Approximate string matching Algorithm

Narendra Kumar
Sr. Lecturer
Computer Science & Engg.,
Galgotia College of Engg. & Tech.
Greater Noida, U.P, India

Vimal Bibhu
Sr. Lecturer
Computer Science & Engg.,
Galgotia College of Engg. & Tech.
Greater Noida, U.P, India

Mohammad Islam
Sr. Lecturer
Computer Science & Engg.,
Galgotia College of Engg. & Tech
Greater Noida, U.P, India

Shashank Bhardwaj
Lecturer,
Master in Computer Application,
Krishna Institute of Engg. & Tech.,
Ghaziabad, U.P, India

Abstract - Approximate string matching is used when a query string is similar to but not identical with desired matches many patterns can be symbolically encoded as strings. Approximate string matching is the process of searching for optimal alignment of two finite-length strings in which comparable patterns may not be obvious; long strings subject to natural variations or random noise, for example, may share subtle, characteristic, underlying patterns of symbols. Use of the term approximate merely emphasizes the fact that a perfect match may not be achievable and that imperfections such as missing and extraneous symbols have to be considered. In many applications, one of the two strings is a prototype string that represents a pattern class and the other is a test string that we wish to analyze and/or classify.

Keywords: DNA - Deoxyribonucleic acid, Lexicon – A method of analysis, Substitution – A method to substitute something in place of other.

I. INTRODUCTION

The problem of string matching is very simply stated. Given a body of text *T [1…n]* we try to find a pattern *P [1…m]* where $m \leq = n$. This can be used to search bodies of text for specific patterns, or in biology, can be used to search strands of DNA for specific sequences of genes. The problem of exact string matching has been extensively researched. However, approximate string matching is a much more complicated problem to solve and has many more real world applications. Unfortunately, in real world

applications the problem is not so cut and dry. This is where approximate string matching comes in. Instead of searching for the string exactly, approximate string matching searches for patterns that are close to *P*. In other words approximate string matching allows for a certain amount of error between the two strings being compared.
One of the earliest applications of approximate string matching was in text searching. Another application of approximate string matching is in biology in DNA searching. One of the newest applications of string matching is to signal

II. OVERVIEW OF THE PROBLEM

Approximate string matching problem is to find approximate similar match for query string from the given database. Database may be organized in different ways. Mathematically the problem can be defined as consider two strings of text *T [1...n]* and *P [1...m]*, and a distance function *D (x [i...j], y [a...b])* where *x [i...j]* and *y [a...b*] denotes substrings of *x* and *y*. *D (x [i...j], y [a...b])* computes the minimal cost of converting *x [i...j]* in to *y [a…b*]. There are three operations we can perform to convert *x* into *y*, each with a cost of 1.

A. SUBSTITUTION

To perform a substitution we simply take one character in *x* and change it to match a character in *y*.

B. INSERTION

An insertion is when a character is simply inserted into *x* to match the character in *y* at the same position.

C. DELETION

This is the opposite of insertion. As the name suggests, it is the act of removing a character in *x*.

The final input to the approximate string matching problem is k, the maximum allowable error. Then the problem is to calculate the set of *P [i…j]* such that $D (T[x…y], P [i…j]) \leq k$.

III. APPROXIMATE MATCHING TECHNIQUE

There are two broad classes of schemes for approximate string matching that could be used for fine searches:

1. String similarity measures and
2. Phonetic coding

String similarity measures compute a numerical estimate of the similarity between two strings; such computation might be based on the number of characters they have in common, or the number of steps required transforming one into another. These measures are often referred to as *edit distances*. These measures can be used to rank a set of strings—that is, a *lexicon*—with respect to a query string. Such schemes are generally regarded as appropriate to

spelling correction, where around 80% of human errors are a single insertion, omission, or exchange. Phonetic coding, on the other hand, assign a phonetic code to each string; two strings are judged to be similar if they have the same code, and dissimilar otherwise. Phonetic schemes have been regarded as appropriate to personal name matching because it is possible for names that sound similar to have very different written forms. There are also several candidate coarse search schemes. One scheme would be to bucket the lexicon strings according to their phonetic code, and retrieve the bucket with the same code as the query string.
Another scheme is the use of n-*grams*, that is, indexing each string in a lexicon according to the character substrings it contains. Yet another scheme, which to our knowledge has not previously been applied to approximate matching, is to “permute” a lexicon by adding to it every rotation of every word, and find answers by binary search in the lexicon that result.

IV. NEW SIMILARITY MEASURE

During development of an algorithm for approximate matching by using N-gram technique fallowing assumptions are made,

1. For a given input string of length N only strings of length of (N-N/2) to (N+N/2) are selected from the database
2. There is no need to compare all N-gram (for N=1, 2, 3 …) of input string with the N-gram of strings stored in database.

a). Only those strings are considered for 2-gram comparison for which more than 50% of 1-grams are matched with 1-gram of input string form strings selected in assumption 1.
b). only those strings are considered for 3-gram comparison for which more than 40% of 2-gram are matched with 2-gram of input string form strings selected in step a.
c). only those strings are considered for 4-gram comparison for which more than 30% of 3-gram are matched with 3-gram of input string form strings selected in step b

All the string displayed as approximate matched strings for which 25% of 4-gram are matched with 4-gram of input string from strings selected in step c.
Similarly further comparisons provide accurate result with less time complexity.
This similarity measurer gives good results after implementing it into the algorithm,Still it required further refinement that we have made in modified similarity measurer.
We have seen that the % of N-gram that required being satisfied at different level of approximation is First approximation: 50% of 1-gram and Second approximation: 40% of 2-gram and Third approximation: 30% of 3-gram and Fourth approximation: 25% of 4-gram
Now we consider the case of string of length N=27, for which
Number of 1-gram =27
Number of 2-gram =26
Number of 3-gram =25
Number of 4-gram =24

Then 25% of 4-gram will return 6 that is 6, 4-gram of query string must be matched with target strings from database which are displayed as final results. By the definition of N-gram six 4-gram formed by at least 9(lower bound), 1-grams (when all 4-grams are exists in continuity) or at max 24(upper bound), 1-grams (when all 4-grams are exists separately). But our first approximation criteria require 50% of 1-gram should be matched that is 13, 1-gram should be matched, which does not satisfied lower bound on fourth approximation. Thus we get variation in results, so to remove this variation we change % of N-gram matched in different approximation. Following table show variation in approximation criteria through which we draw important results for designing an efficient similarity measurer.

Table 1: Variation in approximation criteria

N-gram	Variation-1	Variation-2	Variation-3
1-gram	75%	50%	25%
2-gram	60%	40%	20%
3-gram	50%	30%	15%
4-gram	40%	25%	10%

On the basis of results of these variations shown in table 1 approximation criteria, we are able to draw important conclusion that is the approximation criteria should vary with length of query string.

V. MODIFIED SIMILARITY MEASURE

1. For a given input string of length N only strings of length of (N-N/2) to (N+N/2) are selected from the database.

2. There is no need to compare all N-gram (for N=1, 2, 3 …) of input string with the N-gram of strings stored in database.

3. If length of the query string is less than and equal to six than
   i. Only those strings are considered for 2-gram comparison for which more than 75% of 1-grams are matched with 1-gram of input string form strings selected in assumption 1.
   ii. All the string displayed as approximate matched strings for which 50 of 2-gram are matched with 2-gram of input string from strings selected in step (i).

4. If length of the query string is less than and equal to Nine than

i. Only those strings are considered for 2-gram comparison for which more than 50% of 1-grams are matched with 1-gram of input string form strings selected in assumption 1.
ii. Only those strings are considered for 3-gram comparison for which more than 40% of 2-gram are matched with 2-gram of input string form strings selected in step (i).
iii. Only those strings are considered for 4-gram comparison for which more than 30% of 3-gram are matched with 3-gram of input string form strings selected in step (ii)
iv. All the string displayed as approximate matched strings for which 25% of 4-gram are matched with 4-gram of input string from strings selected in step (iii).

5. If length of the query string greater than Nine then

i. Only those strings are considered for 2-gram comparison for which more than 25% of 1-grams are matched with 1-gram of input string form strings selected in assumption 1. Also keep the track of total number of 1-gram matched (C1).
ii. Only those strings are considered for 3-gram comparison for which more than 20% of 2-gram are matched with 2-gram of input string form strings selected in step (i). Also keep the track of total number of 2-gram matched (C2).
iii. Only those strings are considered for 4-gram comparison for which more than 15% of 3-gram are matched with 3-gram of input string form strings selected in step (ii). Also keep the track of total number of 3-gram matched (C3).

iv. Only those strings are considered for Next comparison for which more than 10% of 4-gram are matched with 4-gram of input string form strings selected in step (iii). Also keep the track of total number of 4-gram matched (C4).
v. All the string displayed as approximate matched strings for which the condition ((C1/C4) < 3.5 OR (C2/C4) < 2.5 OR (C3/C4) < 1.5) is satisfied.

VI. PPROPOSED ALGORITHM

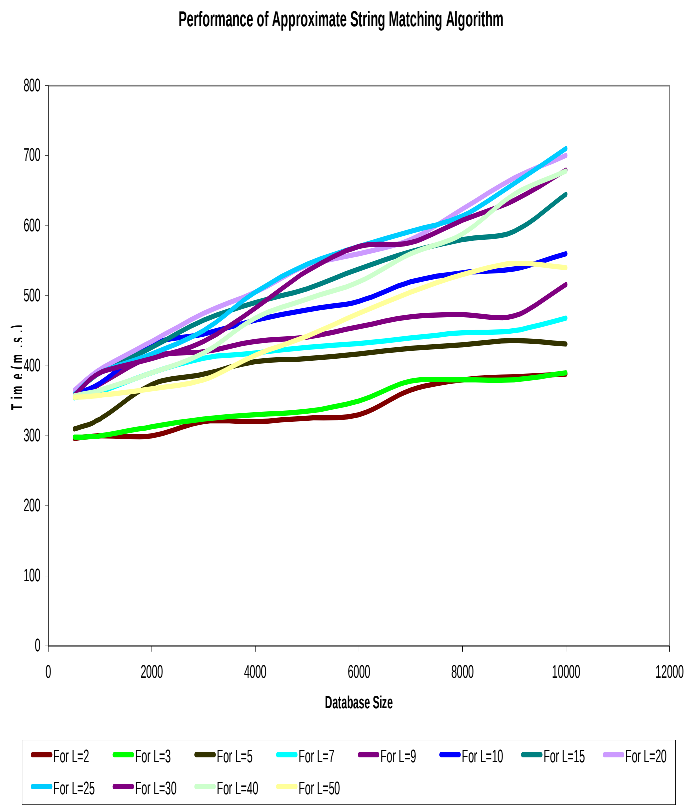
Database:

Trademarkinformation(TextId,Trademarkname, Textlength),

Input string: S,
Length of input string: N,
Average text length: L,

1. For input string call a procedure to evaluate N-gram (N=1, 2….)
2. For strings from database call a procedure to evaluate N-gram (N=1, 2….)
3. Select strings from database for which L>= (N-N/2) and L<=(N+N/2)
4. call modified similarity measurer procedure
5. Display the results.

VII. PERFORMANCE MATCHING GRAPH



CONCLUSION:

We have shown that the n-gram technique with some modification provides excellent search results and we get linear time relationship for finding the approximate match from the database of different sizes. We have shown that n-gram indexing provides an excellent coarse search mechanism for identifying approximate matches in a large lexicon. By varying n, the index can be kept small or query evaluation can be fast, with small n giving better effectiveness. We have also variation in time for accessing the approximate match from the database during the operation of algorithm. Basically the factor depend on the no matched find by the program from the database and further this variation depend on the number of strings in the database of the particular length Thus this time variation is depend on the length of query string, L and number of strings in the database of the length U where (L-L/2) <=U <= (L+L/2).

REFERENCES:

[1] 1. C.L. Borgman and S.L. Siegfried, 'Getty's Synoname and its cousins: A survey of applications of personal name-matching algorithms', *Journal of the American Society for Information Science*, **43**, (7), 459–476,(1992).

[2] P.A.V. Hall and G.R. Dowling, 'Approximate string matching', *Computing Surveys*, **12**, (4), 381–402, (1980).

[3] K. Kukich, 'Techniques for automatically correcting words in text', *Computing Surveys*, **24**, (4), 377–440, (1992).

[4] National Institute of Standards and Technology. *Proc. Text Retrieval Conference (TREC)*, Washington, November 1992. Special Publication 500-207.

[5] H.J. Rogers and P. Willett, 'Searching for historical word forms in text databases using spelling correction methods: reverse error and phonetic coding methods', *Journal of Documentation*, **47**, (4), 333–353, (1991).

[6] G. Salton, *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*, Addison-Wesley, Reading, MA, 1989.

[7] S. Wu and U. Manber, 'Fast text searching allowing errors', *Communications of the ACM*, **35**, (10), 83–91, (1992).

[8] E. Ukkonen, 'Approximate string-matching with n-grams and maximal matches', *Theoretical Computer Science*, **92**, 191–211, (1992).

[9] T.N. Gadd, "Fisching fore werds': Phonetic retrieval of written text in information systems', *Program: automated library and information systems*, **22**, (3), 222–237, (1988).

[10] T.N. Gadd, 'PHONIX: The algorithm', *Program: automated library and information systems*, **24**, (4), 363– 366, (1990).

[11] T.C. Bell, A. Moffat, C.G. Nevill-Manning, I.H.Witten, and J. Zobel, 'Data compression in full-text retrieval systems', *Journal of the American Society for Information Science*, **44**, (9), 508–531, (October 1993).

[12] J. Zobel, A. Moffat, and R. Sacks-Davis, 'An efficient indexing technique for full-text database systems', *Proc. International Conference on Very Large Databases*, Vancouver, Canada, August 1992, pp. 352–362.

AUTHORS PROFILE

Narendra Kumar has received his Bachelor of Technology and Master in Technology in Computer Science & Engineering from UP Technical University, Lucknow, India. He is working as Senior Lecturer in the Department of Computer Science & Engineering at Galgotia College of Science & Technology, Greater Noida, India. He is a member of various Technical Societies viz. Computer Society of India (CSI), Indian Society of Technical Education (ISTE). He published many research papers in various Conferences. His main research interests include: Wireless Sensor Network, Distributed & Mobile Computing and Middleware.

Vimal Bibhu has received his Bachelor of Science in Chemistry(Hons.) from Magadh University, Bodh Gaya, Bihar, India, Post Graduate Diploma in Information Technology from IGNOU, New Delhi, India, Master in Computer Application from IGNOU, New Delhi, India and Master in Technology in Computer Science & Engineering from CDAC Noida, Affiliated to Guru Gobind Singh Indraprastha University, New Delhi, India. He is working as Sr. Lecturer in the Department of Computer Science & Engineering at Galgotia's College of Engineering & Technology, Greater Noida, Uttar Pradesh India. He is a member of various Technical Societies viz. International Association of Computer Science & Information Technology (IACSIT), International Association of Engineers (IEANG). He published many research papers in various International Journals and Conferences.

Mohammad Islam has received his Bachelor of Technology and he is also perusing Master of Technology in Computer Science & Engineering from Uttar Pradesh Technical University, Lucknow. He is working as Senior Lecturer in the Department of Computer Science & Engineering at Galgotia College of Engineering & Technology, Greater Noida, India. He is a member of various Technical Societies viz. Computer Society of India (CSI), Indian Society of Technical Education (ISTE). He published many research papers in various Conferences. His main research interests include: Wireless Sensor Network, Distributed & Mobile Computing and Middleware.

Shashank Bhardwaj has received his Bachelor of Technology in information Technology and he is also perusing Master of Technology in Computer Science & Engineering from Uttar Pradesh Technical University, Lucknow. He is currently working on the post of Lecturer in department of Master of Computer Application at Krishna Institute of Engineering & Technology, Ghaziabad, India. He is a member of various Technical Societies viz. Computer Society of India (CSI), Indian Society of Technical Education (ISTE). He published many research papers in various Conferences. His main research interests include: Wireless Sensor Network, Distributed & Mobile Computing and Middleware.