Australian Government
Department of Defence
Defence Science and
Technology Organisation

Resolving Partial Name Mentions Using String Metrics

Jyotsna Das and Poh Lian Choong

Command, Control, Communications and Intelligence Division
Defence Science and Technology Organisation

DSTO-RR-0318

ABSTRACT

Information Extraction is concerned with discovering entities, relationships and events from text. Before relationships and events can be discovered accurately, it is critical to resolve all mentions of the same entity. This process is known as coreference resolution. Coreferenced mentions of entities can occur in a number of forms including pronominal mentions; partial name mentions; and through the use of honorifics. This report focuses on addressing the problem of resolving partial name mentions to their canonical form within a text document using character-based string metrics. Based on a review and investigation of some of the main character-based string metrics, we developed a method to resolve partial name mentions within a document. This method applies the Jaro-Winkler string comparator and a variation of the Smith-Waterman string similarity measure. The method was applied to name mentions sourced from a sample of emails with a precision of 97%, and news articles with a precision of 100%.

Published by

Command, Control, Communications and Intelligence Division
DSTO Defence Science and Technology Organisation
PO Box 1500
Edinburgh South Australia 5111 Australia

Telephone: (08) 8259 5555
Fax: (08) 8259 6567

AR-014-065
December 2007

APPROVED FOR PUBLIC RELEASE

Resolving Partial Name Mentions Using String Metrics

Executive Summary

This report presents results of research undertaken in information extraction under Tasks INT 07/020 and NS 07/201.

Information Extraction is concerned with discovering entities, relationships and events from text. Before relationships and events can be discovered accurately, it is critical to resolve all mentions of the same entity. This process is known as coreference resolution. Coreferenced mentions of entities can occur in a number of forms including pronominal mentions such as *he*, *she*, or *I*; partial name mentions such as *Bush* for the entity *George W Bush*; and through the use of honorifics such as *the President*.

This report focuses on addressing the problem of resolving partial name mentions to their canonical form within a text document.

The report reviews some of the main character-based string metrics. It describes some investigations we did on name mentions using these metrics to determine if they could be applied to the problem of resolving partial name mentions occurring within a text document.

Based on our findings, we developed and tested a method to address the resolution of partial name mentions in text documents. The method is based on the selective application to this problem of two well known character-based string metrics. The first is the Jaro-Winkler string comparator and the second is a variation of the Smith-Waterman string similarity measure.

The method was applied to name mentions extracted from a sample of news articles and emails with highly accurate results. A precision of 100% was obtained for name mentions extracted from the news articles, and 97% for those extracted from the emails.

Authors

Jyotsna Das
Command, Control, Communications and Intelligence Division

Since joining the Information Technology Division (now Command, Control, Communications and Intelligence Division) of DSTO, Jyotsna Das has worked on R&D projects aimed at facilitating the integration of Database and Geospatial Information Systems into wider applications. More recently she has worked in content extraction, including the evaluation of commercial and other systems, and the development and deployment of prototype systems to address specific client requirements. Her current research interests include application of statistical and machine learning techniques to content extraction and text mining.

Poh Lian Choong
Command, Control, Communications and Intelligence Division

Dr. Poh Lian Choong received her B.E. (Hons) degree in electrical and electronic engineering from the University of Auckland, New Zealand and Ph.D. degree from the University of Western Australia, Australia. Between 1998 - 2000, she has worked in Surveillance Systems Division as a Research Scientist and from 2001 she has worked in the Command and Control Division (now C3ID) as a Senior Research Scientist. Her research interests include: (1) content extraction; (2) data mining; and (3) development of Artificial Neural Networks, Syntactic Pattern Recognition, Maximum Entropy and Bayesian methodologies in application to automatic pattern recognition and classification.

Contents

1. Introduction

Information Extraction is concerned with discovering entities, relationships and events from text. To accurately capture the text content, it is necessary to resolve all mentions of the same entity. This process is referred to as coreference resolution. Coreference resolution is an essential element of discovering entities. Hence it is also critical to the accurate discovery of relationships and hence events.

Coreferenced mentions of named entities can occur in a number of forms, including:

- Pronouns such as *he*, *she* or *they*
- Partial names or nicknames such as *Bush* for the complete entity mention *George W. Bush*.
- Through the use of honorifics such as *The President*.

Different techniques are needed to resolve these types of coreference mentions. This report will be concerned with reviewing techniques that address the second type of coreference mention.

The problem of coreference resolution of partial names and nicknames is closely related to name matching, record linkage and duplicate detection. Approaches that address these have been applied to coreference resolution with some success [5]. These fall under two broad categories – Character based and Token based string metrics. This report focuses on the application of some character-based string metrics to this problem. These are reviewed in Sections 2 to 5.

2. Levenshtein Minimum Edit Distance

The *Levenshtein Minimum Edit Distance* [5] is one of the simplest metrics for determining string similarity. When comparing two strings, this method considers the minimal set of possible string edit operations needed to convert one string of the pair to the other string. The edit operations can be a combination of character *insert*, *delete*, *substitution*, each of which carry a penalty (generally 1), and *copy* (or match) which carries no penalty. The distance between a pair of strings is defined as the minimum number of edit operations needed to be performed on the first string of the pair in order to convert it to the second string.

The algorithm is detailed in the following section.

2.1 The Algorithm

The *Levenshtein Minimum Edit Distance* is measured in terms of the lowest cost sequence of edit operations that convert source string *s* to target string *t*. String edit operations include:

- Character *insert*

- Character *delete*
- Character *substitution*
- Character *copy*

Each operation is assigned a cost. The Levenshtein function assigns a unit cost to the edit operations insertion, deletion and substitution; and a cost of zero to copy.

The sequence of operations is represented as a two-dimensional edit distance array, D, of string edit distances. To compute the array, let the two strings to be compared be denoted by X and Y with lengths m and n respectively. Let D_{ij} represent the minimum edit distance obtained by aligning a substring of X ending at position i with a substring of Y ending in position j. Then array D is initialised as follows:

$$\left.\begin{array}{c} D_{ij} = 0 \quad for \quad i = j = 0 \\ D_{i0} = i \quad for \quad i \neq 0 \\ D_{0j} = j \quad for \quad j \neq 0 \end{array}\right\} \qquad (1)$$

The following recurrence relation can be used to compute elements of the array D:

$$D_{ij} = \min\begin{cases} D_{i-1\,j-1} + d(X_i, Y_j) \ \textit{for copying or substituting a character in string } X \\ D_{ij-1} + 1 \quad \textit{for inserting a character in string } X \\ D_{i-1\,j} + 1 \quad \textit{for deleting a character in string } X \end{cases} \qquad (2)$$

where, when a character in String X is copied:

$$X_i = Y_j \quad and \quad d(X_i, Y_j) = 0 \qquad (3)$$

and, when a character in String X is substituted:

$$X_i \neq Y_j \quad and \quad d(X_i, Y_j) = 1 \qquad (4)$$

The Levenshtein Minimum Edit Distance is found in the cell D_{mn}.

A dynamic programming algorithm based on this recurrence relation is used to calculate the *Levenshtein Minimum Edit Distance*.

Table 1 shows the output from the application of the above recurrence relation to the example strings in "OSAMA" and "USSAMA". The array values are computed with insert, delete, and substitute costs set to 1; and copy cost set to 0. The resulting minimum edit distance for the example strings is 2 (shown in enlarged, bold font in Table 1). This corresponds to the value in the array cell D_{56}, where the cell row and column indexes start at 0.

Table 1: Levenshtein Minimum Edit Distance array for strings "OSAMA" and "USSAMA"

		U	S	S	A	M	A
	0	1	2	3	4	5	6
O	1	1	2	3	4	5	6
S	2	2	1	2	3	4	5
A	3	3	2	2	2	3	4
M	4	4	3	3	3	2	3
A	5	5	4	4	3	3	**2**

The *Levenshtein Minimum Edit Distance* ranges from a value of zero upwards, with zero representing an exact match between the two strings being compared.

A trace back through the array identifies a possible set of string edits needed to transform the source string to the target string. The trace back process is discussed in the following section.

2.2 Trace Back Path for the Levenshtein Minimum Edit Distance

Once values of the edit distance metrics (D) array cells are determined, a trace back path can be constructed to find an optimal set of string edits traversing the array in reverse from the seed cell, D_{mn} where m is length of the source string, and n is the length of the target string. There can be more than one trace back path – the target string can be constructed from a source string by more than one set of string edits. An algorithm for constructing one such trace back path is as follows:

```
Begin
      Set seedcell indexes to values of cell Dmn;
      While i>0 and j>0
      Find-Parentcell;
      Set Parentcell indexes to the seedcell indexes;
      End-of-While;
End;
```

For a given cell *(i, j)*, a parent cell is found as follows:

```
Begin (Find-Parentcell)
      Set seedcell to Dij;
      Check for Insert Operation:
            If Dij = Dij-1 + InsertCost
            If value in Dij-1 <= value in seedcell
            Then set seedcell to Dij-1;
      Check for Delete Operation:
            If Dij = Di-1j + DeleteCost
                  If value in Di-1j <= value in seedcell
                        Then set seedcell to Di-1j;
      Check for Substitute or Copy Operation:
            If Dij = Di-1j-1  or Dij = Di-1j-1 + SubstituteCost
                  If (i-1>0 and j-1>0)
                        If value in Di-1j-1 <= value in seedcell
                              Then set seedcell to Di-1j-1;
            If (j-1=0 and i-1>0) (out of columns so force a delete)
                  Then set seedcell to Di-1j;
            If (i-1=0 and j-1>0) (out of rows so force an insert)
                  Then set seedcell to Dij-1;
            If (i-1=0 and j-1=0)(end of trace back)
                  Then set seedcell to Di-1j-1;
End (Find-Parentcell);
```

Thus each cell containing a non-zero value has at least one parent and may have up to three. For any cell $D_{ij} > 0$, a *trace back path* will start from this cell, stepping successively from it to a parent cell, and terminating as soon as the next step in the path reaches a cell with indices (*i,j*) that are zero.

For the example strings in Table 1, the trace back path for the *Levenshtein Minimum Edit Distance* metric calculation is shown in Table 2.

Table 2: Levenshtein Trace Back Path for strings "OSAMA" and "USSAMA"

		U	S	S	A	M	A
	0	1	2	3	4	5	6
O	1	1					
S	2		1	2			
A	3				2		
M	4					2	
A	5						**2**

An optimal trace back path has been identified to transform the source string to the target string and is shown in Figure 1.

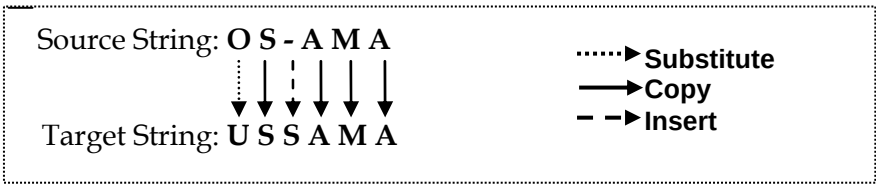


Figure 1: String edit operations on "OSAMA" to convert it to "USSAMA"

To transform the source string, "OSAMA" to the target string "USSAMA", the trace back path identified in Table 2 involves:

- one character *substitution* operation represented by a dotted arrow
- four character *copy* operations represented by the solid arrows
- one character *insert* operation represented by the dashed arrow

2.3 Normalised Levenshtein Minimum Edit Distance

When pairs of strings are compared using *Levenshtein Minimum Edit Distance*, the raw score can be an inaccurate indicator of the closeness of the pairs of strings being compared. A larger score should indicate a worse match, but this may not always be the case. For example, consider the string pairs ("abcd", "abbd") and ("abcdefg", "abbdefg"). The *Levenshtein Minimum Edit Distance* for both pairs is 1, representing a single substitution operation on the third character in both string pairs. Yet, based on string size, the longer string pair represents a proportionally closer match.

One way of addressing this apparent inconsistency is to use the length of the longer of the two

strings to "normalise" the *Levenshtein Minimum Edit Distance*. This will have two benefits:

1. It will restrict the value of the metric to fall in the range 0-1;

2. It will provide a more effective yardstick of the extent of the match between string pairs.

For two strings S_1 and S_2 of lengths $|S_1|$ and $|S_2|$ respectively, the *Normalised Levenshtein Minimum Edit Distance* (N_{mn}) is computed as follows:

1. Compute the Levenshtein Minimum Edit Distance (D_{mn}) as described in Section 2.1.

2. Compute the Normalised Levenshtein Minimum Edit Distance as follows:

$$N_{mn} = \frac{D_{mn}}{Max(|S_1|, |S_2|)} \qquad (5)$$

The values of the *Normalised Levenshtein Minimum Edit Distance* range from 0 to 1, with 0 indicating an exact match between the two strings being compared. In the example discussed in Figure 1, the *Normalised Levenshtein Minimum Edit Distance* for the two pairs of strings is 0.25 and 0.1429 respectively, corroborating the fact that the longer pair represents a proportionally closer match.

3. Detecting Longest Common Sub-sequences

The *Smith-Waterman Similarity Measure* is a classical method used in molecular sequence analysis [6][7] to find local alignments in two strings. By doing so, it allows the string metric to compensate for mismatching text at the beginning and end of strings. An implementation of the method as a dynamic programming scheme is described in the following section, and is based on that described in [5][7].

3.1 The Algorithm

The Smith-Waterman method compares segments of all possible lengths in two strings, and identifies common sub-sequences. The longest common subsequence maximises the similarity measure. A scoring matrix is computed for all possible matches and string edit operations. The cost of opening a gap (i.e. inserting the first character when aligning) can be given a higher cost than that of extending that gap (i.e. inserting additional characters)[1]. A positive contribution is assigned to the score for each character match, and a penalty for character insertions, deletions and substitutions. Thus a high metric value represents a better match.

[1] This is known as *affine gap* cost [5]. An adaptation of the Smith-Waterman metric that incorporates gap cost is discussed in [12], however is not pursued in this report.

Let the two strings to be compared be denoted by X and Y with lengths m and n respectively. Let D_{ij} be the maximum score obtainable by aligning a substring of X ending at position i with a substring of Y ending at position j. The following recurrence relation computes elements of array D as follows:

Assign initial cell values:

$$\left.\begin{aligned} D_{i0} &= 0 \\ D_{0j} &= 0 \end{aligned}\right\} \qquad (6)$$

If $X(i) = Y(j)$,

$$D_{ij} = \max\begin{cases} 0 \\ D_{i-1,j} - G & for\ insert \\ D_{i,j-1} - G & for\ delete \\ D_{i-1,j-1} + h & for\ copy/match \end{cases} \qquad (7)$$

If $X(i) \neq Y(j)$,

$$D_{ij} = \max\begin{cases} 0 \\ D_{i-1,j} - G & for\ insert \\ D_{i,j-1} - G & for\ delete \\ D_{i-1,j-1} - r & for\ substitute \end{cases} \qquad (8)$$

Where:

h is the positive contribution for a match;

G is the Gap Cost for an insertion or deletion.

r is the cost of substituting a character.

A dynamic programming algorithm based on this relation is used to compute the *Smith-Waterman Similarity Measure*.

For the example strings in Table 1, the Smith-Waterman similarity metric array is shown in Table 3. The metric array was computed using a default value for the insert, delete, substitute and match costs of 1. The largest resulting value for the Smith-Waterman similarity metric is 4. This represents the termination point of the best matching substrings between the two strings that are being compared.

The Smith-Waterman similarity metric is useful for discovering significant close matches between substrings of X and Y, where significant is defined in terms of a chosen threshold score. The threshold will depend on the costs of the various string edit operations, and the context. According to Irving [7] for unstructured text a relatively small threshold is sufficient, whereas for more structured or formatted text, a higher threshold will be necessary to correctly discover closely matching substrings.

In contrast to the Levenshtein minimum edit distance, the Smith-Waterman metric ranges from a value of zero upwards, with a high value representing a closer match between the two

strings being compared.

Table 3: Smith-Waterman String Similarity Metric array for strings "OSAMA" and "USSAMA"

	U	S	S	A	M	A
O	0	0	0	0	0	0
S	0	1	1	0	0	0
A	0	0	0	2	1	1
M	0	0	0	1	3	2
A	0	0	0	1	2	**4**

Once values of the *D* array are determined, a trace back path can be constructed to find an optimal local alignment ending at position *i* in string *X* and position *j* in string *Y*. An algorithm for constructing the trace back path based on that outlined in Irving's paper [7] has been implemented and is described in the following section.

3.2 The Trace Back Path for the Smith-Waterman Similarity Measure

Once values of the edit distance metric (D) array cells are determined, a trace back path can be constructed to find an optimal set of string edits by traversing the array in reverse from the seed cell, Dij where i ≤ m, the length of the source string, and j ≤ n, the length of the target string. The seed cell is the cell with the largest value that is greater than the nominated threshold to define "significant" near-matching substrings. The default threshold is 1. There can be more than one trace back path – the target string can be constructed from a source string by more than one set of string edits.

An algorithm for constructing one such trace back path is as follows:

```
Begin
      Set threshold value for trace back (default is 1);
      Set seedcell indexes to values of cell Dmn;
      While seedcell value > 0
      Find Parentcell;
      End-of-While;
End;
```

For a given seed cell D_{ij} a parent cell is found as follows:

```
Begin (Find Parentcell)
      Start with seedcell Dij;
      Source and Target characters match in seedcell:
```

```
      If X_i = Y_j (character copy or match)
            Then set seedcell to D_i-1j-1;
      Source and Target characters do not match in seedcell:
      Else
            If D_ij = D_i-1j - DeleteCost (possible delete)
                  Then set seedcell to D_i-1j;
            If D_ij = D_ij-1 - InsertCost (possible insert)
                  Then set seedcell to D_ij-1;
            If D_ij = D_i-1j-1 – SubstituteCost (possible substitution)
                  Then set seedcell to D_i-1j-1;
End (Find Parentcell);
```

Thus each cell containing a non-zero value has at least one parent cell and may have up to three parent cells. For any cell $D_{ij} > 0$, a *trace back path* will start from this cell, stepping successively from it to a parent cell, and terminating as soon as the next step in the path reaches a cell with a zero value.

If there are multiple maxima in the metric array, the seed cell is set to the cell corresponding to the later character position in the strings. Thus a larger significant near-matching substring will subsume smaller locally aligned ones.

For the example strings in Table 1, the trace back path for the Smith-Waterman similarity metric calculation is shown in Table 4.

Table 4: Smith-Waterman Trace Back Path for strings "OSAMA" and "USSAMA"

	U	S	S	A	M	A
O	0	0	0	0	0	0
S	0		1			
A	0			2		
M	0				3	
A	0					**4**

In this example an optimal trace back path identifies the common substrings in the two strings being compared to be "SAMA".

3.2 Normalised Smith-Waterman Similarity Measure

Similar to the normalisation requirements described in Section 2.3, when pairs of strings are

compared using the *Smith-Waterman Similarity Measure*, the raw score can be an inaccurate indicator of the extent of match between the pairs of strings being compared. A larger score indicates a superior match, which may not always be the case. For example, consider the string pairs ("abcd", "abcd") and ("abcdef", "abcdgh"). The *Smith-Waterman Similarity Measure* has values 4 for both string pairs, with the common substring identified as "abcd". The shorter string pair in this example matches exactly while the longer string pair only matches partially, thus a similarity metric of 4 for both can be misleading.

One way of addressing this apparent inconsistency is to normalise the Smith-Waterman Similarity Metric by the maximum value obtained if the strings in the pair are compared individually to themselves. This will compensate for proportional differences in lengths of the significant matching substrings in the two strings. The value of the metric will fall in the range 0-1; and it will provide a more effective yardstick of the extent of the match between string pairs.

The Normalised Smith-Waterman Similarity Measure (S_{mn}) is computed as follows:

For two strings S_1 and S_2 respectively:

1. Compute the Smith-Waterman Similarity Metric (D_{mn}) as described in Section 3.1.
2. Compute the Smith-Waterman Similarity Metric (D'_{mn}) for the string pair S_1 and S_1;
3. Compute the Smith-Waterman Similarity Metric (D''_{mn}) for the string pair S_2 and S_2;
4. Compute the Normalised Smith-Waterman Similarity Measure (ND_{mn}) as follows:

$$ND_{mn} = \frac{D_{mn}}{Max\left(D'_{mn}, D''_{mn}\right)} \qquad (9)$$

The Normalised Smith-Waterman Similarity Metric can range from 0-1 with a value of 1 indicating a perfect match between the strings in the pair. In the examples above, the normalised Smith-Waterman similarity metric for the first string pair ("abcd", "abcd") is 1, whereas for the second string pair ("abcdef", "abcdgh") is 0.6667 respectively.

4. Comparing Strings with Spelling Anomalies

The *Jaro String Comparator* was developed to identify and link observations or records in two files that correspond to the same individual, in the absence of unique identifiers to allow straightforward matching of records [8],[9]. It is based on matching name strings. The string comparator accounts for length of strings and partially accounts for altered spelling errors and

is based on the number and order of common characters between two strings [5].

4.1 The Method

The Jaro method is based on the count of common characters between two strings. A character is counted as common if it occurs in the other string within a given distance that depends on the string length.

Let $A = (a_1\, a_2 \ldots a_m)$ and $B = (b_1\, b_2 \ldots b_n)$ be two strings. The search range distance d is defined as follows:

$$d = \frac{\max(|m|,|n|)}{2} - 1 \qquad (10)$$

where $|.|$ represents the string length.

So d is less than half the length of the longer string. The character a_i is counted as *in common* with character b_j iff $a_i = b_j$, and $i\text{-}d \le j \le i+d$. Furthermore characters are counted as *in common* only once, so that $A = (x\ a\ b\ c)$, and $B = (x\ x\ p\ q\ r)$ have only one common character, x.

The value of the string comparator metric is further determined by a count of *transpositions*. Transpositions are determined by pairs of *in common* characters that are out of order. Let $A' = (a'_1\, a'_2 \ldots a'_{m'})$ and $B' = (b'_1\, b'_2 \ldots b'_{n'})$ be the *in common* characters for strings A and B respectively. A half transposition occurs for A' and B' at position i if $a'_i \neq b'_i$. Thus the no of transpositions t are defined as:

$$t = \frac{1}{2} \times Number\ of\ half\ \ transpositions \qquad (11)$$

If c is the set of *in common* characters in strings A and B of lengths m and n respectively; and t the number of transpositions, then the *Jaro String Comparator* is defined as follows:

$$S_J = \frac{1}{3}\left(\frac{c}{m} + \frac{c}{n} + \frac{c-t}{c}\right) \qquad (12)$$

If strings A and B are identical $m = n = c$ and $t = 0$, so $S_J = 1$. If A and B differ slightly then $c < m$ and/or $c < n$, and $t > 0$, so $S_J < 1$. If A and B differ entirely with no *in common* characters, then $S_J = 0$. Thus, the *Jaro String Comparator* will range from 0 to 1, with a higher value signifying a greater similarity between the pair of strings being compared.

By using a search range to identify *in common* characters, the comparator allows for errors in positioning of characters to the extent of the search range. Examples of this are typographical errors.

Consider the two strings compared in Table 1:

A = (o **s a m a**) $\Rightarrow$ $m = 5$

B = (u **s** s **a m a**) $\Rightarrow$ $n = 6$

The strings have four *in common* characters:

$$\left.\begin{array}{l} A' = (\text{s a m a}) \\ B' = (\text{s a m a}) \end{array}\right\} \Rightarrow \quad c = 4 \qquad (13)$$

None of the *in common* characters in the two strings are out of order, so there are zero transpositions and $t = 0$. Accordingly, the *Jaro String Comparator* will be:

$$S_J = \frac{1}{3}\left(\frac{4}{5} + \frac{4}{6} + \frac{4-0}{4}\right) \Rightarrow \quad S_J = 0.8222 \qquad (14)$$

5. Winker Enhancement of the Jaro String Comparator for Common Prefixes

The *Jaro String Comparator* takes into account spelling variations between a pair of strings being compared. Winkler proposed an enhancement to the Jaro metric based on his observation that typographical errors occur more commonly toward the end of a string [13]. Winkler's enhancement assigns a higher comparator score to strings with common prefixes as they are more likely to be similar.

5.1 The Algorithm

The Winkler enhancement makes an adjustment to the *Jaro String Comparator* when up to the first four character prefixes of each string match in the string pair being compared. The *Jaro-Winkler String Comparator* is calculated as follows:

Let $A = (a_1\, a_2 \ldots a_m)$, $B = (b_1\, b_2 \ldots b_n)$ be the two strings being compared.

First calculate the *Jaro String Comparator* S_J as specified in section 4.1.

Then compare the first four prefix characters of the two strings as follows:

$$p = \min(4, p') \qquad (15)$$

Where p' is the length of the longest prefix in the string pair that agrees exactly where an exact agreement is defined by $a_i = b_j$.

The Jaro-Winkler String Comparator SW is:

$$S_W = S_J + \frac{p(1 - S_J)}{10} \qquad (16)$$

The *Jaro-Winkler String Comparator* is higher than the *Jaro String Comparator* for string pairs that share a common prefix. If strings A and B are identical then $S_J = 1$ and S_W will also be 1. At the other extreme if A and B have no in common characters then $S_J = 0$ and $S_W = 0$. If A and B differ but with some *in common* characters, then S_J will fall between 0 and 1 as will S_W. Like the *Jaro String Comparator*, the *Jaro-Winkler String Comparator* will also range from 0 to 1, with a higher value signifying a greater similarity between the pair of strings being compared.

For the two strings compared in Table 1, there is no common prefix. In this example p is zero,

and the Jaro-Winkler comparator score reduces to the Jaro comparator score, which is 0.8222. However, for the strings "abcdefgh" and "abcdefgh ijklmnopqrst", the *Jaro String Comparator* score is 0.7937 while the *Jaro-Winkler String Comparator* score is 0.8762, note that the adjustment to the Jaro score is only made for up to the first four common prefix characters of the strings.

6. Preliminary Investigation of String Metrics

A number of experiments were conducted to determine how effectively the different distance metrics and comparators discussed in sections 2-5 performed in detecting coreference mentions in a given text document. Lists of name mentions were extracted from news articles and websites. These methods were applied to name pairs from these lists. The comparison was based on name mentions falling under the following three cases:

1. Name mentions with common sub-strings
2. Name mentions with typographical errors
3. Name mentions with a similar sound

The results are discussed in the following sections 6.1-6.3, and shown in Table 5-Table 7. The columns in these tables are as follows:

Source String: The first string in the string pair being compared.

Target String: The second string in the string pair being compared.

LEV: The Levenshtein Minimum Edit Distance between the string pair.

LEV Norm: The Normalised Levenshtein Minimum Edit Distance between the string pair.

S-W: The Smith-Waterman Distance metric for the string pair. The insert, delete, substitute and copy costs were set to 1.

Common Substring: The common substring identified by the Smith-Waterman trace back process for each string in the pair.

S-W Norm: The Normalised Smith-Waterman distance metric for the string pair.

JARO: The Jaro string comparator metric for the string pair.

J-W: The Jaro-Winkler string comparator metric for the string pair.

6.1 Matching Name Mentions With Common Sub-strings

The list of name mentions selected to test this case included a subset of names that were likely candidates for coreference mentions, and comprised the following five name pairs:

Pair 1: (Abu Bakar Bashir, Abu Rusdan)

Pair 2: (Abu Bakar Bashir, Bashir)

Pair 3: (Abu Bakar Bashir, Bashir's)

Pair 4: (Megawati Sukarnoputri, Ms Megawati)

Pair 5: (Mohammad Saleh, Justice Saleh)

Table 5 shows the results obtained for this list of name pairs for each of the methods.

Table 5:Results from matching name mentions with common sub-strings

Source String	Target String	LEV	LEV Norm	S-W	Common Substring	S-W Norm	JARO	J-W
Abu Bakar Bashir	Abu Rusdan	10	0.625	4	abu |abu	0.25	0.6649	0.7989
Abu Bakar Bashir	Bashir	10	0.625	6	bashir|bashir	0.375	0.4514	0.4514
Abu Bakar Bashir	Bashir's	12	0.75	6	bashir|bashir	0.375	0.5	0.5
Megawati Sukarnoputri	Ms Megawati	16	0.7619	8	megawati|megawati	0.3810	0.6618	0.6956
Mohammad Saleh	Justice Saleh	8	0.5714	6	saleh|saleh	0.4286	0.5745	0.5745

Levenshtein's distance metric is not successful at discriminating between the extent of match in the name pairs ("Abu Bakar Bashir", "Abu Rusdan") and ("Abu Bakar Bashir", "Bashir"). The distance metric for the pair ("Megawati Sukarnoputri", "Ms Megawati") is higher indicating a worse match than for the former two name pairs. With the lowest distance metric, the name pair ("Mohammad Saleh", "Justice Saleh") is matched more closely than other name pairs.

The Smith-Waterman distance metric indicates a closer match between the second name pair compared to the first, and identifies a longer common sub-string; this is more clearly evident with the normalised Smith-Waterman metric values. The normalised distance metric is the same for the third name pair as for the second, but is higher for the fourth and higher still for the fifth name pair, indicating that the name pair ("Mohammad Saleh", "Justice Saleh") is the closest match.

The Jaro string comparator value also indicates a closer match between the second name pair. The Jaro-Winkler comparator value is misleading for the first name pair where the common prefix matches even though the name mentions differ. The Jaro comparator value is highest for the name pair ("Megawati Sukarnoputri", "Ms Megawati") indicating a closer match. The Jaro-Winkler comparator does not indicate the extent of the match as accurately for these example name pairs.

Of the different string metrics compared, the Smith-Waterman distance metric shows some promise in discovering similarity between name pairs with common substrings. The Jaro comparator is able to do this to a lesser extent.

6.2 Matching Name Mentions with Typographical Errors

A list of name mentions selected to test this case were taken from a sample of websites dealing

with spelling errors in names[2]. The list includes the following name pairs:

Pair 1: (Hawk, Hayk)

Pair 2: (Barnett Brouherd, Barnett Broughard)

Pair 3: (Wilson, Wilsen)

Pair 4: (bin Laden, bin Ladin)

Pair 5: (Franz Liszt, Franz Listz)

These examples typically require one or two string edits to revert from the source to the target name. Table 6 shows the results obtained for this list of name pairs for each of the methods applied.

Table 6: Results from matching name mentions with typographical errors

Source String	Target String	LEV	LEV NORM	S-W	Common Substring	S-W NORM	JARO	J-W
Hayk	Hawk	1	0.25	2	hayk|hawk	0.5	0.8333	0.8667
Barnett Broughard	Barnett Brouherd	2	0.1176	13	barnett broughard|barnett brouherd	0.7647	0.9400	0.9640
Condoleeza Rice	Condoleezza Rice	1	0.0625	14	condoleeza rice|condoleezza rice	0.875	0.9792	0.9875
bin Laden	bin Ladin	1	0.1111	7	bin laden|bin ladin	0.7778	0.9259	0.9556
Franz Listz	Franz Liszt	2	0.1818	9	franz listz|franz lisz	0.8182	0.9697	0.9818

The Levenshtein distance metric gives some indication of the extent of typographical errors. The normalised Levenshtein metric is more reliable as it compensates for the string length. The longer name strings score lower implying a closer match. The differences are still only marginal for the five name pairs compared.

The Smith-Waterman distance metric is not as reliable in determining similarity between the typographical variations in names. As almost all the name pairs require one or two string edits, we would expect the name pairs to consistently score high. This has not occurred. The normalised form of the Smith-Waterman is also not consistently giving high scores.

The Jaro string comparator appears to have more success in dealing with typographical errors. The Jaro comparator values are consistently high for all the names pairs. The Jaro-Winkler string comparator, by compensating for common prefixes in the name pairs has even higher values for the name pairs, again indicating a very high extent of similarity.

[2] Including, http://www.elisanet.fi/markus.lang/oops.html

For matching name mentions with typographical errors, the Jaro and Jaro-Winkler string comparators perform better than either the Levenshtein and the Smith-Waterman string distance metrics.

6.3 Matching Similar Sounding Name Mentions

A list of name mentions selected to test this case were taken from a sample of websites dealing with spelling errors in names. The list includes the following name pairs:

Pair 1: (Abu Bakar Bashir, Abu Bakar Baasyir)

Pair 2: (Osama bin Laden, Usama bin Ladin)

Pair 3: (Megawati Sukarnoputri, Megawati Soekarnoputri)

Pair 4: (Mohamed, Muhammed)

Pair 5: (Mohamad, Muhammed)

These examples typically require two or three string edits to revert from the source to the target name. Table 7 shows the results obtained for this list of name pairs for each of the methods applied.

Table 7: Results from matching similar sounding names

Source String	Target String	LEV	LEV NORM	S-W	Common Substring	S-W NORM	JARO	JARO-WINKLER
Abu Bakar Bashir	Abu Bakar Baasyir	2	0.1176	13	abu bakar bashir|abu bakar baasyir	0.7647	0.9400	0.9640
Osama bin Laden	Usama bin Ladin	2	0.1333	12	sama bin laden|sama bin ladin	0.8000	0.9111	0.9111
Megawati Sukarnoputri	Megawati Soekarnoputri	2	0.09091	18	megawati sukarnoputri|megawati soekarnoputri	0.8182	0.9372	0.9623
Mohamed	Muhammed	2	0.25	4	hamed|hammed	0.5	0.8690	0.8821
Mohamad	Muhammed	3	0.375	3	ham|ham	0.375	0.7798	0.8018

The Levenshtein distance metric is not consistently low indicating some variation in the extent of match between the name pairs. The normalised Levenshtein metric is similarly not giving a consistent view, particularly for shorter name pairs.

Similarly, the Smith-Waterman distance metric is not consistently high for the five name pairs tested. The same applies for the normalised Smith-Waterman distance metric.

The Jaro string comparator does provide a more consistent indication of similarity between

name pairs. The similarity is more salient when complete name pairs are considered. The Jaro-Winkler comparator improves on the Jaro comparator in all the example name pairs. This can be explained by the fact that similar sounding names are likely to share common prefixes.

When matching name mentions, the Jaro and Jaro-Winkler string comparator are more consistent in detecting matches in similar sounding name pairs than the Levenshtein and Smith-Waterman distance metrics. Notably, all these methods show some promise when matching name pairs that include first and last names. The confidence in metric values is thus likely to be higher when complete name mentions are matched.

7. Using String Metrics to Resolve Partial Name Mentions

Our early investigations of string metrics to discover matching name mentions indicated that the Normalised Smith-Waterman metrics was useful to detect name mentions with common substrings. It could be used to detect the presence of common tokens in multi-token name strings. The Jaro-Winkler was similarly effective when typographical errors were prevalent in text, and was more useful for matching single token name mentions. In the light of these findings we experimented with these two metrics to see if they could be combined in some fashion to detect and link all partial name mentions in a text document to their canonical form.

7.1 Problem Definition

In a text document where all the name mentions have been identified beforehand, the aim was to investigate how the Normalised Smith-Waterman and Jaro-Winkler string metrics could be combined to help resolve all *partial* name mentions to their *canonical* form.

For a given pair of name mentions, we define the *canonical* name mention to be the name mention with the greater number of tokens. Thus a *partial* name mention is the name mention with the smaller number of tokens. For example, in the name mentions pair: (***Richard A. Boucher***, ***Boucher***), ***Richard A. Boucher*** is the *canonical* name mention, and ***Boucher*** the *partial* name mention. When both name mentions in the pair have the same number of tokens, the *canonical* name is defined as the first in the pair, with the second being the *partial* name.

The resolution of partial names is a part of the broader coreference resolution problem. Note that coreference resolution of partial names has been addressed in the intra-document context.

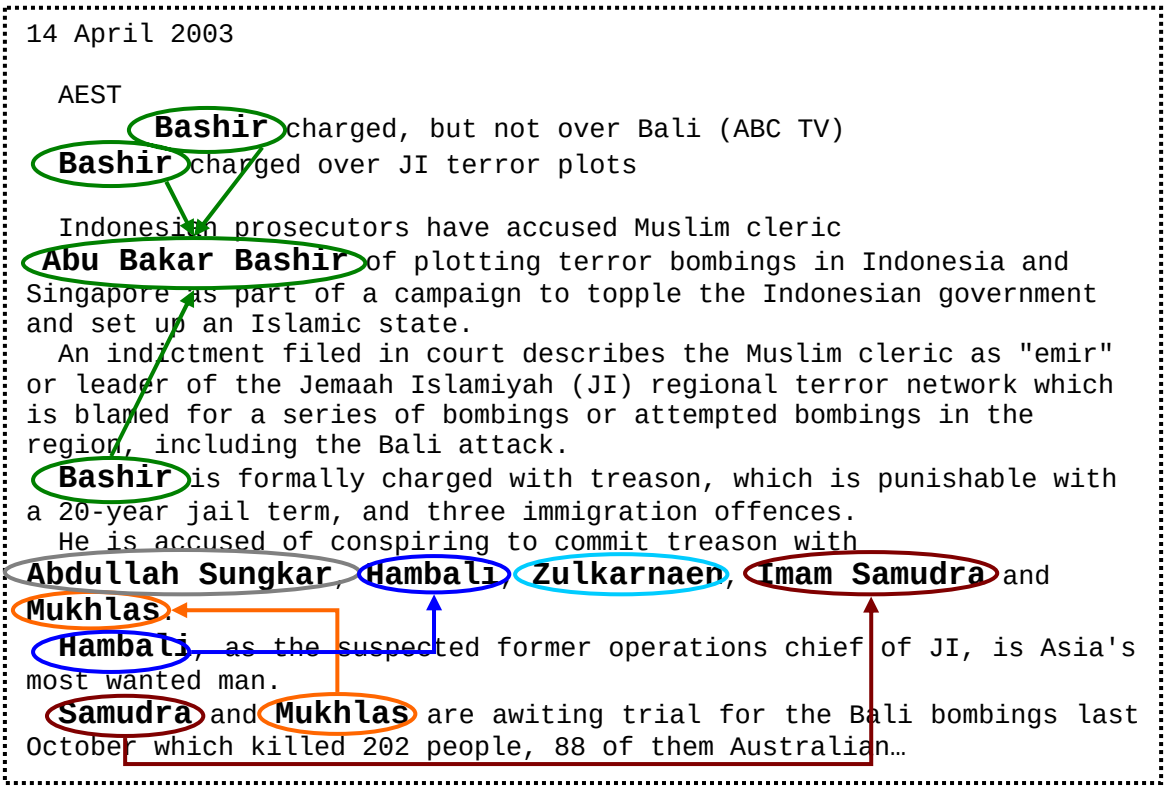


Figure 2: Resolving name mentions in a news except

To illustrate, consider the excerpt from a news article[3] in Figure 2. In this news excerpt, person name mentions are highlighted in bold and larger font size, and are listed in the *Person Name Mention* column of Table 8. The aim was to investigate the viability of selective application of the Normalised Smith-Waterman and Jaro-Winkler string metrics to identify all canonical and partial name mentions in this list, and associate *partial* name mentions to their *canonical* form. The outcome we wanted was to identify and link all *partial* name mentions to their *canonical* form. This is formalised by setting the **AssignedID** of each *partial* name to the **ID** of its *canonical* equivalent, shown by the coloured arrows in Table 8.

Every person name mention found in the news excerpt has an **ID** that is pre-assigned by the Information Extraction system that discovers it. In this example the **ID** for each name mention reflects its relative position in the text. The **AssignedID** for each name mention provides the link to its *canonical* form. It is a placeholder initially, defaulting to the **ID** for each name. When a *partial* name is found, the **AssignedID** for that name is set to the **ID** of its *canonical* name.

In the example in Table 8 the first occurrence of "Bashir" has an **ID** of 1. Its **AssignedID** will be set to 3 to indicate that "Bashir", a *partial* name mention with the *canonical* name "Abu Bakar Bashir" that has an **ID** of 3. Similarly, we would expect to find that 6 out of the 12 person name mentions identified are canonical with the remaining 6 being *partial* name mentions. The arrows link all *partial* name mentions to their *canonical* name mention.

[3] News excerpt sourced from the Australian Broadcasting Corporation, 14/04/2003.

Table 8: Linking AssignedIDs to their canonical IDs from matching person name mentions extracted from the news excerpt in Figure 2

Person Name Mention	ID	AssignedID
Abu Bakar Bashir	**3**	3
Bashir	1	**3**
Bashir	2	**3**
Bashir	4	**3**
Abdullah Sungkar	5	5
Hambali	**6**	6
Hambali	10	**6**
Zulkarnaen	7	7
Imam Samudra	**8**	8
Samudra	11	**8**
Mukhlas	**9**	9
Mukhlas	12	**9**

The method we propose should identify *partial* name mentions in a text document, and normalise them to their *canonical* name. It is based on the Normalised Smith-Waterman and Jaro-Winkler metrics.

7.2 Characteristics of Name Strings

Name mentions in text typically comprise one or more tokens. When the number of tokens in the name pair being matched differs, string distance between the name strings cannot be relied on to accurately identify a match. For example, consider the strings "John Smith" and "Smith". As shown in Table 9 the Normalised Smith-Waterman string edit distance between these name strings is 0.5. The Jaro-Winkler score is 0.4333 for the same comparison. Neither score supports a strong match, although it is not unreasonable to expect that if these two name mentions appear in the same text fragment they are likely to refer to the same named entity.

Table 9: Results from Token-wise comparison of the name mentions "John Smith" and "Smith"

Name1/Token	Name2	Normalised Smith-Waterman Score	Jaro-Winkler Score
John Smith	Smith	0.5	0.4333
John	Smith	0.2	0.0
Smith	Smith	1.0	1.0

The scores for splitting the first name into its token and comparing each token with the second

name are also shown. The first token "John" results in a low Normalised Smith-Waterman score of 0.2. The Jaro-Winkler score of 0 for the same two tokens indicates no match. However, a token-wise comparison with the second token, "Smith" produces a perfect match for both string distance metrics.

Clearly both scores can contribute to the decision that the two strings are close in terms of their string edit distance, and that "Smith" is a *partial* name mention of "John Smith". The method proposed to resolve *partial* name mentions achieves this by doing a specialised comparison to determine string similarity between name mentions. The string metric used for the comparison is selected according to the token size of the names (or token combinations) being compared. Based on these comparisons, a decision is made on whether name pairs are considered close enough to be linked as a (*canonical* name, *partial* name) pair.

7.3 String Similarity Thresholds

In order to use the Jaro-Winkler or Normalised Smith-Waterman metrics to accurately predict string similarity between a pair of names it was necessary to determine threshold scores. These were based on comparisons of token combinations taken from name mentions sourced from news articles[4] and emails[5].

Name pairs were first identified from the sample documents. For each name pair, the name with the greater number of tokens was fragmented into token combinations containing the same number of tokens as the other name. This resulted in a comparison of 1848 (Token-combination, Name) pairs, 1295 of these were single-token and 553 were multi-token.

The Jaro-Winkler metric was used to compare single-token pairs. An inspection of the Jaro-Winkler scores for these indicated that a threshold value of 0.9 could correctly identify matches in the 1295 single-token string pairs that were compared.

Figure 3 shows the Jaro-Winkler scores from comparing the 1295 string pairs using this threshold value. The Jaro-Winkler scores for the string pair comparison fall into three succinct groups. Scores along the top section of the graph, plotted in black, are greater than 0.9 and correspond to matching string pairs; scores in the middle section, plotted in red, correspond to non-matching string pairs that have one or more common characters; finally string pairs with no common characters gave a result of 0, and are plotted in red along the x-axis.

The Normalised Smith-Waterman metric was used to compare multi-token pairs. The Normalised Smith-Waterman scores for the 553 string pairs that were compared indicated that a threshold value of 0.8 could correctly identify matching multi-token string pairs.

[4]The news articles were sourced from The Australian Broadcasting Corporation (www.abc.net.au) .; The Australian newspapers (www.news.com.au); CNN (www.cnn.com); and The New York Times (www.nytimes.com).

[5] The emails were sourced from the Enron Email Dataset (www.cs.cmu.edu/~enron/)

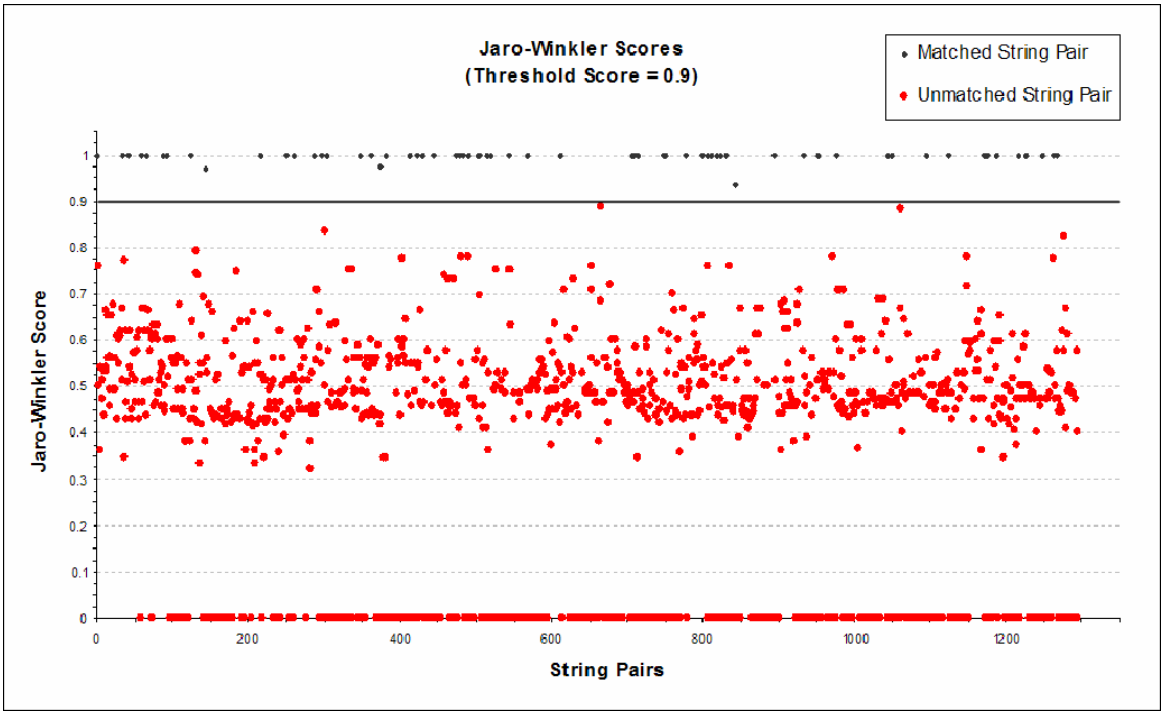


Figure 3: Jaro-Winkler scores from comparing 1295 single-token string pairs

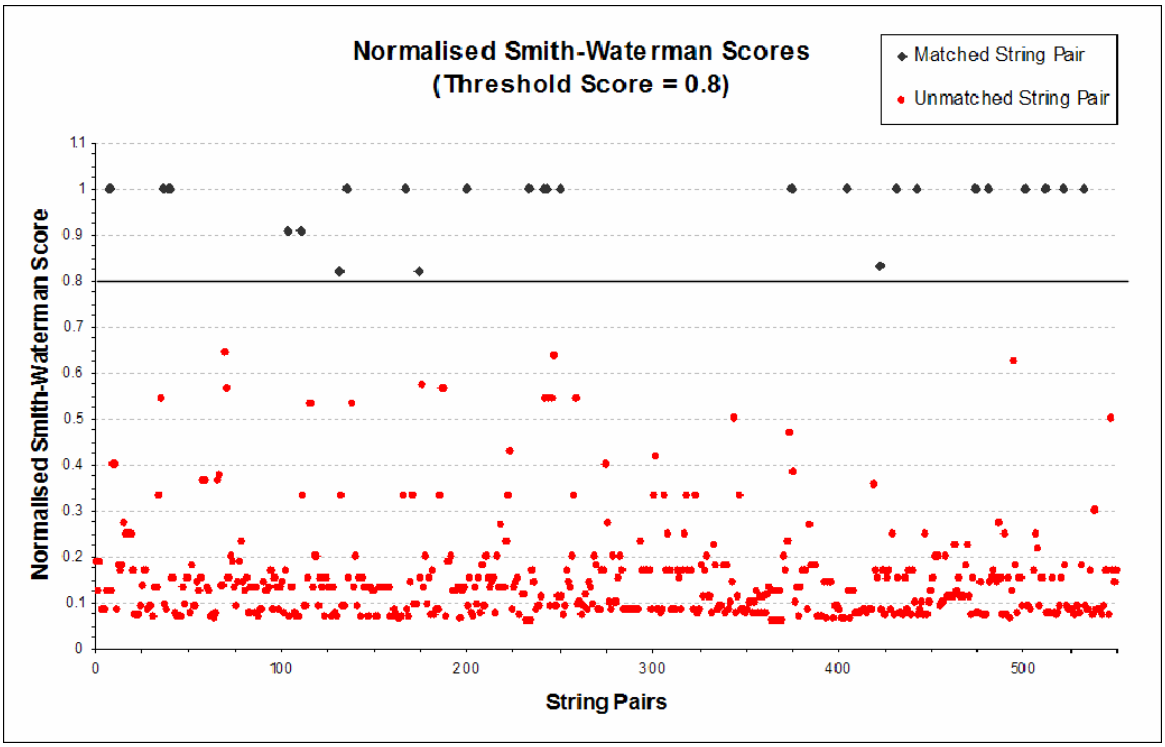


Figure 4: Normalised Smith-Waterman scores from comparing 553 multi-token string pairs

Figure 4 shows the Normalised Smith-Waterman scores from comparing 533 multi-token string pairs using a threshold value of 0.8. The scores fall into two groups. Scores along the

top section of the graph, plotted in black, are greater than 0.8 and correspond to matching string pairs; scores in the lower section of the graph, plotted in red, correspond to non-matching string pairs.

The precision was calculated for both sets of string pairs. The results for the 1295 single-token string pairs compared using the Jaro-Winkler metric with a threshold score of 0.9 are shown in the second column of Table 10.

Table 10: Results from comparing single- and multi-token string pairs

	Single-Token string pairs compared using Jaro-Winkler metric	**Multi-Token string pairs compared using Smith-Waterman metric**
# String Pairs Compared	1295	553
# Correct Results	1295	553
Precision (True Positives)	100%	100%

The results for the 553 multi-token string pairs compared using the Normalised Smith-Waterman metric with a threshold score of 0.8 are shown in the third column of Table 10.

The results show high precision and lend support to our use of the Jaro-Winkler metric to compare single-token strings, and the Normalised Smith-Waterman metric for multi-token strings. This is not surprising in the light of earlier experiments which showed the strengths of the Jaro-Winkler metric in handling typographical errors, and the Normalised Smith-Waterman in detecting common substrings between name mentions.

The results also corroborate setting threshold scores at 0.9 for Jaro-Winkler, and 0.8 for the Normalised Smith-Waterman metrics respectively. In the sample we used, these values were reliable discriminators for determining when a pair of strings could be considered to match.

7.4 Resolving Partial Name Mentions in a Text Document

The application of the Normalised Smith-Waterman and Jaro-Winkler metrics to compare name mentions sourced from sample news media and email corpora led to another important observation. The two metrics worked effectively when the number of tokens in the pair of names being compared was the same. However, when comparing a pair of names with different token sizes, the two string similarity measures could not reliably identify and resolve partial names.

Additional processing was required to account for comparisons of name pairs with different token sizes. This was based on decomposing the *canonical* name into combinations of the same token size as the *other* name, and then using string similarity measures on (*token combination*, *other* name) pairs to determine if the *other* name represented a *partial* name reference to the *canonical* name.

The processing sequence for handling partial name resolution of name mentions sourced from a text document is illustrated in Figure 5. The processing sequence shown is applied to each Name Pair identified in a given text document in order. Sections 7.4.1-7.4.4 elaborate on the individual processes identified in Figure 5.

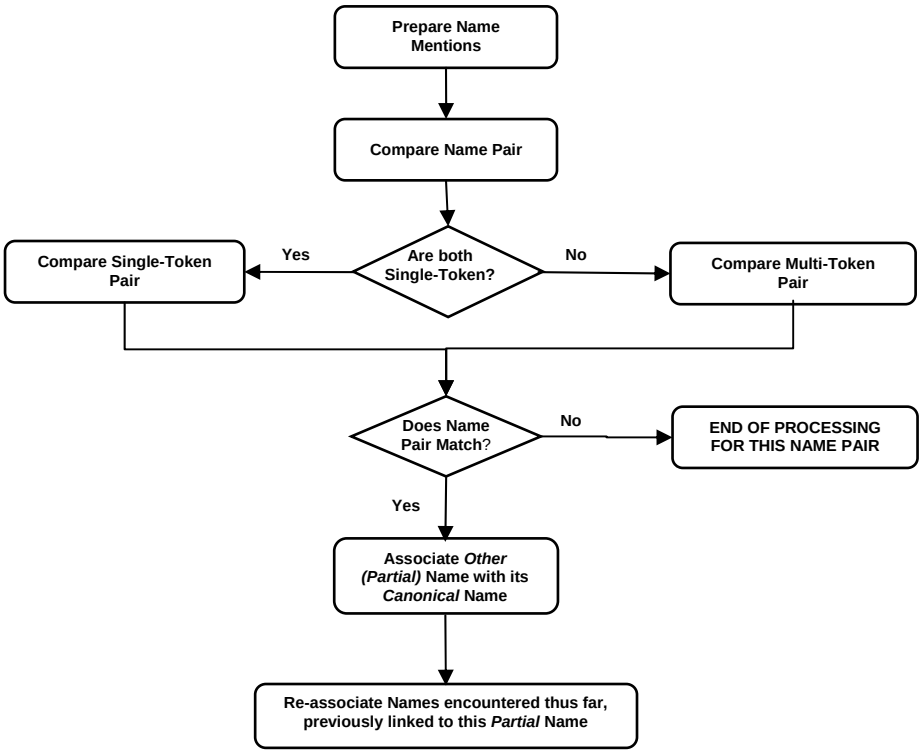


Figure 5: The processing sequence for resolving partial name mentions in a text document

7.4.1 Prepare Name Mentions

The method we developed required all name mentions to be detected and tagged with a unique identifier (**ID**) for each given document. The names were captured in the order in which they appeared in the document. Each name was normalised by removing any extra whitespaces and any embedded punctuation. For each document a list of names was prepared with the following attributes:

ID: A unique identifier for every name mention, indicating its ordinal position in text.

String: The name string.

AssignedID: A placeholder for linking each partial name mention to its canonical form, initially set to the ID value for each name mention. This value is reset to the ID of the canonical form for each partial name coreference mention discovered.

1. Each name was paired with every other name in the order that it occurred in the text.

2. For each name pair the *canonical* and *other* name was identified based on token size (i.e. the number of tokens comprising the name). The name with the bigger token size was assigned as the *canonical* name and the remaining name was assigned as the *other* name. If both names had the same number of tokens, the first name was assigned as the *canonical* name and the second name is assigned as the *other* name.

   To illustrate consider the name pair (***Abu Bakar Bashir***, ***Bashir***). The first name, ***Abu Bakar Bashir***, has a token size of 3 while the second name, ***Bashir***, has a token size of 1. So ***Abu Bakar Bashir*** will be identified as *canonical*, and ***Bashir*** as the *other* name in the pair.

7.4.2 Compare Name Pair

Comparison of a name pair encompasses a number of sub-processes to handle differences in token sizes between the names. These are described in sections 7.4.2.1-7.4.2.3.

7.4.2.1 Create Token Combinations From Canonical Name

After the *canonical* and *other* name had been identified, we used the token size of the *other* name to generate *token combinations* from the *canonical* name. If both names had the same token size, a single *token combination* was generated and was identical to the *canonical* name.

In the example discussed in section 7.4.1, the *other* name, ***Bashir***, has a single token. Therefore, single-token combinations will be generated from the *canonical* name, ***Abu Bakar Bashir*** as follows:

Abu
Bakar
Bashir

7.4.2.2 Process Combination-Name Pair

Once *token combinations* had been generated from the *canonical* name, we then performed pair wise comparisons of each *combination-name* pair based on their token size. The procedure for comparing a single-token *combination-name* pair is described in section 7.4.2.2.1, and a multi-token *combination-name* pair is described in section 7.4.2.2.2.

7.4.2.2.1 Compare Single-Token Combination-Name Pair

If the *combination-name* pair comprised single-token strings, we used the Jaro-Winkler metric to determine if they matched. We computed the Jaro-Winkler score (S_W) for the single-token *combination-name* pair. The Jaro-Winkler threshold score ($S_{Threshold}$) was set to 0.9. A match was defined as follows:

$$\left.\begin{array}{rl} S_W \geq S_{Threshold} \Rightarrow & Match \\ S_W < S_{Threshold} \Rightarrow & Non-match \end{array}\right\} \qquad (17)$$

Thus the single-token *combination-name* pair was considered to match if the Jaro-Winkler score from comparing them is greater than or equal to the Jaro-Winkler threshold value of 0.9.

7.4.2.2.2 Compare Multi-Token Combination-Name Pair

If the *combination-name* pair comprised multi-token strings, we used the Normalised Smith-Waterman metric to determine if they matched. We computed the Normalised Smith-Waterman score (N_S) for the multi-token *combination-name* pair. The Normalised Smith-Waterman threshold score ($N_{Threshold}$) was set to 0.8. A match was defined as follows:

$$\left.\begin{array}{rl} N_S \geq N_{Threshold} & \Rightarrow \quad Match \\ N_S < N_{Threshold} & \Rightarrow \quad Non-match \end{array}\right\} \quad (18)$$

Thus the multi-token *combination-name* pair was considered to match if the Normalised Smith-Waterman score from comparing the names was greater than or equal to the Normalised Smith-Waterman threshold value of 0.8.

7.4.2.3 Determine Whether Name Pair Matches

When a *combination-name* pair was found to match, this signified that the *other* name was a *partial* name reference of the *canonical* name. No further processing was required for other *token combinations* derived from the same *canonical* name. However, processing for the matched name pair continued as described in sections 7.4.3-7.4.4.

When a *combination-name* pair was found not to match, other *token combinations* derived from the same *canonical* name needed to be compared to check for a match until they were exhausted. After all *combination-name* pairs had been compared if no match was found, this signified that the *canonical* and *other* name were not linked. No further processing was required for this name pair.

7.4.3 Associate *Partial* Name to its *Canonical* Name

When a name pair was found to match, this implied that the *other* name was a *partial* name reference of the *canonical* name. To formalise this link we set the **AssignedID** of the *partial* name in the name pair to the **ID** of the *canonical* name.

To illustrate with the example discussed in sections 7.4.1 and 7.4.2.1, prior to applying our method, the **ID** and **AssignedID** for the two names will be as shown in Table 11.

Table 11: Example of Prepared Name Mentions

Name Mention	ID	AssignedID
Abu Bakar Bashir	1	1
Bashir	2	2

After comparing the name mentions, as the name pair matches we would reset the **AssignedID** for the *other* name, ***Bashir***, to 1 as shown in Table 12.

Table 12: Example of Associating a Partial Name to its Canonical Name

Name Mention	ID	AssignedID
Abu Bakar Bashir	1	1
Bashir	2	1

7.4.4 Re-associate Names Previously Linked to Current *Partial* Name

When the **AssignedID** of a name was modified, it was identified as a *partial* name that was linked to a *canonical* name. All other names in the text document that were previously linked to this *partial* name now need to be re-linked to its *canonical* name. We used the following process to perform the required re-linking:

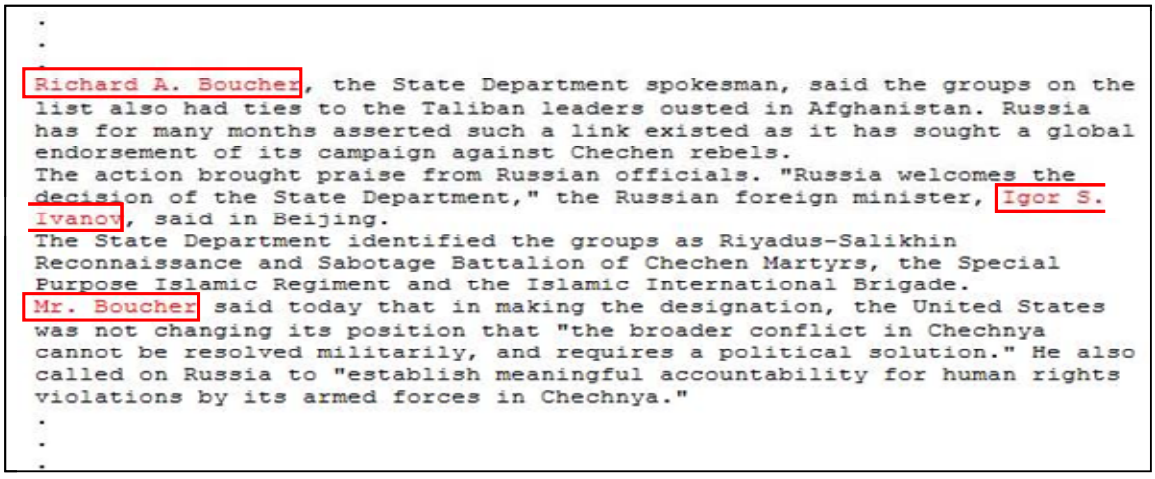
1. Traverse through all the names encountered thus far in chronological order, and identify those names whose **AssignedIDs** were set to the **ID** of the current *partial* name. These were previously identified as *partial* name references of the current *partial* name.

2. Reset **AssignedIDs** of these names to the **ID** of the current *canonical* name, so they are correctly linked to their newly discovered *canonical* name

This is the final step in the processing sequence for a given name pair found to match.

7.5 An Example Application

To illustrate how the method works, consider the example excerpt taken from a New York Times news article in Figure 6. All name mentions in the excerpt have been marked in red. Before applying the method, all name mentions in the text need to be extracted together with their **ID** which is unique and indicates their chronological position.



. . .

Richard A. Boucher, the State Department spokesman, said the groups on the list also had ties to the Taliban leaders ousted in Afghanistan. Russia has for many months asserted such a link existed as it has sought a global endorsement of its campaign against Chechen rebels.

The action brought praise from Russian officials. "Russia welcomes the decision of the State Department," the Russian foreign minister, Igor S. Ivanov, said in Beijing.

The State Department identified the groups as Riyadus-Salikhin Reconnaissance and Sabotage Battalion of Chechen Martyrs, the Special Purpose Islamic Regiment and the Islamic International Brigade.

Mr. Boucher said today that in making the designation, the United States was not changing its position that "the broader conflict in Chechnya cannot be resolved militarily, and requires a political solution." He also called on Russia to "establish meaningful accountability for human rights violations by its armed forces in Chechnya."

. . .

Figure 6: A sample document[6] with name mentions identified

[6] Source: New York Times - www.nytimes.com; dated 1 March 2003.

The method described in sections 7.4.1-7.4.4 will be applied to process the names identified in the news passage in Figure 6 as follows:

7.5.1 Prepare Name Mentions

The names are normalised and allocated **ID**s and **AssignedIDs** as shown in Table 13. The normalisation involves removing extra spaces and punctuation between tokens comprising a name mention.

Table 13: Name mentions extracted from the sample news article in Figure 6

Name Mention	ID	AssignedID
Richard A Boucher	1	1
Igor S Ivanov	2	2
Boucher[7]	3	3

Each name is compared with every other name in the order that it occurs in the text. As three names have been identified in the news excerpt, the method will require 3 pair wise comparisons or iterations through the processing sequence described in sections 7.5.2-7.5.4. The 3 name pairs are listed in Table 14.

7.5.2 Iteration 1 of Compare Name Pair

Consider the first name pair, (***Richard A Boucher, Igor S Ivanov***). Both names are multi-token with the same token size. The first name will be identified as *canonical*.

7.5.2.1 Create Token Combinations From Canonical Name

Both names have equal token size of 3, resulting in a single *token combination*.

7.5.2.2 Process Combination-Name Pair

The *combination-name* pair are multi-token, so they are compared using the Normalised Smith -Waterman metric. The Normalised Smith-Waterman scores for this comparison are shown in Table 14. The score, 0.0588 is less than the threshold value, so the name pair does not satisfy the criterion for a match.

Table 14: Normalised Smith-Waterman scores for comparing name pairs with both multi-token names of equal token size

Name Pair	Normalised Smith-Waterman Score	Threshold Value
Richard A Boucher, Igor S Ivanov	0.0588	0.80
Richard A Boucher, Boucher		
Igor S Ivanov, Boucher		

[7] Note that the honorific has been removed prior to applying the method.

7.5.2.3 Determine Whether Name Pair Matches

As none of *combination-name* pairs match, this implies that the name pair does not match. No further processing is required for this name pair.

7.5.3 Iteration 2 of Compare Name Pair

Consider the second name pair, (***Richard A Boucher***, ***Boucher***). The first name has a bigger token size and will be identified as *canonical*, the second name will be identified as *other*.

7.5.3.1 Create Token Combinations From Canonical Name

The *other* name has a token size of 1. We use this token size to generate token combinations of the *canonical* name. This will result in the following 3 token combinations for the *canonical* name ***Richard A Boucher***:

Richard
A
Boucher

7.5.3.2 Process Combination-Name Pair

The *combination-name* pair are single-token, so they are compared using the Jaro -Winkler metric. The Jaro-Winkler scores for this comparison are shown in Table 15. The scores indicate a perfect match between the last *combination-name* pair.

Table 15: Jaro-Winkler scores from comparing combination-name pairs.

Combination-Name Pair	Jaro-Winkler Score	Threshold Value
Richard, Boucher	0.6191	0.90
A, Boucher	0.0000	0.90
Boucher, Boucher	1.0000	0.90

7.5.3.3 Determine Whether Name Pair Matches

A perfect match between the combination-name pair in the previous step signifies that the *other* name, ***Boucher***, is a *partial* name reference of the *canonical* name, ***Richard A Boucher***.

7.5.3.4 Associate Partial Name to its Canonical Name

The **AssignedID** for the *partial* name ***Boucher*** is reset to the **ID** of its *canonical* name, ***Richard A Boucher*** as shown in Table 16.

Table 16: Association of Partial Name to its Canonical Name

Name Mention	ID	AssignedID
Richard A Boucher	1	1
Igor S Ivanov	2	2
Boucher	3	**1**

7.5.3.5 Re-associate Names Previously Linked to Current Partial Name

We traverse through the list of names occurring prior to ***Boucher*** in chronological order to verify if any have been identified as a *partial* name reference of ***Boucher***. As there are none, no further action is required.

7.5.4 Iteration 3 of Compare Name Pair

Consider the third name pair, (***Igor S Ivanov***, ***Boucher***). The first name has a bigger token size and will be identified as *canonical*, the second name will be identified as *other*.

7.5.4.1 Create Token Combinations From Canonical Name

The *other* name has a token size of 1. We use this token size to generate token combinations of the *canonical* name. This will result in the following 3 token combinations from the *canonical* name ***Igor S Ivanov***:

Igor
S
Ivanov

7.5.4.2 Process Combination-Name Pair

The *combination-name* pair are single-token, so they are compared using the Jaro -Winkler metric. The Jaro-Winkler score for this comparison is shown in Table 17.

Table 17: Jaro-Winkler scores from comparing token combinations

Combination-Name Pair	Jaro-Winkler Score	Threshold Value
Igor, Boucher	0.4643	0.90
S, Boucher	0.0000	0.90
Ivanov, Boucher	0.0000	0.90

The Jaro-Winkler scores for all the *combination-name* pairs are lower than the threshold value indicating that none of the *combination-name* pairs match.

7.5.4.3 Determine Whether Name Pair Matches

As none of *combination-name* pairs match, the name pair does not match. No further processing is required for this name pair.

Furthermore, as we have exhausted all name pairs, no further processing is required.

The results from applying our method to resolve *partial* names identified in the sample news excerpt in Figure 6 correctly detected one *partial* name, ***Boucher***, and linked it to its *canonical* name, ***Richard A Boucher***, as shown in Table 16.

7.6 Application of the Method to Resolve Partial Name Mentions in Emails and News Articles

The method described in sections 7.4.1-7.4.4 was applied to resolve *partial* name mentions identified in a sample of 15 emails[8] and 8 news articles[9]. A set of 111 names were identified in the 8 news articles; and 253 names in the 15 emails.

Due to characteristics unique to each of the two corpora, the precision was calculated separately for name mentions occurring in emails and news articles. The language used in emails tended to have an informal style, with names mentions often abbreviated, for example, ***Jason W*** (instead of ***Jason Wolfe***) or ***Josh*** (instead of ***Joshua***). The method accommodated some of these characteristics.

The results for 111 name mentions sourced from news articles, and 253 name mentions sourced from sample emails are shown in Table 18.

Table 18: Results from Partial Name Resolution

Text Source	# Name Mentions	# True Positives	Precision
News Articles	111	111	100%
Emails	253	245	97%

The results show that all *partial* name mentions taken from the sample news articles have been correctly linked to their *canonical* name mentions. The method is able to handle typographical errors, for example, ***Khalid Shaikh Mohammed*** and ***Khalid Sheik Mohammed*** correctly resolved to ***Khalid Sheikh Mohammed***, which is identified as their *canonical* form.

All *partial* name mentions were also correctly linked to their *canonical* form irrespective of the order in which they occurred in the text. For example, in a news article, although the *partial* name ***Karzai*** occurred before its *canonical* form, ***Hamid Karzai***, it was correctly linked.

Results for name mentions occurring in the sample emails are also sound. The method could correctly resolve majority of the *partial* name mentions. The errors are discussed in section 7.6.1.

The results support the validity of the method we developed to resolve *partial* name mentions. This method relies on string comparisons based on the Jaro-Winkler string comparator and the Normalised Smith-Waterman string similarity metric.

[8] The emails were sourced from the Enron Email Dataset (www.cs.cmu.edu/~enron/).
[9] The news articles were sourced from The Australian Broadcasting Corporation (www.abc.net.au) .; The Australian newspapers (www.news.com.au); CNN (www.cnn.com); and The New York Times (www.nytimes.com).

7.6.1 Errors in Resolving Partial Name Mentions

In the previous section we saw that the method we developed to resolve *partial* name mentions in text resulted in 8 erroneous resolutions. These fell into the following two distinct categories:

Errors requiring exception handling:

1. The *partial* name mention ***Jason W*** did not resolve to its *canonical* name ***Jason Wolfe***.

2. As a consequence of this, a following *partial* name mention, ***Jason***, resolved incorrectly to ***Jason W*** instead of ***Jason Wolfe***.

Errors that require context sensitive handling:

1. The *partial* name mention ***Ken*** resolved to ***Ken Harrison*** instead of its correct *canonical* name ***Ken Rice***.

2. The previous error was repeated in the same email.

3. The *partial* name ***Mamamia*** was erroneously resolved to itself when it should have been resolved to ***Momie*** which was expressly stated to be an alias for ***Mamamia***.

4. The *partial* name ***Mamia*** was erroneously resolved to itself when it should have been resolved to ***Momie***, also expressly stated to be an alias.

5. The *partial* name ***Mia*** was erroneously resolved to itself when it should have been resolved to ***Momie***, again expressly stated to be an alias.

6. The *partial* name ***David*** was incorrectly resolved to ***Pate David Herrold*** when based on the context, it did not have a *canonical* name.

Errors in the first category require special handling of name mentions that include initials, and can be addressed by a method based on string comparisons.

The second category of errors relies on context to be resolved, and therefore is outside the scope of our method which is based on string comparisons. These errors were identified by reading the emails containing these names, and understanding the context in which they occur. The method we propose relies primarily on string matching. It is not designed to incorporate context surrounding name mentions, and is therefore not suitable for addressing context-based errors. A different, context based approach is required to address these errors.

8. Conclusion

This report focuses on the application of character-based string metrics to address the problem of coreference resolution of partial names that occur within a document. This problem is closely related to that of name matching, record linkage and duplicate detection. Character-

based string metrics have been applied to these problems, and therefore lend themselves to partial name resolution. A number of character-based string metrics are discussed. These include the Levenshtein minimum edit distance; the Smith-Waterman similarity metric; the Jaro string comparator; and the Winkler enhancement of the Jaro string comparator.

We have used a modified version of the Smith-Waterman string comparator normalised on the length of the significant matching substrings in the two strings being compared. By compensating for proportional differences in lengths of the significant matching substring, the value of the metric provides an effective measure of the extent of the match between their parent strings.

Our investigations suggested that the Normalised Smith-Waterman and Jaro-Winkler metrics could be successfully combined to address the problem of resolving partial name mentions. We developed a method based on these two string metrics and tested it on names sourced from a sample set of news articles and emails. The results support the validity of the method to resolve partial name mentions occurring in a document.

The method did produce a small number of erroneous resolutions in the email corpus. The errors fell in two distinct categories. The first type of error was from a failure to match name mentions that included initials. The second type of error resulted from ignoring the context surrounding the name mentions. The former can be tackled by incoporating special handling for name mentions with embedded initials, and will be addressed in future work. The latter requires an approach that takes cognisance of context. The method we have proposed is based strictly on string matching and is therefore not suited to address this type of error.

There was a preponderance of partial name mentions in news articles compared to emails. Emails typically had a far greater incidence of first and second order pronouns. Future effort on entity normalisation will also be directed to addressing pronominal resolution.

Other future work will investigate adapting the proposed method to resolve partial organisation and place names.

9. References

[1] Linguistic Data Consortium, *ACE (Automatic Content Extraction) English Annotation Guidelines for Entities*, Version 5.6.1, 2005, http://projects.ldc.upenn.edu/ace/docs/English-Entities-Guidelines_v5.6.1.pdf.

[2] Pirkola, A., Keskustalo, H., Leppänen, E., Känsälä, A., Järvelin, K., Targeted s-gram matching: a novel n-gram matching technique for cross- and mono-lingual word form variants, Information Research, **7**(2), January 2002, also available at (http://informationr.net/ir/7-2/paper126.html).

[3] Jurafsky, D., Martin, J.H., *Speech and Language Processing*, Prentice Hall, New Jersey, 2000.

[4] McCallum, A., *Information Extraction: Coreference and Relationship Extraction*, (http://www.cs.umass.edu/~mccallum/courses/inlp2004/lect22-ie2.pdf).

[5] Bilenko, M., Mooney, R., Cohen, W., Ravikumar, P., Fienberg, S., *Adaptive Name Matching in Information Integration*, IEEE Intelligent Systems, **18**(5), 16-23, September/October 2003.

[6] Smith, T.F. and Waterman, M.S., *Identification of Common Molecular Subsequences*, Journal of Molecular Biology, **147**, 195-197, 1981.

[7] Irving, R.W., *Plagiarism and Collusion Detection using the Smith-Waterman Algorithm*, TR-2004-164, DCS Technical Report, 1-24, Dept of Computing Science, University of Glasgow, 2004.

[8] Jaro, M. A., Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida, Journal of the American Statistical Association, **84**(406), 414-420, June 1989.

[9] Jaro, M.A., *Probabilistic Linkage of Large Public Health Data Files*, Statistics in Medicine, **14**(5-7), 491-498, March-April 1995.

[10] Winkler, W. E., *The State of Record Linkage and Current Research Problems*, Statistical Society of Canada, Proceedings of the Survey Methods Section, 73-80, 1999; (longer version RR99/03, US Bureau of Census, 1999, also available at: http://www.census.gov/srd/www/byname.html

[11] Winkler, W. E., *String Comparator Metrics and Enhanced Decision Rules in the Felligi-Sunter Model of Record Linkage*, Proceedings of the Section on Survey Research Methods, American Statistical Assn., 354-359, 1990.

[12] Monge, A.E., *An Adaptive and Efficient Algorithm for Detecting Approximately Duplicate Database Records*, California State University, Long Beach, 2000, available at: http://www.cecs.csulb.edu/~monge/Papers/is-special-issues.ps

[13] Yancey, W. E., *An Adaptive String Comparator for Record Linkage*, Research Report Series (Statistics #2004-02), US Bureau of Census, 2004.

[14] Elmagarmid, A.K., Iperirotis, P.G. and Verykios, V.S., "Duplicate Record Detection: A Survey", IEEE Transactions on Knowledge and Data Engineering, **19**(1), January 2007.

[15] Cohen, W. W., Ravikumar, P., Fienberg, S. E., *A Comparison of String Metrics for Matching Names and Records*, Proceedings of the IJCAI-2003 Workshop on Information, 2003, also available at: http://www.isi.edu/info-agents/workshops/ijcai03/papers/Cohen-p.pdf

Page classification: UNCLASSIFIED

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION
DOCUMENT CONTROL DATA

1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)

2. TITLE

Resolving Partial Name Mentions Using String Metrics

3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION)

Document	(U)
Title	(U)
Abstract	(U)

4. AUTHOR(S)

Jyotsna Das and Poh Lian Choong

5. CORPORATE AUTHOR

Defence Science and Technology Organisation
PO Box 1500
Edinburgh South Australia 5111 Australia

6a. DSTO NUMBER	6b. AR NUMBER	6c. TYPE OF REPORT	7. DOCUMENT DATE
DSTO-RR-0318	AR- 014-065	Research Report	December 2007

8. FILE NUMBER	9. TASK NUMBER	10. TASK SPONSOR	11. NO. OF PAGES	12. NO. OF REFERENCES
2007/1079052	INT 07/020 and NS 07/201	ASCP and EXEC DIR CTSTC	34	15

13. URL on the World Wide Web

http://www.dsto.defence.gov.au/corporate/reports/DSTO-RR-0318.pdf

14. RELEASE AUTHORITY

Chief, Command, Control, Communications and Intelligence Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, EDINBURGH, SA 5111

16. DELIBERATE ANNOUNCEMENT

No Limitations

17. CITATION IN OTHER DOCUMENTS No

18. DSTO RESEARCH LIBRARY THESAURUS http://web-vic.dsto.defence.gov.au/workareas/library/resources/dsto_thesaurus.htm

Information Extraction, Coreference Resolution, String Edit Distance, String Similarity Measure, String Comparator.

19. ABSTRACT

Information Extraction is concerned with discovering entities, relationships and events from text. Before relationships and events can be discovered accurately, it is critical to resolve all mentions of the same entity. This process is known as coreference resolution. Coreferenced mentions of entities can occur in a number of forms including pronominal mentions; partial name mentions; and through the use of honorifics. This report focuses on addressing the problem of resolving partial name mentions to their canonical form within a text document using character-based string metrics. Based on a review and investigation of some of the main character-based string metrics, we developed a method to resolve partial name mentions within a document. This method applies the Jaro-Winkler string comparator and a variation of the Smith-Waterman string similarity measure. The method was applied to name mentions sourced from a sample of emails with a precision of 97%, and news articles with a precision of 100%.