1. All string comparators take value 1 when the strings agree character-by-character.

Table 3. Comparison of string comparators using last names and first names

Two strings		String comparator Values			
		Jaro	Winkler	Bigram	Edit
SHACKLEFORD	SHACKELFORD	0.970	0.982	0.800	0.818
DUNNINGHAM	CUNNIGHAM	0.867	0.867	0.917	0.889
NICHLESON	NICHULSON	0.926	0.956	0.667	0.889
JONES	JOHNSON	0.867	0.893	0.167	0.667
MASSEY	MASSIE	0.889	0.933	0.600	0.667
ABROMS	ABRAMS	0.889	0.922	0.600	0.833
HARDIN	MARTINEZ	0.778	0.778	0.286	0.143
ITMAN	SMITH	0.467	0.467	0.200	0.000
JERALDINE	GERALDINE	0.926	0.926	0.875	0.889
MARHTA	MARTHA	0.944	0.961	0.400	0.667
MICHELLE	MICHAEL	0.833	0.900	0.500	0.625
JULIES	JULIUS	0.889	0.933	0.800	0.833
TANYA	TONYA	0.867	0.880	0.500	0.800
DWAYNE	DUANE	0.778	0.800	0.200	0.500
SEAN	SUSAN	0.667	0.667	0.200	0.400
JON	JOHN	0.778	0.822	0.333	0.750
JON	JAN	0.778	0.800	0.000	0.667

2.3. BigMatch technology

In this section, we consider the situation where we must match a moderate size file B of 100 million records with a large file having upwards of 4 billion records. An example of large files might be a Social Security Administrative Numident file having 700 million records, a U.S. Decennial Census file having 310 million records, or a California quarterly employment file for 20 years having 1 billion records. If the California employment data has 2-3% percent typographical error in the Social Security Number (SSN) in each quarter, then it is possible that most individuals have two breaks in their 20-year time series. The main ways of correcting the SSNs are by using a combination of name, date-of-birth, and address information. The primary time-independent information is name and date-of-birth because address varies considerably over time. Name variations such as maiden names are sometimes available in the main files or in auxiliary files. Name, date-of-birth, and address can also contain significant typographical error. If first name has 10% typographical error rate and last name, day-of-birth, month-of-birth, and year-of-birth have 5% typographical error rates, then exact character-by-character matching across successive quarters could miss 25% of matches.

With classical matching, 10 blocking passes might be performed on the pair of files. For each matching pass, the two files are sorted according to the blocking criteria and then passed through the matching software. To complete the matching 20 passes would need to be made on each file. BigMatch technology alleviates the limitations of the classical matching situation (Yancey and Winkler 2004). Only the smaller B file and appropriate indexes are held in memory. In addition to a copy of the B-file, two sets of indexes are created for each set of blocking criteria. The first index corresponds to the basic quick sort method. The second index gives a very fast method of retrieving and comparing the records information from the B-file with individual records from the A-file. A B-file of 100 million records with associated sets of indexes can reside in 12 gigabytes of memory. Only one pass is made on the B-file and on the A-file. The pass on the A-file is input/output pass only. The possibly very large A-file is never sorted. Several output streams are created for each blocking criteria. Each individual A-record is compared to all of the appropriate B-records according to the set of blocking criteria. No pair is compared more than once. If the A-file contains 1 billion records, then the BigMatch technology may need only 4 terabytes of disk storage in contrast with 16 or more terabytes using conventional matching. Although the BigMatch software effectively makes 10-blocking passes simultaneously, it is nearly as fast as a classical matching program that only makes a single pass against a pair of files. It processes 300,000+ pairs per second. It saves the cpu-time of multiple sorts of the large file that may contain a billion or more records. One sort of a billion-record file on an exceptionally fast machine may take 8+ hours. The biggest saving is often from the reduction in the amount of skilled intervention by programmers who must track a large number of files, make multiple runs, and put together information across multiple runs.

3. Empirical applications

Our main application is for the Decennial Census (300+ million × 300+ million = ~10^{17} pairs) and other large administrative lists having on the order of 1 billion records. After several tests using 2000 Decennial Census data in 2006-2008 for which that had follow-up of a sample of matches, we made very minor refinements to the software to facilitate additional matching applications. In the following, we describe the matching strategies that were developed on 2000 data and are being used for the 2010 Decennial Census. The 2010 data have similar collection modes and errors to that of the 2000 data.

3.1. 2000 Decennial Census

The basic file is the main 2000 Decennial Census file with approximately 300 million records. Each record in the file contains first name, middle initial, last name, address, census block id, date-of-birth, age, sex, relationship to head of household, and race. There is a household identifier that identifies different individuals that are in the same housing unit.

An additional file is the main ACE file of approximately 750,000 individuals representing a complete enumeration of a large sample of blocks. The ACE is matched against the Census file by blocks in order to determine overlap of files. The overlap is in turn used to estimate undercount and overcount in the Decennial Census files. The ACE is data-captured via laptop computer and the Decennial file is data-captured via a scanning technology that may induce additional typographical error beyond the types of error encountered with transcription and keypunch. Because the true match status of each ACE record is known, some of the (individual field agreement) probabilities of P(A | M) can be estimated based on matching ACE with Census. We are primarily interested in estimating P(A | M) for matching the Census with itself and for estimating the number of "missed matches" when the Census is matched with itself using a set of blocking criteria. The ACE file and some other follow-up files were used in refining the blocking strategy.

4.2. Sets of Blocking Criteria and Estimates of Missed Matches

During the initial phase, we investigated eleven blocking sets of criteria. There were 606,411 true matches identified from matching the ACE against the Census.

Table 4 Blocking Criteria and Number of Matches in Set of Pairs

1. Zip, 1st char surname	546,648
2. 1st char surname, 1st char first name, date-of-birth	424,972
3. phone (10 digits)	461,491
4. 1st three char surname, 1st three char phone, house number	436,212
5. 1st three char first name, 1st three char ZIP, house number	485,917
6. 1st three char last name, 1st three char ZIP, 1st three char phone	471,691
7. 1st char last name = 1st char first name (2-way switch) 1st three char ZIP, 1st three char phone	31,649
8. 1st three char ZIP, day-of-birth, month-of-birth	434,518
9. ZIP, house number	514,572
10. 1st three char last name, 1st three char first name, month-of-birth	448,073
11. 1st three char last name, 1st three char first name	522,584

With eleven blocking criteria, 1350 matches were missed. With the best four {1, 3, 11, 9}, 2766 matches were missed. With the best five {1, 3, 11, 9, 8}, 1966 were missed. Some of the most difficult missed matches were children in a household headed by a single or separated mother. The children were listed under two last names, date-of-birth was missing in the ACE

file, and street address was missing in the Census file. It is interesting to observe the high rate of typographical error that may, at least partially, be due to scanning error. The matching children have no 3-grams in common. Two records have a 3-gram in common if any three consecutive characters from one record can be located in another record. It is unlikely that these most difficult-to-match record pairs could be identified through any computerized procedure that uses only the information in the Census and ACE files.

Table 4. Example of missed matches (artificial data)

	Household 1		Household 2	
	First	Last	First	Last
HeadH	Julia	Smoth	Julia	Smith
Child1	Jerome	Jones	Gerone	Smlth
Child2	Shyline	Jones	Shayleene	Smith
Child3	Chrstal	Jcnes	Magret	Smith

The point of the initial tests was to determine whether we could find nearly all (~99.8%) matches with a particular set of blocking criteria. We had three other, very similar sets that we also tested. After tests in 2006-2008, we finalized on the following set of blocking strategy that is used for 2010 production matching.

Table 5. Final 2010 Decennial Census Blocking Strategy

1. Phone Number
2. State, County, BlockID, first initial of first name, first initial of last name
3. First Initial, Last Initial, Month-of-birth, Year-of-birth
4. State, County, LocalCensusOfficeID, first two letters of first name, first two letters of last name, sex, AgeGroup (0s, 10s, 20s, …)

The original set of blocking criteria with 11 blocking passes yields approximately 4×10^{12} pairs containing 99.7+% of true matches. Using the final set of blocking criteria, BigMatch does detailed comparison of 1×10^{12} pairs among the 10^{17} pairs using 40 cpus of an SGI Linux machine in approximately 15 hours. Each cpu processes ~400,000 pairs per second during the entire matching process. Based on a number of personal communications with Professor Dongwon Lee (Lee 2007) BigMatch is 40 times as fast as parallel software from Penn State (Kim and Lee 2007) and 50 times as fast as parallel software from Stanford (Kawai et al. 2006). Kim and Lee (2007) did a direct comparison between the PSU and Stanford methods.

BigMatch is written in portable C. Upon recompilation is runs fine on the SGI Linux machine, a standard Windows PC, a MacIntosh, Linux Blade servers, and HP machines running the VMS operating system. Because we have done some direct comparisons with some commercial matching software, BigMatch is on the order 80+ times as fast as some commercial software. We are unaware of any commercial software that runs on reasonably standard computer systems (but not on proprietary supercomputers) that are faster than 80 times as slow as BigMatch.

3.2. Voter registration data from Oregon and Washington

Maintaining voter registration databases is (VRDs) of great interest to a number of individuals. There have been two Federal laws (NVRA 1993, HAVA 2002) mandating certain maintenance

requirements on the lists in terms of updating from a variety of sources and removing duplicates from the main VRDs. Almost all states have adopted very simplistic exact character-by-character matching that have been developed for matching department of motor vehicle files with other files including VRD files (National Research Council 2009). To facilitate adoption of newer, more powerful methods Alvarez et al. (2009) described the basic matching scenarios for comparing lists from the States of Oregon and Washington and some extended matching scenarios using modern record linkage.

In the following, we describe some of the details in Alvarez et al. (2009) and some extensions using BigMatch technology. The OR VRD file contains approximately 2 million records and the WA VRD file contains approximately 3.5 million records. The common fields available for matching consist of first name, middle initial (or name), last name, and date-of-birth. If we perform exact character-by-character matching of OR with WA, we obtain approximately 8300 matches. The initial exact character-by-character match was performed by State of Oregon VRD staff in several steps on a MacIntosh computer in 140+ minutes (cpu time). We note that many individuals having common names such as John Smith (30,000 or more on national lists) can co-incidentally agree on date-of-birth even when the pair of records refer to different individuals. This type of co-incidental agreement on date-of-birth is well known among certain groups (see e.g., McDonald and Levitt, 2008). Indeed, by comparing individuals having the same first and last names across OR and WA, we can expect ~498 co-incidental agreements on date-of-birth (private communication from Winkler to Alvarez, Jonas, and Wright in June 2009).

After some quick trial-and-error preliminary matching, Winkler (2009b) developed the following matching strategy for the OR and WA VRD files. Matching parameters varied somewhat across the different blocking passes but were very straightforward to estimate.

Table 6. OR-WA VRD File Blocking and Matching Strategy

1. block on date-of-birth and first character surname match on last name, first name, middle initial
2. block on month-of-birth, day-of-birth, first three char last name match on last name, first name, middle initial, year-of-birth (matching on year-of-birth allows slight discrepancies)
3. block on first 3 char last name, first 3 char first name match on 2nd part of last name (i.e., skip first 3 char), match on 2nd part of first name, middle initial, month-of-birth, day-of-birth, year-of-birth (matching on month, day is exact, match on year allows discrepancy)

To better compare with outputs from the main IBM Entity Analytics Software (Jeff Butcher and Jeff Jonas communication in April 2009) Winkler (2009a) created a new version of BigMatch that allowed certain comparisons of nicknames with other names (Robert versus Bob, Susan versus Sue etc.) 'on-the-fly'. The new comparisons moved certain ideas from 1990 (also 2000, 2010) Decennial Census pre-processing routines into BigMatch but slowed BigMatch by 35%. BigMatch, with its three blocking passes, processed 194 million pairs in the OR-WA match in 9.5 minutes to obtain approximately 26,000 pairs that might be followed up. Table 7 provides a sample of the types of pairs that are found by BigMatch beyond the 8300 found during the exact character-by-character match. The last pair in Table 7 has a possible switched month and day of birth.

Table 7. Artificial Examples Representative of Real Examples from OR-WA VRD Match

	First	Last	Date-of-Birth (MMDDYY)
1a.	Robert	Smith	032151
1b.	Robrt	Smoth	032151
2a.	Robert	Smith	041875
2b.	Bob	Smith	041876
3a.	Susan	Jones	061068
3b.	Susan	Janes	100668

4. Concluding Remarks

For relatively small files having a few million records each, BigMatch is remarkably fast even on standard Windows PCs.

References

Alvarez, R. M., Jonas, J., Winkler, W. E., and Wright, R. (2009), "Interstate Voter Registration Database Matching: The Oregon-Washington 2008 Pilot Project", *Electronic Voting Technology*.

Baxter, R., Christen, P. and Churches, T. (2003), "A Comparison of Fast Blocking Methods for Record Linkage," *Proceedings of the ACM Workshop on Data Cleaning, Record Linkage and Object Identification*, Washington, DC, August 2003.

Cohen, W. W., Ravikumar, P., and Fienberg, S. E. (2003a), "A Comparison of String Metrics for Matching Names and Addresses," *International Joint Conference on Artificial Intelligence, Proceedings of the Workshop on Information Integration on the Web*, Acapulco, Mexico, August 2003.

Cohen, W. W., Ravikumar, P., and Fienberg, S. E. (2003b), "A Comparison of String Distance Metrics for Name-Matching Tasks," *Proceedings of the ACM Workshop on Data Cleaning, Record Linkage and Object Identification*, Washington DC, August 2003.

Fellegi, I. P., and Sunter, A. B. (1969), "A Theory for Record Linkage," *Journal of the American Statistical Association*, 64, 1183-1210.

Hall, P. A. V. and Dowling, G. R. (1980), "Approximate String Comparison," *Association of Computing Machinery, Computing Surveys*, 12, 381-402.

Herzog, T. N., Scheuren, F., and Winkler, W.E., (2007), *Data Quality and Record Linkage Techniques*, New York, N. Y.: Springer.

Kawai, H., Garcia-Molina, H., Benjelloun, O., Menestrina, D., Whang, E., and Gong, H. (2006), "P-Swoosh: Parallel Algorithm for Generic Entity Resolution," Stanford University CS technical report.

Kim, H.-S., and Lee, D. (2007), "Parallel Linkage," CIKM '07.

McDonald, M. P. and Levitt, J. (2008), "Seeing Double Voting: An Extension of the Birthday Problem," *Election Law Journal*, 7 (2), 111-122.

National Research Council Committee on Voter Registration Data, (2009), "Improving State Voter Registration Databases," National Academy Press (also available at http://www.nap.edu/catalog.php?record_id=12788).

Newcombe, H. B., Kennedy, J. M. Axford, S. J., and James, A. P. (1959), "Automatic Linkage of Vital Records," *Science*, 130, 954-959.

Newcombe, H.B., and Kennedy, J. M. (1962) "Record Linkage: Making Maximum Use of the Discriminating Power of Identifying Information" *Communications of the Association for Computing Machinery*, .5, 563-567.

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

Winkler, W. E. (1995), "Matching and Record Linkage," in B. G. Cox *et al*. (ed.) *Business Survey Methods*, New York: J. Wiley, 355-384 (also available at http://www.fcsm.gov/working-papers/wwinkler.pdf).

Winkler, W. E. (1999). "The State of Record Linkage and Current Research Problems," *Statistical Society of Canada, Proceedings of the Survey Methods Section*, 73-80 (longer version also available at http://www.census.gov/srd/www/byyear.html).

Winkler, W. E. (2004), "Approximate String Comparator Search Strategies for Very Large Administrative Lists," *Proceedings of the Section on Survey Research Methods, American Statistical Association*, CD-ROM (also report 2005/06 at http://www.census.gov/srd/www/byyear.html).

Winkler, W. E. (2009a). "BigMatch Version 2009.06.18". July 2009 version created by W. Yancey. Both undocumented.

Winkler, W. E. (2009b), "Preliminary Matching Results of OR and WA VRD Files," e-mail communication with output files and explanation to R. M. Alavarez, J. Jonas, and R. Wright on June 21, 2009.

Yancey, W.E., and Winkler, W. E. (2004), "BigMatch Software," computer system, documentation available at http://www.census.gov/srd/www/byyear.html.