Efficient Approximate Search on String Collections

(Tutorial)

Marios Hadjieleftheriou
AT&T Labs–Research
180 Park Ave Bldg 102
Florham Park NJ 07932
Phone:+19733607082, Fax:+19733608077
marioh@research.att.com

Chen Li
UC Irvine
Bren Hall, Room 2092
Irvine CA 92697
Phone:+19498249470, Fax:+9498244056
chenli@ics.uci.edu

ABSTRACT

This tutorial provides a comprehensive overview of recent research progress on the important problem of approximate search in string collections. We identify existing indexes, search algorithms, filtering strategies, selectivity-estimation techniques and other work, and comment on their respective merits and limitations.

1. MOTIVATION

Text data is ubiquitous. Management of string data in databases and information systems has taken on particular importance recently. This tutorial focuses on the following problem: Given a collection of strings, efficiently identify the ones similar to a given query string. Such a query is called an "approximate string search." This problem is of great interest for a variety of applications, as illustrated by the following examples.

Data Cleaning: Information from multiple data sources often have numerous inconsistencies. For example, the same real-world entity can be represented in slightly different formats, such as "`PO Box 23, Main St.`" and "`P.O. Box 23, Main St`". Errors can also be introduced due to irregularities in the data-collection process, from human mistakes, and many other causes. For these reasons, one of the main goals of data cleaning is to find similar entities within a collection, or all similar pairs of entities across a number of collections.

Query Relaxation: Often enough, users might pose SQL queries to a DBMS that contain selection predicates that do not match all of the relevant data within the database exactly. The reasons are possible errors in the query, inconsistencies in the data, limited knowledge about the data, and more. By supporting query relaxation, the DBMS can return data of potential interest to the user, based on query predicate similarity (e.g., returning "`Steven Spielberg`" as an answer to the query "`Steve Spielberg`").

Spell Checking: Given an input document, a spell checker finds potential candidates for a possibly mistyped word by performing an approximate string search in its dictionary.

Interactive Search: A very recent important application is to provide answers to query results in real-time, as users are typing their query (e.g., a Google search box with a drop-down suggestion menu that updates as users type). Such interactive-search boxes are ubiquitous and have shown to be very important in practice, because they limit the number of errors made by users and also reduce the number of query reformulations submitted in order to find the one that will yield satisfying results. The drawback of almost all existing, interactive techniques is that they support only prefix or substring matches, without regard for fuzzy, approximate searching; if users make a spelling mistake, they are presented with an empty suggestion box. One reason is that interactive approximate string search has attracted little attention and is not a trivial problem to solve, given the expensive nature of string similarity functions and ranking techniques.

These applications require approximate-string-search algorithms with a high real-time performance. For instance, consider a spell checker such as those used by Gmail, Hotmail, or Yahoo! Mail, which need to be invoked numerous times per second, in order to support the millions of concurrent users using these services. Each spell checking request needs to be processed as fast as possible. Clearly, higher throughput allows the server to serve a much larger number of users seamlessly. Another example is a business search on a *local-search* engine (e.g., YellowPages, Yahoo! Local, and Superpages). It is very often the case that users misspell business names (e.g., "Wall-mart" instead of "Wal-mart"), and hence approximate string search in the context of local-search is essential. Performing approximate string searching efficiently over the very large string collections present in these applications is fundamental in order to be able to sustain thousands of user requests per second.

A closely related problem is that of selectivity estimation for approximate-string-matching queries. It is of great interest to be able to efficiently and accurately evaluate the selectivity of selection queries for the purpose of query optimization (in order to design efficient query-execution plans). Clearly, the selectivity of approximate string matching queries depends highly on the similarity function used. Hence, a variety of selectivity-estimation algorithms have already been proposed in the literature, for different similarity functions and based on a diverse number of techniques (e.g., histograms, sampling, and clustering).

VLDB '09, August 24-28, 2009, Lyon, France

2. TUTORIAL OUTLINE

First, we will *motivate* the problem by using real examples and industrial-strength demos of approximate-search queries and various similarity functions. Then, we will focus on *list-merging search algorithms* [26, 21, 11] and *variable-length grams* [22, 29]. Next, we will focus on the emerging problem of *interactive approximate search* [14, 9]. Then, we will present a detailed explanation of *filtering techniques* for efficient candidate generation [10, 28, 6, 1, 4]. The final part of the tutorial will be devoted to *selectivity-estimation* techniques [16, 19, 20, 24, 12]. We will conclude the tutorial by outlining other *related work* [2, 3, 13, 23].

3. REFERENCES

[1] A. Arasu, V. Ganti, and R. Kaushik. Efficient exact set-similarity joins. In *VLDB*, pages 918–929, 2006.

[2] A. Arasu, S. Chaudhuri, K. Ganjam, and R. Kaushik. Incorporating string transformations in record matching. In *SIGMOD*, pages 1231–1234, 2008.

[3] A. Behm, S. Ji, C. Li, and J. Lu. Space-Constrained Gram-Based Indexing for Efficient Approximate String Search. In *ICDE*, 2009.

[4] K. Chakrabarti, S. Chaudhuri, V. Ganti, and D. Xin. An efficient filter for approximate membership checking. In *SIGMOD*, pages 805–818, 2008.

[5] S. Chaudhuri, V. Ganti, and R. Motwani. Robust identification of fuzzy duplicates. In *ICDE*, pages 865–876, 2005.

[6] S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In *ICDE*, page 5, 2006.

[7] S. Chaudhuri, B.-C. Chen, V. Ganti, and R. Kaushik. Example-driven design of efficient record matching queries. In *VLDB*, pages 327–338, 2007.

[8] S. Chaudhuri, A. D. Sarma, V. Ganti, and R. Kaushik. Leveraging aggregate constraints for deduplication. In *SIGMOD*, pages 437–448, 2007.

[9] S. Chaudhuri, R. Kaushik. Extending Autocompletion to Tolerate Errors. In *SIGMOD*, 2009.

[10] L. Gravano, P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. Approximate string joins in a database (almost) for free. In *VLDB*, pages 491–500, 2001.

[11] M. Hadjieleftheriou, A. Chandel, N. Koudas, and D. Srivastava. Fast indexes and algorithms for set similarity selection queries. In *ICDE*, pages 267–276, 2008.

[12] M. Hadjieleftheriou, X. Yu, N. Koudas, and D. Srivastava. Hashed samples: Selectivity estimators for set similarity selection queries. In *VLDB*, 2008.

[13] M. Hadjieleftheriou, N. Koudas, D. Srivastava. Incremental Maintenance of Length Normalized Indexes for Approximate String Matching. In *SIGMOD*, 2009.

[14] S. Ji, G. Li, C. Li, and J. Feng. Efficient Interactive Fuzzy Keyword Search. In *WWW*, 2009.

[15] L. Jin, N. Koudas, C. Li, and A. K. H. Tung. Indexing mixed types for approximate retrieval. In *VLDB*, pages 793–804, 2005.

[16] L. Jin and C. Li. Selectivity estimation for fuzzy string predicates in large data sets. In *VLDB*, pages 397–408, 2005.

[17] N. Koudas, C. Li, A. K. H. Tung, and R. Vernica. Relaxing join and selection queries. In *VLDB*, pages 199–210, 2006.

[18] N. Koudas, S. Sarawagi, and D. Srivastava. Record linkage: similarity measures and algorithms. In *SIGMOD*, pages 802–803, 2006.

[19] H. Lee, R. T. Ng, and K. Shim. Extending q-grams to estimate selectivity of string matching with low edit distance. In *VLDB*, pages 195–206, 2007.

[20] H. Lee, R. T. Ng, K. Shim. Approximate substring selectivity estimation. In *EDBT*, pages 827–838, 2009.

[21] C. Li, J. Lu, and Y. Lu. Efficient merging and filtering algorithms for approximate string searches. In *ICDE*, 2008.

[22] C. Li, B. Wang, and X. Yang. VGRAM: Improving performance of approximate queries on string collections using variable-length grams. In *VLDB*, pages 303–314, 2007.

[23] G. Li, S. Ji, C. Li, and J. Feng. Efficient type-ahead search on relational data: a TASTIER approach. In *SIGMOD*, 2009.

[24] A. Mazeika, M. H. Böhlen, N. Koudas, and D. Srivastava. Estimating the selectivity of approximate string queries. *ACM Transactions on Database Systems*, 32(2):12, 2007.

[25] G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001.

[26] S. Sarawagi and A. Kirpal. Efficient set joins on similarity predicates. In *SIGMOD*, pages 743–754, 2004.

[27] C. Xiao, W. Wang, and X. Lin. Ed-join: An efficient algorithm for similarity joins with edit distance constraints. In *VLDB*, 2008.

[28] C. Xiao, W. Wang, X. Lin, and J. X. Yu. Efficient similarity joins for near duplicate detection. In *WWW*, pages 131–140, 2008.

[29] X. Yang, B. Wang, and C. Li. Cost-based variable-length-gram selection for string collections to support approximate queries efficiently. In *SIGMOD*, 2008.