Web Data Integration Using Approximate String Join

Yingping Huang Gregory Madey
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556
{yhuang3, gmadey}@nd.edu

ABSTRACT

Web data integration is an important preprocessing step for web mining. It is highly likely that several records on the web whose textual representations differ may represent the same real world entity. These records are called approximate duplicates. Data integration seeks to identify such approximate duplicates and merge them into integrated records. Many existing data integration algorithms make use of approximate string join, which seeks to (approximately) find all pairs of strings whose distances are less than a certain threshold. In this paper, we propose a new mapping method to detect pairs of strings with similarity above a certain threshold. In our method, each string is first mapped to a point in a high dimensional grid space, then pairs of points whose distances are 1 are identified. We implement it using Oracle SQL and PL/SQL. Finally, we evaluate this method using real data sets. Experimental results suggest that our method is both accurate and efficient.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems—*Relational databases, Textual databases*

General Terms

Algorithms,Performance,Experimentation

Keywords

data integration, approximate string join

1. INTRODUCTION

Web data integration is an important preprocessing step for web mining and data analysis. Web data is dirty due to user input errors, different flavors of abbreviations, etc. It is highly likely that two or more records which differ somewhat in textual representation actually represent the same real world entity. Such database records are called approximate duplicates. Data integration seeks to identify such approximate duplicates. Once approximate duplicates are identified, they can be merged into integrated records. Data integration sometimes is also called data cleansing, record linkage, etc.

In many existing data cleansing algorithms, approximate string processing is a fundamental step [2, 1, 3]. In these algorithms, a certain metric is used to define the distance between two strings. Possible metrics include edit distance, q-gram distance, and the vector cosine similarity metric. Once the string distance is defined, these algorithms try to find pairs of strings whose distances are below a certain threshold. In this paper, we use the vector cosine similarity metric.

Once the metric is specified, it is required to detect all pairs of strings such that their similarity is above a user specified threshold. In [7], this operation is referred to as a text join. A baseline approach to find all pairs of approximate duplicates is to apply a nested loop to compute the similarity between each pair of strings. In practice, it is computationally expensive. In this paper, we present a new method to address the approximate string join problem. The details of the algorithm are presented in Section 3.

WWW2004, May 17–22, 2004, New York, New York, USA.
ACM 1-58113-912-8/04/0005.

2. RELATED WORK

Li [6] proposed a mapping algorithm for efficient record linkage. In their approach, each string is mapped to a point in a high dimensional Euclidean space using FastMap. Then a similarity join algorithm proposed by Hjaltason and Samet [5] is used to identify close pairs of points in the hosting Euclidean space. The similarity join algorithm is very sensitive to the dimension of the hosting space. Thus when the dimension of the hosting space is large, the similarity join algorithm becomes very inefficient.

Gravano [7] presented a sampling approach for performing text join (as described later) and implemented it in an unmodified RDBMS. In their approach, each string is represented by a sparse vector in a high dimensional space. The dimension of the space is the distinct number of tokens in all the strings. A low dimensional subspace is used to calculate the similarity of strings. The accuracy of this approach depends on the dimension of the subspace.

3. OUR APPROACH

With review of the drawbacks of both Li's and Gravano's approaches, we designed a new algorithm which somewhat combines the two approaches. We first form the database of strings into a (1,2)-B metric space (as described later) and then map the (1,2)-B metric space into a high dimensional grid space instead of an Euclidean space. In the next step, pairs of points with distance 1 are identified. A metric space $M = (X, D)$ is called a (1,2)-B metric space, if the distance between any two different points is either 1 or 2, and for any points in X, there are at most B points within distance 1.

We can consider the database of strings to be a (1,2)-B metric space as follows. For two distinct strings s and t, if their similarity is above the user specified threshold, we define their distance to be 1, otherwise, their distance is 2. We also assume that each string has at most B (where B is reasonably large) other strings that are similar to it.

Guruswami and Indyk [4] proved that a (1,2)-B metric space can be isometrically embedded into a high dimensional grid space, as

shown in Lemma 1. In Lemma 1, l_∞^d is a vector space $\{0,1,2\}^d$ with the maximum distance, i.e., for any two vectors $x = (x_1, ..., x_d)$ and $y = (y_1, ..., y_d)$, their distance $\delta(x,y)$ is $\max_{1 \le i \le d} |x_i - y_i|$.

LEMMA 1. *A (1,2)-B metric space M=(X,D) can be isometrically embedded into l_∞^d, where $N = |X|$ and $d = O(B \log N)$.*

For each $1 \le i \le d$, we choose a subset S_i of X such that each element of X is included in S_i independently with probability $\frac{1}{B}$. Define mapping $F : M \to l_\infty^d$ by

$$F(x) = (D(x, S_1), D(x, S_2), ..., D(x, S_d))$$

It can be shown that F is isometric with high probability. The construction time of the mapping F is computationally expensive, which is not desirable. To improve the efficiency of the algorithm, some heuristic is necessary.

Using the tf.idf scheme from the information retrieval field, every string x can be represented by a vector $x = (x_1, ..., x_n)$ where n is the number of tokens obtained from all the strings. Let $A_{N \times n}$ denote the matrix formed by the vector representations of all the strings, where each row of A is a representation of a string. Also let $(S_i)_{N' \times n}$ denote the sub-matrix of A obtained from a subset of rows of A, where the expected value of N', $E(N') = \frac{N}{B}$.

From the construction of the mapping in the proof of the above lemma, we may need to compute AS_i^t for each $i \in \{1, ..., d\}$. Since we only need to compute $D(x, S_i)$, only the largest entry in each row of AS_i^t interests us. To compute AS_i^t is computationally expensive, and an estimation of AS_i^t would suffice to solve the original problem. For simplicity, we remove the subscript of S_i and denote S_i by S.

Let $A = (a_1 ... a_n)$ where a_j is the j-th column of A. Similarly, let $S = (s_1 ... s_n)$ where s_j is the j-th column of S. Then $AS^t = \sum_{j=1}^{n} a_j s_j^t$, i.e., AS^t is the sum of n matrices with rank 1. The representation of AS^t suggests that we can approximate AS^t by the sum of a subset of those rank 1 matrices.

The following is the algorithm to approximate the i-th coordinate of each x in the hosting space:

1. Choose an positive integer s and uniformly randomly choose s columns from A. Denote the columns by $a_{j_1}, ..., a_{j_s}$.
2. Compute $R = \sum_{k=1}^{s} a_{j_k} s_{j_k}^t$
3. For each row p of R, choose σ largest entries. Let q denote each such column.
4. Compute the similarity $sim(x_p, x_q)$ of strings x_p and x_q for each q in the above step.
5. Let x_q be the string which achieves the largest similarity in the above step.
6. If $sim(x_p, x_q) = 1$, then $(x_p)_i = 0$, else if $sim(x_p, x_q) > \phi$, then $(x_p)_i = 1$, else $(x_p)_i = 2$.

Repeat the above process d times. We obtain a mapping from the database of strings to the d-dimensional grid space $\{0,1,2\}^d$. The magnitude of $d = O(logN)$, and is much less than n, the total number of distinct tokens, in practice.

Denote the image of the mapping in $\{0,1,2\}^d$ by V. After we map the database of strings into the grid space $\{0,1,2\}^d$, we need to find all pairs of vectors in V, such that their distances are ≤ 1. This step can be easily done with a simple join. Note that the distance defined in $\{0,1,2\}^d$ is the maximal distance as mentioned in previous sections.

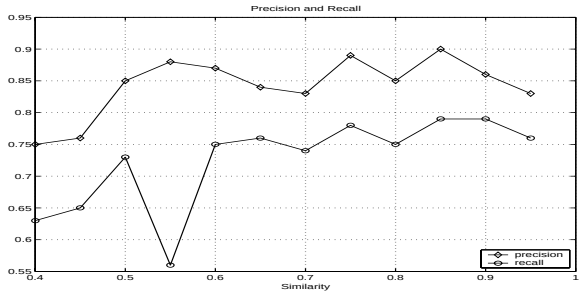

Figure 1: Precision and recall v.s. similarity

4. IMPLEMENTATION AND EXPERIMENTS

We implemented the algorithm using Oracle SQL and PL/SQL. We evaluated our algorithm against real world data. We downloaded the international movie data sets from Internet and used it to test the effectiveness and efficiency of our approximate text join algorithm. There are several parameters in our algorithm. In this experiment, we choose $B = 10$, $s = 4$ and $t = 4$. More experiments are still under way for different combinations of these parameters. Figure 1 shows that our algorithm is both efficient and accurate, in terms of recall and precision.

Our algorithm has some potential advantages over the algorithms presented in [7] and [6]. In [7], a large sample of the tuples is often necessary to obtain satisfactory precision and recall. Our algorithm computes the largest entry for each row of the matrix multiplication, a small sample suffices. In [6], the distance join algorithm used in the resulting Euclidean hosting space is very sensitive to the dimension. The processing time increases very quickly when the dimension increases. In our algorithm, the processing time is almost linear to the dimension of the hosting grid space.

5. REFERENCES

[1] L. Gravano and P. Ipeirotis. Approximate string joins in a database (almost) for free. In *Proc. 27th Int. Conf. on VLDB*, pages 491–500, 2001.

[2] L. Gravano and P. Ipeirotis. Using q-grams in a dbms for approximate string processing. In *IEEE Data Engineering Bulletin 24(4)*, pages 28–34, 2001.

[3] L. Gravano and P. Ipeirotis. Text joins for data cleansing and integration in an rdbms. In *Proc. Int. Conf. on Data Engineering*, 2003.

[4] V. Guruswami and P. Indyk. Embeddings and non-approximability of geometric problems. In *Proc. 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 537–538, 2003.

[5] G. Hjaltason and H. Samet. Incremental distance join algorithms for spatial databases. In *Proc. ACM-SIGMOD*, 1998.

[6] L. Jin, C. Li, and S. Mehrotra. Efficient record linkage in large data sets. In *Proc. 8th International Conference on Database Systems for Advanced Applications (DASFAA)*, 2003.

[7] N. K. L. Gravano, P. Ipeirotis and D. Srivastava. Text join in an rdbms for web data integration. *Proc. 12th international WWW conf.*, 2003.