Similarity Search Using Pre-Search in UniRef100 Database

Maulika S Patel[1], Himanshu S Mazumdar[2]

[1] *Head, Department of Computer Engineering,*
G H Patel College of Engineering & Technology, Vallabh Vidyanagar, India
And Research Scholar, Research & Development Center,D D University, Nadiad, India, maulika.sandip@gmail.com

[2] *Head, Research & Development Center*
Dharmsinh Desai University, Nadiad, India
Senior Member, IEEE, hsmazumdar@hotmail.com

Abstract

Sequence similarity in biological databases is used to characterize a newly discovered protein and confirming the existence of its homologs. This is often computationally very expensive. We have implemented a new algorithm that performs sequence similarity search using a pre-search phase. The proposed algorithm works in three phases. As a pre-preparation for Pre-Search, we locate a sequence, similar to the query sequence to extract all common words between the former and the latter. In the second phase, the pre-search phase, we locate all sequenes containing any of the randomly chosen common words. The list is further scanned in the third phase and the results obtained from the second phase are refined using Similarity Search (SS) algorithm, described in the paper. We have preprocessed the Uniref100.FASTA protein database containing 9,757,328 records downloaded from uniprot.org, to suit our application of sequence similarity search. The algorithm is simple and can be applied in various perspectives. These include searching in DNA and protein sequence databases, motif finding, and gene identification search. Pre-Search reduces the search space using much faster simpler algorithm. In large database search, its effect could be phenomenal.

Keywords: *UniRef100 protein database, sequence similarity, sequence alignment, Pre-search, sequence similarity, similarity score, randomized algorithm*

1. Introduction

"Members of the 1000 Genomes Project plan to sequence as many as 1,100 samples by the end of this summer as part of the project's main sequencing effort" [17] as heard by the attendees of the Biology of Genomes meeting on May 14, 2010. Genome sequencing leaves a plethora of sequence information, both protein and DNA. More important is the manner in which these large repositories are managed and used to access information. It is common in molecular biology to try to discover the function of a DNA or protein sequence by relating it to other sequences [8]. Proteins function through their conformation. It is theorized that proteins that share a similar sequence generally share the same basic structure [14]. Similarity is defined as the extent to which nucleotide or protein sequences are related. Similarity is relative in nature and depends on the application or researchers' needs. With increase of similarity criteria, group size reduces. Protein sequence similarity typically implies homology, which in return may imply structural and functional similarities. Homology modeling approaches have used similarity measures as one of

the parameters deciding homology. Hence sequence similarity remains the most demanding and widely addressed problem so far. Being able to quickly identify the similar sequences from very large databases for a newly obtained sequence can provide significant clue to its function. In this paper, we propose a new similarity search (SS) algorithm along with a short literature review discussing popular sequence similarity algorithms, both in the alignment and alignment-free category. In spite of the fact that the processing speed is increasing, the speed of the algorithms remains an important consideration because the databases are also increasing in size [15]. It is very obvious that use of heuristics in guiding the similarity search can probably yield promising results. Although, there do exist standard and popular algorithms for sequence similarity, we have found a need for development of algorithms that take into account the growing database sizes. We attempt to generate an exhaustive search on the Uniref100.FASTA protein database downloaded on 31st Jan 2010 and contain 9,757,328 records.

The proposed method of sequence similarity works in three phases. In the first phase, we locate a sequence, similar to the query sequence to extract all common words between the former and the latter. This search is performed in nearby locations of the query sequence location. The justification for the same is provided in Section 5. The heuristics used in this method is to use these two sequences for finding similarity between the query sequence and the sequences in the databases. We have used two algorithms. One is accurate in finding the similar sequences, but computationally expensive. The other is approximately accurate but is an order of magnitude faster than the first one. The motivation behind the development of this algorithm is the simple fact that there exists more number of dissimilar sequences in the database than the similar ones. Hence, computationally intensive exhaustive searching should be avoided. The speed is achieved at the cost of accuracy, but the accuracy can be further increased by suitably modifying the heuristics. With large databases like the UniRef100.FASTA, any saving through the search algorithm has substantial gain. The flowchart of the entire similarity search method is shown in Fig. 1.

The paper is organized as follows. Section 2 briefs about the public protein databases and the need for sequence analysis. A brief literature review on the current state of the art sequence comparison methods is presented in Section 3. Section 4 describes the algorithm proposed for sequence similarity. This section also presents a theoretical analysis of the algorithm. Section 5 covers experimental results for sequence similarity. Section 6 has concluding remarks and an idea of the ongoing work.

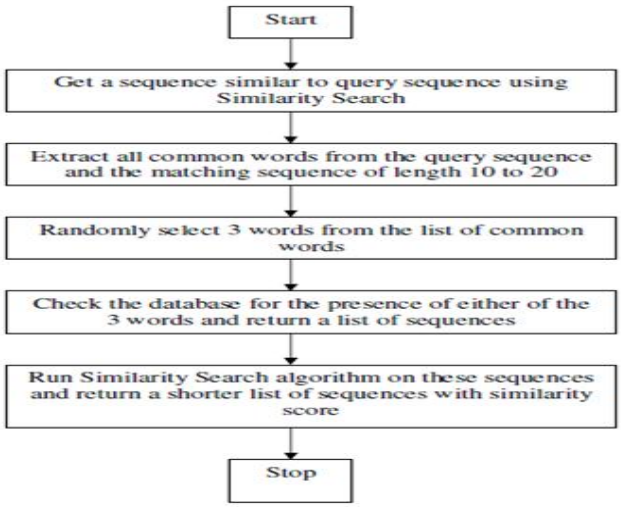

Figure 1. Flowchart Showing the Steps Involved in Performing Sequence Similarity Search

2. Sequence Databases and Sequence Analysis

Biology in the 21st century is being transformed from a purely lab-based science to an information science as well [14]. Computational biology [9] tasks such as multiple sequence alignment [18], sequence similarity [19], motif finding, and structure prediction [20] have attracted many researchers. The biological databases are very rich in nature in terms of content and size. The primary public protein databases are Protein Data Bank (PDB) and SWISS-PROT. The genomic databases are GenBank in USA, DNA Data Bank in Japan (DDBJ) and European Molecular Biology Laboratory DNA database (EMBL) in Europe.

This availability of free and massive data provides ample opportunities to computational biologists to experiment and test their softwares and tools for the tasks of biological relevance. This also ensures that machine learning tools that heavily depend on the training data may be used effectively. The growth rate of protein and genomic databases can be visualized in [12], [14]. As more and more whole genomes are sequenced, the need for a central, publicly available and easily accessible archive for deposition, searching and analysis of sequence data continues to grow [5]. With the exponential increase in the size of the genomic and proteomic databases, there exists a need to hierarchically maintain these databases for fast and relevant retrieval. Sequence similarity is also important for classifying the existing proteins and also categorizing the newly invented proteins.

Proteins are polymers of amino acids. Proteins can be viewed as a sequence of strings over a 20 letter alphabet; wherein each alphabet corresponds to an amino acid. Each protein has a unique sequence of amino acids, which specifies protein shape and function. Proteins are involved in various functions of cell. Proteins are complex molecular structures, an important feature, yet not well understood, is how they bind to each other and when. We are interested in exploiting the string nature of a protein sequence, which allows us to compare sequences, identify substrings, and produce a list of similar sequences.

In a search for similarity in a huge database as UniRef100.Fasta, generally a very small subset of the database will be homologous to the query sequence. Therefore, we are normally interested in listing those sequences that resemble the query sequence to an acceptable threshold. These sequences might have strong similarity with the query or might loosely resemble the query. In literature, there exists mention of two types of similarity; Global and Local. The global similarity approaches basically try to align the two sequences. While in local similarity, we try to identify the substrings or sub parts of the sequences that have a match. Local similarity measures are useful for database searches.

3. Literature Review

We, in this paper, discuss the various methods of sequence analysis of biological sequences. There are mainly two approaches having mention in literature, for this purpose; Sequence alignment based approach and Alignment-free approach.

Extensive work and research has been done in the alignment based approach. Sequence alignment method, though very powerful and popular, leave space for further research, to optimize the computational complexity arising out of the alignment process. An example of the pair wise sequence alignment and multiple sequence alignment is given below.

Alignment based methods such as those based on Dynamic programming [Needleman, smith] [7], FASTA [10] and BLAST [6] have been developed for identifying sequence similarity. BLAST has been widely used by biologists for sequence analysis [6]. These tools are largely dependent on heuristics, and they fail on queries which do not have very similar sequences deposited in the database. BLAST gives approximately accurate results and its

speed is therefore achieved at the cost of some degree of precision [15]. Also alignment based methods suffer from the drawback of the increasing computational complexity with the increase in the number of the sequences as well as the size of the sequences. Therefore the research into alignment free sequence analysis is emerging at a greater speed than before. Also this is necessary to overcome critical limitations of sequence analysis by alignment. In [1], it has been mentioned that although the pace of work in this area is increasing sharply, the total number of published reports proposing or using alignment-free metrics is relatively small, still under the one hundred mark.

More details and information about the alignment based sequence analysis can be found from [8]. Moving further, we mention the current research on the alignment free sequence analysis and present our algorithm for the same. The review of various methods used for alignment free sequence analysis appears in [1]. One of the approaches makes use of hidden Markov models. The method is based on the concept of comparing the similarity/dissimilarity between two constructed Markov models. The distances between DNA sequences are calculated without prior alignment. The similarity between two sequences is calculated by computing the log-likelihood difference between the two Markov models with the same observation sequence [2].

A program, CD-hit [3] was developed that forms clusters of protein sequences and efficiently handles large databases. This algorithm works by finding the minimum number of identical short substrings, called 'words', such as dipeptides, tripeptides, and so on, shared by two proteins. Sequence similarity is a function of this number [3].

Another method that falls under the alignment free sequence analysis is based on comparing the frequencies of all fixed-length words in the two sequences. A sequence similarity score, the D2z score, is used to detect functional and/or evolutionary similarities among regulatory sequences, in an alignment-free manner. The score is 'normalized', under the commonly used assumption of sequences being generated by Markov chains, and can hence be compared across different sequence pairs [4].

As demonstrated in this document, the numbering for sections upper case Arabic numerals, then upper case Arabic numerals, separated by periods. Initial paragraphs after the section title are not indented. Only the initial, introductory paragraph has a drop cap.

4. Materials and Methods

4.1 Database

We have downloaded the UniRef100 database of protein sequences from uniprot.org [13] in the FASTA format. We have preprocessed the database and extracted all the protein sequences to suit the application. The preprocessed database consists of 9,757,328 protein sequences and the total size is 3.21 GB. This database is used for testing the SS algorithm proposed here. We have observed that the sequences are clustered but some members of the cluster are located far away from the clusters. We have used this feature of the UniRef100 database in the fasta format to test our algorithm. Our algorithm is able to locate the remotely located cluster members in the database.

4.2 Algorithm – Pre-Search

We describe the pre-search algorithm that operates on the query sequence and the database containing 9,757,328 million sequences. The resultant output is a short list of sequences that resemble the query sequence. The algorithm is described in Fig. 2.

ALGORITHM: PRE-SEARCH

Input: The query sequence, the first similar sequence, and the sequence database

Output: A list of sequences for input to the Similarity Search algorithm

1. Extract all common words between the query sequence and the first similar sequence
2. Randomly choose 3 words from the common words list.
3. Check for the presence of either of the words in all the sequences in the database
    a. If word present, save the sequence
       Else ignore the sequence

Figure 2. Algorithm – Pre-Search which reduces the search space

We explain the working of the algorithm with the following example:

Suppose that we want to find similar sequences to the query sequence A = "IYSLAHYHSFNSLIFYEPVEIIGYDNKSSLVLVKRLITRMYQQKSLISSVNDSNQNES". We run Pre-Search algorithm and find 10 substrings of this sequence. Let the set of such substrings be S = {IYSLAHYHSFN, SFNSLIFYEPVEI, VEIIGYDNKSSL, LISSVNDSNQN…}. 3 words from this list are randomly chosen and all the sequences that contain two of the three strings in S are listed. This list is much smaller in size as compared to the main database. The SS algorithm is executed on this list.

We explain our notion of selecting 3 words, in Pre-Search Algorithm (Fig. 2). Probability of a word of length say 15, occurring in a database of 3 Giga characters is $1/(11*10^9)$. Therefore, ideally one word is sufficient for searching for near similar sequences. However, we are dealing with biological databases. Assuming a 0.1% mutation, it is possible that this 15-character word is mutated at some location. In such a case, we will not be able to retrieve the sequence containing the 15 character mutated word. But, if we select, two words, the probability that both the words are mutated in the same sequence is far less. And moving further, if we consider 3 words, probability that all the three words are mutated together in the same sequence is very rare.

4.3 Algorithm: Similarity Search (SS)

The SS algorithm is described in Fig. 3. The sequences obtained from Pre-Search are given as input, one at a time, to the SS Algorithm. The output of the SS algorithm is a list of sequences that resemble to the query sequence and have similarity score above a certain threshold.

Algorithm: Similarity Search (SS)

Input: A sequence 'A', which is the query sequence and a sequence 'B' obtained from Pre-Search

Output: Similarity score for short-listed sequences

Description: Sequence A slides over sequence B, one character each pass. In each pass, characters are matched from left to right and a score Sn is assigned for each pass. This process is repeated for each B obtained in Pre-Search

Algorithm:
Score Sn=0; k=1;

1. Compare matching pairs of A and B from left to right.
    a. Add k to Sn for each mathc.
    b. Double k if previous pair is matched or reset it to 1, otherwise
    c. Limit the value of k to 10,000
    d. Normalize Sn for the overlapping length of A and B
2. Repeat step 1 till the last character of A is matched with last character of B.
3. Get the maximum value of Sn as the final similarity score.
4. Based on a suitable threshold, such as 150, either accept the sequence as similar, or reject otherwise.

Figure 3. Algorithm – Similarity Search (SS)

We explain the working of the above algorithm with an example.

If A= SLVLVKRLITRMYQQK, and B= IIGYDNKSSLVLVKRLITR. The algorithm starts with comparing S with N, then KS with NL, then LKS with NLK and so on. The maximum score will be obtained when SLVLVKRLITR is compared with SLVLVKRLITR. Continuing with the procedure, SL will be compared with TR, and lastly, S will be compared with R.

4.4 Scoring Method

The scoring method to assign similarity scores to the sequences is simple yet effective. The scoring method is explained in Fig. 3.

4.5 Algorithm Analysis

Randomized algorithms are the ones that involve some sort of randomness in the algorithm. This also means that the output is dependent on the value of the random variables. Hence, for the same input, randomized algorithms may produce different output on independent executions. Given a computational problem, it may be difficult to formulate a deterministic algorithm with good running time [11], [16]. The solution to this problem is the use of efficient heuristics or opting for randomized or approximation algorithms. The problem of sequence similarity search in a database of over 9 million sequences, each of variable length from 50 to 5000 amino acids, is expected to take considerable time with a

deterministic algorithm. Randomized algorithms are preferred over the deterministic ones because they are simple, fast and provide optimum output with a very high probability. Executing the algorithm several times may increase the probability of achieving correct result. It is also known that analysis of running time or probability of getting a correct answer is usually difficult with randomized algorithms. However, we have experimentally verified the correctness of the algorithms.

Pre – search requires computing 10 substrings of random and different lengths from the query sequence. This process is trivial and requires constant time. Now enlisting the sequences in the database containing any of the 3 randomly chosen substrings, requires 9 million instructions of the type, if sequence A *contains* substring B, then keep A. This is observed to be fast.

In Similarity search, we are comparing two sequences by advancing one sequence, character by character, over the other. So given the sequences of length m and n, the number of comparisons involved is equal to the number of overlapping characters at a particular time. This number starts from 1, increases to the min (m, n) and then reduces to 1, which is proportional to n^2, if $n < m$. This can be considered computationally expensive in terms of the size of the sequence and the total number of sequences in the search space. However, we run the SS algorithm on those sequences produced as output from the Pre-Search algorithm. Hence, this overhead is on a very small subset of the entire 9 million sequences and for the sake of accuracy, is acceptable.

5. Results and Discussion

We have tested our algorithm by deriving the query sequence form the database itself and noting the location of the query sequence. The program was allowed to run for several hours, as a result of which, we have accumulated similar sequences of more than 400 query sequences. This process could be continued for extensive testing. However, keeping the space constraints in mind, we present here the results of 5 query sequences in Fig. 4. The sequences are numbered 1, 2, 3, 4 and 5. Their lengths are 103, 112, 386, 123 and 370 amino acid residues respectively. Since the algorithm involves randomness, we provide four different execution instances for each query sequence.

Results of our algorithm suggest that similarity search using Pre-search can be performed an order of magnitude faster without significant loss in the accuracy of the search results.

We have randomly chosen 3 common substrings. We check for the presence of the either of them in the sequence database. This number has experimentally proved to suffice our requirement of retrieving the sequences for further refinement. We have also justified the use of 3 common substrings. However, this heuristic can be further improved upon for higher level of accuracy. Also, in the first phase, we are looking for a similar sequence in the neighborhood. There is a need for clarification here regarding two points. 1) How to obtain the neighborhood information of the query sequence and 2) What if the neighborhood information is not available or the query sequence is a new sequence, not existing in the database. The first question can be answered as follows. UniRef100 FASTA database has been very useful in development of our similarity search algorithms. We have observed that the database is organized in a manner so as the similar appearing sequences are clustered around. So, in order to achieve speed, we opted to search the first similar sequence around the query sequence location. However, we have discovered that there are some members of the cluster which are located far away from the cluster. Our algorithms are able to detect these members of the cluster. The answer to the second question is rather non-trivial. We have listed all 15-character words and their frequencies in the database. We have used this

information to guide the similarity search in case the query sequence is a new sequence, not existing in the database. However, the detail description of this new algorithm is out of scope of this paper.

The fundamental underlying principle for sequence comparison is that similar sequences will share word composition to some extent, which is then assessed by various techniques [1]. The Pre-Search algorithm proposed in the paper is length dependent in the sense that the sequences are checked to contain substrings of small fixed length words. This could be thought of as a weak departure from the idea of alignment since sharing k-words is equivalent to recognizing an alignment between identical segments [1]. We still feel that the algorithm proposed here falls under the category of alignment free sequence comparison, as there is no attempt to do either pair wise or multiple sequence alignment. We have discussed our method to rate the sequences retrieved.

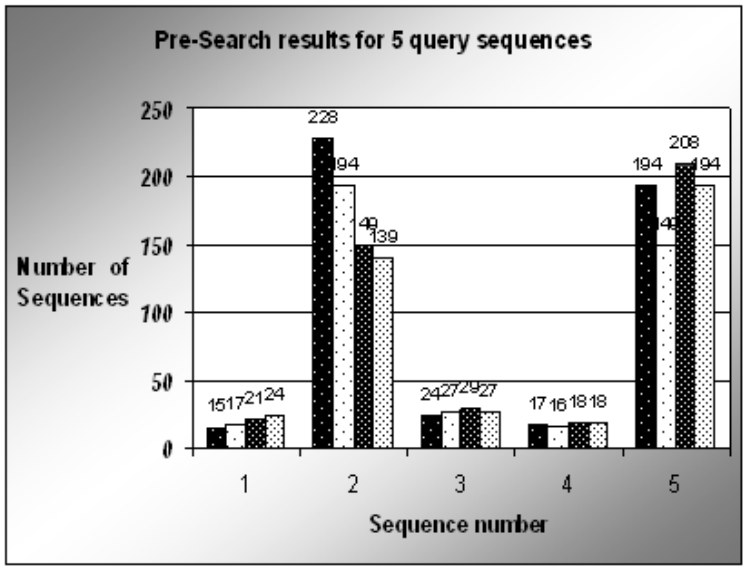

Figure. 4. Pre-Search results for 5 query sequences and 4 execution instances for each.

6. Conclusion

We have proposed a randomized algorithm that rapidly retrieves the relevant sequences based on the words in the query sequence. The pre-search algorithm involves randomness, produces a short list of sequences for input to the SS algorithm, and is very fast. Making multiple runs of the Pre-Search algorithm ensures that it covers all the possible sequences that can be input to the SS algorithm. Only these relevant sequences are considered for execution in the SS algorithm. This greatly reduces the time complexity of retrieving similar sequences. The SS algorithm is a deterministic one, and highly accurate in judging whether a given sequence is similar to the query sequence or not.

We are currently working on enhancing the proposed randomized Pre-Search algorithm using 15-character words and their frequencies in the database. These 15 amino acid residue substrings are used to fully index the UniRef100 database.

References

[1] Susana Vinga and Jonas Almeida, Alignment-free sequence comparison—a review, Vol. 19 no. 4 2003, pages 513–523, DOI:10.1093/bioinformatics/btg005

[2] Tuan D. Pham , and Johannes Zuegg, A probabilistic measure for alignment-free sequence comparison, Bioinformatics Advance Access published on December 12, 2004, DOI 10.1093/bioinformatics/bth426, Bioinformatics 20: 3455-3461.

[3] Weizhong Li, and Adam Godzik, Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences, Bioinformatics Advance Access published on July 1, 2006, DOI 10.1093/bioinformatics/btl158, Bioinformatics 22: 1658-1659.

[4] Miriam R. Kantorovitz , Gene E. Robinson , and Saurabh Sinha, A statistical method for alignment-free comparison of regulatory sequences, Bioinformatics 23: Vol. 23 ISMB/ECCB 2007, pages i249–i255

[5] Europe's leading laboratory for basic research in molecular biology, ww.ebi.ac.uk/embl

[6] Altschul, S. F. et al. (1990) Basic Local Alignment Search Tool. J. Mol. Biol., 215, 403–410.

[7] C. Setubal and J. Meidanis, "Introduction to Computational Molecular Biology", Cengage Learning, 1997

[8] Michael S. Waterman, Applications of Combinatorics to Molecular Biology, Handbook of Combinatorics, 1995 Elsevier Science B.V.

[9] Canillo, H., and D. Lipman, The multiple sequence alignment problem in biology, SIAMJ Appl. Math, 1981. 48, 1073-1082.

[10] William R. Pearson and David J. Lipman, Improved tools for biological sequence comparison, Vol. 85, pp. 2444-2448, April 1988, Biochemistry

[11] Cormen, Thomas H., Charles E. Leiserson, and Ronald L. Rivest, Introduction to Algorithms, 2nd ed., Prentice-Hall of India, 2004.

[12] The PDB archive containing information about experimentally-determined structures of proteins, nucleic acids, and complex assemblies, www.pdb.org

[13] Database in fasta format- Uniref100.fasta, downloaded on 31st Jan 2010 from www.uniprot.org

[14] The National Center for Biotechnology Information provides access to biomedical and genomic information. http://www.ncbi.nlm.nih.gov/

[15] Clare Sansom. Database searching with DNA and protein sequences: An introduction. Briefings in Bioinformatics (2000) Vol.1, No.1 (22-32).

[16] Brassard Gilles and Paul Bratley, Fundamentals of Algorithmmics, Prentice-Hall of India, 2004

[17] http://www.genomeweb.com

[18] Canillo, H., and D. Lipman, The multiple sequence alignment problem in biology, SIAMJ Appl. Math, 1981. 48, 1073-1082.

[19] Setubal and J. Meidanis, "Introduction to Computational Molecular Biology", Cengage Learning, 1997

[20] J Cheng, A Tegge and P Baldi, "Machine learning methods for protein structure prediction", IEEE Reviews in Biomedical Engineering, Vol I, 2008.

Authors

Himanshu S. Mazumdar acquired his Bachelor's degree in Engineering in the year 1968 and PhD in Computer Engineering in the year 2004. He has over 30 years of extensive experience in Airborne and Ground based instrumentation in Rocket, Satellite, Space Shuttle, Radio Telescope, Infrared Telescope, Industrial Automation, Process Control, Robotics, Communication, Signal and Image processing projects. Long design experience of Analog, Digital, Embedded Controller, VC++, Power Electronics and PC based Systems. He has worked in important space missions in University College London, NASA's Space Shuttle and Indian Space Research Organization. He has worked with Physical Research Laboratory, Ahmedabad, India in the capacity of the Head, Electronics Division and Computer Science Laboratory during 1968 – 1997. He joined as Professor and honorary Head of R&D Center of Dharmsinh Desai Institute of Technology, Nadiad, India. He has worked as Director, Research and Development, Defense Training & Technologies Champaign, IL, USA.

Presently, he is working as Professor and Head of R&D Center of Dharmsinh Desai Institute of Technology, Nadiad, India. His current research interests include Computational Biology and Intelligent Imaging.

Maulika S. Patel received her Bachelor and Master degrees in Computer Engineering in 1997 and 2004 respectively. She is currently a PhD student in the Research and Development Center, Dharmsinh Desai University, Nadiad, India. Her research interests include development of algorithms in Computational Biology. She is working as Head of the Department in Computer Engineering at G H Patel College of Engineering & Technology, Vallabh Vidyanagar, India.