Applications of Approximate Word Matching in Information Retrieval

James C. French Allison L. Powell *
Department of Computer Science
University of Virginia
Charlottesville, Virginia 22903
{french|alp4g}@cs.virginia.edu

Eric Schulman
National Radio Astronomy Observatory†
520 Edgemont Road
Charlottesville, VA 22903-2475
eschulma@nrao.edu

Technical Report No. CS-97-01
January 10, 1997

Abstract

As more online databases are integrated into digital libraries, the issue of quality control of the data becomes increasingly important, especially as it relates to the effective retrieval of information. The need to discover and reconcile variant forms of strings in bibliographic entries, i.e., authority work, will become more difficult. Spelling variants, misspellings, and transliteration differences will all increase the difficulty of retrieving information. Approximate string matching has traditionally been used to help with this problem. In this paper we introduce the notion of approximate word matching and show how it can be used to improve detection and categorization of variant forms.

1 Introduction

There are increasingly more online databases in the current climate of electronic publishing. The challenge is to integrate them into coherent digital libraries that let users have unimpeded access to accurate information. As the pace of electronic publication accelerates there will be increasing reliance on automated techniques to aid information providers as they seek to reach this goal.

For a number of years there has been an increasing emphasis on data quality in online databases [10]. In this paper we look at techniques to aid in detecting variant forms of strings in bibliographic databases. This is called authority work [2] and results in the creation of authority files that maintain the correspondence between all the allowable forms for strings in a particular bibliographic field, say, author or journal name.

Another problem arises when bibliographic databases are integrated. The different component databases might use different authority conventions. Users familiar with one set of conventions will expect their usual forms to retrieve relevant information from the entire collection when searching.

*This work supported in part by Dept. of Energy Grant no. DE-FG05-95ER25254 and NSF grant CDA-9529253.

†The National Radio Astronomy Observatory is a facility of the National Science Foundation operated under cooperative agreement by Associated Universities, Inc.

Therefore, a necessary part of the integration will be the creation of a joint authority file in which classes of equivalent strings are maintained. These equivalence classes can be assigned a canonical form which, in principle, could be substituted for the original strings in the combined database. In practice, this will generally be impractical or impossible because of intellectual property constraints. So a mapping will have to be maintained between searchs and systems that hides this heterogeneity from users. Tools are needed to automate this process. Techniques that underlie effective tools are the topic of this paper.

The remainder of the paper is organized as follows. First we discuss a particular problem to provide a framework for discussion. Then we show how approximate string matching can be used to help with the problem. We then propose approximate word matching as a mechanism for overcoming some of the remaining shortcomings. We conclude with a comparison of performance and propose a refinement to strict distance based approaches that we have been using in a large-scale application.

2 A Specific Problem

For the purposes of exposition we will consider a concrete problem where uncertainty in the data requires attention. We would like to automate the process of specifying canonical forms for text strings occurring in a bibliographic database. This is a step toward automating authority control.

We are collaborating with astronomers at the Smithsonian Astrophysical Observatory who have provided us with bibliographic records from the Astrophysics Data System (ADS)[1]. The ADS bibliographic records contain very standard bibliographic information including author(s), title, journal, date of publication, and institutional affiliations for the authors. Suppose we wish to create an index of publications by author affiliation. Among other things we will have to create a list of affiliations. Since the ADS does not currently have such an author affiliation list, we will need to derive it from the data. This work is part of a larger project described in [5]. Related work in astronomical sociology is described by Schulman [11, 12]. Others have considered similar problems with variant forms in bibliographic fields, for example, author names [13] and titles [15].

To attack this problem we must do two things: (1) decide on a set of canonical affiliation strings; and (2) assign each affiliation string in the database to one of these canonical strings. A related application is to decide the set of canonical forms and then replace all variant instances with the standard form. The two subproblems outlined above still apply.

Our approach to this problem is to apply a clustering algorithm to the set of strings to bring variants together. There is absolutely no way that this can be completely automated using lexical techniques alone. Consider the following two strings for example.

```
Leander McCormick Observatory, Charlottesville
Leander J. McCormick Observatory, Charlottesville
```

While we might reasonably deduce that they designate the same place, there is no way to determine that McCormick Observatory is really part of the Astronomy Department at the University of Virginia based on lexical considerations alone. Because we cannot completely automate the process, we concentrate on clustering the affiliations with the goal of minimizing the amount of manual inspection necessary to make final assignments. So our approach is to:

1. extract the strings;

2. cluster them as aggressively as possible consistent with the goal of minimizing misclassification;

3. have a domain expert review the outcome of step 2 with some automated aids and iterate until a final list has been produced; and

4. optionally, synthesize programs to (a) substitute the canonical forms for the variant forms in the original database and (b) screen newly entered data for conformance with the standard forms.

The second step above is the topic of the present paper. A case study of a specific application of steps one through three is detailed in a companion paper [5]. The fourth step is the topic of a forthcoming paper.

3 Clustering Approach

We will use examples from the ADS throughout the discussion that follows. We assume that we have extracted all the unique strings, $\{s_i\}$, together with the occurrence frequency of each string. The input to our clustering algorithm is a set of pairs $\{(s_i, f_i)\}$ denoting the strings and their frequencies. Figure 1 gives a sample of these pairs derived from all the variants of the institution *University of Virginia* occurring in a sample of 146,000 records from the ADS database. The variety of forms is typical of the variation occurring with all frequently referenced institutions. Note that the table also includes unique strings due to misspellings.

Given some distance metric $d(x, y)$, our heuristic clustering algorithm proceeds as follows.

Step 1: Sort the strings in descending order by frequency.

Step 2: Starting with the first (next) string, compute the distance $d(s_i, s_j)$ for all strings s_j, $j > i$. When $d(s_i, s_j) < \delta$, where δ is a threshold parameter, (1) insert s_j into cluster C_i and (2) remove s_j from further consideration.

Note: By construction, $f_i \geq f_k \ \forall s_k \in C_i$, that is, s_i is the most frequently ocurring string in the cluster C_i so it is the algorithm's nominee for the canonical label for that cluster.

Step 3: Increment i and repeat step 2.

The choice of the distance metric, $d(x, y)$, plays a crucial role in our clustering approach. We are trying to coalesce similar strings while accounting for misspellings and other variant forms. The next section discusses the well known edit distance metric [14, 8] and our proposed metric in more detail.

There is one remaining issue that needs some discussion. There are some variants that will defeat any attempt to organize strings by their similarity, specifically abbreviations and acronyms. It may be necessary to perform some domain specific transformations on the strings before clustering them. In some cases this step can drastically reduce the manual effort required to resolve the final assignment of strings to standard forms. An example of this phenomenon is discussed in [5].

Affiliation string	Number of occurrences
Univ. of Virgina, Charlottesville, VA, US	1
Univ. of Virginia, Charlottesvill, VA, US	1
Univ. of Virginia, Charlottesville, VA, US	44
Univ. of Virginia, Charlottsville, VA, US	1
Univ. of Virginia, VA, US	1
University of VA., Charlottesville	1
University of Virginia, Charlottesville, VA, US	23
University of Virginia, Charlottesville, Virginia, US	1
University of Virginia, Virginia, US	1
Virgina Univ., Charlottesville, VA, US	1
Virgina, University, Charlottesville, VA	1
Virginia Univ.	2
Virginia Univ., Charlottesville	58
Virginia Univ., Charlottesville, VA	1
Virginia Univ., Charlottesville, VA, US	4
Virginia University, Charlottesville	1
Virginia University, Charlottesville, VA	1
Virginia, University	57
Virginia, University, Charlottesville	204
Virginia, University, Charlottesville, VA	77
Virginia, University, Charlottesville, Va.	83

Figure 1: Raw affiliation strings for the University of Virginia

3.1 Edit Distance

The edit distance, $e(u, v)$, of two strings is a measure of similarity that is given by the minimal number of simple edit operations needed to transform u into v or vice versa. The four simple edit operations considered here are insertion of one character, deletion of one character, substitution of one character by another, and transposing two adjacent characters.

Edit distance has traditionally been used in approximate string matching [6], spelling error detection and correction [7], and more recently has been shown to be more effective than Soundex for phonetic string matching [16]. The edit distance measure is robust to spelling variants such as could occur when non-Roman alphabets are transliterated (e.g., Chebyshev and Tchebysheff have an edit distance of 3) or when misspellings occur.

Now there is the issue of how to set the threshold, δ, when using edit distance. An absolute threshold is problematic. A threshold of 5 seems reasonable when comparing two strings of length 45 and 50 characters respectively, but is not viable if the strings are of 5 and 10 characters. We suggest a relative threshold of the form $\delta = \alpha \min(|u|, |v|)$, that is, the threshold is some fraction, α, of the length of the shorter string. Otherwise short strings will erroneously appear similar to many unrelated longer strings.

One final consideration is that since $\epsilon(u, v)$ is calculated in $\Theta(mn)$ time by a dynamic pro-

gramming algorithm, it is a fairly expensive distance metric to compute. Next, we show when the computation can be avoided. But first we need the following lemma.

Lemma: $| |u| - |v| | \leq e(u,v) \leq \max(|u|, |v|)$.

Proof: The lower bound follows when one string is a substring of the other. In that case it is only necessary to delete (insert) the excess characters from (into) the longer (shorter) string. The upper bound follows from the fact that if u and v are disjoint it will be necessary to edit every character in the longer string. □

To simplify the notation somewhat, let m denote the length of the shorter string and n denote the length of the longer. Recall that we have suggested the decision rule $e(u,v) \leq \alpha m$ to protect shorter strings. We state the following result using this simplified notation.

Theorem: $e(u,v) \leq \alpha m$ only if $n - m \leq \alpha m$.

Proof: From the lemma, $n - m \leq e(u,v) \leq \alpha m$ and the result is immediate. □

Since $n - m \leq \alpha m$ is a necessary condition for $e(u,v) \leq \alpha m$, the expensive computation of $e(u,v)$ can be avoided when $n - m > \alpha m$. The following is an example of the benefit. In one of our subprocessing steps in [5], we computed the upper triangular matrix of pairwise edit distances for an 8,380 word vocabulary. On a Sun Ultra this took 938 seconds to compute. That time was reduced to 716 seconds after applying the test above. This was a net reduction in CPU time of 222 seconds or a 24% reduction in processing time.

3.2 Approximate Word Matching

Although edit distance is robust to spelling variants, it can be completely defeated by permutations of words. Consider the five strings in Figure 2(a). They clearly belong in three classes, $\{s_1, s_2\}$, $\{s_3, s_4\}$, and $\{s_5\}$. Each of these classes represents a separate institution and we would like to judge them as "similar" in the clustering algorithm. Figure 2(b) shows the pairwise edit distances for the strings of Figure 2(a). The length of each string is shown in parentheses. Note that $e(s_4, s_5) \ll e(s_3, s_4)$ so that *University of Vermont* is much more similar to *University of Virginia* than is *Virginia, University*. Also, in general, the edit distances are over one half the string length. Any threshold large enough to accept these strings as similar would admit a large number of unrelated strings.

The problem here is mainly due to word permutations, although spelling variants still play a role. The conventional representation of a string s is a sequence of characters, $s =< c_i >$. A more useful representation is to think of s as a set of words, $s = \{w_i\}$, where each "word", $w_i =< c_{ij} >$, is a sequence of characters. This representation provides more flexibility in the choice of comparators. We should also note that this set representation does not allow duplicate words. This has been advantageous in our work but there may be applications where a multiset is more appropriate.

The decision as to which tokens denote words is domain dependent. For the present purposes, we use a blank as the word separator and exclude punctuation. In general, punctuation will require more care in handling. This is discussed further in [5].

Our approximate word matching approach is to find a minimum distance matching of the words in u and v. The idea is to pair up the words so that the sum of the edit distances is minimized.

s_1: Moskovskii Gosudarstvennyi Pedagogicheskii Institut, Moscow
s_2: Moskovskij Pedagogicheskij Gosudarstvennyj University, Moscow
s_3: Virginia, University
s_4: University of Virginia
s_5: University of Vermont

(a)

$e(u, v)$

	s_1 (59)	s_2 (61)	s_3 (20)	s_4 (22)	s_5 (21)
s_1	0	36	50	50	50
s_2		0	45	52	51
s_3			0	17	16
s_4				0	5
s_5					0

(b)

$w(u, v)$

	s_1 (55)	s_2 (57)	s_3 (18)	s_4 (20)	s_5 (19)
s_1	0	11	56	52	52
s_2		0	48	44	44
s_3			0	2	7
s_4				0	5
s_5					0

(c)

Figure 2: A Comparative Example of $e(u, v)$ and $w(u, v)$

Note that if u and v contain a different number of words, then the cost of each excess word, w, is the edit distance between it and the null string, $e(w, \lambda) = |w|$, which is simply the length of the word. Our proposed word edit distance $w(u, v)$ is the sum of the edit distances resulting from a minimal matching. Figure 2(c) shows $w(u, v)$ for the strings of Figure 2(a). As with edit distance, we use a relative threshold for $w(u, v)$. However, we adjust the "length" of the new string representation to reflect the loss of blanks and perhaps other punctuation. The revised lengths are also shown in parentheses in Figure 2(c). Using a relatively conservative threshold of 0.2 of the length of the shorter string, it can be seen from Figure 2(c) that the strings of Figure 2(a) will be assigned to the correct clusters. Moreover, such a conservative threshold will allow relatively few unrelated strings to be grouped together.

While it is not generally true that $w(u, v) < e(u, v)$, when word permutations are considered it is almost always the case that $w(u, v) \ll e(u, v)$. In addition, when there are no word permutations involved, $w(u, v) \approx e(u, v)$ for strings with the same number of words. For these reasons, $w(u, v)$ will often characterize similarity better than $e(u, v)$ for particular applications. The practical implication is that a lower threshold can often be used in the clustering algorithm and this will result in fewer misclassifications.

3.3 Performance Evaluation

We compared the performance of $e(u, v)$ and $w(u, v)$ as distance functions in our clustering approach. We used a subset of the ADS affiliation data to compare the quality of clusters produced

using $e(u,v)$ and $w(u,v)$.[1] To get the subset, we selected all the affiliations located in the state of Virginia in the U.S. together with a few from West Virginia that were chosen because they are particularly hard to distinguish from some Virginia affiliations based solely on lexical considerations. There are 120 strings in this test set representating 57 institutions.

We hand clustered the test set into 57 distinct classes, one for each individual institution. Then we clustered the strings using $e(u,v)$ and $w(u,v)$ at several different threshold levels. The resulting machine-generated partitions were then compared to the hand-clustered partition along two dimensions — cluster purity and total misclassifications.

Cluster purity is a measure of self-consistency. It is the ratio of the number of clusters containing no misclassifications to the total number of clusters produced. The total number of strings misclassified is a measure of error and can be further broken down into: (a) the number of items placed into clusters when they should not have been; and (b) the number of items not placed in a cluster when they should have been. We regard category (a) as more serious because these items are difficult to locate manually and remove, whereas we can provide tools to help with the placement of those items occurring in category (b).

Distance measure	**Relative distance** (α)	**Purity of clusters**	**Total misclassified**	**Number incorrectly placed**	**Number not placed**
$e(u,v)$	0.20	78/79	29	1	28
	0.35	65/69	30	8	22
	0.50	55/62	30	13	17
$w(u,v)$	0.20	75/76	25	1	24
	0.35	62/66	23	4	19
	0.50	50/58	23	10	13

Table 1: Clustering Experiments using $e(u,v)$ and $w(u,v)$

The results of these experiments are shown in Table 1. Recall that α is the threshold parameter, the fraction of the shorter string that we will tolerate editing. Since we require $d(u,v) \leq \alpha m$, we must have $\frac{d(u,v)}{m} \leq \alpha$ for each of the distance measures. The distance metric $w(u,v)$ consistently produced better results over the entire range of thresholds examined. Had the strings in the benchmark all contained the same number of words, the approximate word matching approach would have been even better.

3.4 One Final Approach

There are still situations that can cause problems when using $w(u,v)$. Because the measure is the sum of the component edit distances, it may so happen that all the difference is due to one component. For example, consider the following strings.

```
University of California, Davis
University of California, Irvine
```

[1] This test set is available at ftp://ftp.cs.virginia.edu/pub/french/ads/va.benchmark.tar.

Here we have $w(u, v) = 4$ but all the distance is due to e ("`Davis`", "`Irvine`"). [2] These two strings would be judged similar at a threshold $\alpha = .2$, but they are clearly two different institutions.

We can address this problem by constraining the allowable edit distance between two components. Damerau[3] has noted that over 80% of all spelling errors are due to one of the four simple edit operations (insertion, deletion, substitution, and transposition). This has also been confirmed by Morgan[9]. So whenever $e(u, v) = 1$ it is almost certainly a spelling variant or misspelling. This observation leads to the following approach. First, as before, we find a minimal cost matching of the components in the string. Next we determine what components match sufficiently well. This is simply a bound on the allowable edit distance for a component. If we constrain the component difference to be ≤ 1 then we will almost certainly be dealing exclusively with spelling variants (e.g., center, centre, color, colour) or misspellings; a larger threshold will allow more variability and may be appropriate for some applications. The threshold is just a parameter of the algorithm and could be set to other values. After we determine how many words match sufficiently well, we compute a Jaccard coefficient, the ratio of the matching words in u and v to all the words in u and v, to use as a similarity function for clustering.

Similarity coefficient	**Purity of clusters**	**Total misclassified**	**Number incorrectly placed**	**Number not placed**
0.75	78/78	24	0	24
0.65	68/69	15	1	14
0.50	56/59	11	5	6
0.40	48/54	13	10	3

Table 2: Clustering Experiment using the Jaccard Coefficient

The results of this experiment using the Virginia affiliations benchmark are shown in Table 2. Although the threshold parameters in Tables 1 and 2 are not comparable, it is easy to see that the outcome at threshold .65 in Table 2 misplaces 1 item as do the approaches in Table 1 at threshold .2, but it correctly places 9 more items and does it with fewer total clusters.

4 Conclusion

There are many opportunities to exploit existing online databases as new techniques are developed in the field of information retrieval. Before this evolution can take place, we need to find ways to improve the quality of the data and to make it easier for data providers to integrate new information into their systems. This paper has discussed techniques for improving data quality and data access by detecting variable forms of strings and collecting them together under a standard form. This can be thought of as the semi-automatic generation of an authority file[5]. Our procedure still requires manual intervention, but after we generate authority files, we can create classifiers to reduce the burden of detecting variant forms as new data is acquired by a system.

[2] It is also the case here that $e(u, v) = 4$ because there are no word permutations.

We have given a method for speeding up the evaluation of edit distance when a threshold is being employed for clustering. We have also proposed a new way to evaluate string distance — word approximation — and shown that it can lead to superior results in some applications. We have also shown how word approximation can be adapted with component thresholds to improve the performance even more.

Affiliation cluster/string	Number of occurrences
Virginia, University, Charlottesville	502
Virginia, University, Charlottesville	431
University of Virginia, Charlottesville	70
University of Virginia, Charlottesville, Virginia	1
Virginia, University	59
University of Virginia	2
University of Virginia	1
University of Virginia, Virginia	1
University of VA., Charlottesville	1

Figure 3: Automatic affiliation clusters for the University of Virginia

We are currently applying these methods to the bibliographic data in the Astrophysics Data System (ADS) and the Networked Computer Science Technical Report Library (NCSTRL)[4] databases, two real-world online databases. The methods described in this paper were used in the pilot study [5] from which the records in Figure 1 were drawn; the effect of these methods on those records is shown in Figure 3. The 21 unique affiliation variants of Figure 1 have been grouped into four clusters. These are high quality clusters containing no misclassifications that are used in the second stage, manually supervised, clustering phase of our methodology. The methods are effective and efficient enough to be used in production environments. In the study reported in [5], we extracted 20,168 unique affiliation strings from our sample of 146,000 records. We were able to reduce these to 8,928 strings by using extremely conservative thresholds. As a result the error rate is so low that we could choose to ignore it.

One of our goals is to automatically generate auxiliary access structures for browsing online databases. The techniques described in this paper represent necessary steps to that end.

Acknowledgement. We would like to thank Stephen S. Murray, Guenther Eichhorn and Michael J. Kurtz of the Smithsonian Astrophysical Observatory for providing us access to the data in the Astrophysics Data System (ADS).

References

[1] A. Accomazzi, G. Eichhorn, M. J. Kurtz, C. S. Grant, and S. S. Murray. The ADS Article Service Data Holdings and Access Method. In G. Hunt and H. Payne, editors, *Astronomical*

Data Analysis Software and Systems VI, A.S.P. Conference Series, 1997. in press.

[2] L. Auld. Authority Control: An Eighty-Year Review. *Library Resources & Technical Services*, 26:319–330, 1982.

[3] F. J. Damerau. A Technique for Computer Detection and Correction of Spelling Errors. *Communications of the ACM*, 7(3):171–176, Mar. 1964.

[4] J. R. Davis. Creating a Networked Computer Science Technical Report Library. *D-Lib Magazine*, Sept. 1995.

[5] J. C. French, A. L. Powell, and E. Schulman. Automating the Construction of Authority Files in Digital Libraries: A Case Study. Technical Report CS-97-02, Department of Computer Science, University of Virginia, January 1997.

[6] P. A. V. Hall and G. R. Dowling. Approximate String Matching. *Computing Surveys*, 12(4):381–402, Dec. 1980.

[7] K. Kukich. Techniques for Automatically Correcting Words in Text. *Computing Surveys*, 24(4):377–440, Dec. 1992.

[8] R. Lowrance and R. A. Wagner. An Extension of the String-to-String Correction Problem. *Journal of the ACM*, 22(2):177–183, Apr. 1975.

[9] H. L. Morgan. Spelling Correction in Systems Programs. *Communications of the ACM*, 13(2):90–94, Feb. 1970.

[10] E. T. O'Neill and D. Vizine-Goetz. Quality Control in Online Databases. *Annual Review of Information Science and Technology*, 23:125–156, 1988.

[11] E. Schulman, J. C. French, A. L. Powell, S. S. Murray, G. Eichhorn, and M. J. Kurtz. The Sociology of Astronomical Publication Using ADS and ADAMS. In G. Hunt and H. Payne, editors, *Astronomical Data Analysis Software and Systems VI*, A.S.P. Conference Series, 1997. in press.

[12] E. Schulman, A. L. Powell, J. C. French, G. Eichhorn, M. J. Kurtz, and S. S. Murray. Using the ADS Database to Study Trends in Astronomical Publication. In *Bulletin of the American Astronomical Society*, volume 4, 1996.

[13] S. L. Siegfried and J. Bernstein. Synoname: The Getty's New Approach to Pattern Matching for Personal Names. *Computers and the Humanities*, 25(4):211–226, 1991.

[14] R. A. Wagner and M. J. Fischer. The String-to-String Correction Problem. *Journal of the ACM*, 21(1):168–173, Jan. 1974.

[15] M. E. Williams and L. Lannom. Lack of Standardization of the Journal Title Data Element in Databases. *Journal of the American Society for Information Science*, 32(3):229–233, May 1981.

[16] J. Zobel and P. Dart. Phonetic String Matching: Lessons from Information Retrieval. In *Proc. 19th Inter. Conf. on Research and Development in Information Retrieval (SIGIR'96)*, pages 166–172, Aug. 1996.