Approximate String Processing

Approximate String Processing

Marios Hadjieleftheriou

AT&T Labs - Research
180 Park Ave
Florham Park, NJ 07932
USA
marioh@research.att.com

Divesh Srivastava

AT&T Labs - Research
180 Park Ave
Florham Park, NJ 07932
USA
divesh@research.att.com

the essence of knowledge

Boston – Delft

Foundations and Trends® in Databases

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
USA
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is M. Hadjieleftheriou and D. Srivastava, Approximate String Processing, Foundation and Trends® in Databases, vol 2, no 4, pp 267–402, 2009

ISBN: 978-1-60198-418-0

Foundations and Trends® in Databases

Volume 2 Issue 4, 2009

Editorial Board

Editorial Scope

Foundations and Trends® in Databases covers a breadth of topics relating to the management of large volumes of data. The journal targets the full scope of issues in data management, from theoretical foundations, to languages and modeling, to algorithms, system architecture, and applications. The list of topics below illustrates some of the intended coverage, though it is by no means exhaustive:

- Data Models and Query Languages
- Query Processing and Optimization
- Storage, Access Methods, and Indexing
- Transaction Management, Concurrency Control and Recovery
- Deductive Databases
- Parallel and Distributed Database Systems
- Database Design and Tuning
- Metadata Management
- Object Management
- Trigger Processing and Active Databases
- Data Mining and OLAP
- Approximate and Interactive Query Processing
- Data Warehousing
- Adaptive Query Processing
- Data Stream Management
- Search and Query Integration
- XML and Semi-Structured Data
- Web Services and Middleware
- Data Integration and Exchange
- Private and Secure Data Management
- Peer-to-Peer, Sensornet and Mobile Data Management
- Scientific and Spatial Data Management
- Data Brokering and Publish/Subscribe
- Data Cleaning and Information Extraction
- Probabilistic Data Management

Information for Librarians

Foundations and Trends® in Databases, 2009, Volume 2, 4 issues. ISSN paper version 1931-7883. ISSN online version 1931-7891. Also available as a combined paper and online subscription.

Foundations and Trends® in
Databases
Vol. 2, No. 4 (2009) 267–402

DOI: 10.1561/1900000010

Approximate String Processing

Marios Hadjieleftheriou[1] and Divesh Srivastava[2]

[1] *AT&T Labs - Research, 180 Park Ave, Florham Park, NJ, 07932, USA, marioh@research.att.com*
[2] *AT&T Labs - Research, 180 Park Ave, Florham Park, NJ, 07932, USA, divesh@research.att.com*

Abstract

One of the most important primitive data types in modern data processing is text. Text data are known to have a variety of inconsistencies (e.g., spelling mistakes and representational variations). For that reason, there exists a large body of literature related to approximate processing of text. This monograph focuses specifically on the problem of *approximate string matching*, where, given a set of strings S and a query string v, the goal is to find all strings $s \in S$ that have a user specified degree of similarity to v. Set S could be, for example, a corpus of documents, a set of web pages, or an attribute of a relational table. The similarity between strings is always defined with respect to a similarity function that is chosen based on the characteristics of the data and application at hand. This work presents a survey of indexing techniques and algorithms specifically designed for approximate string matching. We concentrate on inverted indexes, filtering techniques, and tree data structures that can be used to evaluate a variety of set based and edit based similarity functions. We focus on all-match and top-k flavors of selection and join queries, and discuss the applicability, advantages and disadvantages of each technique for every query type.

Contents

1
Introduction

Arguably, one of the most important primitive data types in modern data processing is strings. Short strings comprise the largest percentage of data in relational database systems, long strings are used to represent proteins and DNA sequences in biological applications, as well as HTML and XML documents on the Web. In fact this very monograph is safely stored in multiple formats (HTML, PDF, TeX, etc.) as a collection of very long strings. Searching through string datasets is a fundamental operation in almost every application domain. For example, in SQL query processing, information retrieval on the Web, genomic research on DNA sequences, product search in eCommerce applications, and local business search on online maps. Hence, a plethora of specialized indexes, algorithms, and techniques have been developed for searching through strings.

Due to the complexity of collecting, storing and managing strings, string datasets almost always contain representational inconsistencies, spelling mistakes, and a variety of other errors. For example, a representational inconsistency occurs when the query string is ‘Doctors Without Borders’ and the data entry is stored as ‘Doctors w/o Borders’. A spelling mistake occurs when the user mistypes the query as ‘Doctors

Witout Borders'. Even though exact string and substring processing have been studied extensively in the past and a variety of efficient string searching algorithms have been developed, it is clear that approximate string processing is fundamental for retrieving the most relevant results for a given query, and ultimately improving user satisfaction.

How many times have we posed a keyword query to our favorite search engine, only to be confronted by a search engine suggestion for a spelling mistake? In a sense, correcting spelling mistakes in the query is not a very hard problem. Most search engines use pre-built dictionaries and query logs in order to present users with meaningful suggestions. On the other hand though, even if the query is correct (or the search engine corrects the query) spelling mistakes and various other inconsistencies can still exist in the web pages we are searching for, hindering effective searching. Efficient processing of string similarity as a primitive operator has become an essential component of many successful applications dealing with processing of strings. Applications are not limited to the realm of information retrieval and *selection queries* only. A variety of other applications heavily depend on robust processing of *join queries*. Such applications include, but are not limited to, record linkage, entity resolution, data cleaning, data integration, and text analytics.

The fundamental *approximate text processing problem* is defined as follows:

Definition 1.1 (Approximate Text Matching). Given a text T and a query string v one desires to identify all substrings of T that have a user specified degree of similarity to v.

Here, the similarity of strings is defined with respect to a particular similarity function that is chosen based on specific characteristics of the data and application at hand. There exist a large number of similarity functions specifically designed for strings. All similarity functions fall under two main categories, *set based* and *edit based*. Set based similarity functions (e.g., Jaccard, Cosine) consider strings as sets of tokens (e.g., q-grams or words), and the similarity is evaluated with respect to the number, position and importance of common tokens. Edit based similarity functions (e.g., Edit Distance, Hamming) evaluate the similarity of strings as a function of the total number of edit operations that are necessary to convert one string into the other. Edit operations can be insertions, deletions, replacements, and transpositions of characters or tokens.

Approximate text processing has two flavors, online and offline. In the online version, the query can be pre-processed but the text cannot, and the query is answered without using an index. A survey on existing work for this problem was conducted by Navarro [54]. In the offline version of the problem the text is pre-processed and the query is answered using an index. A review of existing work for this problem was conducted by Chan et al. [16].

Here, we focus on a special case of the fundamental approximate text processing problem:

Definition 1.2 (Approximate String Matching). Given a set of strings S and a query string v, one desires to identify all strings $s \in S$ that have a user specified degree of similarity to v.

The approximate string matching problem (which is also referred to as the approximate dictionary matching problem in related literature) is inherently simpler than the text matching problem, since the former relates to retrieving strings that are similar to the query as a whole, while the latter relates to retrieving strings that *contain a substring* that is similar to the query. Clearly, a solution for the text matching problem will yield a solution for the string matching problem. Nevertheless, due to the simpler nature of approximate string matching, there is a variety of specialized algorithms for solving the problem that are faster, simpler, and with smaller space requirements than well-known solutions for text matching. The purpose of this work is to provide an overview of concepts, techniques and algorithms related specifically to the approximate string matching problem.

To date, the field of approximate string matching has been developing at a very fast pace. There now exists a gamut of specialized data structures and algorithms for a variety of string similarity functions and application domains that can scale to millions of strings and can provide answers at interactive speeds. Previous experience has shown that for most complex problems there is almost never a one size fits all solution. Given the importance of strings in a wide array of applications, it is safe to assume that different application domains will benefit from specialized solutions.

There are four fundamental primitives that characterize an indexing solution for approximate string matching:

- The similarity function: As already discussed, there are two types of similarity functions for strings, set based and edit based.
- String tokenization: Tokenization is the process of decomposing a string into a set of primitive components, called tokens. For example, in a particular application a primitive component might refer to a word, while in some other application a primitive component might refer to a whole sentence. There are two fundamental tokenization schemes, overlapping and non-overlapping tokenization.
- The query type: There are two fundamental query types, selections and joins. Selection queries retrieve strings similar to a given query string. Join queries retrieve all similar pairs of strings between two sets of strings. There are also two flavors of selection and join queries, all-match and top-k queries. All-match queries retrieve all strings (or pairs of strings) within a user specified similarity threshold. Top-k queries retrieve the k most similar strings (or pairs of strings).
- The underlying index structure: There are two fundamental indexing schemes, inverted indexes and trees. An inverted index consists of a set of lists, one list per token in the token universe produced by the tokenization scheme. A tree organizes strings into a hierarchical structure specifically designed to answer particular queries.

Every approximate string indexing technique falls within the space of the above parametrization. Different parameters can be used to solve a variety of problems, and the right choice of parameters — or combination thereof — is dependent only on the application at hand.

This work explains in detail the available choices for each primitive, in an effort to delineate the application space related to every choice.

For example, consider a relevant document retrieval application that uses cosine similarity and token frequency/inverse document frequency weights[1] to retrieve the most relevant documents to a keyword query. The application uses a set based similarity function, implying a word-based, non-overlapping tokenization for keyword identification, a clear focus on selection queries, and most probably an underlying inverted index on keywords. Notice that this particular application is not related to approximate matching of keywords. A misspelled keyword, either in the query or the documents, will miss relevant answers. Clearly, to support approximate matching of keywords, a relevant document retrieval engine will have to use a combination of primitives.

As another example, consider an application that produces query completion suggestions interactively, as the user is typing a query in a text box. Usually, query completion is based on the most popular queries present in the query logs. A simple way to enable query suggestions based on approximate matching of keywords as the user is typing (in order to account for spelling mistakes) is to use edit distance to match what the user has typed so far as an approximate substring of any string in the query logs. This application setting implies an edit based similarity, possibly overlapping tokenization for enabling identification of errors on a per keyword level, focus on selection queries, and either an inverted index structure built on string signatures tailored for edit distance, or specialized trie structures.

The monograph is organized into eight sections. In the first four sections we discuss in detail the fundamental primitives that characterize any approximate string matching indexing technique. Section 2 presents in detail some of the most widely used similarity functions for strings. Section 3 discusses string tokenization schemes. Section 4 gives a formal definition of the four primitive query types on strings. Finally, Section 5 discusses the two basic types of data structures used to answer approximate string matching queries. The next three sections are dedicated to specialized indexing techniques and algorithms for

[1] Token frequency is also referred to as term frequency.

approximate string matching. Section 6 discusses set based similarity algorithms using inverted indexes. Section 7 discusses set based similarity algorithms using filtering algorithms. Finally, Section 8 discusses edit based similarity algorithms using both inverted indexes and filtering algorithms. Section 9 concludes the monograph.

References

[1] S. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *Journal of Molecular Biology*, vol. 215, pp. 403–410, October 1990.

[2] A. Amir, Y. Aumann, G. M. Landau, M. Lewenstein, and N. Lewenstein, "Pattern matching with swaps," in *IEEE Symposium on Foundations of Computer Science (FOCS)*, p. 144, 1997.

[3] A. Andoni and K. Onak, "Approximating edit distance in near-linear time," in *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pp. 199–204, 2009.

[4] A. Arasu, S. Chaudhuri, and R. Kaushik, "Transformation-based framework for record matching," in *Proceedings of International Conference on Data Engineering (ICDE)*, pp. 40–49, 2008.

[5] A. Arasu, S. Chaudhuri, and R. Kaushik, "Learning string transformations from examples," *Proceedings of the VLDB Endowment (PVLDB)*, vol. 2, no. 1, pp. 514–525, 2009.

[6] A. Arasu, V. Ganti, and R. Kaushik, "Efficient exact set-similarity joins," in *Proceedings of Very Large Data Bases (VLDB)*, pp. 918–929, 2006.

[7] A. N. Arslan and Ö. Eğecioğlu, "Dictionary look-up within small edit distance," in *Proceedings of the Annual International Conference on Computing and Combinatorics (COCOON)*, pp. 127–136, 2002.

[8] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval.* Addison Wesley, May 1999.

[9] Z. Bar-Yossef, T. S. Jayram, R. Krauthgamer, and R. Kumar, "Approximating edit distance efficiently," in *IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 550–559, 2004.

[10] R. J. Bayardo, Y. Ma, and R. Srikant, "Scaling up all pairs similarity search," in *WWW*, pp. 131–140, 2007.

[11] A. Behm, S. Ji, C. Li, and J. Lu, "Space-constrained gram-based indexing for efficient approximate string search," in *Proceedings of International Conference on Data Engineering (ICDE)*, pp. 604–615, 2009.

[12] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. E. Whang, and J. Widom, "Swoosh: A generic approach to entity resolution," *The VLDB Journal*, vol. 18, no. 1, pp. 255–276, 2009.

[13] G. Brodal and S. Venkatesh, "Improved bounds for dictionary look-up with one error," *Information Processing Letters*, vol. 75, no. 1–2, pp. 57–59, 2000.

[14] G. S. Brodal and L. Gasieniec, "Approximate dictionary queries," in *Proceedings of the Annual Symposium on Combinatorial Pattern Matching (CPM)*, pp. 65–74, 1996.

[15] H.-L. Chan, T.-W. Lam, W.-K. Sung, S.-L. Tam, and S.-S. Wong, "Compressed indexes for approximate string matching," in *Proceedings of the Annual European Symposium (ESA)*, pp. 208–219, 2006.

[16] H.-L. Chan, T.-W. Lam, W.-K. Sung, S.-L. Tam, and S.-S. Wong, "A linear size index for approximate pattern matching," in *Proceedings of the Annual Symposium on Combinatorial Pattern Matching (CPM)*, pp. 49–59, 2006.

[17] W. I. Chang and E. L. Lawler, "Approximate string matching in sublinear expected time," in *IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 116–124, vol. 1, 1990.

[18] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani, "Robust and efficient fuzzy match for online data cleaning," in *Proceedings of ACM Management of Data (SIGMOD)*, pp. 313–324, 2003.

[19] S. Chaudhuri, V. Ganti, and R. Kaushik, "A primitive operator for similarity joins in data cleaning," in *Proceedings of International Conference on Data Engineering (ICDE)*, p. 5, 2006.

[20] S. Chaudhuri and R. Kaushik, "Extending autocompletion to tolerate errors," in *Proceedings of ACM Management of Data (SIGMOD)*, pp. 707–718, 2009.

[21] S. Chaudhuri, A. D. Sarma, V. Ganti, and R. Kaushik, "Leveraging aggregate constraints for deduplication," in *Proceedings of ACM Management of Data (SIGMOD)*, pp. 437–448, 2007.

[22] A. Cobbs, "Fast approximate matching using suffix trees," in *Proceedings of the Annual Symposium on Combinatorial Pattern Matching (CPM)*, pp. 41–54, 1995.

[23] R. Cole, L.-A. Gottlieb, and M. Lewenstein, "Dictionary matching and indexing with errors and don't cares," in *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pp. 91–100, 2004.

[24] D. Comer, "The ubiquitous B-tree," *ACM Computing Surveys*, vol. 11, no. 2, pp. 121–137, 1979.

[25] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*. McGraw-Hill Higher Education, 2001.

[26] G. Cormode and S. Muthukrishnan, "The string edit distance matching problem with moves," *ACM Transactions on Algorithms (TALG)*, vol. 3, no. 1, pp. 1–19, 2007.

[27] M. G. Elfeky, A. K. Elmagarmid, and V. S. Verykios, "Tailor: A record linkage tool box," in *Proceedings of International Conference on Data Engineering (ICDE)*, pp. 17–28, 2002.

[28] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, "Duplicate record detection: A survey," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 19, no. 1, pp. 1–16, 2007.

[29] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algorithms for middleware," in *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pp. 102–113, 2001.

[30] P. Ferragina and R. Grossi, "The string b-tree: A new data structure for string search in external memory and its applications," *Journal of the ACM (JACM)*, vol. 46, pp. 236–280, 1999.

[31] 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 *Proceedings of Very Large Data Bases (VLDB)*, pp. 491–500, 2001.

[32] R. Grossi and F. Luccio, "Simple and efficient string matching with k mismatches," *Information Processing Letters*, vol. 33, no. 3, pp. 113–120, 1989.

[33] M. Hadjieleftheriou, A. Chandel, N. Koudas, and D. Srivastava, "Fast indexes and algorithms for set similarity selection queries," in *Proceedings of International Conference on Data Engineering (ICDE)*, 2008.

[34] M. Hadjieleftheriou, N. Koudas, and D. Srivastava, "Incremental maintenance of length normalized indexes for approximate string matching," in *Proceedings of ACM Management of Data (SIGMOD)*, pp. 429–440, 2009.

[35] A. Halevy, A. Rajaraman, and J. Ordille, "Data integration: The teenage years," in *Proceedings of Very Large Data Bases (VLDB)*, pp. 9–16, 2006.

[36] M. C. Harrison, "Implementation of the substring test by hashing," *Communications of the ACM*, vol. 14, no. 12, pp. 777–779, 1971.

[37] W.-K. Hon, T.-W. Lam, R. Shah, S.-L. Tam, and J. S. Vitter, "Cache-oblivious index for approximate string matching," in *Proceedings of the Annual Symposium on Combinatorial Pattern Matching (CPM)*, pp. 40–51, 2007.

[38] S. Ji, G. Li, C. Li, and J. Feng, "Efficient interactive fuzzy keyword search," in *WWW*, pp. 371–380, 2009.

[39] P. Jokinen and E. Ukkonen, "Two algorithms for approximate string matching in static texts," in *Proceedings of the Mathematical Foundations of Computer Science (MFCS)*, pp. 240–248, 1991.

[40] R. M. Karp and M. O. Rabin, "Efficient randomized pattern-matching algorithms," *IBM Journal of Research and Development*, vol. 31, no. 2, pp. 249–260, 1987.

[41] J. D. Kececioglu and D. Sankoff, "Exact and approximation algorithms for the inversion distance between two chromosomes," in *Proceedings of the Annual Symposium on Combinatorial Pattern Matching (CPM)*, pp. 87–105, 1993.

[42] D. K. Kim, J.-S. Lee, K. Park, and Y. Cho, "Efficient algorithms for approximate string matching with swaps," *Journal of Complexity*, vol. 15, no. 1, pp. 128–147, 1999.

[43] D. E. Knuth, "The art of computer programming, volume 3 (2nd ed.)," in *Sorting and Searching*, Addison Wesley Longman Publishing Co., Inc., 1998.

[44] N. Koudas, A. Marathe, and D. Srivastava, "Flexible string matching against large databases in practice," in *Proceedings of Very Large Data Bases (VLDB)*, pp. 1078–1086, 2004.

[45] N. Koudas, A. Marathe, and D. Srivastava, "Propagating updates in SPIDER," in *Proceedings of International Conference on Data Engineering (ICDE)*, pp. 1146–1153, 2007.

[46] N. Lester, A. Moffat, and J. Zobel, "Efficient online index construction for text databases," *ACM Transactions on Database Systems (TODS)*, vol. 33, no. 3, pp. 1–33, 2008.

[47] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals (in Russian)," *Doklady Akademii Nauk SSSR*, vol. 163, no. 4, pp. 845–848, 1965.

[48] C. Li, J. Lu, and Y. Lu, "Efficient merging and filtering algorithms for approximate string searches," in *Proceedings of International Conference on Data Engineering (ICDE)*, pp. 257–266, 2008.

[49] C. Li, B. Wang, and X. Yang, "Vgram: Improving performance of approximate queries on string collections using variable-length grams," in *Proceedings of Very Large Data Bases (VLDB)*, pp. 303–314, 2007.

[50] J. P. Linderman, Personal Communication, 2011.

[51] U. Manber and S. Wu, "An algorithm for approximate membership checking with application to password security," *Information Processing Letters*, vol. 50, no. 4, pp. 191–197, 1994.

[52] W. J. Masek and M. Paterson, "A faster algorithm computing string edit distances," *Journal of Computer and System Sciences*, vol. 20, no. 1, pp. 18–31, 1980.

[53] M. L. Minsky and S. A. Papert, *Perceptrons*: *An Introduction to Computational Geometry*. MIT Press, 1969.

[54] G. Navarro, "A guided tour to approximate string matching," *ACM Computing Surveys*, vol. 33, no. 1, pp. 31–88, 2001.

[55] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of Molecular Biology*, vol. 48, no. 3, pp. 443–453, 1970.

[56] R. Ostrovsky and Y. Rabani, "Low distortion embeddings for edit distance," *Journal of the ACM*, vol. 54, no. 5, pp. 23–36, 2007.

[57] O. Owolabi and D. R. McGregor, "Fast approximate string matching," *Software — Practice & Experience*, vol. 18, no. 4, pp. 387–393, 1988.

[58] P. A. Pevzner and M. S. Waterman, "A fast filtration algorithm for the substring matching problem," in *Proceedings of the Annual Symposium on Combinatorial Pattern Matching (CPM)*, pp. 197–214, 1993.

[59] S. Puglisi, W. F. Smyth, and A. H. Turpin, "A taxonomy of suffix array construction algorithms," *ACM Computing Surveys*, vol. 39, no. 2, p. 4, 2007.

[60] S. Sarawagi and A. Kirpal, "Efficient set joins on similarity predicates," in *Proceedings of ACM Management of Data (SIGMOD)*, pp. 743–754, 2004.

[61] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *Journal of Molecular Biology*, vol. 147, pp. 195–197, 1981.

[62] G. A. Stephen, *String Searching Algorithms.* World Scientific Publishing Co., 1998.

[63] E. Sutinen and J. Tarhio, "On using q-gram locations in approximate string matching," in *Proceedings of the Annual European Symposium (ESA)*, pp. 327–340, 1995.

[64] T. Takaoka, "Approximate pattern matching with samples," in *Proceedings of the International Symposium on Algorithms and Computation (ISAAC)*, pp. 234–242, 1994.

[65] W. F. Tichy, "The string-to-string correction problem with block moves," *ACM Transactions on Computer Systems (TOCS)*, vol. 2, no. 4, pp. 309–321, 1984.

[66] E. Ukkonen, "Algorithms for approximate string matching," *Information and Control*, vol. 64, no. 1–3, pp. 100–118, 1985.

[67] E. Ukkonen, "Approximate string-matching with q-grams and maximal matches," *Theoretical Computer Science*, vol. 92, no. 1, pp. 191–211, 1992.

[68] E. Ukkonen, "Approximate string matching over suffix trees," in *Proceedings of the Annual Symposium on Combinatorial Pattern Matching (CPM)*, pp. 228–242, 1993.

[69] R. Vernica and C. Li, "Efficient top-k algorithms for fuzzy search in string collections," in *Proceedings of the International Workshop on Keyword Search on Structured Data (KEYS)*, pp. 9–14, 2009.

[70] R. A. Wagner and M. J. Fischer, "The string-to-string correction problem," *Journal of the ACM*, vol. 21, no. 1, pp. 168–173, 1974.

[71] I. H. Witten, T. C. Bell, and A. Moffat, *Managing Gigabytes*: *Compressing and Indexing Documents and Images.* John Wiley & Sons, Inc., 1994.

[72] S. Wu and U. Manber, "Fast text searching: Allowing errors," *Communications of the ACM (CACM)*, vol. 35, no. 10, pp. 83–91, 1992.

[73] C. Xiao, W. Wang, and X. Lin, "Ed-join: An efficient algorithm for similarity joins with edit distance constraints," in *Proceedings of the VLDB Endowment (PVLDB)*, vol. 1, no. 1, pp. 933–944, 2008.

[74] C. Xiao, W. Wang, X. Lin, and H. Shang, "Top-k set similarity joins," in *Proceedings of International Conference on Data Engineering (ICDE)*, pp. 916–927, 2009.

[75] C. Xiao, W. Wang, X. Lin, and J. X. Yu, "Efficient similarity joins for near duplicate detection," in *WWW*, pp. 131–140, 2008.

[76] X. Yang, B. Wang, and C. Li, "Cost-based variable-length-gram selection for string collections to support approximate queries efficiently," in *Proceedings of ACM Management of Data (SIGMOD)*, pp. 353–364, 2008.

[77] A. C. Yao and F. F. Yao, "Dictionary look-up with one error," *Journal of Algorithms*, vol. 25, no. 1, pp. 194–202, 1997.

[78] Z. Zhang, M. Hadjieleftheriou, B. C. Ooi, and D. Srivastava, "B^{ed}-tree: An all-purpose index structure for string similarity search based on edit distance," in *Proceedings of ACM Management of Data (SIGMOD)*, 2010.

[79] J. Zobel and A. Moffat, "Inverted files for text search engines," *ACM Computing Surveys*, vol. 38, no. 2, p. 6, 2006.