Proceedings of Statistics Canada Symposium 2014
Beyond traditional survey taking: adapting to a changing world

Useful functionalities for record linkage

Martin Lachance[1]

Abstract

There is a wide range of character-string comparators in the record-linkage field. Comparison problems arise when factors affect the composition of the strings (for example, the use of a nickname instead of a given name, and typographical errors). In these cases, more sophisticated comparators must be used. Such tools help to reduce the number of potentially missed links. Unfortunately, some of the gains may be false links. In order to improve the matches, three sophisticated string comparators were developed; they are described in this paper. They are the Lachance comparator and its derivatives, the multi-word comparator and the multi-type comparator. This set of tools is currently available in a deterministic record-linkage prototype known as MixMatch. This application can use prior knowledge to reduce the volume of false links generated during matching. This paper also proposes a link-strength indicator.

Keywords: Pre-processing; string comparator; false links.

1. Introduction

Administrative data are likely to play a more prominent role in surveys over the next several years, for purposes such as reducing respondent burden, lowering collection costs, etc. However, record linkage (probabilistic, deterministic and other types) has gained functionalities over the years. This paper introduces additional tools. The needs identified and the functionalities presented in this paper are the product of 10 years of experience as a record-linkage user, mainly with a view to improving frame coverage, or of constant interaction with other users. The five major needs I have identified are as follows: (1) simplify data formats, (2) develop sophisticated string comparators, (3) improve pre-processing, (4) limit linkage errors, and (5) provide quality indicators. In this paper, I present a list of suggestions aimed at addressing these needs, along with the functionalities that I have developed, implemented and tested.

2. Addressing the needs

2.1 Simplifying data formats

One of the first problems that arise when using administrative data is how to overcome difficulties with the data format. This is especially true when multiple data sources are used in an effort to improve the coverage of a sampling frame. There is no guarantee that all the sources will be in a similar format for linkage with the frame. In addition, the record-linkage application may force users to accept content, field type and field length constraints. The application may require well-defined input variables, like first name, last name, and date of birth. The data then have to be manipulated to comply with these constraints. Once this is done, it may be difficult to consider cross comparisons, such as an inverted first name and last name compared with a full name.

Being able to treat each record in a source as a single variable, i.e., a single string of characters, provides a great deal of flexibility, whether records represent people, households or something else. Users can partition this string into as many ‘pieces’ as they like and then combine them at will to identify potential matches. Table 2.1-1 shows an

[1]Martin Lachance, Statistics Canada, 100, Tunney’s Pasture Driveway, Ottawa, Ontario, Canada, K1A 0T6, Martin2.Lachance@statcan.gc.ca.

example of partitioning, which allows us to compare the month and the day of the year, in case of an inversion in the data, and to manage inversions of first and last names. Comparisons are made possible by partitioning and using a simple concatenation operator, '+'.

Table 2.1-1
Comparisons of partitioned single-character strings

		File 1					File 2			
Records		Martin	Lachance	2014 - 10 - 30			Lachance Martin	2014 / 30 / 10		
Partitions		firstname1	lastname1	Y1	M1	D1	name2	Y2	M2	D2
Successive comparisons between the files	1	(firstname1+lastname1) and (Y1+M1+D1)					(name2) and (Y2+M2+D2)			
	2	(lastname1+firstname1) and (Y1+M1+D1)								
	3	(firstname1+lastname1) and (Y1+D1+M1)								
	4	(lastname1+firstname1) and (Y1+D1+M1)								

2.2 Developing sophisticated string comparators

A lot of character-string comparators attempt to overcome phonetic, typographical and other errors. Most of them treat each character string to be compared as a single sequence of characters, i.e., a single word when the spaces between the words are ignored. For example, 'Martin Lachance' would be transformed into a single word, 'MARTINLACHANCE'. These are sophisticated single-word comparators. We differentiate between them and sophisticated multi-word comparators, which are capable of individually comparing each word in a character string. Multi-word comparators not only improve the flexibility of comparisons, but also may help establish stronger links than single-word comparators. In addition, being able to manage numbers in character strings (for example, '100' in '100 Symposium Avenue') using multi-word comparators yields even more flexibility. I will refer to comparators of this kind as sophisticated multi-type comparators.

This section describes three character-string comparators. They are part of a range of comparators available in the MixMatch deterministic record-linkage prototype (Lachance 2014). They are the Lachance comparator, the multi-word comparator and the multi-type comparator. The last two are based on the Lachance comparator.

2.2.1 The Lachance comparator

First, it is important to define what I will call a 'legitimate' match. A legitimate match is an imperfect match, based on pre-established criteria, between two character strings, each treated as a single word. In addition to perfect matches, the Lachance comparator identifies legitimate matches that meet pre-set tolerance parameters. The algorithm associated with the Lachance comparator is based on a search tree. It involves identifying as many common characters as possible between two single words, provided the characters appear in the same order in each word. On the basis of the number of common characters, a degree of similarity between the two words is calculated using a simple formula. The degree of similarity, or score, is equal to the ratio of twice the number of common characters that appear in the same order in each word to the sum of the lengths of the two words. For example, a comparison of 'MARRTINLACHANCE' and 'MARTINLCHANCE' would yield a ratio of 26/28, or 0.93. Table 2.2.1-1 presents a few examples of comparators from the literature (Porter and Winkler 1997) and compares their scores with those of the Lachance comparator.

The tolerance parameters of the Lachance comparator have two components: the minimum number of characters required in order to obtain a match (perfect or legitimate) and the degree of similarity expressed as a percentage. The Lachance comparator has been used at Statistics Canada for several years, and parameters of a minimum of four characters with 85% similarity generally provide the best gains while limiting the number of false links. In addition, since the performance of the algorithm slows as the length of the words to be compared increases, and since multi-word comparators provide greater flexibility, it is better to start with a multi-word string comparator when the character string consists of more than one word.

Table 2.2.1-1
Comparisons involving first and last names

Character strings compared		String comparator scores			
		Jaro	Winkler	Lynch	**Lachance**
SHACKLEFORD	SHACKELFORD	0.970	0.982	0.989	0.909
DUNNINGHAM	CUNNIGHAM	0.896	0.896	0.931	0.842
NICHLESON	NICHULSON	0.926	0.956	0.977	0.889
JONES	JOHNSON	0.790	0.832	0.874	0.667
MASSEY	MASSIE	0.889	0.933	0.953	0.833
DWAYNE	DUANE	0.822	0.840	0.896	0.727
SEAN	SUSAN	0.783	0.805	0.845	0.667

2.2.2 Multi-word and multi-type comparators

The multi-word string comparator follows a two-step process. First, it is based on the successive identification of words that match perfectly, and then of words that match legitimately, followed by the identification of words included within one another, starting at the beginning (for example, Martin compared with M). This is the 'agree' component of the comparator. Second, all remaining words are deemed to be conflicts or extras. This is the 'disagree' component. The comparator forms pairs with the words remaining on each side; these are the conflicts. After as many pairs as possible have been formed with each remaining word appearing only once, any words left on one side are the extras. Table 2.2.2-1 shows a fictitious example in which the legitimate component is defined by applying the Lachance comparator. The tolerance parameters, set by the user, determine which pairs of character strings will be considered links. Note that while the example produces a link, the same tolerance parameters would not have produced a link if 'M. MARTIN RICHARD LACHANCE' had been compared with 'LACNANCE MARTIN RENE,' because of the conflict between 'RICHARD' and 'RENE.'

Table 2.2.2-1
Application of the multi-word Lachance string comparator

Character strings compared	**Tolerance parameters**				**Comparison status**
	Set by the user		Components identified	Parameter status	
'M. MARTIN R. LACHANCE' versus 'LACNANCE MARTIN RENE'	Agree	1 perfect match, minimum of 4 characters	MARTIN, MARTIN	Met	Link
		Legitimate matches: minimum of 4 characters, 85% in common	LACHANCE, LACNANCE	1 pair identified	
		Inclusions: minimum of 1 character	RENE, R	1 inclusion identified	
	Disagree	0 conflicts permitted	None!	Met	
		(no constraint on the extras)	M	1 word identified	

The multi-word string comparator has a variant: the multi-type string comparator. It has the same tolerance parameters as the multi-word string comparator, but it imposes one additional constraint. It requires a minimum number of perfect matches for isolated numbers (for example, '100' in '100 Symposium Avenue'). This comparator may prove useful for such tasks as address processing.

2.3 Improving pre-processing

Even with sophisticated string comparators, pre-processing of the data is almost inevitable in many cases. Pre-processing can be broken down into two main steps. The first step is a data clean-up, which essentially involves removing undesirable elements from the character strings (e.g., periods and commas). The second step is recoding, in which the focus is on standardizing values to boost matches, which might otherwise be missed. The conversion of given names to nicknames, such as 'William' to 'Bill,' or vice versa depending on the software being used, is a typical example of recoding.

Table 2.3-1
Use of various string comparators following various data recodings

Character strings compared	**Pre-processing**				**String comparator used**	**Comparison status**
	After clean-up	**After recoding**				
		#	String 1	String 2		
'WILLIAM R. SYMPOSIUM' versus 'RON BILL SYMPOSIUM'	'R.' becomes 'R'	1	WILLRSYMP	RONBSYMP	Perfect	No link
		2	BILL R SYMPOSIUM	RON BILL SYMPOSIUM	Perfect	No link
		2	BILL R SYMPOSIUM	RON BILL SYMPOSIUM	Multi-word (parameters specified)	Link

Generally, we identify only perfect matches after the data are recoded. Nevertheless, there may be some advantages to considering sophisticated string comparators, such as greater flexibility and the possibility of gaining links, as shown by the example in Table 2.3-1. Note that while this also involves risks, it is still possible to limit linkage errors.

2.4 Limiting linkage errors

Pre-processing and the use of sophisticated string comparators can produce links that are difficult to identify, such as the link between 'MARTIN LACHANCE' and 'MARTIN LACAHNCE.' However, they can also produce their share of false links. For example, consider two names that are perfectly valid and distinct but are matched because they are very similar, such as 'MARTIN LACHANCE' and 'MARTIN LACASSE,' or 'ALEXANDRE' and 'ALEXANDRA,' representing two people of opposite sex. So, how can we prevent false links? An attractive option is to take advantage of prior knowledge and incorporate it into the matching process in the form of a list, which I refer to here as an exclusion list. Table 2.4-1 contains an exclusion list covering first names with similar spellings. The idea is to list pairs of character substrings that may turn up on opposite sides, in one of the two character strings being compared; for example, 'GÉRARD' would be on one side and 'GÉRALD' on the other. If the substrings are found, the result is no link for the strings being compared.

Table 2.4-1
Exclusion list for first names

Words in the first character string	Words in the second character string
DAVIS	DAVID
MARC	MARCO
GERALD	GERARD
…	…

The benefits of preparing such lists are substantial. The lists help to prevent false links, and exclusion lists can be shared among record-linkage projects, thereby reducing the amount of manual processing for all concerned. With the addition of such lists, the process of comparing character strings following pre-processing ultimately has three components: a prevention component (exclusion lists), a recoding component (conversion lists) and the string comparator used. The examples in Table 2.4-2 summarize this sequence.

Table 2.4-2
Complete evaluation of character-string pairs

	Pair #1	Pair #2	Pair #3
Original strings	MARTIN LACHNACE MARTY LACHANCE	MARCO SYMPOSIUM MARC SYMPOSIUM	M. NOVEMBER M. DECEMBER
After application of an exclusion list	MARTIN LACHNACE MARTY LACHANCE	Exclusion: MARC, MARCO	M NOVEMBER M DECEMBER
After recoding (conversions)	MARTY LACHNACE MARTY LACHANCE		M NOVEMBER M DECEMBER
After use of the Lachance string comparator (4, 85%)	MARTYLACHNACE MARTYLACHANCE		MNOVEMBER MDECEMBER
Comparison result	Link	No link	No link

2.5 Providing quality indicators

Low false-match rates and low missed-match rates are favourable global quality indicators in a record-linkage process. At the component level of the linkage process, such as the character-string comparators, the outcome of a matching attempt is generally expressed as a percentage indicating the strength of the link (see Table 2.2.1-1). However, it is possible to provide a measure of link strength that is more informative than a simple percentage and that can be determined for each string comparator. The PLICE quality indicator was designed to satisfy this definition. PLICE expresses the outcome of several small matches within a process of comparing two character strings. It is defined in Table 2.5-1. It applies to both single-word string comparators (such as the Lachance comparator) and multi-word or multi-type string comparators. For example, comparators such as the Jaro and the Winkler comparators, which are similar to the Lachance comparator, would contribute only to the 'L' in PLICE, i.e., the 'legitimate' component. In particular, this indicator makes it easier to separate links by simultaneously placing them in descending order for the 'agree' part and ascending order for the 'disagree' part.

Table 2.5-1
Definition of the PLICE indicator

	Letter	Definition	Examples of links found	
			Martin Henry Lachance versus Lachance Martin Harry	John Symposium versus J. R. Symposuim
Agree	**P**	Number of **P**erfect matches observed	2	0
	L	Number of **L**egitimate matches observed	0	1
	I	Number of **I**nclusions observed	0	1
Disagree	**C**	Number of **C**onflicts observed (if accepted in the use of a multi-word comparator)	1	0
	E	Number of **E**xtra words	0	1

3. Application

All the suggestions made in Section 2 were implemented and incorporated in the MixMatch prototype, including the PLICE quality indicator. MixMatch performs deterministic linkages. It is a Statistics Canada in-house application; its name is based on the expression 'mix and match,' which indicates its high level of flexibility. Recently, it was completely rewritten in the SAS programming language as part of a research project, on the basis of a previous version in the C programming language that had been updated over several years. The SAS version is a significant improvement. Of course, the application has evolved a great deal over the years, but, since the earliest versions, its development has always taken place through constant interaction with users so that it could be tailored as much as

possible to their actual needs. As a result of this approach, I have identified three qualities that are near and dear to users: simplicity, flexibility and performance. These are the reasons for the success of the application to date.

The operating principle behind MixMatch is relatively simple. Three types of linkage are possible: (1) linking records from two files; (2) identifying records in the same file that are similar to each other; and (3), given already formed record pairs, identifying the pairs that constitute valid links. No matter how the record pairs are formed, the application applies a set of logical rules in sequence to evaluate them. For example, the first logical rule might involve identifying pairs of individuals who are perfect matches for last name, first name, date of birth and postal code. The rule would have four conditions. The second rule might involve identifying which of the remaining pairs are perfect matches for date of birth and postal code; the multi-word string comparator described in this article would be used on a combination of the last name and the first name, with specific tolerance parameters. More rules could follow suit. It is worth noting that the order in which the conditions for a rule are evaluated was optimized to reduce evaluation time, depending on the complexity of the comparators used. Better still, the application also ensures that each condition is evaluated only once, whether it occurs in more than one rule or not. In the end, record pairs that do not satisfy any of the rules are not considered by the application to be links. Lastly, the sequence of rules, provided by the user, combined with the PLICE strength indicators for each linked pair, can be used to order the links.

4. Conclusion

The proposed options for meeting the major needs identified in this paper were implemented and tested at Statistics Canada. The deterministic record-linkage application MixMatch treats the data from records as a single character string that the user can partition as he or she sees fit. Sophisticated string comparators have been developed, such as the Lachance comparator and the multi-word and multi-type string comparators; these are generating a high level of interest. The internal structure of MixMatch also provides users with a great deal of flexibility in pre-processing data. In particular, a sequence of words can be recoded iteratively into another sequence. The application also enables error prevention using exclusion lists. In addition, a detailed link-strength indicator, PLICE, was developed and implemented. In summary, the overall approach of simplicity, flexibility and performance played a key role in developing and deploying the tools presented in this paper. That said, while MixMatch was constructed to control processing time, there is still room for improvement. And, beyond this prototype, the ideas presented in this article can also be used in probabilistic record linkage. In fact, the incorporation of MixMatch's features into Statistics Canada's official record-linkage application, G-Link (Statistics Canada 2014), has begun.

References

Lachance, M. (2014), "MixMatch 1.2 - User Guide", unpublished document, Ottawa, Canada: Statistics Canada.

Porter, E. H., and Winkler, W. E. (1997), "Approximate String Comparison and its Effect on an Advanced Record Linkage System", Record Linkage Techniques - 1997, *Proceedings of an International Workshop and Exposition, Federal Committee on Statistical Methodology*, pp. 190-199.

Statistics Canada (2014), "User Guide for G-Link version 3.0", Ottawa, Canada: Statistics Canada.