The field matching problem: Algorithms and applications

Alvaro E. Monge and **Charles P. Elkan**

Department of Computer Science and Engineering
University of California, San Diego
La Jolla, California 92093–0114

{amonge,elkan}@cs.ucsd.edu

Abstract

To combine information from heterogeneous sources, equivalent data in the multiple sources must be identified. This task is the field matching problem. Specifically, the task is to determine whether or not two syntactic values are alternative designations of the same semantic entity. For example the addresses *Dept. of Comput. Sci. and Eng., University of California, San Diego, 9500 Gilman Dr. Dept. 0114, La Jolla, CA 92093* and *UCSD, Computer Science and Engineering Department, CA 92093-0114* do designate the same department. This paper describes three field matching algorithms, and evaluates their performance on real-world datasets. One proposed method is the well-known Smith-Waterman algorithm for comparing DNA and protein sequences. Several applications of field matching in knowledge discovery are described briefly, including WEBFIND, which is a new software tool that discovers scientific papers published on the worldwide web. WEBFIND uses external information sources to guide its search for authors and papers. Like many other worldwide web tools, WEBFIND needs to solve the field matching problem in order to navigate between information sources.

Introduction

In many knowledge discovery and database mining applications there is a need to combine information from heterogeneous sources. The core issue is to identify equivalent data in the multiple sources. Through this operation, one can navigate from one source to another. This is what is commonly known as a "join" between two tables in relational databases.

In order to perform a join between two relations, one must first determine which columns refer to the same category of entities. This is known as the schema matching problem (Batini *et al.* 1986; Kim *et al.* 1993). Given a solution to the schema matching problem, one still needs to determine whether two specific tuples, i.e. field values, are equivalent. This is the problem studied in this paper. In general the field matching problem is to determine whether or not two field values are syntactic alternatives that designate the same semantic entity.

A solution to the field matching problem can be applied to solve the schema matching problem. The "information learning agent" (ILA) of Etzioni and Perkowitz (1995) learns the schema of other information sources based on the known schema of one source. To learn a new schema, equivalent pieces of information must be detected. The ILA can match (206) 616-1845 and 616.1845 for example, but details of the matching method are not given by Etzioni and Perkowitz (1995). General matching methods are the topic of this paper.

The field matching problem

Many information sources, e.g. relational databases or worldwide web pages, provide information about the same real-world entities, but designate these entities differently. We refer to a designator of an entity as a *field*. Table 1 contains examples of fields designating academic institutions. These examples show that fields can be made up of subfields delimited by separators such as newlines, commas, or spaces. A subfield is itself a field and may be made up of subsubfields, and so on. Two fields are *equivalent* if they are equal semantically, that is if they both designate the same semantic entity. Equivalence may sometimes be a question of degree, so we allow a function solving the field matching problem to return a value between 0.0 and 1.0, where 1.0 means certain equivalence and 0.0 means certain non-equivalence.

There has been little previous research on the field matching problem, although it has been recognized as important in industry for decades. For example tax agencies must do field matching to correlate different pieces of information about the same taxpayer when social security numbers are missing or incorrect. In general, field matching is the central issue in the so-called "merge/purge" task (Hernandez and Stolfo 1995): identifying and combining multiple records, from one database or many, that concern the same entity but are distinct because of data entry errors.

Published previous work deals with special cases of the field matching problem, involving customer addresses (Ace *et al.* 1992), census records (Slaven 1992),

or variant entries in a lexicon (Jacquemin and Royaute 1994). The most similar work to ours is due to Hernandez and Stolfo (1995), for the "merge/purge" task. After clustering tuples using indices, Hernandez and Stolfo (1995) use domain-specific equational axioms to identify semantically equivalent tuples inside each cluster. This approach depends on knowledge supplied by a human. We consider domain-independent methods here.

Field matching algorithms

The input to a field matching algorithm is the two fields being tested for semantic equivalence. In the work described here, stop words in the set {*and in for the of on & - /*} are removed before matching, but this is not critical.

A basic field matching algorithm

An atomic string is a sequence of alphanumeric characters delimited by punctuation characters. A simple definition of the degree to which two fields match is the number of their matching atomic strings divided by their average number of atomic strings. Two atomic strings match if they are the same string or if one is a prefix of the other. For example, consider the fields A = *"Comput. Sci. & Eng. Dept., University of California, San Diego"* and B = *"Department of Computer Science, Univ. Calif., San Diego"*. After removing stop words, $k = 6$ strings in the first field match some string in the second field, namely *Comput., Sci., San, Diego, Univ.* and *Calif.* No matches exist for *Eng.* and *Dept.* The overall matching score is $k/(|A| + |B|)/2) = 0.8$.

The algorithm to compute the basic matching score is straightforward. First, the atomic strings of each field are extracted and sorted. Second, each atomic string of one field is searched for in the other field's list of strings. The number of matched atomic strings is recorded. The complexity of the algorithm is dominated by the sort of the atomic strings, which uses time $O(n \log n)$ where n is the maximum number of atomic strings in either field.

The basic field matching algorithm does not take into account abbreviations which are not prefixes. It also does not use information regarding the ordering of the subfields. The algorithms described in the next two subsections attempt to overcome these limitations.

A recursive field matching algorithm

The algorithm here uses the recursive structure of typical textual fields. The base case is that A and B match with degree 1.0 if they are the same atomic string or one abbreviates the other; otherwise their degree of match is 0.0. Each subfield of A is assumed to correspond to the subfield of B with which it has highest score. The score of matching A and B then equals the mean of these maximum scores:

$$match(A, B) = \frac{1}{|A|} \sum_{i=1}^{|A|} \max_{j=1}^{|B|} match(A_i, B_j).$$

Matching of abbreviations uses four patterns:

(i) the abbreviation is a prefix of its expansion, e.g. "Univ." abbreviates "University", or

(ii) the abbreviation combines a prefix and a suffix of its expansion, e.g. "Dept." matches "Department", or

(iii) the abbreviation is an acronym for its expansion, e.g. "UCSD" abbreviates "University of California, San Diego", or

(iv) the abbreviation is a concatenation of prefixes from its expansion, e.g. "Caltech" matches "California Institute of Technology".

Note that case (iii) is a special case of case (iv).

The recursive field matching algorithm has quadratic time complexity. Given A and B, every subfield in A must be compared with every subfield in B. At the lowest level, each atomic string of A is compared with each atomic string of B. An important optimization is to apply memoization to remember the results of recursive calls which have already been made.

The Smith-Waterman algorithm

This method (Smith and Waterman 1981) is a dynamic programming algorithm. It was first developed to find optimal alignments between related DNA or protein sequences.

The Smith-Waterman algorithm has three main adjustable parameters. Given the alphabet Σ, the first parameter is a $|\Sigma| \times |\Sigma|$ matrix of match scores for each pair of symbols in the alphabet. The other parameters are penalties for starting a gap in an alignment, and for continuing a gap.

For matching textual fields we define Σ to be just the lower case and upper case alphabetic characters, the ten digits, and three punctuation symbols space, comma, and period. Our experiments use Smith-Waterman algorithm parameter values chosen in preliminary experiments as being intuitively reasonable and providing good results. The match score matrix is symmetric with all entries -5 except that an exact match scores 5, and approximate matches score 2. An approximate match occurs between two characters if they are both in one of the following subsets: {*d t*} {*g j*} {*l r*} {*m n*} {*b p v*} {*a e ı o u*} {*, .*}. The gap start and continue penalties are 5.0 and 1.0 respectively.

Since the Smith-Waterman algorithm allows for gaps of unmatched characters, it should perform well for many abbreviations, and when fields have missing information or minor syntactical differences. On the other hand, subfields out of order will certainly cause problems.

It is important to note that the Smith-Waterman algorithm and the basic algorithm are symmetric, but

internet host	*institution*
cs.ucsd.edu	computer science department, university of california, san diego
cs.stanford.edu	computer science department, stanford university, palo alto, california
(INSPEC)	Dept. of Comput. Sci., California Univ., San Diego, La Jolla, CA, USA.
(INSPEC)	Dept. of Comput. Sci. Stanford Univ., CA, USA.

Table 1: Example of NETFIND and INSPEC fields.

group size	*UCSD*	*Stanford*	*mixed*
1	31	44	81
2	9	11	25
3	4	7	14
4	1	0	2

Table 2: Equivalent groups in the three datasets

the recursive algorithm is asymmetric when two fields have different numbers of subfields.

An experimental comparison

Each algorithm was tested using three datasets. The first dataset contains 65 fields describing various academic departments at UCSD, and the second dataset contains 87 fields from Stanford, all taken from the INSPEC bibliographic database. The third dataset is the union of the UCSD and Stanford datasets and an additional 29 UCSD and Stanford fields from NETFIND, a service that gives internet host addresses and email addresses (Schwartz and Pu 1994).

For each dataset we identified groups of equivalent fields. For example, the two Stanford fields in Table 1 form a group. Table 2 shows the number of equivalent groups of each size in each dataset. A group of size 1 is a single field for which there was no match in a dataset.

The performance of a field matching algorithm can be evaluated by viewing the problem in information retrieval terms (Salton and McGill 1983). Given a set of possibly equivalent fields, consider each field in turn to be a query, and rank all other fields according to their degree of match as computed by the field matching algorithm. The accuracy of the algorithm then corresponds to retrieval effectiveness as measured by precision and recall. Recall is the proportion of relevant information actually retrieved, while precision is the proportion of retrieved information that is relevant. The amount of retrieved information varies based on what threshold is chosen for match scores. Typically, as recall is increased from 0% to 100%, precision decreases from 100% to 0%.

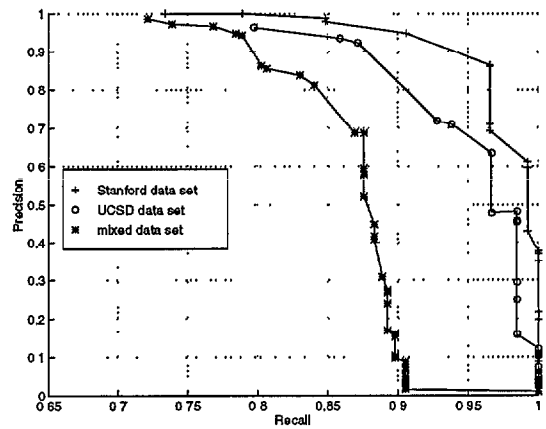

Figure 1: Basic algorithm: recall v. precision.

Figures 1 to 3 show recall versus precision for each algorithm. All algorithms perform worst on the mixed dataset, which is the most diverse. Both the Smith-Waterman algorithm and the recursive algorithm can achieve 100% recall, while the basic algorithm can only achieve 90% recall. The expected tradeoff between recall and precision is visible.

For all levels of recall, the Smith-Waterman algorithm has lower precision than the other two algorithms. The recursive and basic algorithms have similar performance. It appears that these datasets do not require the complexity of the recursive algorithm.

The WEBFIND application

WEBFIND is an application that discovers scientific papers made available by their authors on the worldwide web. The external information sources integrated by WEBFIND are MELVYL and NETFIND. MELVYL is a University of California library service that includes comprehensive databases of bibliographic records, including INSPEC (University of California 1996).

A WEBFIND search starts with the user providing keywords to identify the paper, exactly as he or she would in searching INSPEC directly. A paper can be identified using any combination of the names of its authors, words from its title or abstract, or other bibliographic information. After the user confirms that the right paper has been identified, WEBFIND queries INSPEC to find the institutional affiliation of the principal author of the paper. Then, WEBFIND uses NETFIND to provide the internet address of a host computer with the same institutional affiliation. WEBFIND then uses a search algorithm to discover a worldwide web server on this host, then an author's home page, and finally the location of the wanted paper.

Since institutions are designated very differently in INSPEC and NETFIND, it is non-trivial to decide when an INSPEC institution corresponds to a NETFIND in-

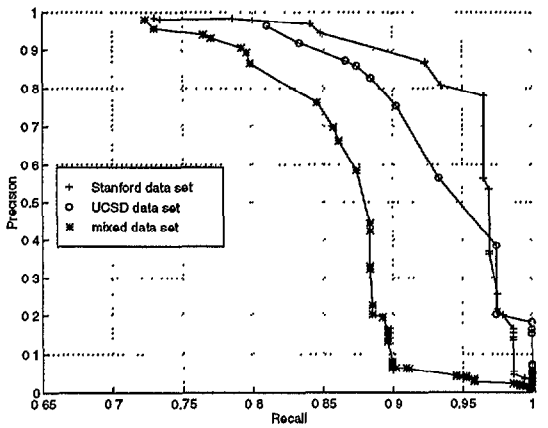

Figure 2: Recursive algorithm: recall v. precision.

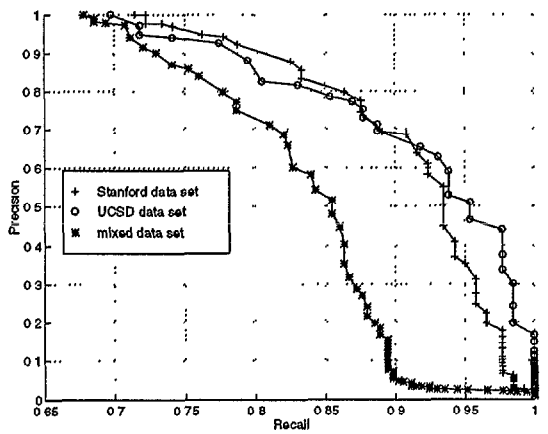

Figure 3: Smith-Waterman algorithm: recall v. precision.

stitution. WEBFIND uses the recursive field matching algorithm to do this.

Monge and Elkan (1996) provide a more lengthy description of WEBFIND, including experiments which show that WEBFIND is successful at finding worldwide web servers and finding web pages designated for authors. WEBFIND is less successful at finding actual papers, most of all because many authors have not yet published their papers on the worldwide web.

Conclusion

This study addresses the problem of reconciling information from heterogeneous sources. Such sources may represent entities differently, so identifying equivalent information is difficult. Future work will use an algorithm that is a hybrid of the Smith-Waterman method and the recursive method described above. This algorithm will take into account field order while allowing for missing fields. Parameter values for the Smith-Waterman component will be learned automatically by tuning on sets of representative data.

Acknowledgments

Alvaro Monge's work is supported by an AT&T Cooperative Research Program Fellowship.

References

J. Ace, B. Marvel, and B. Richer. Matchmaker...matchmaker...find me the address (exact address match processing). *Telephone Engineer and Management*, 96(8):50,52–53, April 1992.

C. Batini, M. Lenzerini, and S. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, December 1986.

Oren Etzioni and Mike Perkowitz. Category translation: learning to understand information on the internet. *International Joint Conference on AI*, pages 930–936, 1995.

M. Hernandez and S. Stolfo. The merge/purge problem for large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 127–138, May 1995.

C. Jacquemin and J. Royaute. Retrieving terms and their variants in a lexicalized unification-based framework. In *Proceedings of the ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 132–141, July 1994.

W. Kim, I. Choi, S. Gala, and M. Scheevel. On resolving schematic heterogeneity in multidatabase systems. *Distributed and Parallel Databases*, 1(3):251–279, July 1993.

Alvaro E. Monge and Charles P. Elkan. Integrating external information sources to guide worldwide web information retrieval. Technical Report CS96-474, Department of Computer Science and Engineering, University of California, San Diego, January 1996. http://www.cs.ucsd.edu/users/amonge/Papers/WebFind-CS96-474.ps.gz.

Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval.* McGraw Hill, 1983.

Michael Schwartz and Calton Pu. Applying an information gathering architecture to Netfind: a white pages tool for a changing and growing internet. *IEEE/ACM Transactions on Networking*, 2(5):426–439, October 1994.

B.E. Slaven. The set theory matching system: an application to ethnographic research. *Social Science Computer Review*, 10(2):215–229, Summer 1992.

T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

Division of Library Automation University of California. Melvyl system welcome page. URL, May 1996. http://www.dla.ucop.edu/.