Bigmatch: A Program for Large-Scale Record Linkage[1]

William E. Yancey
U.S. Census Bureau

Abstract

The Bigmatch program is designed to extract plausible matches from a large file using several blocking criteria without having to sort the file before each blocking run. The program has also been useful for file unduplication.

Key words: record linkage

1. Introduction

The Bigmatch program is designed to handle efficiently large-scale record linkage projects by taking advantage of the large amounts of core memory that are available on modern computers. The initial purpose of the program was to extract from a large file records that are plausible matches to records in a moderate size file. The limitation is that the moderate size file must be able to be read into core memory.

2. Record Linkage Basics

A record linkage program to find probable matching records from two files A, B should require two types of input data: matching field data and blocking field data. Under the Fellegi-Sunter model [2], the user specifies n matching fields and their matching parameters. For each matching field, the input parameters are the probabilities

$$\Pr(\gamma_i = 1|M)$$
$$\Pr(\gamma_i = 1|U)$$

designating the probabilities that the two records agree on their matching field values given that the records are a match (resp. non-match). For each matching filed, these parameters determine a comparison weight

$$w_i = \begin{cases} a_i & \text{if field values agree} \\ d_i & \text{if field values disagree} \end{cases}$$

where the agreement weight a_i is given by

$$a_i = \log \frac{\Pr(\gamma_i = 1|M)}{\Pr(\gamma_i = 1|U)}$$

and the disagreement weight d_i is given by

$$\begin{aligned} d_i &= \log \frac{\Pr(\gamma_i = 0|M)}{\Pr(\gamma_i = 0|U)} \\ &= \log \frac{1 - \Pr(\gamma_i = 1|M)}{1 - \Pr(\gamma_i = 1|U)}. \end{aligned}$$

Under the conditional independence assumption, the comparison weight w for the record pair is the sum of the field comparison weights

$$w = \sum_{i=1}^{n} w_i.$$

Record pairs with the highest comparison weights are considered the most likely matches and those with low weights are unlikely matches.

The blocking field variables are used to restrict the number of record pairs that have their comparison weights computed. If we do not use blocking, then two files of size N would require N^2 comparison weight comparisons. For large values of N, this becomes an expensive number of computations, most of them resulting in very low and irrelevant comparison weights. Therefore one restricts the compared record pairs to pairs that have some features in common.

[1] This report is released to inform interested parties of ongoing research and to encourage discussion of work in progress. The views expressed on technical issues are those of the author and not necessarily those of the U.S. Census Bureau.

This is done by specifying blocking field variables. The values of the blocking field variables for a given record form the record's blocking key and only record pairs that have the same blocking key are compared. For example, with Census records one can use for blocking variables some geographical coding (such as a zip code) or one can use personal data (such as the first few characters of a first or last name). We will call a choice of blocking field variables a blocking criterion.

However, typographical error may affect the values of the blocking variables as well as any of the matching variables, so that there may be many matching record pairs that do not agree on a particular blocking key. Hence the choice of blocking criteria is something of an art and since any choice may result in overlooking some pairs of likely matched records, a record linkage project usually involves successive runs using different blocking criteria. Recent matching projects have used up to 10 blocking passes.

2.1 The Standard Matcher

For the standard Census Bureau matching software [3], the user specifies the blocking variables, the matching variables and their matching parameters. The blocking variables determine the keys on which records must agree before they have their matching variables compared. The two files are presorted according to their key values and the records with the same key value are read in. All of the record pairs with the same blocking key value have their matching variables compared and their matching weight computed based on the input matching parameters. Once the entire block of matching weights for the given key are computed, the record linkage software uses a linear sum assignment algorithm to select the one-to-one linking that maximizes the sum of the comparison weights.

If it is desired to use another blocking criterion to bring together potential matching records, the files have to be sorted by the keys defined by the new blocking criterion before the record linkage software can be run again. For moderate size files and multiple blocking criteria, this procedure has worked very well. However, for very large files, the sorting procedure can take prohibitively long. For example, even on a fast machine, sorting a file of 300 million records can take about 12 hours. Also sorting a large file on disk requires about disk space of about three times the size of the file. If the record linkage project uses 10 blocking passes, each requiring a separate file sort, the sorting task can take too long. It can also require excessive disk space and each separate sort must be set up by the user. The Bigmatch program was designed to eliminate the sorting of a very large file.

3. The Bigmatch Program

The original purpose of the Bigmatch program was to perform record linkage operations for a large file A and a moderate size file B. The program can run through several blocking passes at once without ever sorting the large A or externally sorting the B file. By not having to sort the large A file for each blocking run, using the Bigmatch program can save cpu time, disk storage requirements, and operator intervention time. For each blocking pass the program extracts from the A file a subset of records that have a sufficiently high comparison weight to be a plausible match with some record in the B file. The trade-offs with the standard Census matcher are that the B file must be small enough to fit entirely into core memory and that Bigmatch does not compute one-to-one matching within the blocks.

As with the standard matcher, the user provides the blocking and matching variables. However, with Bigmatch the blocking

variables and matching fields and parameters are provided for several blocking runs at once. Each blocking run can have its own blocking fields, matching fields, matching field parameters, and output cutoff values. After reading in this blocking strategy data, the program reads the entire B file into memory. Thus the size of the B file is limited by the amount of available core memory. For each blocking criterion, the keys from the B file are found, each key has a list of the B file records with that key, and the key lists are sorted in memory. The key sorting is done using a version of Quicksort optimized for strings by Bentley and Sedgewick [1]. Once this is done, the program reads in records from the A file one at a time. For each blocking criterion, the A record key is found, this key is search for in the B file key list, and if the key is found in the list then the A record has its matching weight computed with each B record with the same key. After processing the A file record for each blocking criterion, the next A record is read in.

The program produces two kinds of output files. For each blocking criterion, there is a file of A records that had a matching weight above a cutoff value for some B record with the same key. Thus for each blocking criterion, the Bigmatch program produces a subfile of the A file that contains plausible matching records. Presumably one of these subfiles of plausible matching records is of more manageable size, so if desired, the standard Census matcher can then be used with such a subfile and the B file to produce a set of one-to-one linked records. The other type of file prints out the pairs of matching field values of the A and B records with matching weights above a cutoff value. These files are intended for analytic purposes to assess how the program is doing.

The Bigmatch program is written in C code and has been compiled an run on UNIX, VAX, and Windows platforms. On our UNIX workstation it has been timed to compute comparisons for 100 million record pairs per second.

3.1 File Unduplication

As it turns out, a lot of the application of the Bigmatch program has been for a somewhat different purpose: finding duplicate records within a file. In this case, the two files A, B are the same file. This means that the file to be unduplicated has to fit into core memory. For example, a 100 million record Census file and its blocking key tables can fit into 4 gigabytes of memory. At the Census Bureau, we have UNIX workstations with 32 gigabytes of memory. If the file has multiply duplicated records, since the Bigmatch program does not have one-to-one matching, all likely duplicated pairs are reported. For this application, the output files containing the matching field data for the record pairs are probably more relevant than the selected subfiles. On the Census UNIX machine, a 600K record file can be run though 10 blocking passes without any intermediate user intervention in only slightly more time than it takes to run one blocking pass with the standard matcher.

4. Program Enhancements

4.1 Used Record List

If a pair of records is very similar, they are likely to be brought together under several different blocking criteria. For each new A file record, the Bigmatch program keeps track of which B records have been compared to it. Thus if the same B record comes up in a subsequent blocking run, it does not get the comparison weight computation repeated. This can save some computing time. It also has implications for blocking strategy design. Often blocking runs are designed so that the first blocking criterion is very specific so that only very similar records

are brought together. Subsequent blocking runs can relax the blocking criteria to allow a wider selection of records to be compared. If the blocking criteria are defined so that the pairs of records B_i brought together by the i^{th} blocking run are nested

$$B_1 \subset B_2 \subset \cdots \subset B_m$$

then the output of the second blocking run will have only records not found in the first blocking run. In general, each successive blocking run will have only new records output.

4.2 Blank Keys

The purpose of blocking is to bring together a reasonable number of records that show some common characteristics. However, if blocking fields are chosen that happen to have blank values frequently in the file, then the blocking scheme can bring together a lot of pairs of records with blank keys, possibly resulting in a lot of calculation involving record pairs with nothing in common. For each blocking run, the Bigmatch program allows the user to designate blocking fields as essential so that if any of the designated key components are blank, then the record is skipped for comparison computation.

4.3 Unduplication

If the Bigmatch program is being used to detect duplicate records within a file, the program avoids some redundant computation. This requires that each record in the file has a unique sequence number. By comparing record sequence numbers, the program can avoid computing matching data for a record paired with itself (a, a) and for a given record pair (a, b), avoid computing for the symmetric pair (b, a). This saves some time and avoids some unnecessary output. However, if a record is duplicated k times, there can still be $\binom{k}{2}$ corresponding record pairs printed out.

5. Program Code

The C code for the Bigmatch program can be obtained by contacting william.e.yancey@census.gov or william.e.winkler@census.gov. A manual for using the program can be found at http://www.census.gov/srd/www/byname.html.

REFERENCES

Bentley, J.L., and Sedgewick, R.A. (1996). "Fast Algorithms for Searching and Sorting Strings." *Proceedings of the Eights ACM-SIAM Symposium on Discrete Algorithms.* pp.360–369.

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

Winkler, W. E. (1995). "Matching and Record Linkage." in B. G. Cox *et. al.* (ed.) *Business Survey Methods.* New York: J. Wiley, pp. 355–384