12-5-2006

Comparison of Methods Used for Aligning Protein Sequences

Sangeetha Madangopal

COMPARISON OF METHODS USED FOR ALIGNING PROTEIN SEQUENCES

by
SANGEETHA MADANGOPAL

Under the Direction of Saeid Belkasim and Robert Harrison

ABSTRACT

Comparing protein sequences is an essential procedure that has many applications in the field of bioinformatics. The recent advances in computational capabilities and algorithm design, simplified the comparison procedure of protein sequences from several databases. Various algorithms have emerged using state of the art approaches to match protein sequences based on structural and functional properties of the amino acids. The matching involves structural alignment, and this alignment may be global; comprising of the whole length of the protein, or local; comprising of the sub-sequences of the proteins.

Families of related proteins are found by clustering sequence alignments. The frequency distributions of the amino acids within these different clusters define the sequence profile. The best alignment algorithm uses these profiles. In this thesis, we have studied different profile alignment algorithms where the cost function for comparing two profiles is changed. These are compared to the FFAS3 (Fold and Function Assignment) algorithm.

INDEX WORDS: Protein sequence alignment, matching-algorithms, properties of amino acids, FFAS3 algorithm

COMPARISON OF METHODS USED FOR ALIGNING PROTEIN SEQUENCES

by

SANGEETHA MADANGOPAL

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Science

in the College of Arts and Sciences

Georgia State University

2006

COMPARISON OF METHODS USED FOR ALIGNING

PROTEIN SEQUENCES

by

SANGEETHA MADANGOPAL

Major Professors: Saeid Belkasim
Robert Harrison
Committee: Raj Sunderraman
A.P. Preethy

Electronic Version Approved:

Office of Graduate Studies
College of Arts and Sciences
Georgia State University
December 2006

Acknowledgements

I take this moment to express my hearty thanks to all those who were a part in bringing my work to a successful completion.

I would like to thank my advisors Dr. Saeid Belkasim, and Dr. Robert Harrison for their immense interest in guiding me.

I feel happy and proud to have worked under Dr. Robert Harrison. He never hesitated to explain me, any number of times, things that I did not understand. His constant encouragement and motivation, made me work with full dedication. He taught me things not only related to the subject, but also about ethics in the professional world.

Dr. Saeid Belkasim, was always kind to me. He helped me to get started of with my work. He took all efforts to throw light on various projects that were undertaken by our thesis group, by conducting thesis meetings. This helped me think of various ways that I could do my thesis work.

Dr. Rajshekar Sunderraman, and Dr. Preethy made my committee a perfect one.

I will never forget to thank my parents and my brother, for their everlasting love and care for me. I wouldn't have earned anything in life without them.

Last, but not the least, I would like to extend my sincere thanks to my beloved friends; Harsha Vardhan, Shanthi, Ruchi, Sugandhi, Nofiya, Neelima, and Pranay Kumar, for their moral support.

Dedicated to my advisors and my parents!!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Proteins are large organic compounds called polyamides arranged in a linear chain and joined by peptide bonds [1-8]. The sequence of an amino acid in a protein is defined by its gene and is encoded in the genetic code. Proteins have structure, and can work together to achieve a particular function [9] [11-19].

A huge number of proteins exist in our life supporting system. Their role is vital for every living organism. Biologists have discovered a number of proteins and classified them according to their functions and structures. Today, we have a number of databases that hold information about proteins. Yet, we have a number of proteins that are still in the process of being discovered [57-59].

Sequence alignment is a way of identifying new proteins by comparing it with the existing ones to find similarities in structure and function. It helps in classifying the new proteins based on the pre-existing knowledge of the known ones. Not only this, applications where the biologists need to align the sequences are many; some of them are solving protein structures and comparing sequences to deduce the functions of proteins, to find ancestral connections, understanding enzyme mechanisms in a cell, analyzing structural receptors and molecules involved in the process of cell signaling, in identifying molecular surfaces of protein-protein and protein-DNA interactions, in protein engineering – a branch of study that deals with artificial synthesis of proteins for commercial purposes, an example is manufacture of insulin, in clustering of families and super families for the purpose of classification of proteins , and in other studies like metabolic computing and comparative genome analysis [10][22-35][48-52].

There are a number of methods available to do the sequence alignment of proteins. These are usually done using dynamic programming, heuristic approaches or probabilistic methods. With the advancement in computation and algorithm design, search for a better method with greater accuracy is needed to help the biologists in their studies [33-46].

So, in this thesis we have developed a strategy for comparison of different standard alignment methods to find out the best among them. Finding the best one can make a biologist's work accurate. The dynamic algorithms considered in this thesis are Maximum Entropy Kernel, FFAS, Central Limit measure, and Information Measure algorithms. These algorithms basically work on the dot matrix plot of the two input sequences. The task of the dynamic programming is to obtain maximum information about the alignment sequence from the dot matrix plot. This information is returned in the form of a score value.

The results or score values of these methods are compared against the FATCAT [68-74] results. By comparing the results, we come to know the effect of scoring function of algorithms on the accuracy of the alignment sequence. Scoring function is nothing but the main function that decides on how the dynamic program interprets the dot matrix plot.

Therefore, analyzing the scoring function results in finding the optimal algorithm for sequence alignment.

Chapter 2

BACKGROUND

In this chapter, we shall see a brief description of some of the concepts that are necessary in understanding the procedures involved in this thesis work.

2.1 Properties of proteins and its evolution

Studies show that if genes are similar by evolution, then their sequences are similar, also if proteins are related by function, then their sequences are similar.

Traditionally, protein evolution has been studied by computing the similarity of sequences within groups of homologous proteins. Homologous proteins are those that have same ancestors. It is believed that the proteins evolved to have different structures because of the point mutations; which are changes in the sequences at a single point, or gaps which are due to insertions or deletions at that point of the sequence. Studies show that proteins usually follow a predefined way of evolution in order to preserve its conservativeness. This can be because of the physiochemical properties of the bonding nature of amino acids [52][54-58].

Proteins are said to have "active sites" where are responsible for interacting with other proteins [53]. The chemical properties of these sites is highly sensitive, meaning that in order for two proteins to interact, their active sites have to correlate very closely in their chemical properties with one another [59].

As was stated earlier, proteins try to preserve their conservativeness. This means that evolutionary changes occur in protein sequences only when they are subject to substitutions between amino acids with similar properties resulting in changes that are

less likely to affect the overall structure and function of the protein. For instance, consider hydrophobic amino acids of similar sizes. They tend to substitute for each other quite well because they occupy positions within the hydrophobic core of the protein where tight packing and hydrophobic nature of the amino acids strongly affect the stability of the protein structure [12].

Protein sequences from within the same evolutionary family usually show substitutions between amino acids with similar physiochemical properties.

Some of the physiochemical properties of the amino acids are given below:

Hydrophobic: A, G, P, I, L, V, C, M, W, F

Polar: S, T, N, Q, Y

Aromatic: W, F, Y, H

Basic: H, K, R

Acidic: D, E

The list of the amino acids found in a protein sequence and its abbreviation are given in the table below for reference:

alanine A	histidine H
arginine R	isoleucine I
asparagine N	leucine L
aspartic acid D	lysine K
asparagine or aspartic acid* B	methionine M
cysteine C	phenylalanine F
glutamic acid E	serine S
glutamine Q	threonine T
glutamine or glutamic acid* Z	tryptophan W
glycine G	tyrosine Y
proline P	valine V

Table 1: Amino acids and their abbreviations

It can be observed that some amino acids have dual nature. For example, ‘Y’ is both polar and aromatic.

Finding the close match between two sequences not only involves finding the exact match of sequences, but also the ones that strongly co-relate to each other based on their physiochemical properties.

These physiochemical properties determine the score matrix which is a substitution matrix used to assign scores to the protein sequence alignment. The first row and the first column of this matrix is the sequence list of the 20 amino acids. Its entries in the matrix are the score values that should be used in calculation of protein alignment. The entries have score values according to their physiochemical properties. For exact match, they have a higher score, and for every miss match they have a small or negative score assigned. Gaps have negative value.

A brief description of the protein databases and search tools that helps in understanding this thesis is presented below:

2.2 A brief description of - 'BLOSUM', a protein database:

BLOSUM stands for BLOck Substitution Matrices. BLOSUM matrices originated from paper Henikoff and Henikoff [13]. BLOSUM matrices are generally used to find alignment scores between evolutionarily related protein sequences.

The main purpose o f its origin is to obtain a better performance measure to find the difference between two protein sequences that are distantly related proteins.

It uses the BLOCKS [52] database to search for differences among sequences in conserved regions within a protein family. BLOCKS are defined to be multiple aligned ungapped segments that correspond to the most highly conserved regions of proteins.

BLOSUM first collects all the sequences from the BLOCKS database and then for each of it, it finds the sums of the number of amino acids in each site to find their frequencies that indicates how often different pairs of amino acids are found together.

Different levels of the BLOSUM matrices are obtained by weighting the degree of similarity between sequences, differentially. The contributions of multiple entries of closely related sequences are reduced based on the similarities of the protein sequences. For example, consider BLOSUM62 matrix, which is calculated from protein blocks that have sequences of more than 62% similarity, the contribution of these sequences is weighted to sum to one.

As a matter of fact, BLOSUM and other matrices are constant (stationary) and cannot emphasize features that are specific to a given protein family.

2.3 A brief description of - 'FASTA', a protein matching tool:

FASTA is a well known Protein sequence alignment software tool that was first described (as FASTP) by David J. Lipman and William R. Pearson [15].

FASTA stands for "FAST Alignment" .It takes as its input an amino-acid sequence and searches for the corresponding sequences in its protein database by using a local sequence alignment method to find its corresponding matches.

The FASTA program is said to follow a heuristic approach which results in the increase of performance speed.

FASTA works as follows: It first, observes the pattern for word hits; which is defined as word to word matches of a given length of protein sequence, and then marks the matches observed before performing a more time consuming optimized search using a local alignment algorithm called Smith-Watermann [16] algorithm. The size taken for a word is considered to be the parameter called *'ktup'*, which controls the overall performance and speed of the program. It is observed that, increasing the *'ktup'* value decreases the number of background hits that are found. The program looks for segments that contain a cluster of closely matching hits by the word hits that are returned from the previous stage. It then searches these segments for a possible match.

The FASTA tool takes its input in a text format known as FASTA format. An example of FASTA format is as shown below [17]:

```
 >gi|5524211|gb|AAD44166.1| cytochrome b [Elephas maximus maximus]
LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATAFMGYVLPWGQMSFWGATVITNLFSAIPYIGTNLV
EWIWGGFSVDKATLNRFFAFHFILPFTMVALAGVHLTFLHETGSNNPLGLTSDSDKIPFHPYYTIKDFLG
LLILILLLLLLALLSPDMLGDPDNHMPADPLNTPLHIKPEWYFLFAYAILRSVPNKLGGVLALFLSIVIL
GLMPFLHTSKHRSMMLRPLSQALFWTLTMDLLTLTWIGSQPVEYPYTIIGQMASILYFSIILAFLPIAGX
IENY
```

Figure 1: An example showing FASTA format

2.4 A brief description of - 'PSI-BLAST', a protein matching tool:

PSI-BLAST is a protein matching tool which given a protein sequence, finds a family or cluster of related proteins and from that a sequence profile. PSI-BLAST is a profile-profile matching tool, which means it looks for the profile values while aligning and not for the exact matches. The profile values are estimated from the BLAST by repeated estimation. The profile is nothing but a string of frequency or probability values.

BLAST stands for Basic Local Alignment Search Tool. It is one of the algorithms that is used for comparing biological sequences such as the amino-acids of different proteins or even the nucleotides of DNA sequences. Unlike the PSI-BLAST, the BLAST is a symbol by symbol match, which means the algorithm looks for exact matches of amino acids in the given two sequences.

The remaining part of this section explains about the BLAST search:

A '*BLAST search'* enables a biologist to pose a query to compare a given sequence with a database of known sequences. It also identifies sequences that resemble the query sequence given a certain criteria like sequence length, and so on [18].

For example, to find out the similarity of genes in humans and mouse, a biologist will run the query on BLAST for the new gene that he finds in the mouse. The results of the test will show the close matches of the protein sequence found in humans [19].

BLAST searches for high scoring sequence alignments between the query sequence and sequences in the database using a heuristic approach that approximates the Smith-Waterman algorithm [16]. The Smith-Waterman approach is too slow for searching large genomic databases such as GenBank [20]. Therefore, the BLAST algorithm uses a heuristic approach that is slightly less accurate than Smith-Waterman. The speed and its relative good accuracy of BLAST makes it the key factors of the BLAST programs and no wonder why it is one of the widely used tools in bioinformatics.

2.5 A brief description of - 'SCOP', a protein database:

SCOP stands for **S**tructural **C**lassification **O**f **P**roteins (SCOP), it is a database in which largely manual classification of proteins based on similarities in amino acid sequences can be observed [21].

It is known that: SCOP utilizes a hierarchical method to organize the classification of proteins, allowing a four character code to be assigned to any protein domain it comes across.

The four classification levels of SCOP, as given by its authors Alexey and John-Marc are as follows [60]:

1. class - a wide description of the structural content of proteins
2. fold – indicates a broad structural similarity but with no proof of a homologous relationship
3. super family – has sufficient structural similarity but no detectable sequence similarity to indicate a divergent evolutionary relationship
4. family – has sufficient sequence similarity which can be identified either directly or indirectly through a transitive search.
5. domains – are independent-folding units of compact structure

It is stated that the **SCOP** database which is created by manual inspection aims to provide a detailed and well-defined description of the structural and evolutionary relationships among all proteins whose structure is already known [21].

Chapter 3

DYNAMIC PROGRAMMING

In this chapter we will see how a general dynamic program is designed, and how it is used in case of sequence alignment.

3.1 General Dynamic Programming:

In computer science, dynamic programming is defined as a method that is used for reducing the runtime of the algorithms by using the properties of *overlapping sub problems, optimal substructure* and *memorization.*

Optimal substructure means using optimal solutions of sub problems to find the optimal solutions for the overall problem.

For example, the shortest path to a goal-vertex from a given node in an acyclic graph can be found by first finding the shortest path to the goal-vertex from all its adjacent nodes, and then using the results obtained we can trace back to pick the best overall path.

In general, a problem with optimal substructure can be solved by using a three-step process as shown below [61]:

1. Divide the problem into smaller sub problems.

2. Solve the sub problems recursively.

3. Use the optimal solutions obtained from sub problems to construct an optimal solution for the overall problem.

By recursively dividing the problem, we mean that sub problems are in turn solved by dividing them into sub problems, and so on, until we reach some simple case that is easy enough to solve.

Overlapping sub problems means that the same sub problems are used to solve many different larger problems in the future computations.

For example: Consider the problem of finding the Fibonacci series. The problem involves computation of the previous sub-problem, for example: F4=F3+F2, but F2 has been previously computed. So finding F2 can be treated as a sub-problem, whose results can be used to find F4.

Memorization is stated as the concept of storing the previously solved results of sub-problems to be used in future to compute the overall problem. This reduces re-calculating of the results, thus increasing the efficiency of the algorithm.

For example: Consider the same problem stated above for overlapping sub problems. We stated that re-computing F2 can be avoided if it has already been computed. This can be done by storing all the results of the sub-problems. This is what we mean by memorization.

Dynamic programming usually follows one of the following approaches:

- *Top-down approach:* The overall problem is broken into sub problems, and these sub problems are solved and their solutions are remembered in case they need to be used in future computations.

- *Bottom-up approach:* All sub problems that might be needed are solved in advance and then combined to build up solutions to larger problems. This method uses a lot of memory space, and also makes it hard to predict the sub problems in advance.

3.2 Dynamic Programming in case of sequence alignment:

The technique of dynamic programming can be used to produce global alignments and local alignments in protein sequences.

In general, protein alignments use a matrix called substitution matrix or score matrix to assign scores to amino-acid matches or mismatches, and it assigns a gap penalty which is usually a negative score for matching an amino acid in one sequence to a gap in the other. Gaps are inserted between the protein sequences so as to align the sequences with identical or similar characters in successive columns.

Gap penalties are used during sequence alignment to calculate the overall score of alignments [62]. The size of the gap penalty used relative to the entries in the similarity matrix or score matrix, affects the alignment that is finally selected. Selecting a higher gap penalty results in few gaps because it will cause less favorable characters to be aligned.

Methods to improve the accuracy of sequence alignment are focused on improving the scoring functions like the Dayhoff-type matrix [64].

Calculating score values:

Given two protein sequences of length N and M, respectively, a scoring matrix of dimensions N * M can be constructed. Each element X_{ij} of this scoring matrix is the score obtained by substituting residue '*i*' in the first sequence with residue '*j*' in the second sequence.

Substitution scores are said to be calculated from standard residue matrices like BLOSUM score matrix [13]. This scoring matrix can also be manually constructed by comparing the sequence profiles for every aligned position. In either case, our ultimate aim is to align the sequences to optimize the overall alignment score. This alignment score obtained is nothing but the sum of the scores corresponding to the matched residues subtracted from the penalties for occurrences of mis-matched residues. Here the term residues refer to the amino acids.

The maximum score for the alignment of two sequences is obtained by working forward along each sequence step by step and storing its corresponding score values in the form of a dot matrix plot. To obtain the alignment and the best score, we have to trace back the prominent alignment line which usually appears as the diagonal in the dot matrix plot.

CHAPTER 4

SEQUENCE ALIGNMENT ALGORITHMS

In this chapter, we will introduce some basic protein sequence alignment algorithms that were considered in this thesis.

1. FFAS3- Fold and Function Assignment System: a dot product method

This fold assignment method is based on the profile-profile matching algorithm. This algorithm is said to have two dimensional weighting schemes which takes into account the topology of the evolutionary tree of proteins in the family of homologous proteins.

This paragraph describes the general steps involved in finding the score in case of the FFAS: FFAS aligns two profiles using a standard local dynamic programming algorithm. The value of the comparison score between positions 'n' and 'm' from the two profiles is computed as a dot product of the n-th column of the first profile and the m-th column of the second profile. After assigning values to all positions, then the matrix is normalized. The optimal alignment is calculated using dynamic programming as described in section 3.2.

In the last step the raw alignment score obtained with dynamic programming is then translated into the final FFAS score by comparing it with that of the raw scores that were obtained before for pairs of unrelated proteins. This is the general approach of finding the score used by the FFAS server. The input data for FFAS server are amino-acid sequences in FASTA format [62]. If P_1 and P_2 are two profiles, then

Score of the FFAS3 method is: $P_1.P_2$, i.e, dot product of P_1 and P_2, where P_1 and P_2, are the two protein profiles.

2. **Maximum Entropy kernel**: a probability distribution measure

Shannon defines the principle of maximum entropy as a method for analyzing the available information in order to determine a unique probability distribution [55]. He defined a property of a probability distribution, $H(\boldsymbol{p}) = -\sum p_i \log p_i$, which he called entropy [54].

The principle of maximum entropy uses this measure of $H(\boldsymbol{p})$ to rank probability distributions. It states that the '*least biased'* distribution that encodes certain given information is that which maximizes the Shannon entropy $H(\boldsymbol{p})$ [56] while remaining consistent with the given information.

The score value of kernel method is given by,

$Score = P_1P_2 + 0.01/(1-(P_1 - P_2)^2)$, where P_1 and P_2 are the two protein profiles.

3**. Central Limit Algorithm**: a Z- score normalization method

A central limit theorem is defined as a set of weak-convergences resulting in probability theory. Scoring functions are usually normalized for the length of a protein chain because a longer protein chain results in a higher value produced by the scoring function.

The scoring function used is: $Score = e^{-Z}$ where 'z' is the normalization.

The central limit theorem states that sample means are normally distributed irrespective of the shape of the input population considered for large samples and for any sample size with normally distributed population [57], thus sample means can be analyzed by using Z scores $Z = (X - \mu)/\sigma$

4. Information Measure: a probability distribution measure

Information Measure is defined as "a system of measurement of information based on the probabilities of the information-bearing events" [65]. It is an optimal method.

Information measure can be understood as: Given a set of 'S' things and for each set of D measurements, to form a partition of the set of things, or, a partition of the D-dimensioned measurement space in which each thing may be represented by a point, such that the things within each subset, or region of measurement space, information measure is defined as a set contained in the S x D attribute measurements [66].

The score function for information measure is given by [69]:

$$Score = \ln\left(\frac{(P_1 \bullet P_2)^2}{(P_1 \bullet P_1) \bullet (P_2 \bullet P_2)}\right)$$

A note on Dot Matrix Plot:

This section describes the need to use dot matrix plot for identifying alignment and, finding score.

It is easy to visualize certain sequence features such as insertions, deletions and substitutions from a dot-matrix plot. So, in this section we shall see how a dot matrix is plotted from two sequences. All the algorithms mentioned above obtain their score values from the dot matrix plot.

The dot matrix plot is constructed using two sequence, one written along the top row and the other as the leftmost column in a two dimensional matrix. A dot is placed at places where the characters match. Some implementations vary the size or intensity of the dot

depending on the degree of similarity of the two matches. It is observed that the dot plots of very closely related sequences appear as a single diagonal line along the matrix. One can easily generate the equivalent picture of any sequence if its chain is given.

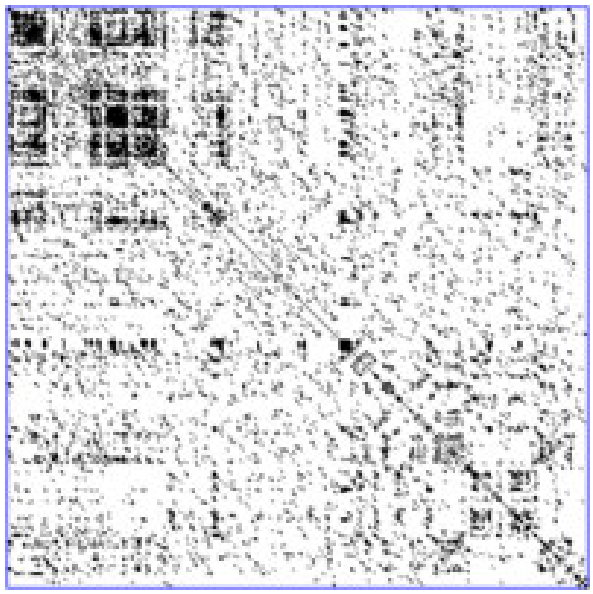

Figure 2[67]: A DNA dot plot of a human zinc finger transcription factor

Chapter 5

METHODOLOGY OF COMPARING ALGORITHMS

In this section, we will be seeing the various steps undertaken to compare the protein sequences that are obtained as a result of various algorithms mentioned above.

Step 1: Collection of protein names.

In order to test the accuracy of the algorithms, a random collection of input data is essential.

For this purpose, SCOP was used to collect various ranges of the input data. About 120 protein names that belonged to different families were obtained from SCOP.

It helped in making a wide range collection of data possible, by making us pick proteins that belonged to a variety of classes.

Criteria used to select protein names where: family name, sequence length > 300, and identity in a vast range, ranging from 15% to 90%.

Step 2: Obtain protein sequences.

After obtaining the names of the proteins from SCOP, the next task was to obtain their alignments from FATCAT.

FATCAT was chosen for this, because it is expected to have accuracy relatively better than the other known methods.

FATCAT is very easy to use; we just need to know the name of the protein that we need to compare. It then looks for all matches in the database and gives us all possible results.

```
Align d1ea5a_.pdb 532 with d1f6wa_.pdb 533
Twists 0 ini-len 448 ini-rmsd 2.35 opt-equ 510 opt-rmsd 3.02 chain-rmsd 2.35 Score 1093.59 align-len 552 gaps 42 (7.61%)
P-value 0.00e+00 Afp-num 90122 Identity 29.35% Similarity 42.75%

Chain 1:    1 SELLVNTKSGKVMGTRVPVLS--SHISAFLGIPFAEPPVGNMRFRRPEPKKPWSGVWNASTYPNNCQQYV
              1111111111111111111111  111111111111111   11111111111111111111111111111
Chain 2:    1 KLGAVYTEGGFVEGVNKKLGLLGDSVDIFKGIPFAAPT---KALENPQPHPGWQGTLKAKNFKKRCLQAT

Chain 1:   69 DEQFPGFSGSEMWNPNREMSEDCLYLNIWVPSP---RPKSTTVMVWIYGGGFYSGSSTL------DVYNG
              111           1111111111111111111   11111111111111111111111      11111
Chain 2:   68 ITQ-----------DSTYGDEDCLYLNIWVPQGRKQVSRDLPVMIWIYGGAFLMGSGHGANFLNNYLYDG

Chain 1:  130 KYLAYTEEVVLVSLSYRVGAFGFLALHGSQEAPGNVGLLDQRMALQWVHDNIQFFGGDPKTVTIFGESAG
              11111111111111111111111111111111 11111111111111111111111111111111111111
Chain 2:  127 EEIATRGNVIVVTFNYRVGPLGFLSTGDAN-LPGNYGLRDQHMAIAWVKRNIAAFGGDPDNITLFGESAG

Chain 1:  200 GASVGMHILSPGSRDLFRRAILQSGSPNCPWASVSVAEGRRRAVELGRNLNCNLNSDEELIHCLREKKPQ
              1111111111111111111111111111111111111  11111111111111111111111111111111
Chain 2:  196 GASVSLQTLSPYNKGLIRRAISQSGVALSPWVIQK--NPLFWAKKVAEKVGCPVGDAARMAQCLKVTDPR

Chain 1:  270 ELIDVEWNVLP--FDSIFR-FSFVPVIDGEFFPTSLESMLNSGNFKKTQILLGVNKDEGSFFLLYGAPGF
              111111111    111111 1111111111111111111  11111111111111111111111 1111
Chain 2:  264 ALTLAYKVPLAGLEYPMLHYVGFVPVIDGDFIPDDPINLY--ANAADIDYIAGTNNMDGHIFASI-DMPA

Chain 1:  337 SKDSESKISREDFMSGVKLSVPHANDLGLDAVTLQYTDWMD-DNNGIKNRDGLDDIVGDHNVICPLMHFV
              11111111111111111111111111111111111111111 11111111111111111111111111111
Chain 2:  331 INKGNKKVTEEDFYKLVSEFTITKGLRGAKTTFDVYTESWAQDPSQENKKKTVVDFETDVLFLVPTEIAL

Chain 1:  406 NKYTKFG--NGTYLYFFNHRASNLVWPEWMGVIHGYEIEFVFGLPLVKELNYTAEEEALSRRIMHYWATF
              1111111  1111111111111111111111111111111111111111111111111111111111111
Chain 2:  401 AQHRANAKSAKTYAYLFSHPSRMPVYPKWVGADHADDIQYVFGKPFATPTGYRPQDRTVSKAMIAYWTNF

Chain 1:  474 AKTGNPNEPH-SQESKWPLFTTKEQKFIDLNTEPM--KVHQRLRVQMCVFWNQFLPKLLNAT
              1111111111 11111111111111111111111  11111111111111111111111
Chain 2:  471 AKTGDPNMGDSAVPTHWEPYTTENSGYLEITKKMGSSSMKRSLRTNFLRYWTLTYLALPTVT
```

Figure 3[23]: A typical output from FATCAT for 1EA5A and 1F6WA proteins

Step 3: Obtain the results from the algorithms, and save it in separate folders.

This is done automatically with the help of a module that is written in Perl.

Step 4: Changing the format of the input files.

Designed a module that changes the format of the results obtained from FATCAT to discard the fields that are of no interest to us. The fields that were selected to be retained are the name of the proteins, their rmsd values, identity value and the two protein sequences. This module was written in Perl.

Step 5: Comparing the sequences in two files.

A module that reads in the two input files; one from FATCAT, and another from the test algorithms, that corresponds to the same pair of proteins is used to obtain the number of hits in the two pair of sequences. The number of hits corresponds to the number of times the sequence pairs found in both the files. The following section clearly explains how the comparison is done.

Comparison is done as follows:

- Read the first file from FATCAT and move its contents to an array. Manipulate the array to obtain only the sequence into another array that is easy to compare.
- Do the same thing to the second file which is obtained from the algorithms mentioned.
- Manipulate the sequences to make them start at the same sequence position.

Having the two arrays that start at the same position, and making them have same sequence length, makes it easy to compare the sequences and assign scores to them.

Now, there is a problem encountered while doing this. Comparing the sequences with just their starting sequences matching did not do what we expected. Though the problem seems to be trivial, it is not. This is because insertions made the task difficult. Whenever an insertion was encountered, the module missed to look for matching pairs in the subsequent columns, thus accounting for mismatch.

The output of this module is as shown below:

1luga-1znca.align

Identity from FATCAT:32.22

Extracted sequences from FATCAT:

```
WGYGK********HNGPEHWHKDFPIAKGERQSPVDIDTHTAKYDPSLKPLSVS*YDQATSLRILNNGH
WCYEVQAESSNYPCLVPVKW***GGNCQKDRQSPINIVTTKAKVDKKLGRFFFSGYDKKQTWTVQNNGH

FNVEFDDSQDKAVLKGGPLDGTYRLIQFHFHWGSLDGQGSEHTVDKKKYAAELHLVHWNTKYGDFGKAV
VMMLLEN***KASISGGGLPAPYQAKQLHLHWSDLPYKGSEHSLDGEHFAMEMHIVHEKEKGTSRNVKE

*QQPDGLAVLGIFLKVGS*AKPGLQKVVDVLDSIKTKGKSADFTNFDPRGLLPE***SLDYWTYPGSLT
QDPEDEIAVLAFLVEAGTQVNEGFQPLVEALSNIPKPEMSTTMAESSLLDLLPKEEKLRHYFRYLGSLT

PPLLECVTWIVLKEPISVSSEQVLKFRK*LNFNGEGEPEELMVDNWRPAQPLKNRQIKA
PTCDEKVVWTVFREPIQLHREQILAFSQKLYYDK**EQTVSMKDNVRPLQQLGQRTVIK
```

Concatenated into two arrays:

Arr[1]: First sequence moved to array[1][n]

```
WGYGK********HNGPEHWHKDFPIAKGERQSPVDIDTHTAKYDPSLKPLSVS*YDQATSLRILNNGHFNV
EFDDSQDKAVLKGGPLDGTYRLIQFHFHWGSLDGQGSEHTVDKKKYAAELHLVHWNTKYGDFGKAV*QQP

DGLAVLGIFLKVGS*AKPGLQKVVDVLDSIKTKGKSADFTNFDPRGLLPE***SLDYWTYPGSLTPPLLECV
TWIVLKEPISVSSEQVLKFRK*LNFNGEGEPEELMVDNWRPAQPLKNRQIKA
```

Arr[2]: Second sequence moved to array[2][n]

```
WCYEVQAESSNYPCLVPVKW***GGNCQKDRQSPINIVTTKAKVDKKLGRFFFSGYDKKQTWTVQNNGHVMM
LLEN***KASISGGGLPAPYQAKQLHLHWSDLPYKGSEHSLDGEHFAMEMHIVHEKEKGTSRNVKEQDPE

DEIAVLAFLVEAGTQVNEGFQPLVEALSNIPKPEMSTTMAESSLLDLLPKEEKLRHYFRYLGSLTPTCDEKV
VWTVFREPIQLHREQILAFSQKLYYDK**EQTVSMKDNVRPLQQLGQRTVIK
```

Score value from algorithm profile_SOM:

Score is: 4.16187

Extracted sequence from Maximum Entropy Kernel:

```
S*HHWGYGKHNGPEHWHKDFP*****IAKGERQSPVDIDTHTAKYDPSLKPLSVS*YDQATSLRILNNGHAF
NVEFDDS
AESHWCYEVQAESSNYPCLVPVKWGGNCQKDRQSPINIVTTKAKVDKKLGRFFFSGYDKKQTWTVQNNGHSV
MMLLE**


DKAVLKGGPLDGTYRLIQFHFHWGSLDGQGSEHTVDKKKYAAELHLVHWNTKY**GDFGKAVQQPDGLAVLG
IFLKVG*
NKASISGGGLPAPYQAKQLHLHWSDLPYKGSEHSLDGEHFAMEMHIVHEKEKGTSRNVKEAQDPEDEIAVLA
FLVEAGT


AKPGLQKVVDVLDSIKTKGKSADFTNFDPRGLLP*E*S*LDYWTYPGSLTTPPLLECVTWIVLKEPISVSSE
QVLKFRK
VNEGFQPLVEALSNIPKPEMSTTMAESSLLDLLPKEEKLRHYFRYLGSLTTPTCDEKVVWTVFREPIQLHRE
QILAFSQ

NFNGEGEPEELMVDNWRPAQPLKNRQIKA
LYY*DKEQTVSMKDNVRPLQQLGQRTVIK
```

Concatenated into two arrays:

Arr2[1]: First sequence moved to array2[1][n]

```
S*HHWGYGKHNGPEHWHKDFP*****IAKGERQSPVDIDTHTAKYDPSLKPLSVS*YDQATSLRILNNGHAF
NVEFDDSDKAVLKGGPLDGTYRLIQFHFHWGSLDGQGSEHTVDKKKYAAELHLVHWNTKY**GDFGKAVQ

QPDGLAVLGIFLKVG*AKPGLQKVVDVLDSIKTKGKSADFTNFDPRGLLP*E*S*LDYWTYPGSLTTPPLLE
CVTWIVLKEPISVSSEQVLKFRKNFNGEGEPEELMVDNWRPAQPLKNRQIKA
```

Arr2[2]: Second sequence moved to array2[2][n]

```
AESHWCYEVQAESSNYPCLVPVKWGGNCQKDRQSPINIVTTKAKVDKKLGRFFFSGYDKKQTWTVQNNGHSV
MMLLE**NKASISGGGLPAPYQAKQLHLHWSDLPYKGSEHSLDGEHFAMEMHIVHEKEKGTSRNVKEAQD

PEDEIAVLAFLVEAGTVNEGFQPLVEALSNIPKPEMSTTMAESSLLDLLPKEEKLRHYFRYLGSLTTPTCDE
KVVWTVFREPIQLHREQILAFSQLYY*DKEQTVSMKDNVRPLQQLGQRTVIK
```

Sequences obtained after matching the starting positions:

Sequence 1 from FATCAT:

→**WGYGKHN**GPEHWHKDFP*****IAKGERQSPVDIDTHTAKYDPSLKPLSVS*YDQATSLRILNNGHAFNV
EFDDSDKAVLKGGPLDGTYRLIQFHFHWGSLDGQGSEHTVDKKKYAAELHLVHWNTKY**GDFGKAVQQPDG
LAVLGIFLKVG*AKPGLQKVVDVLDSIKTKGKSADFTNFDPRGLLP*E*S*LDYWTYPGSLTTPPLLECVTW
IVLKEPISVSSEQVLKFRKNFNGEGEPEELMVDNWRPAQPLKNRQIKA

WCYEVQAESSNYPCLVPVKWGGNCQKDRQSPINIVTTKAKVDKKLGRFFFSGYDKKQTWTVQNNGHSVMMLL
E**NKASISGGGLPAPYQAKQLHLHWSDLPYKGSEHSLDGEHFAMEMHIVHEKEKGTSRNVKEAQDPEDEIA
VLAFLVEAGTVNEGFQPLVEALSNIPKPEMSTTMAESSLLDLLPKEEKLRHYFRYLGSLTTPTCDEKVVWTV
FREPIQLHREQILAFSQLYY*DKEQTVSMKDNVRPLQQLGQRTVIK

Sequence 2 from Maximum Entropy Kernel:

→**WGYGK*****HNGPEHWHKDFPIAKGERQSPVDIDTHTAKYDPSLKPLSVS*YDQATSLRILNNGHF
NVEFDDSQDKAVLKGGPLDGTYRLIQFHFHWGSLDGQGSEHTVDKKKYAAELHLVHWNTKYGDFGKAV*QQP
DGLAVLGIFLKVGS*AKPGLQKVVDVLDSIKTKGKSADFTNFDPRGLLPE***SLDYWTYPGSLTPPLLECV
TWIVLKEPISVSSEQVLKFRK*LNFNGEGEPEELMVDNWRPAQPLKNRQIKA

WCYEVQAESSNYPCLVPVKW***GGNCQKDRQSPINIVTTKAKVDKKLGRFFFSGYDKKQTWTVQNNGHVMM
LLEN***KASISGGGLPAPYQAKQLHLHWSDLPYKGSEHSLDGEHFAMEMHIVHEKEKGTSRNVKEQDPEDE
IAVLAFLVEAGTQVNEGFQPLVEALSNIPKPEMSTTMAESSLLDLLPKEEKLRHYFRYLGSLTPTCDEKVVW
TVFREPIQLHREQILAFSQKLYYDK**EQTVSMKDNVRPLQQLGQRTVIK

Number of matches, Score = 9,

Length of the sequences=262

Comparison found to be not trivial:

From the result obtained above, it can be noticed that, because of the insertions at different places the module fails to check for the subsequent matches in the following columns.

In other words, this module just looks for one-to-one match with respect to start position alone. This makes the method weak.

To overcome the above problem, we came up with a different way of comparing the sequences, a method that not only looks for matching start of the sequences, but also compares the sequences with respect to the original sequence which does not have insertions was considered.

The results obtained after modifying the comparison method is significantly good.

The result of the same pair of files is now found to be: **Score=220, Length = 262**

The first pair of sequences obtained after removing the insertions to obtain better results, is as shown below:

The positions are numbered with respect to the initial sequence.

W 1|W 1	I 29|I 34	N 57|N 63	R 85|Q 91	E 113|E 119
G 2|C 2	D 30|V 35	N 58|N 64	L 86|A 92	L 114|M 120
Y 3|Y 3	T 31|T 36	G 59|G 65	I 87|K 93	H 115|H 121
G 4|E 4	H 32|T 37	H 60|H 66	Q 88|Q 94	L 116|I 122
K 5|V 5	T 33|K 38	A 61|S 67	F 89|L 95	V 117|V 123
H 6|Q 6	A 34|A 39	F 62|V 68	H 90|H 96	H 118|H 124
N 7|A 7	K 35|K 40	N 63|M 69	F 91|L 97	W 119|E 125
G 8|E 8	Y 36|V 41	V 64|M 70	H 92|H 98	N 120|K 126
P 9|S 9	D 37|D 42	E 65|L 71	W 93|W 99	T 121|E 127
E 10|S 10	P 38|K 43	F 66|L 72	G 94|S 100	K 122|K 128
H 11|N 11	S 39|K 44	D 67|E 73	S 95|D 101	Y 123|G 129
W 12|Y 12	L 40|L 45	D 68|* 74	L 96|L 102	G 124|R 132
H 13|P 13	K 41|G 46	S 69|* 75	D 97|P 103	D 125|N 133
K 14|C 14	P 42|R 47	Q 70|* 76	G 98|Y 104	F 126|V 134
D 15|L 15	L 43|F 48	D 71|N 77	Q 99|K 105	G 127|K 135
F 16|V 16	S 44|F 49	K 72|K 78	G 100|G 106	K 128|E 136
P 17|P 17	V 45|F 50	A 73|A 79	S 101|S 107	A 129|A 137
I 18|N 23	S 46|S 51	V 74|S 80	E 102|E 108	V 130|Q 138
A 19|C 24	Y 47|Y 53	L 75|I 81	H 103|H 109	Q 131|D 139
K 20|Q 25	D 48|D 54	K 76|S 82	T 104|S 110	Q 132|P 140
G 21|K 26	Q 49|K 55	G 77|G 83	V 105|L 111	P 133|E 141
E 22|D 27	A 50|K 56	G 78|G 84	D 106|D 112	D 134|D 142
R 23|R 28	T 51|Q 57	P 79|G 85	K 107|G 113	G 135|E 143
Q 24|Q 29	S 52|T 58	L 80|L 86	K 108|E 114	L 136|I 144
S 25|S 30	L 53|W 59	D 81|P 87	K 109|H 115	A 137|A 145
P 26|P 31	R 54|T 60	G 82|A 88	Y 110|F 116	V 138|V 146
V 27|I 32	I 55|V 61	T 83|P 89	A 111|A 117	L 139|L 147
D 28|N 33	L 56|Q 62	Y 84|Y 90	A 112|M 118	G 140|A 148

Table 2: Shows the amino acids and their original positions of the protein sequence obtained after removing the gaps

I 141|F 149	A 169|T 178	P 197|T 209	N 225|L 237	A 253|K 265
F 142|L 150	D 170|T 179	L 198|C 210	F 226|Y 238	S 254|S 266
L 143|V 151	F 171|M 180	L 199|D 211	N 227|Y 239	
K 144|E 152	T 172|A 181	E 200|E 212	G 228|* 240	
V 145|A 153	N 173|E 182	C 201|K 213	E 229|D 241	
G 146|G 154	F 174|S 183	V 202|V 214	G 230|K 242	
S 147|Q 156	D 175|S 184	T 203|V 215	E 231|E 243	
A 148|V 157	P 176|L 185	W 204|W 216	P 232|Q 244	
K 149|N 158	R 177|L 186	I 205|T 217	E 233|T 245	
P 150|E 159	G 178|D 187	V 206|V 218	E 234|V 246	
G 151|G 160	L 179|L 188	L 207|F 219	L 235|S 247	
L 152|F 161	L 180|L 189	K 208|R 220	M 236|M 248	
Q 153|Q 162	P 181|P 190	E 209|E 221	V 237|K 249	
K 154|P 163	E 182|E 192	P 210|P 222	D 238|D 250	
V 155|L 164	S 183|K 194	I 211|I 223	N 239|N 251	
V 156|V 165	L 184|R 196	S 212|Q 224	W 240|V 252	
D 157|E 166	D 185|H 197	V 213|L 225	R 241|R 253	
V 158|A 167	Y 186|Y 198	S 214|H 226	P 242|P 254	
L 159|L 168	W 187|F 199	S 215|R 227	A 243|L 255	
D 160|S 169	T 188|R 200	E 216|E 228	Q 244|Q 256	
S 161|N 170	Y 189|Y 201	Q 217|Q 229	P 245|Q 257	
I 162|I 171	P 190|L 202	V 218|I 230	L 246|L 258	
K 163|P 172	G 191|G 203	L 219|L 231	K 247|G 259	
T 164|K 173	S 192|S 204	K 220|A 232	N 248|Q 260	
K 165|P 174	L 193|L 205	F 221|F 233	R 249|R 261	
G 166|E 175	T 194|T 206	R 222|S 234	Q 250|T 262	
K 167|M 176	T 195|T 207	K 223|Q 235	I 251|V 263	
S 168|S 177	P 196|P 208	L 224|K 236	K 252|I 264	

Table 2 [Continuation]

The above two tables show the first amino acid pairs from the two files, and their corresponding sequence positions. Their sequence position helps in comparing the amino acids to the original sequence with respect to their positions.

Summarization of the work done in this chapter:

1. Collect protein names from different families using SCOP
2. Find out the alignments of the proteins obtained in step 1 from FATCAT
3. Convert the file formats to obtain only desired fields
4. Run the tests on programs FFAS3, Maximum entropy kernel, Central Limit, and Information measure for the protein pairs obtained from SCOP.
5. Run the C++ program that reads in the two corresponding files(one from FATCAT and one from our programs), compares and returns the accuracy.
6. Tabulate the results for all the programs.
7. Plot the graph to find out the best method.

Chapter 6

EXPERIMENTAL RESULTS

The results obtained after comparing the protein sequences are given in this chapter. Also the pictures of profiles obtained for higher and low value of identity is given to visualize the performance of the algorithms.

The graphs plotted from the results are shown in this chapter for the various tests. The results are given for about 112 protein sequences. The detailed table is given in the appendix D.

The tables and figures are represented using some terms, knowing which would improve our understanding. The following legend explains the terms used in this chapter.

Legend:

Sequence pair: Is the name of the protein pair that is aligned.

Identity: Refers to the identity of the sequence obtained from FATCAT.

Score: Represents the score obtained by applying our algorithms.

Length: Is the length of the protein sequence that was used for comparison. In cases where the two lengths are different, the shorter of the two is the Length.

Num.of hits: Is the number of matching pairs found in the two set of sequences.

Accuracy: Is (Num.of hits/Length)*100.

6. 1 Diagrammatic View of the profiles for low and high identity values:

It can be observed from the following dot matrix profiles, that the ones with low identity values look fuzzier than the ones with higher identity values. A prominent diagonal shows the matching positions where the two sequences are matched. The score values are based on this diagonal.

Diagrammatic representation of profiles obtained by Maximum Entropy Kernel:

.

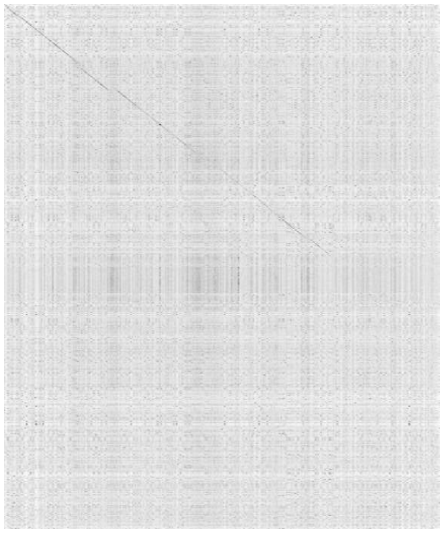

Figure 4: Maximum entropy kernel method, Sequence 1OOYA- 1POIA,
Identity value: 18.39(lowest in the table)

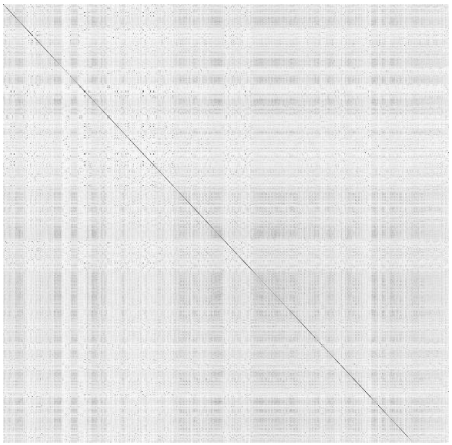

Figure 5: Maximum entropy kernel method, Sequence 1F6WA-1BCE0,
Identity value: 78.42(higher value)

Diagrammatic representation of profiles obtained by FFAS3 method:

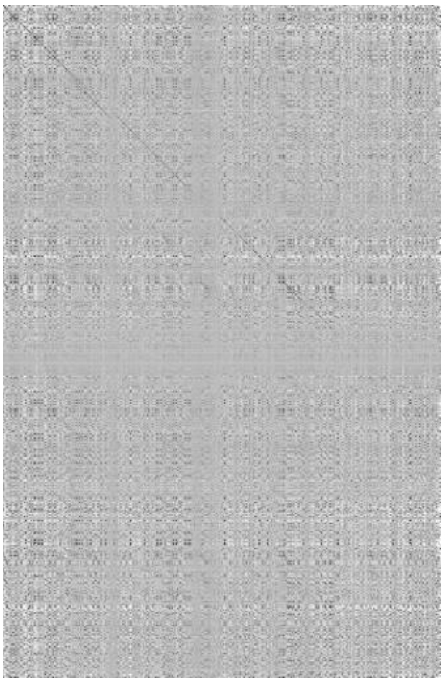

Figure 6: FFAS3 method, Sequence 1OOYA- 1POIA,
Identity value: 18.39 (lowest in the table)

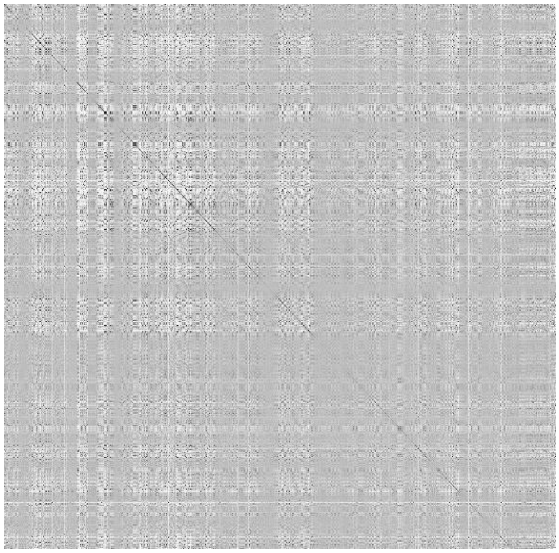

Figure 7: FFAS3 method, Sequence 1F6WA-1BCE0,
Identity value: 78.42 (higher value)

Diagrammatic representation of profiles obtained by Central Limit method:

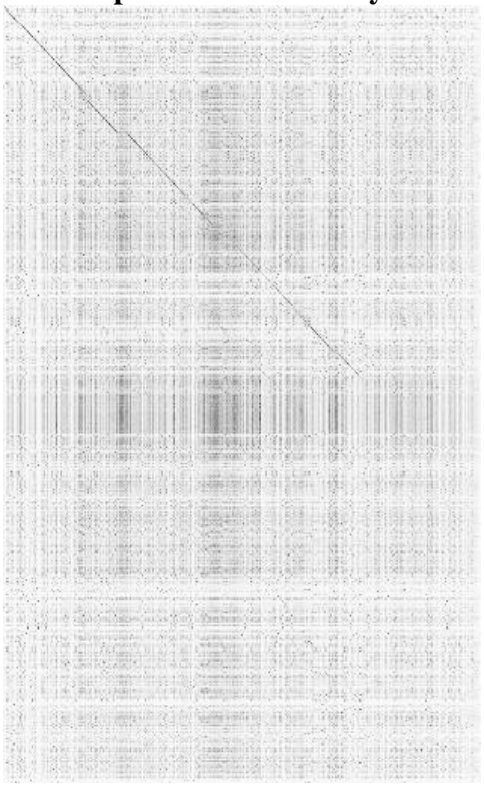

Figure 8: Central Limit method, Sequence 1OOYA- 1POIA,
Identity value: 18.39(lowest in the table)

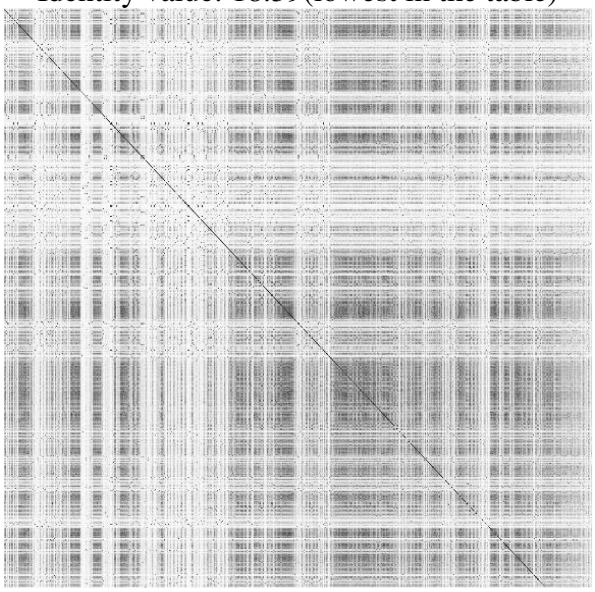

Figure 9: Central limit method, Sequence 1F6WA-1BCE0,
Identity value: 78.42 (higher value)

Diagrammatic representation of profiles obtained by Information Measure:

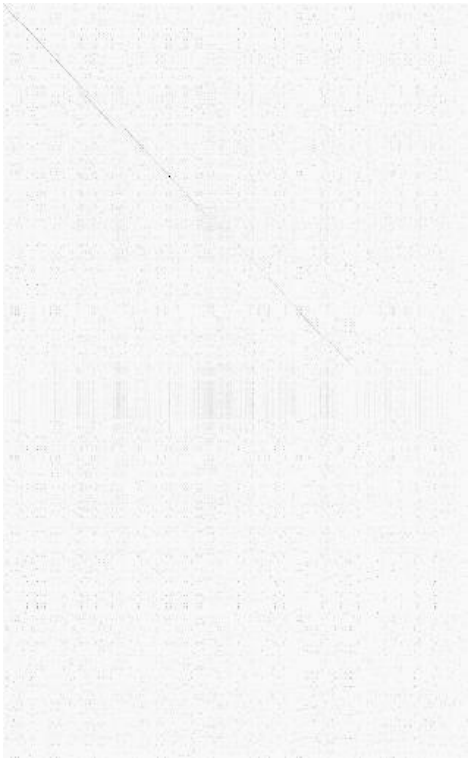

Figure 10: Information Measure, Sequence 1OOYA- 1POIA,
Identity value: 18.39(lowest in the table)

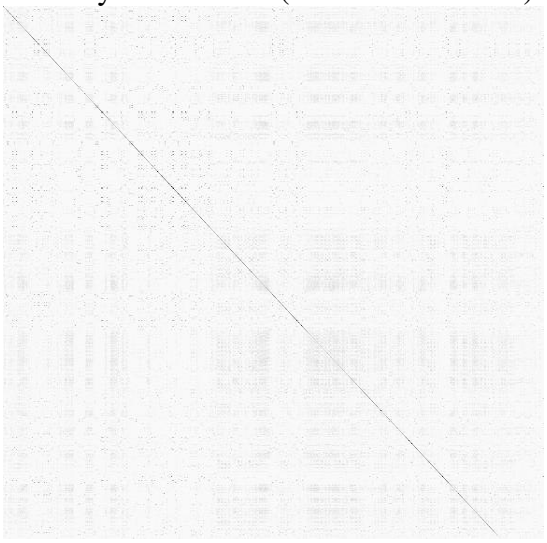

Figure 11: Information measure, Sequence 1F6WA-1BCE0,
Identity value: 78.42(higher value)

From the above figures it can be observed that:

FFAS3 (Figure 5 and 6) looks the fuzziest,

Central Limit method looks the next fuzziest (Figure 7 and 8),

Maximum Entropy Kernel method (Figure 3 and 4) looks sharper and

The Information Measure (Figure 9 and 10) looks sharpest.

The analysis done on the results show that the sharpness increases the accuracy of the results as seen in figure 9 and 10. Even for the alignments with low identity value, we can see that the sharper image yields good results.

6.2 Graphical Analysis of the experiment results

The following graphs illustrate the performance of each method against the identity and score values. The graphs are plotted in two ways: one is identity Vs accuracy and the other one is score Vs accuracy. These graphs help us in understanding the correlation between identity Vs accuracy and score Vs accuracy. It tells us whether the effect of identity or score affects the accuracy of the results. It can be seen that in all the methods that accuracy is approximately related to the score and identity of the methods. The results show that when normalization is done on the graph, we can see that our results perform much better than the results from FATCAT for all the methods however the results can be noticed to be significantly good in case of Information Measure, as seen in figure 18 and 19. The results are better in case of maximum entropy kernel as seen in figure 12 and 13, and somewhat deteriorates in case of central limit method (figure 16 and 17) and FFAS3 (figure 14 and 15).

6.2.1 Maximum Entropy Kernel– line graph

Identity VS Accuracy:

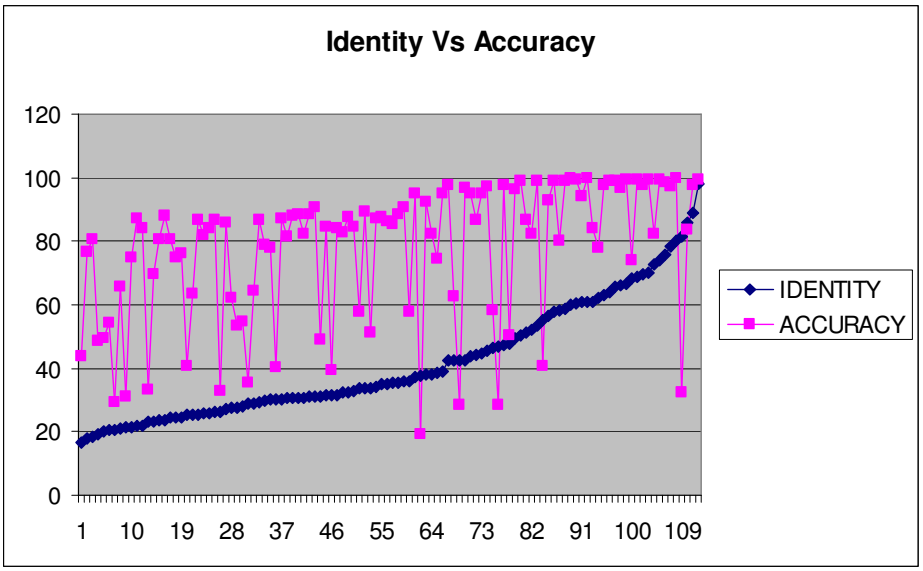

Figure 12: Showing graphical representation of Identity against Accuracy of Maximum Entropy Kernel

Score VS Accuracy:

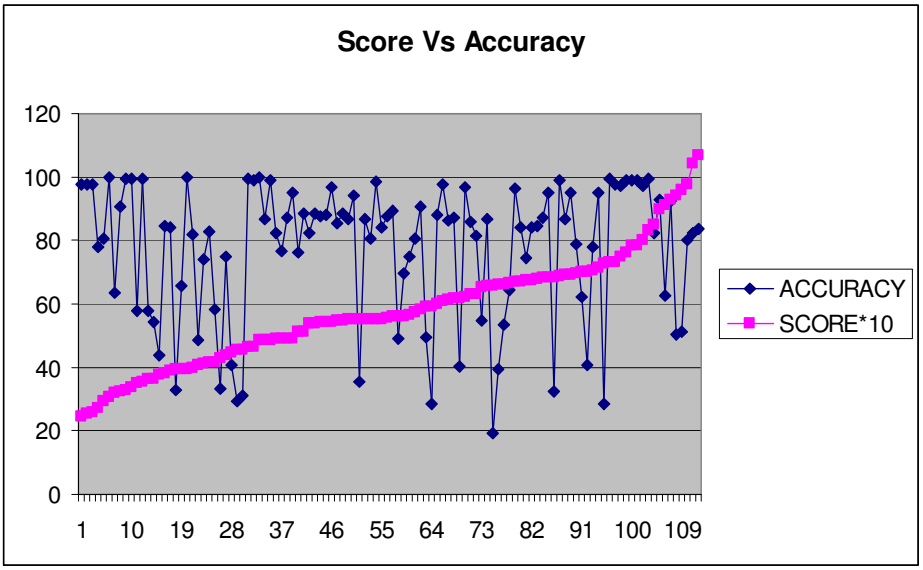

Figure 13: Showing graphical representation of Score against Accuracy of Maximum Entropy Kernel

6.2.2 FFAS3 – Line graph:

Identity Vs Accuracy:

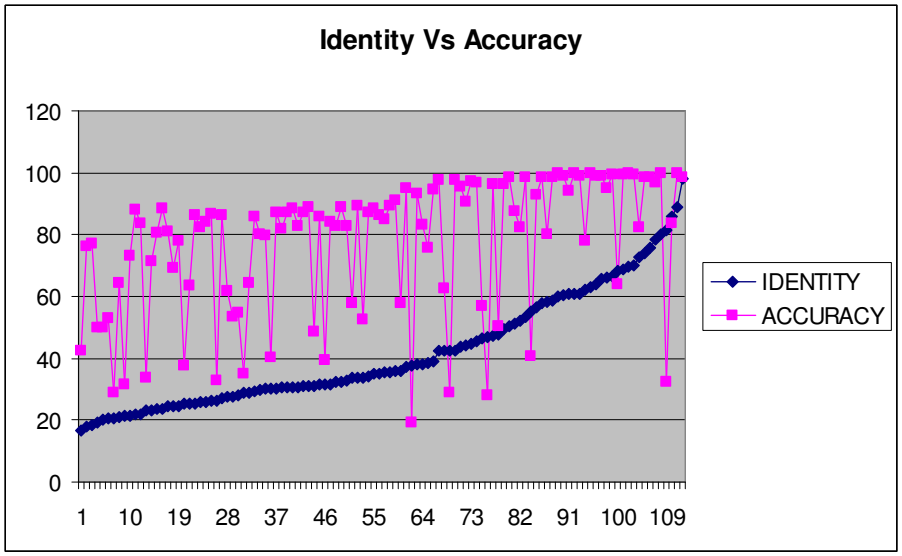

Figure 14: Showing graphical representation of Identity against Accuracy of FFAS3

Score Vs Accuracy:

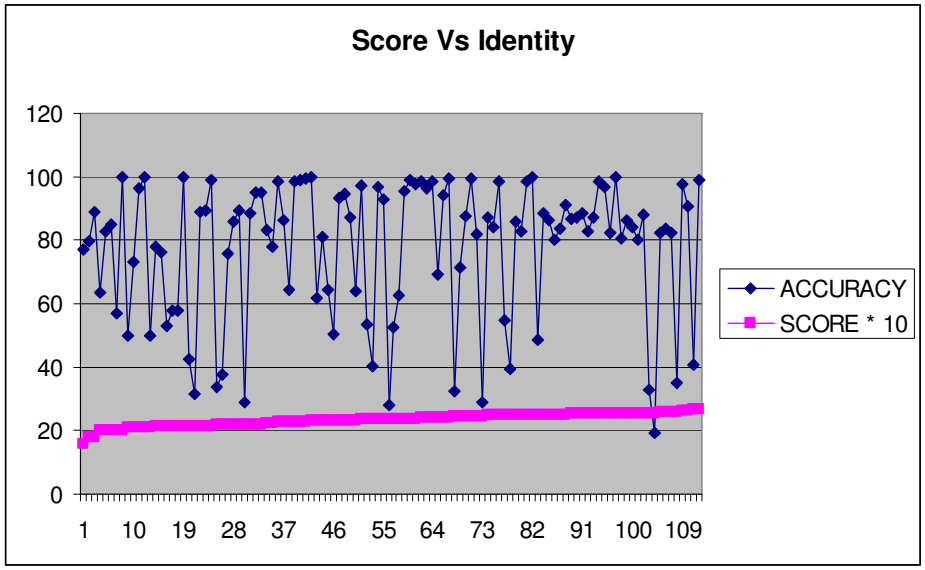

Figure 15: Showing graphical representation of score against Accuracy of FFAS3

6.2.3 CENTRAL LIMIT – Line graph:

Identity Vs Accuracy:

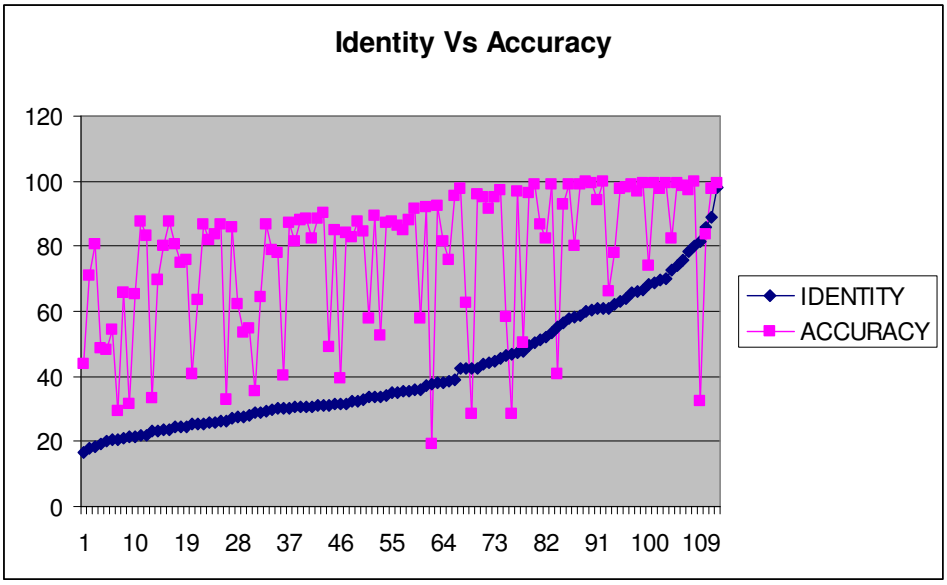

Figure 16: Showing graphical representation of Identity against Accuracy of CENTRAL LIMIT

Score Vs Accuracy:

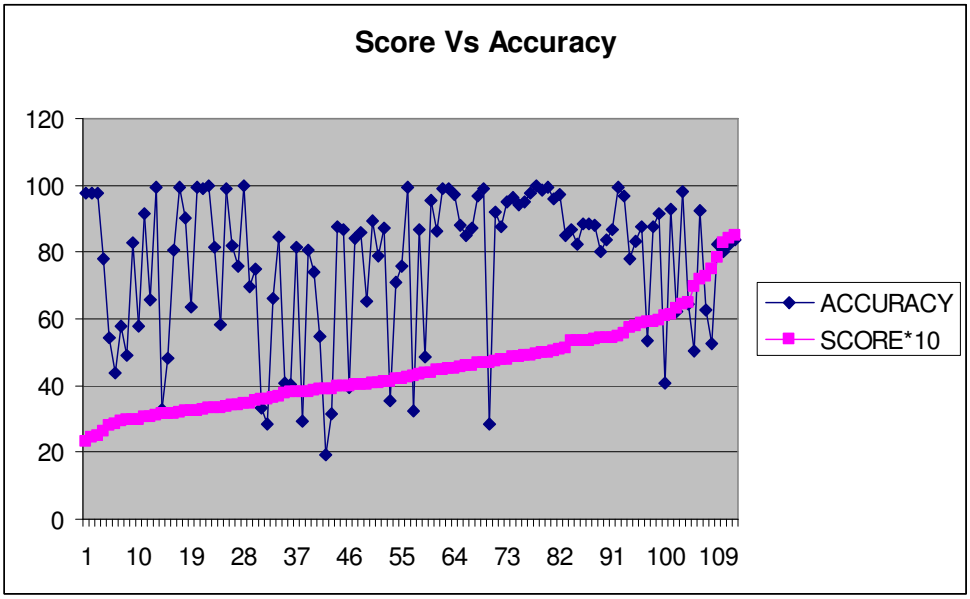

Figure 17: Showing graphical representation of score against Accuracy of CENTRAL LIMIT

6.2.4 INFORMATION MEASURE – Line Graph

Identity Vs Accuracy:

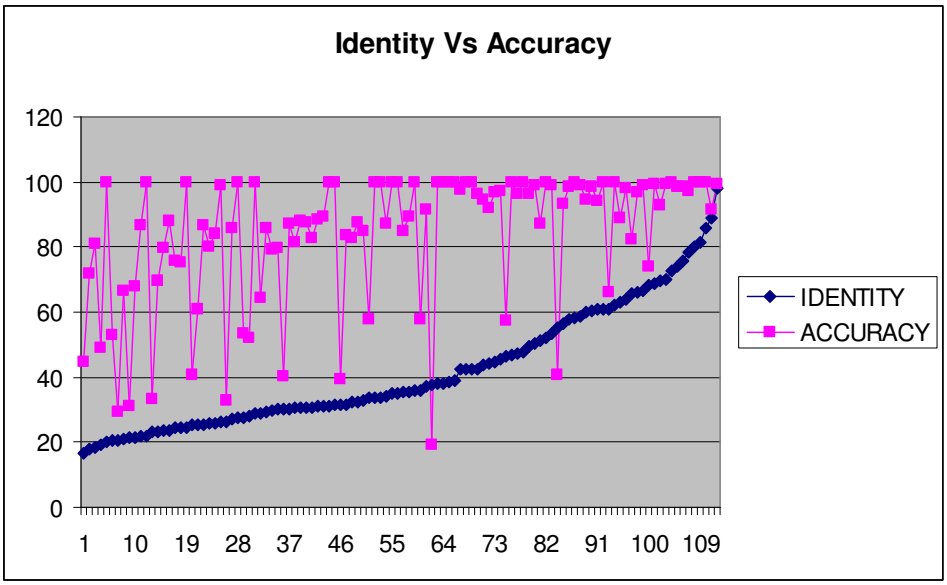

Figure 18: Showing graphical representation of Identity against Accuracy of INFORMATION MEASURE METHOD

Score Vs Accuracy:

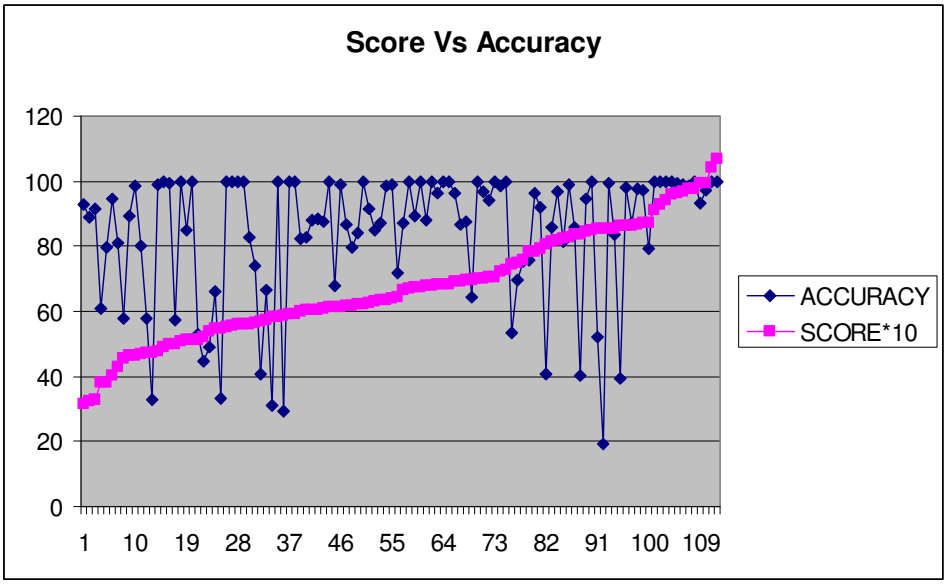

Figure 19: Showing graphical representation of Score against Accuracy of INFORMATION MEASURE METHOD

6. 3 Finding the best method by experimenting with the results:

Step 1: Finding the average performance

The following graph shows the average Accuracy of the four methods.

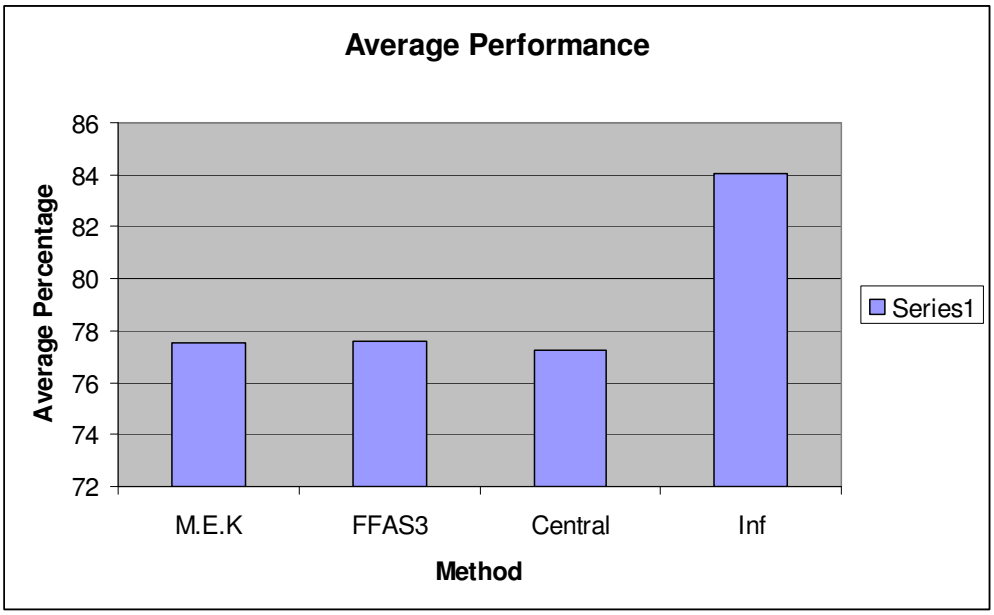

Figure 20: Performance chart, Method Vs Average Accuracy

In the column graph given above, we can see that the Information Measure method outperforms the other methods. There is significant difference in the performance of the Information Measure from the other methods. The other three methods do not have considerable differences in their performances.

Finding the average difference, and its difference from the best method:

Table 7: Showing the average difference of the results, and the difference from the best value.

Legend: Avg. Difference = Summation of the percentages of all the methods divided by 4

Diff. from Best = Avg. Difference subtracted from the best value. The best value means the highest accuracy of the four methods.

Graph plotted to show the difference obtained by subtracting the average value from the maximum value:

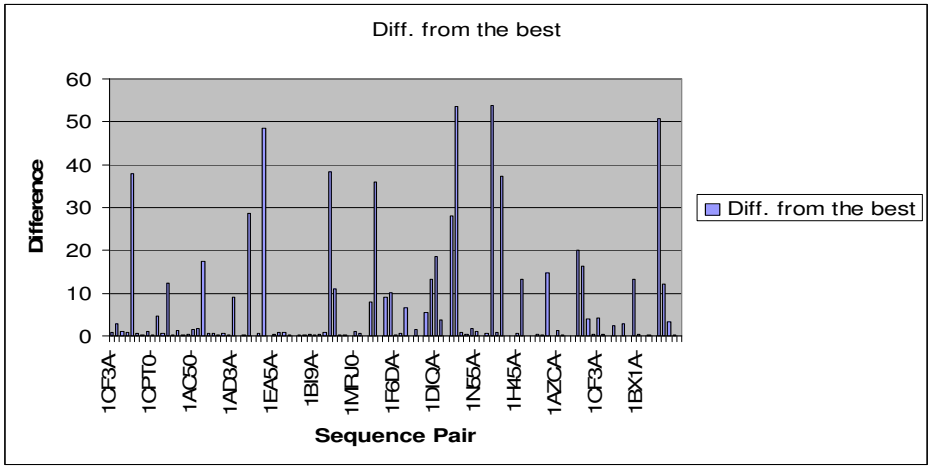

Figure 21: Column graph showing the 'Difference from the Best'

High difference from the best method means that some algorithms are inefficient in aligning the sequences whereas one of them is able to do it. The differences can be seen in figure 21. Finding this helps us know how many times each algorithm performs better than the rest.

Finding the performance of each method:

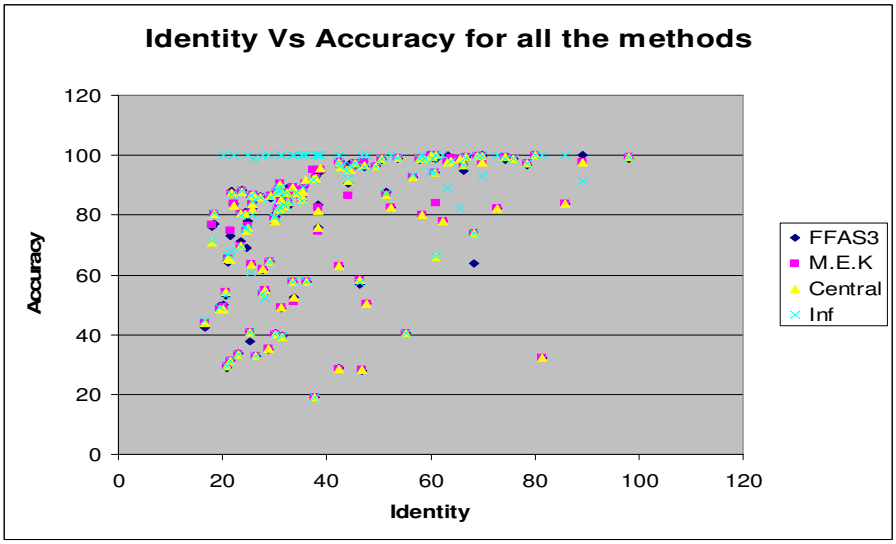

Figure 22: Scatter plot showing the performance of each method

This can be better understood when a scatter plot of all the results of all the methods are plotted against identity as in figure 22. The figure shows that in case of Information Measure the accuracy is significantly good even for low identity values.

It can be seen from the scatter plot given above, that even for low identity values, the Information measure outperforms the other methods.

Finding the error rate:

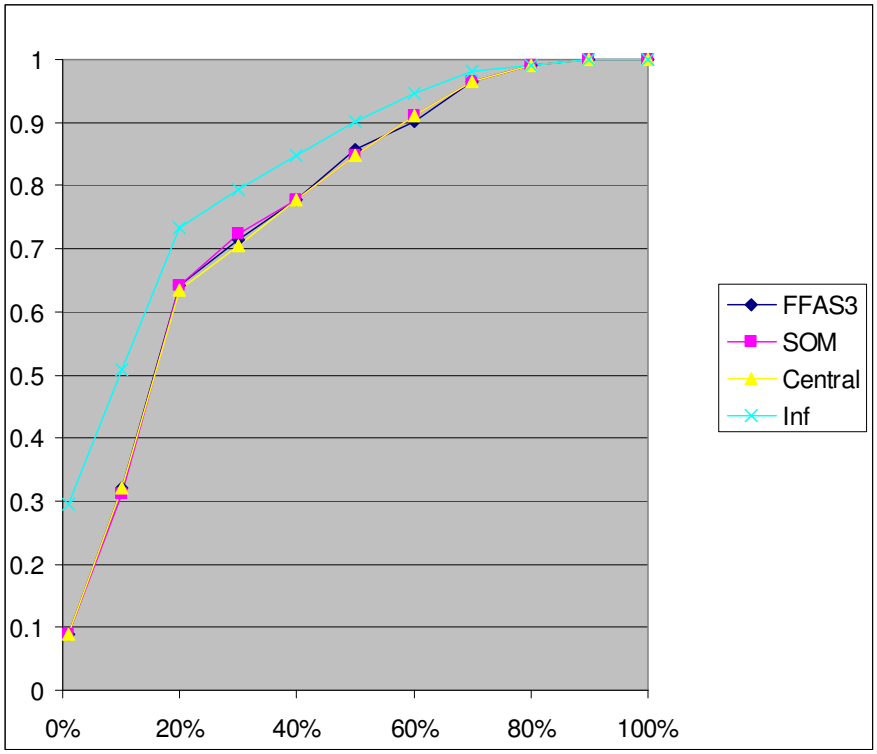

Figure 23: Error rate

The graph given above in figure 23 is used for finding the error rate. The method used to find this is called as Receiver Operating Characteristics plot. It can be clearly observed that the Information Measure has low error rates for any given identity range. The errors are due to the erratic profiles used, and their alignment method being not accurate.

Chapter 7

CONCLUSIONS AND FUTURE ENHANCEMENT

In this chapter, we are presenting a summary of the research results and the methods implemented in this thesis. We first started of by collecting a random set of protein sequences belonging to different families. This was done using SCOP. Then we found out the alignment pairs from FATCAT, which was chosen because it was believed to give good results. Then we designed modules that could do automatic conversion of the file formats obtained from FATCAT and from our algorithms. The next step was to compare the two sequences; one from FATCAT and its corresponding pair obtained from our algorithms.

The results obtained were tabulated, and graphs were plotted to visualize the accuracy of the algorithms. The graphs were plotted for Identity Vs Accuracy pair and Score Vs Accuracy pair. From the graphs, the correlation functions were obtained.

From the various results obtained by using different approaches, we designed various strategies to compare the results, and get the performance of the algorithms. The overall comparison resulted in the following conclusion:

Information Measure – Sharpest

Maximum Entropy Kernel – Moderate

Central Limit – Next fuzziest

FFAS3 – Fuzziest

It is observed that the Information Measure is the accurate of all, and it is an optimal method. So Information Measure method should set the base for further work in this field.

This thesis leads way to many future works. Some of them are:

- Finding different strategies for comparison of methods like use of dynamic programming or string shifting could be another extension of this work.
- Studying more about how Information Measure can be further improved, will result in more accurate results.

REFERENCES

[1] Toward information extraction: identifying protein names from biological papers. Fukuda K, Tamura A, Tsunoda T, Takagi T.,Human Genome Center, University of Tokyo, Japan. 1998

[2] A novel genetic system to detect protein–protein interactions, Stanley Fields and Ok-Kyu Song, Department of Microbiology, State University of New York at Stony Brook, Stony Brook, New York 11794, USA

[3] Murzin A. G., Brenner S. E., Hubbard T., Chothia C. SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.* 247, 1995.

[4] Isolation of phosphorylated peptides and proteins on ion exchange papers. DB Glass, RA Masaracchia, JR Feramisco, BE Kemp - Anal Biochem, 1978 , ncbi.nlm.nih.gov

[5] The structural alignment between two proteins: Is there a unique answer? - A GODZIK - Protein Science, 1996 , intl.proteinscience.org

[6] Intrinsically disordered protein-AK Dunker, JD Lawson, CJ Brown, RM Williams, P - J. Mol. Graph. Model, 2001.

[7] Remarks about protein structure precision, DWJ Cruickshank - Acta Crystallographica Section D Biological Crystallography, 1999, journals.iucr.org

[8] Essential dynamics of proteins - A Amadei, ABM Linssen, HJC Berendsen - Proteins: Struct. Funct. Genet, 1993, doi.wiley.com

[9] Another Human Genome Project: A Private Company's Plan Shocks the Genetics Community, John Travis, Science News, 23 May, 1998.

[10] Structure determination of a new protein from backbone-centered NMR data and NMR-assisted structure prediction**,** K. L. Mayer, Y. Qu, S. Bansal, P. D. LeBlond, F. E. Jenney Jr., P. S. Brereton, M. W. W. Adams, Y. Xu, J. H. Prestegard, Aug 2006

[11] Zhang Y, Skolnick J. (2005). The protein structure prediction problem could be solved using the current PDB library. Proc Natl Acad Sci USA 102(4):1029-34

[12] Comparing protein sequence-based and predicted secondary structure-based methods for identification of remote homologs, V. Geetha, valentine Di Fransesco, Jean Garnier, and Peter J. Munson, 1999.

[13] Performance evalutation of amino-acid substitution matrices. *Proteins*, S. Henikoff and J. G. Henikoff., 1993

[14] Elementary Sequence Analysis, Distance Measures, B. Golding, 1996.

[15] Rapid and sensitive protein similarity searches. Lipman DJ, Pearson WR, 1985

[16] Using Computers for Molecular Biology, Stuart M. Brown, Ph.D, RCR, NYU Medical Center

[17] Example borrowed from: http://en.wikipedia.org/wiki/Fasta_format

[18] Altschul, S.F., Gish, W., Miller, W., Myers, E.W. & Lipman, D.J. "Basic local alignment search tool." J. Mol. Biol. , 1990

[19] Gish, W. & States, D.J. "Identification of protein coding regions by database similarity search." Nature Genet, 1993.

[20] Nucleic Acids Research 2006. http://www.ncbi.nlm.nih.gov/Genbank/

[21]http://www.entelechon.de/index.php?id=leto/letoblog&blogid
A comparison of scoring functions for protein sequence profile alignment, Edgar RC, Sjolander K, Bioinformatics, 2004, Epub 2004.

[22] http://web.mit.edu/esgbio/www/lm/proteins/structure/structure.html

[23] http://fatcat.burnham.org/fatcat-cgi/cgi/struct_neibor/fatcatStructNeibor.pl

[24] Sampling Distribution of the Mean, Dr. Lari Arjomand

[25] Pitfalls of protein sequence analysis, Burkhard Rost, EMBL, Germany, and Alfonso Valencia, CNB-CSIC, Spain, Current Opinion in Biotechnology 1996.

[26] Predicting one-dimensional protein structure by profile based neural networks, Rost B, Meth Enzymol, 1996.

[27] Structure prediction of proteins - where are we now? Current Opinion Biotechnology, Rost B, Sander C, 1994

[28] Bridging the protein sequence-structure gap by structure predictions. Annual Biophysics Bimolecular Structure, Rost B, Sander C, 1996.

[29] The Protein Data Bank: a computer based archival file for macromolecular structures, Bernstein FC, Koetzle TF, Williams GJB, Meyer EF, Brice MD, Rodgers JR, Kennard O, Shimanouchi T, Tasumi M, 1977.

[30] Update on protein structure prediction: results, IRBM workshop, Hubbard T, Tramantano A et al, 1996.

[31] Basic local alignment search tool, Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ, 1990.

[32] Progressive sequence alignment as a a prerequisite to correct phylogenetic trees, Feng D-F, Doolittle RF, 1987

[33] M-Coffee: combining multiple sequence alignment methods with T-Coffee
Iain M. Wallace, Orla O'Sullivan, Desmond G. Higgins and Cedric Notredame, 2006.

[34] Recent progress in multiple sequence alignment: a survey *Pharmacogenomics*, Notredame, C., 2002.

[35] Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments, Hogeweg, P. and Hesper, B., 1984.

[36] MUSCLE: multiple sequence alignment with high accuracy and high throughput *Nucleic Acids Residues,* Edgar, R.C., 2004.

[37] Evaluation of iterative alignment algorithms for multiple alignment *Bioinformatics,* Wallace, I.M., O'Sullivan, O., Higgins, D.G. , 2005

[38] SAGA: sequence alignment by genetic algorithm *Nucleic Acids Res*, . Notredame, C. and Higgins, D.G., 1996.

[39] Motif recognition and alignment for many sequences by comparison of dot-matrices, Vingron, M. and Argos, P., 1991.

[40] DIALIGN: finding local similarities by multiple sequence alignment *Bioinformatics*, Morgenstern, B., Frech, K., Dress, A., Werner, T., 1998.

[41] MAFFT version 5: improvement in accuracy of multiple sequence alignment *Nucleic Acids Res*,, Katoh, K., Kuma, K.I., Toh, H., Miyata, T. 2005.

[42] ProbCons: Probabilistic consistency-based multiple sequence alignment *Genome Res*, . Do, C.B., Mahabhashyam, M.S., Brudno, M., Batzoglou, S., 2005.

[43] BAliBASE: a benchmark alignment database for the evaluation of multiple alignment programs *Nucleic Acids Res*, Thompson, J.D., Plewniak, F., Poch, O. 1999.
[44] HOMSTRAD: a database of protein structure alignments for homologous families *Protein Sci*, Mizuguchi, K., Deane, C.M., Blundell, T.L., Overington, J.P., 1998.

[45] Evaluation of protein multiple alignments by SAM-T99 using the BAliBASE multiple alignment test set *Bioinformatics*, Karplus, K. and Hu, B., 2001.

[46]. Issues in searching molecular sequence databases. *Nature Genet.*, S. F. Altschul, M. S. Boguski, W. Gish, and J. C. Wootton , 1994.

[47] Empirical statistical estimates for sequence similarity searches, W. R. Pearson., 1998.

[48] Rapid and sensitive sequence comparison with FASTP and FASTA., W. R. Pearson., 1990.

[49] Dynamic programming algorithms for biological sequence comparison., W. R. Pearson and W. Miller. , 1992.

[50] The rapid generation of mutation data matrices from protein sequences. D. T. Jones, W. R. Taylor, and J. M. Thornton. 1992

[51] Where did the BLOSUM62 alignment score matrix come from?, Sean R Eddy, Primer, Nature Biotechnology, 2004

[52] BLOCKS Database,1993-1997, Fred Hutchinson Cancer Research Center

[53] Locating Binding Sites in Protein Structures, Paul Labute and Martin Santavy *Chemical Computing Group Inc.*

[54] Guiasu, S. and Shenitzer, A., 1985, 'The principle of maximum entropy, The Mathematical Intelligencer

[55] From definition of maximum entropy method, wikepedia.

[56] Book: Generalized Information Measures and Their Applications, Inder Jeet Taneja, Ph. D. , Departamento de Matemática, Universidade Federal de Santa Catarina, Brazil. Chapter 1

[57] Sampling Distribution of the Mean, Dr. Lari Arjmond, 1996.

[58] An information measure for classification, By C. S. Wallace and D. M. Boulton , 1968.

[59] Baker, D. and Sali, A., Protein Structure Prediction and Structural Genomics (Science, Vol. 294, 5 October 2001), pp. 93-96.

[60] From definition of SCOP, Wikepedia. [61] David B. Wagner. Dynamic Programming. A 1995 introductory article on dynamic programming.

[61] Vingron M, Waterman MS (1994). Sequence alignment and penalty choice. Review of concepts, case studies and implications. *J Mol Biol.*

[62] Jaroszewski, L., Rychlewski, L., Li, Z., Li, W. & Godzik, A. (2005) FFAS03: a server for profile-profile sequence alignments. *Nucl. Acids Res.* **33**, W284-W288

[63] Henk Tijms, *Understanding Probability: Chance Rules in Everyday Life*, Cambridge: Cambridge University Press, 2004

[64] Citeseer, Accessing Distant Homology Between an Aligned Family a proposed Member trough accurate sequence alignment, Weiqing Zhang, John D. Kececioglu, Will Taylor

[65] Thesaurus, the freedictionary.com, definition of information measure.

[66] Wallace, C. S. and Boulton, D. M. (1968). An information measure for classification. Computing Journal, 11, 185-195.

[67] Dot plot of a human zinc-finger transcription factor (GenBank NM_002383) against itself to show self-similarity

[68] Godzik Laboratory, Flexible structure AlignmenT by Chaining Aligned fragment pairs allowing Twists, Yuzhen Ye and Adam Godzik , 2003

[69] Book: Information theory and statistics by Solomon Kullback, Dover Publications, Inc. 1996.

APPENDIX A

PROTEINS

In this chapter, a brief description of proteins, protein sequencing, the role of bioinformatics in identifying protein sequences and its applications are discussed for quick reference.

PROTEINS:

Proteins belong to a class of organic compounds called polyamides. The monomer units in proteins are called α – amino acids. The amino group CO-NH joining two α-amino acids is called peptide link.

Proteins are very complicated molecules. They have 20 different amino acids that can be arranged in any order to make a polypeptide of up to thousands of amino acids long. This variety allows proteins to function as specific enzymes that compose a cell's metabolism.

Proteins constitute 15% of our body mass. They are the vital factors for our life sustenance.

STRUCTURE OF PROTEINS

Proteins have multiple levels of structure ranging from primary to quaternary.

PRIMARY STRUCTURE:

A protein's primary structure is nothing but its order of the amino acids. This order, by convention, is always written from amino end to carboxyl end. The primary structure may be thought of as a complete description of all of the covalent bonding in a polypeptide chain.

An example of a protein primary structure [22] from yeast hexokinase is as follows:

```
  1 A A S X D X S L V E V H X X V F I V P P X I L Q A V V S I A
 31 T T R X D D X D S A A A S I P M V P G W V L K Q V X G S Q A
 61 G S F L A I V M G G G D L E V I L I X L A G Y Q E S S I X A
 91 S R S L A A S M X T T A I P S D L W G N X A X S N A A F S S
121 X E F S S X A G S V P L G F T F X E A G A K E X V I K G Q I
151 T X Q A X A F S L A X L X K L I S A M X N A X F P A G D X X
181 X X V A D I X D S H G I L X X V N Y T D A X I K M G I I F G
211 S G V N A A Y W C D S T X I A D A A D A G X X G G A G X M X
241 V C C X Q D S F R K A F P S L P Q I X Y X X T L N X X S P X
271 A X K T F E K N S X A K N X G Q S L R D V L M X Y K X X G Q
301 X H X X X A X D F X A A N V E N S S Y P A K I Q K L P H F D
331 L R X X X D L F X G D Q G I A X K T X M K X V V R R X L F L
361 I A A Y A F R L V V C X I X A I C Q K K G Y S S G H I A A X
391 G S X R D Y S G F S X N S A T X N X N I Y G W P Q S A X X S
421 K P I X I T P A I D G E G A A X X V I X S I A S S Q X X X A
451 X X S A X X A
```

SECONDARY STRUCTURE:

Secondary structure is the ordered arrangement of amino acids in localized regions of a protein molecule. The two main types of secondary structures are the alpha helix and the anti-parallel beta-pleated sheet. An a-helix is a clockwise spiral with each peptide bond in its trans-conformation and is planar. The amine group of each peptide bond runs upwards and parallel to the axis of the helix. The carbonyl group generally points downwards.

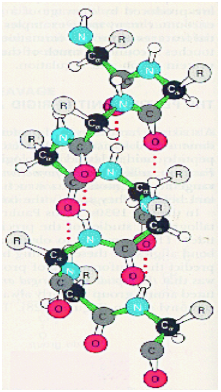

Figure 24[22]: The a-helix spiral structure

The b-pleated sheet usually consists of polypeptide chains with neighboring chains extending perpendicular to each other. As in the case of the a-helix, each peptide bond is '*trans*' and planar. The amine and carbonyl groups of peptide bonds point toward each other and in the same plane.

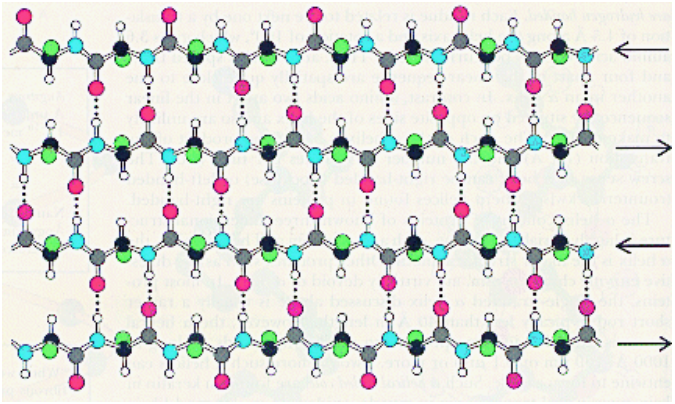

Figure 25 [22]: The b-pleated sheet model

TERTIARY STRUCTURE:

Tertiary structure is the three-dimensional folded structure of the protein. Tertiary structure is largely maintained by disulfide bonds. For a protein composed of a single polypeptide molecule, the tertiary structure is said to be the highest level of structure that is attained.

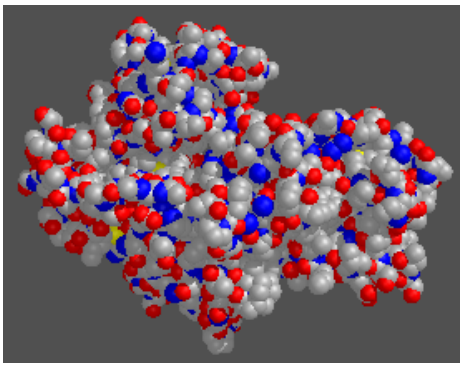

Figure 26 [22]:3-D protein structure

QUATERNARY STRUCTURE:

Quaternary structure is found only if there is more than one polypeptide chain. It is used to describe proteins composed of multiple subunits; each called a 'monomer'.

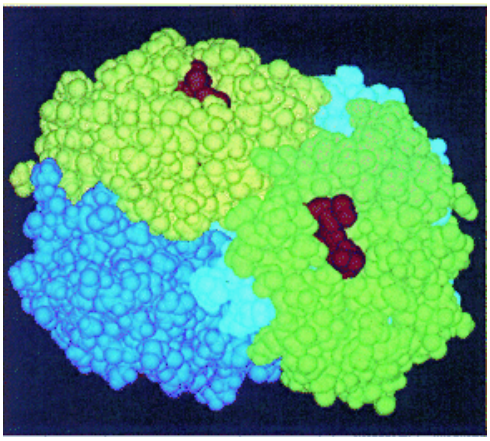

Figure 27 [22]: Quaternary structure of proteins

APPENDIX B

SEQUENCE ALIGNMENT

Aligning protein sequences with other protein sequences is a way to know the family of proteins. It helps in understanding the evolutionary process dating back from millions of years. Not only this, it also gives a way to compare a new sequence against a set of sequences in the protein database, thus co-relating the new sequence to a well-defined structure and/or function of the proteins that matched the sequence. This concept is used in the study of medicine, to study about various causes of diseases, and helps us to find a solution to the problem.

MULTIPLE ALIGNMENTS OF PROTEIN SEQUENCES:

Multiple alignments of protein sequences are important tools in studying proteins. It involves finding matching protein sequences from the database of proteins, given a sample protein sequence. The information we obtain is used in identifying conserved sequence regions. This is used in designing experiments to test and modify the function of specific proteins.

Sequence alignment is a way of arranging the primary sequence of DNA, RNA, or protein to identify regions of similarity that may be a consequence of functional, structural, or evolutionary relationships between the sequences

Sequences can be aligned across their entire length called global alignment, or only in certain regions called local alignment. This is true for pair wise and multiple alignments. Global alignments need to use gaps representing insertions/deletions, while local

alignment can avoid them. Gaps are inserted between the residues so that residues with identical or similar characters are aligned in successive columns.

Sequence alignment should be possible for any group of related proteins. For studying purposes, the initial alignment is considered to be important because it tells us the relevance to our purpose. When considering initial alignment, two extreme cases are possible:

1. If sequences are very identical to each other across their entire length, then they are not of much use to us because they do not exhibit mutation.
2. If sequences are much diverged from each other, they do not help us to co-relate them. But, still, it is possible to identify related proteins if large data is present for experimenting.

A protein sequence is said to be related by homology or convergence. Homologous proteins have common ancestor and common functions. Converged proteins are those that evolve independently to have common sequence features that have a common function.

In protein sequence alignment, the degree of similarity between amino acids occupying a certain position in the sequence can be interpreted as a rough measure of how conserved a particular region is among lineages.

APPENDIX C

SOURCE CODE

Program 1: To change the format of the input files from FATCAT

```
#FS =" "                        // This is an awk program
#BEGIN {print toupper($2)} {print toupper($5)}
/Align/ {print toupper($2)}
/Align/ {print toupper($5)}
/ini-rmsd/ { print "ini.rmsd:" $6}
/opt-rmsd/ {print "opt.rmsd:" $10}
/chain-rmsd/ {print "chain.rmsd:" $12}
/Score/ {print "Score:" $14}
/Identity/ { print "Identity:" $6}
#/1111/ {print "\n"}
/Chain 1:/ {print $4}
#/1111/ {print "\n"}
/Chain 2:/ {print $4"\n"}
#{for(i=1;i<=NF;i++) print "field:    "  $i;
#{print toupper($2)}
#{print toupper($5)}
```

Program 2: To change the format of the input files from our algorithms

```
BEGIN {FS="" }              //This is an awk program
/^8/ { var1[length($2)]=$2;len1=length($2);for (i=1;i<=len1;i++){ print var1[i]}}
/^9/ {S2=$2}
{
len=length($arr1[100])
for(i=1; i<=$len; i++)
{
if ($arr1[i] ==  $arr2[i])
{$count=+1}
}
}
END {print "Count=" $count;print "n=" $n;print "len=" $len1;
for(i=1;i<=$len1;i++)
{
print arr1[i]
}
}
```

Program 3: Awk program to do some pattern matching on the input files

```
#!/bin/ksh
for file in format/*.txt
do
gawk -f format.awk $file|sed "y/-/*/"> temp1
sed -e "s/[_]*.PDB/0/" temp1>temp2
sed -e "s/^D[1-9]/1/" temp2>$file.align
done
```

Program 4: C++ program to find the number of matching pairs in the two input sequence alignment files

```
#include<fstream.h>
#include<iostream.h>
#include<stdio.h>

int main(int argc,char *argv[])
{
      const int MAX=80;
      int
j2=1,j3,p=1,q=1,i,m=1,j,s[100],t,p2=1,q2=1,m2=1,s2[100],seq1=0,seq2=0,c
ount,bingo1,bingo2;
      char
arr[50][100],ch,temp[2][800],temp2[2][800],arr2[50][100],ch2,fat[2][800
],psi[2][800];
        char sfat[2][800],spsi[2][800];

struct orig_arr
{
char protein;
int position;
};
orig_arr
psia1[600],psib1[600],fata2[600],fatb2[600],new_sfat1[600],new_sfat2[60
0],new_sposa[800],new_sposb[800];

      for(i=1;i<=100;i++)
            {
            s[i]='\0';
            s2[i]='\0';
            }
      for(i=1;i<=800;i++)
            {
            temp[1][i]='\0';
            temp[2][i]='\0';
            temp2[1][i]='\0';
            temp2[2][i]='\0';
            }

      for(int i=0;i<50;i++)
            {
      for(int j=0;j<100;j++)
                  {
```

```
                arr[i][j]='\0';
                arr2[i][j]='\0';
                }
            }
ifstream infile(argv[1]);
            while(infile)
            {
            infile.get(ch);
                  while(ch !='\n')
             {
                if(ch=='\0')
                {
                 break;
                }
                else
                {
                  arr[p][q++]=ch;
                  infile.get(ch);
                }
                  }
                s[m++]=q-1;
            p++;
            q=1;
            }
printf("Identity from FATCAT:");
      for(i=1;i<s[7];i++)
      cout<<arr[7][i];
               for(i=1;i<=2;i++)
       {
        for(j=1;j<=s[i];j++)
         {
        cout<<arr[i][j];
        }
        cout<<"\n";
       }
      for(i=10;i<=p;i=i+3)
      for(q=1;q<=70;q++)
            {
            arr[i][q]='\0';
            }
      for(i=7;i<=p;i=i+3)
      s[i]=0;
//for(i=1;i<=p;i++)
//printf("size of[ %d]=%d\n", i,s[i]);
      printf("\nExtracted sequences:\n");
      for(i=8;i<p;i++)
      {
        for(q=1;q<=s[i];q++)
      {
        cout<<arr[i][q];
       }
         cout<<"\n";
      }
      int n=1;
      m=1;
      for(i=8;i<=p;i=i+3)                          //temp is for fatcat
      {
```

```
            for(j=1;j<=s[i];j++)
            {
              temp[1][n++]=arr[i][j];
            }
      }
//for(i=1;i<=n-1;i++)
//printf("%d",temp[1][i]);
      seq1=n-1;
      printf("seq1=%d\n", seq1);
      n=1;
      for(i=9;i<=p;i=i+3)
      {
            for(j=1;j<=s[i];j++)
            {
            temp[2][n++]=arr[i][j];
            }
      }
/*
printf("\nConcatenated into two arrays:\n",n);
        for(i=1;i<=seq1;i++)
        cout<<temp[1][i]<<" "<<temp[2][i]<<"\n";
                for(i=1;i<=2;i++)
                {
            for(j=1;j<=seq1;j++)
            {
            cout<<temp[i][j];
            }
                cout<<"\n";
                }
      */
m2=1;
//*********************************************************************
***********
ifstream infile2(argv[2]);
while(infile2)
      {
                  infile2.get(ch2);
                  while(ch2!='\n')
                  {
                  if(ch2=='\0')
                  { break;  }

                  else
                  {
            arr2[p2][q2++]=ch2;
            infile2.get(ch2);
            }
                  }
         s2[m2++]=q2-1;
         p2++;
         q2=1;
      }
printf("\nScore value from file 2:\n");
      for(i=1;i<=s2[1];i++)
      cout<<arr2[1][i];

      for(i=4;i<=p2;i=i+3)
```

```
      for(q2=1;q2<=s2[i];q2++)
            {
            arr[i][q2]='\0';
            }
      for(i=4;i<=p2;i=i+3)
      s2[i]=0;
//for(i=1;i<=p2;i++)
//printf("size of[ %d]=%d\n", i,s2[i]);
      printf("\nExtracted sequence 2(from psi-blast):\n");
                for(i=2;i<=p2;i++)
                {
            for(q2=1;q2<=s2[i];q2++)
            {
                  cout<<arr2[i][q2];
            }
                cout<<"\n";
                }
     n=1;j=1;
m=1;
for(i=2;i<=p2;i=i+3)
      {
            for(j=1;j<=s2[i];j++)
            {
            temp2[1][n++]=arr2[i][j];
            }

      }
seq2=n-1;
n=1;
      for(i=3;i<=p2;i=i+3)
            {
            for(j=1;j<=s2[i];j++)
                  {
                  temp2[2][n++]=arr2[i][j];
                  }
            //printf("\ns2[%d]=%d\n",i,s[i]);
            }
/*printf("\nConcatenated into array:\n");           //temp2 is for psi-
blast
   for(i=1;i<=2;i++)
  {
   for(j=1;j<=seq2;j++)
   {
    cout<<temp2[i][j];
   }
    cout<<"\n";
   }
//for(i=1;i<=seq2;i++)
//cout<<temp2[1][i]<<" "<<temp2[2][i]<<"\n";
*/
j=1; i=1;n=1;
int l=0,k=0,miscount=0;
count=0;
while(i<=30)
{
      if(count>2)
        break;
```

```
      if(temp2[1][i]!=temp[1][j])// checkout for matching position
     { miscount++;
      i++;
      }
      else
            {
            k=j+1;
            l=i+1;
            if(temp2[1][l]!=temp[1][k])
            i++;
            else
                  {
                      count=count+1;
                  i++;
                  j++;

                  }
            }

}

count=i-count;
//if(miscount>5)
//count=10;
printf("position=%d \n sequence=%d\n",count,seq2);
int seq3=0;

      for(i=1;i<=2;i++)
      for(j=1;j<=800;j++)
      {
            fat[i][j]='\0';
            psi[i][j]='\0';
                sfat[i][j]='\0';
                spsi[i][j]='\0';
           }
n=1;
      for(i=1;i<=2;i++)
      {
      for(j=count;j<=seq2;j++)
            {
                        psi[i][n++]=temp2[i][j];
            }
      n=1;
      }
i=1;
      while(psi[1][i]!='\0')
      {
      i++;
      }
seq3=i-1;
int seq,se;
      if(seq1<seq2)
      seq=seq1;
      else
      seq=seq2;
       printf("\n Seq1= %d\n seq2=%d \nseq=%d", seq1,seq2,seq);
//printf("\n After matching the start of the sequences:\n");
```

```
for(i=1;i<=2;i++)
        {
        for(j=1;j<=seq;j++)
        {
        fat[i][j]=temp[i][j];
        }
        }
/*printf("\nSequence 2 from psi-blast:\n");
for(i=1;i<=2;i++)
        {
        for(j=1;j<=seq;j++)
        {
        psi[i][j]=temp[i][j];
        cout<<psi[i][j];
        }
        printf("\n");
        }
*/
for(i=1;i<=seq;i++)
{
psia1[i].protein='\0';
psia1[i].position=0;
fata2[i].position=0;
psib1[i].position=0;
fatb2[i].position=0;
fata2[i].protein='\0';
psib1[i].protein='\0';
fatb2[i].protein='\0';
new_sfat1[i].protein='\0';
new_sfat1[i].position=0;
new_sfat2[i].position=0;
new_sfat2[i].protein='\0';
}
for(i=1;i<=seq;i++)
{
psia1[i].protein=psi[1][i];
psia1[i].position=i;
fata2[i].position=i;
psib1[i].position=i;
fatb2[i].position=i;
fata2[i].protein=fat[1][i];
psib1[i].protein=psi[2][i];
fatb2[i].protein=fat[2][i];
}
/*for(i=1;i<=seq;i++)
{
cout<<psia1[i].protein<<"
"<<psia1[i].position<<"|"<<psib1[i].protein<<" "<<psib1[i].position<<"
"<<fata2[i].protein<<" "<<fata2[i].position<<"|"<<fatb2[i].protein;
cout<<" "<<fatb2[i].position<<"\n";
}
*/
j=1;
      for(i=1;i<=seq2;i++)
      {
            if(psi[1][i]=='*' && psi[1][i]!='\0')
            continue;
```

```
            else
            {
            new_sposa[j].protein=psi[1][i];
            new_sposa[j].position=j;
            new_sposb[j].protein=psi[2][i];
            new_sposb[j].position=i;
                j++;
            }
      }
      for(i=1;i<=seq2;i++)
      {
            if(fat[1][i]=='*' && fat[2][i]!='\0')
            continue;
            else
            {
            new_sfat1[j2].protein=fat[1][i];
            new_sfat1[j2].position=j2;
            new_sfat2[j2].protein=fat[2][i];
            new_sfat2[j2].position=i;
            j2++;
            }
      }
/*
for(i=1;i<=seq2;i++)
      {
      if(psi[2][i]=='*'&& psi[2][i]!='\0')
      continue;
      else
      spsi[2][j3++]=psi[2][i];
      }
for(i=1;i<=seq;i++)
      {
      cout<<sfat[1][i]<<" "<<sfat[2][i]<<" |
"<<spsi[1][i]<<spsi[2][i]<<"\n";
      }
*/
printf("shrink\n");
/* for(i=1;i<=seq;i++)
      {
      if(new_sposa[i].protein!='\0')
            {
            cout<<new_sposa[i].protein;
            cout<<"
"<<new_sposa[i].position<<"|"<<new_sposb[i].protein<<"
      "<<new_sposb[i].position<<"\n";
            }
      else
      break;
      }
*/
int length;
length=i-1;
bingo1=1;
for(i=1;i<=length;i++)
      {
```

```
if((new_sposa[i].protein==new_sfat1[i].protein)&&(new_sposa[i].position
==new_sfat1[i].position)&&((new_sposa[i].protein!='\0') &&
(new_sfat1[i].protein!='\0')))
            {
             if(new_sposb[i].protein==new_sfat2[i].protein &&
      new_sposb[i].protein!='\0' && new_sfat2[i].protein!='\0')
            {
//     cout<<"\n"<<new_sposa[i].protein<<" "<<new_sposb[i].protein;
//     cout<<" "<<new_sfat1[i].protein<<"
"<<new_sfat2[i].protein<<"\n";
       bingo1++;
                    }
            }
      }
//printf("\nBINGO 1=%d\n",bingo1-1);
printf("\nBINGO 1=%d\n",bingo1-1);
//printf("\nBINGO 2=%d\n",bingo2-1);
printf("Length 1=%d\n",length);
//printf("Lenth 2= %d",length2);
printf("Identity from FATCAT:");
        for(i=1;i<s[7];i++)
        cout<<arr[7][i];
printf("Score value from file 2:\n");
        for(i=1;i<=s2[1];i++)
        cout<<arr2[1][i];
//*********************************************************************
*******
int score=0;
for(j=1;j<=seq;j++)
{
if((fat[1][j]==psi[1][j]) && (fat[2][j]==psi[2][j]))
{score=score+1;
}
}
printf("\n\n Score = %d\n\n", score);
//*********************************************************************
*****
count=0;

/*printf("seq2=%d", seq2);
        for(j=1;j<=seq;j++)
        {
        if(fat[1][j]!='\0')
        count=count+1;
        }
printf("count=%d", count);
*/
}
```

APPENDIX D

EXPERIMENTAL RESULTS

Results from Maximum Entropy Kernel method:

SEQUENCE PAIR	IDENTITY	NUM. OF HITS	LENGTH	SCORE	ACCURACY
1CF3A-1KDGA.align_136	16.54	178	405	3.767959	43.95061728
1MRJ0-1QI7A.align_100	18.01	200	261	4.892712	76.62835249
1OOYA-1POIA.align_1116	18.39	210	261	2.939893	80.45977011
1K6DA-1POIA.align_186	19.43	120	247	4.084214	48.58299595
1AC50-1WHT0.align_17	20.17	229	463	5.911321	49.4600432
1CF3A-1JU2A.align_135	20.61	214	393	3.644429	54.45292621
1CPT0-1ODOA.align_141	20.78	120	409	4.558136	29.3398533
1AC50-1IVYA.align_16	21.17	330	504	3.950794	65.47619048
1CPT0-1DZ4A.align_142	21.58	130	417	4.57	31.17505995
1DLC0-1I5PA.align_145	21.59	170	227	4.362265	74.88986784
1AD3A-1UZBA.align_114	21.83	392	449	6.168434	87.30512249
1AD3A-1UXNA.align_113	22.08	380	453	6.714322	83.88520971
1CPT0-1LFKA.align_130	23.02	135	404	4.27176	33.41584158
1EA5A-1THG0.align_150	23.41	402	576	5.591395	69.79166667
1AD3A-1BI9A.align_18	23.5	363	451	5.519207	80.48780488
1AD3A-1EUHA.align_10	23.81	388	441	5.988865	87.98185941
1AC50-1CPY0.align_15	24.31	381	473	5.72491	80.54968288
1EA5A-1GZ7A.align_154	24.56	426	570	5.663376	74.73684211
1HRDA-1HWXA.align_181	24.74	148	194	5.119852	76.28865979
1CPT0-1JFBA.align_139	25.25	166	408	4.481258	40.68627451
1E3JA-1JVBA.align_148	25.41	115	181	3.192881	63.5359116
1AD3A-1O04A.align_111	25.44	391	452	5.515295	86.50442478
1OS8A-1TON0.align_1117	25.63	195	238	3.980906	81.93277311
1AD3A-1O9JA.align_112	25.66	381	452	5.538068	84.2920354
1AD3A-1BXSA.align_19	26.06	390	449	5.49918	86.8596882
1TON0-1UCY0.align_1126	26.44	86	261	3.93	32.95019157
1EA5A-1QE3A.align_159	27.19	452	526	6.310761	85.93155894
1LVL0-1GRS0.align_195	27.68	139	224	7.010285	62.05357143
1LVL0-1ONFA.align_197	27.73	118	220	6.625127	53.63636364
1K4YA-1BCE0.align_185	28.15	300	547	6.537891	54.84460695
1TON0-1RP2A.align_1124	28.94	83	235	5.510303	35.31914894
1GESA-1LVL0.align_166	29.09	142	220	6.648236	64.54545455
1EA5A-1F6WA.align_152	29.35	478	552	6.55232	86.5942029
1EA5A-1K4YA.align_155	29.85	431	546	6.972468	78.93772894
1A53A-1PII0.align_1	30.12	202	259	2.716951	77.99227799
1F8UA-1BCE0.align_165	30.16	220	547	6.196808	40.21937843

1EA5A-1BCE0.align_140	30.4	477	546	6.821065	87.36263736
1MX1A-1BCE0.align_103	30.66	447	548	6.31741	81.56934307
1O04A-1UZBA.align_1112	30.83	446	506	5.419308	88.14229249
1UZBA-1BXSA.align_129	30.83	447	506	5.473991	88.33992095
1BI9A-1UZBA.align_126	30.84	419	509	5.367789	82.31827112
1O9JA-1UZBA.align_1113	31.03	447	506	5.407814	88.33992095
1I4NA-1PII0.align_182	31.1	230	254	3.244906	90.5511811
1LUGA-1ZNC0.align_192	31.23	132	269	5.590944	49.07063197
1MRJ0-1TFMA.align_101	31.33	211	249	6.810171	84.73895582
1N5MA-1BCE0.align_1111	31.56	214	545	6.622496	39.26605505
1EA5A-1MX1A.align_156	31.68	458	546	6.725425	83.88278388
1LUGA-1ZNCA.align_194	32.22	224	270	4.16187	82.96296296
1MRJ0-1ONKA.align_199	32.28	223	254	5.41391	87.79527559
1PII0-1VC4A.align_1119	32.68	217	257	3.809386	84.43579767
1AK20-1S3GA.align_116	33.51	110	191	3.512316	57.59162304
1BVUA-1HRDA.align_127	33.52	160	179	5.586709	89.38547486
1PZEA-1UXJA.align_1121	33.79	74	145	9.594587	51.03448276
1CPT0-1DLC0.align_137	34.06	200	229	4.897619	87.33624454
1EUZA-1HRDA.align_161	35.2	157	179	5.553004	87.70949721
1B26A-1HRDA.align_124	35.23	152	176	6.125555	86.36363636
1F6DA-1V4VA.align_163	35.37	321	376	5.471936	85.37234043
1MRJ0-1UQ5A.align_102	35.55	227	256	5.123775	88.671875
1LUGA-1RJ6A.align_189	35.98	239	264	5.835612	90.53030303
1AK20-1ZIN0.align_117	36.13	110	191	3.632336	57.59162304
1LEHA-1C1DA.align_120	37.44	193	203	4.920991	95.07389163
1DX4A-1EA5A.align_147	37.68	104	544	6.56946	19.11764706
1GV1A-1PZEA.align_167	37.93	134	145	9.277181	92.4137931
1H45A-1NKXA.align_174	38.3	282	342	4.892506	82.45614035
1DIQA-1QLTA.align_143	38.46	194	260	6.723626	74.61538462
1AW1A-1N55A.align_118	38.89	240	252	6.835295	95.23809524
1B9BA-1N55A.align_125	42.4	244	250	7.334283	97.6
1LVL0-1LADA.align_196	42.42	145	231	9.095095	62.77056277
1A8I0-1L5WA.align_13	42.45	231	808	7.272206	28.58910891
1TON0-1TRNA.align_1125	42.54	221	228	6.22014	96.92982456
1N55A-1BTMA.align_104	44	238	250	7.133661	95.2
1IYXA-1ONEA.align_184	44.11	257	297	6.90091	86.53198653
1N55A-1O5XA.align_106	44.58	237	249	6.933276	95.18072289
1N55A-1NEYA.align_105	45.38	242	249	7.467237	97.18875502
1AK20-1AKY0.align_115	46.35	112	192	4.169659	58.33333333
1A8I0-1YGPA.align_14	46.81	235	831	5.926558	28.27918171
1F6DA-1O6CA.align_162	47.27	357	366	6.093797	97.54098361
1PZEA-1T2DA.align_1110	47.74	78	155	9.421396	50.32258065
1N55A-1R2RA.align_107	49.4	240	249	6.690825	96.38554217
1M6JA-1N55A.align_198	50.58	254	257	6.870615	98.83268482
1H45A-1IEJA.align_160	51.38	284	327	4.848183	86.85015291
1CDOA-1HETA.align_131	52.26	164	199	8.516372	82.4120603

1KEQA-1LUGA.align_187	53.59	235	237	4.856467	99.15611814
1OEPA-1ONEA.align_1114	55.29	119	293	7.016006	40.61433447
1P5HA-1Q7EA.align_1118	56.47	394	425	8.9959	92.70588235
1EA5A-1F8UA.align_153	57.97	527	533	7.825631	98.87429644
1HETA-1M6HA.align_179	58.38	158	197	9.785246	80.20304569
1EA5A-1N5MA.align_157	58.83	526	532	7.834775	98.87218045
1AZCA-1JOI0.align_110	60.16	128	128	3.075229	100
1H45A-1H76A.align_169	60.25	320	322	4.631847	99.37888199
1H45A-1RYOA.align_175	60.87	303	322	5.505967	94.09937888
1AZCA-1NWPA.align_122	60.94	128	128	3.95779	100
1H45A-1JNFA.align_171	60.99	272	323	3.915142	84.21052632
1E3JA-1PL7A.align_149	62.36	139	178	7.054646	78.08988764
1AZCA-1JZGA.align_121	63.28	125	128	2.540213	97.65625
1ONEA-1PDZ0.align_1115	63.95	291	294	7.610665	98.97959184
1CF3A-1GPEA.align_134	65.71	381	385	4.65595	98.96103896
1IDK0-1QCXA.align_183	66.3	347	359	5.448807	96.65738162
1CE2A-1H45A.align_132	66.67	319	321	3.385577	99.37694704
1H45A-1JW1A.align_172	68.22	237	321	4.130954	73.8317757
1N55A-1TCDA.align_109	68.95	247	248	8.332209	99.59677419
1AZCA-1RKRA.align_123	69.77	126	129	2.464808	97.6744186
1KV5A-1N55A.align_188	69.88	248	249	7.29352	99.59839357
1DITA-1HETA.align_144	72.73	163	198	10.40201	82.32323232
1BX1A-1H45A.align_128	74.21	313	315	3.29	99.36507937
1DLC0-1JI6A.align_146	75.77	224	227	5.535889	98.6784141
1F6WA-1BCE0-align_164	78.42	518	533	8.030763	97.18574109
1LUGA-1V9EA.align_191	80.16	257	257	4.840286	100
1A8I0-1L5SA.align_12	81.38	256	795	6.835542	32.20125786
1HETA-1HSOA.align_177	85.86	166	198	10.68392	83.83838384
1AZCA-1DYZA.align_119	89.06	125	128	2.573122	97.65625
1H45A-1LGBC.align_173	98.11	158	159	3.560387	99.37106918

Table 3: Results from Maximum entropy kernel method

Results from FFAS3 method:

SEQUENCE PAIR	IDENTITY	NUM. OF HITS	LENGTH	SCORE	ACCURACY
1OOYA-1POIA.align_1116	18.39	201	261	1.56	77.01149425
1A53A-1PII0.align_1	30.12	206	259	1.8	79.53667954
1I4NA-1PII0.align_182	31.1	226	254	1.81	88.97637795
1E3JA-1JVBA.align_148	25.41	115	181	2.02	63.5359116
1PII0-1VC4A.align_1119	32.68	213	257	2.02	82.87937743
1F6DA-1V4VA.align_163	35.37	320	376	2.02	85.10638298
1AK20-1AKY0.align_115	46.35	109	192	2.03	56.77083333
1AZCA-1RKRA.align_123	69.77	129	129	2.03	100
1K6DA-1POIA.align_186	19.43	123	247	2.09	49.79757085

1DLC0-1I5PA.align 145	21.59	166	227	2.09	73.1277533
1F6DA-1O6CA.align 162	47.27	352	366	2.09	96.17486339
1AZCA-1JZGA.align 121	63.28	128	128	2.09	100
1AC50-1WHT0.align 17	20.17	232	463	2.111795	50.10799136
1HRDA-1HWXA.align 181	24.74	151	194	2.129707	77.83505155
1MRJ0-1QI7A.align 100	18.01	199	261	2.13	76.24521073
1CF3A-1JU2A.align 135	20.61	209	393	2.13	53.18066158
1AK20-1S3GA.align 116	33.51	110	191	2.13	57.59162304
1AK20-1ZIN0.align 117	36.13	110	191	2.13	57.59162304
1AZCA-1DYZA.align 119	89.06	128	128	2.13	100
1CF3A-1KDGA.align 136	16.54	172	405	2.14	42.4691358
1CPT0-1DZ4A.align 142	21.58	131	417	2.16	31.41486811
1MRJ0-1ONKA.align 199	32.28	226	254	2.16	88.97637795
1MRJ0-1UQ5A.align 102	35.55	229	256	2.16	89.453125
1H45A-1H76A.align 169	60.25	319	322	2.16	99.06832298
1CPT0-1LFKA.align 130	23.02	136	404	2.17	33.66336634
1CPT0-1JFBA.align 139	25.25	154	408	2.17	37.74509804
1DIQA-1QLTA.align 143	38.46	197	260	2.173243	75.76923077
1MRJ0-1TFMA.align 101	31.33	214	249	2.178822	85.9437751
1BVUA-1HRDA.align 127	33.52	160	179	2.179657	89.38547486
1CPT0-1ODOA.align 141	20.78	118	409	2.18	28.85085575
1EUZA-1HRDA.align 161	35.2	158	179	2.194751	88.26815642
1LEHA-1C1DA.align 120	37.44	193	203	2.2	95.07389163
1IDK0-1QCXA.align 183	66.3	341	359	2.2	94.98607242
1H45A-1NKXA.align 174	38.3	285	342	2.234349	83.33333333
1E3JA-1PL7A.align 149	62.36	139	178	2.249664	78.08988764
1H45A-1LGBC.align 173	98.11	157	159	2.26	98.74213836
1B26A-1HRDA.align 124	35.23	152	176	2.262336	86.36363636
1AC50-1IVYA.align 16	21.17	324	504	2.27	64.28571429
1BX1A-1H45A.align 128	74.21	310	315	2.27	98.41269841
1CF3A-1GPEA.align 134	65.71	381	385	2.28	98.96103896
1CE2A-1H45A.align 132	66.67	319	321	2.28	99.37694704
1AZCA-1JOI0.align 110	60.16	128	128	2.31	100
1LVL0-1GRS0.align 195	27.68	138	224	2.317521	61.60714286
1AC50-1CPY0.align 15	24.31	383	473	2.32	80.97251586
1GESA-1LVL0.align 166	29.09	142	220	2.32	64.54545455
1PZEA-1T2DA.align 1110	47.74	78	155	2.325589	50.32258065
1GV1A-1PZEA.align 167	37.93	135	145	2.331289	93.10344828
1AW1A-1N55A.align 118	38.89	238	252	2.339672	94.44444444
1CPT0-1DLC0.align 137	34.06	200	229	2.34	87.33624454
1H45A-1JW1A.align 172	68.22	205	321	2.34	63.86292835
1N55A-1O5XA.align 106	44.58	242	249	2.35	97.18875502
1LVL0-1ONFA.align 197	27.73	118	220	2.36	53.63636364
1F8UA-1BCE0.align 165	30.16	221	547	2.36	40.40219378
1N55A-1NEYA.align 105	45.38	241	249	2.36	96.78714859
1P5HA-1Q7EA.align 1118	56.47	394	425	2.36	92.70588235

1A8I0-1YGPA.align 14	46.81	233	831	2.36311	28.03850782
1PZEA-1UXJA.align 1121	33.79	76	145	2.363578	52.4137931
1LVL0-1LADA.align 196	42.42	145	231	2.36398	62.77056277
1N55A-1BTMA.align 104	44	239	250	2.37	95.6
1H45A-1JNFA.align 171	60.99	319	323	2.37	98.76160991
1B9BA-1N55A.align 125	42.4	244	250	2.38	97.6
1M6JA-1N55A.align 198	50.58	253	257	2.39	98.44357977
1N55A-1R2RA.align 107	49.4	240	249	2.41	96.38554217
1DLC0-1JI6A.align 146	75.77	224	227	2.41	98.6784141
1EA5A-1GZ7A.align 154	24.56	394	570	2.43	69.12280702
1H45A-1RYOA.align 175	60.87	303	322	2.43	94.09937888
1KV5A-1N55A.align 188	69.88	248	249	2.43	99.59839357
1A8I0-1L5SA.align 12	81.38	257	795	2.432197	32.32704403
1EA5A-1THG0.align 150	23.41	411	576	2.44	71.35416667
1H45A-1IEJA.align 160	51.38	287	327	2.44	87.7675841
1N55A-1TCDA.align 109	68.95	247	248	2.44	99.59677419
1MX1A-1BCE0.align 103	30.66	448	548	2.45	81.75182482
1A8I0-1L5WA.align 13	42.45	233	808	2.466028	28.83663366
1EA5A-1BCE0.align 140	30.4	477	546	2.48	87.36263736
1EA5A-1MX1A.align 156	31.68	459	546	2.48	84.06593407
1EA5A-1F8UA.align 153	57.97	525	533	2.48	98.49906191
1K4YA-1BCE0.align 185	28.15	299	547	2.49	54.66179159
1N5MA-1BCE0.align 1111	31.56	216	545	2.49	39.63302752
1EA5A-1F6WA.align 152	29.35	473	552	2.5	85.6884058
1LUGA-1ZNCA.align 194	32.22	224	270	2.5	82.96296296
1KEQA-1LUGA.align 187	53.59	234	237	2.5	98.73417722
1AZCA-1NWPA.align 122	60.94	128	128	2.5	100
1LUGA-1ZNC0.align 192	31.23	131	269	2.506702	48.69888476
1AD3A-1EUHA.align 10	23.81	390	441	2.51	88.43537415
1EA5A-1QE3A.align 159	27.19	454	526	2.51	86.31178707
1EA5A-1K4YA.align 155	29.85	438	546	2.51	80.21978022
1AD3A-1UXNA.align 113	22.08	379	453	2.514268	83.66445916
1LUGA-1RJ6A.align 189	35.98	240	264	2.514546	90.90909091
1AD3A-1BXSA.align 19	26.06	389	449	2.52	86.63697105
1UZBA-1BXSA.align 129	30.83	442	506	2.52	87.35177866
1O04A-1UZBA.align 1112	30.83	448	506	2.52	88.53754941
1BI9A-1UZBA.align 126	30.84	419	507	2.52	82.64299803
1O9JA-1UZBA.align 1113	31.03	442	506	2.52	87.35177866
1EA5A-1N5MA.align 157	58.83	525	532	2.52	98.68421053
1F6WA-1BCE0-align 164	78.42	515	533	2.52	96.62288931
1OS8A-1TON0.align 1117	25.63	196	238	2.53	82.35294118
1LUGA-1V9EA.align 191	80.16	257	257	2.53	100
1AD3A-1BI9A.align 18	23.5	363	451	2.55	80.48780488
1AD3A-1O04A.align 111	25.44	389	452	2.55	86.0619469
1AD3A-1O9JA.align 112	25.66	381	452	2.55	84.2920354
1HETA-1M6HA.align 179	58.38	158	197	2.554671	80.20304569

1AD3A-1UZBA.align 114	21.83	395	449	2.56	87.97327394
1TON0-1UCY0.align 1126	26.44	86	261	2.56	32.95019157
1DX4A-1EA5A.align 147	37.68	104	544	2.56	19.11764706
1DITA-1HETA.align 144	72.73	163	198	2.566516	82.32323232
1HETA-1HSOA.align 177	85.86	166	198	2.571579	83.83838384
1CDOA-1HETA.align 131	52.26	164	199	2.571689	82.4120603
1TON0-1RP2A.align 1124	28.94	82	235	2.58644	34.89361702
1TON0-1TRNA.align 1125	42.54	223	228	2.62	97.80701754
1IYXA-1ONEA.align 184	44.11	269	297	2.63	90.57239057
1OEPA-1ONEA.align 1114	55.29	119	293	2.65	40.61433447
1ONEA-1PDZ0.align 1115	63.95	291	294	2.68	98.97959184

Table 4: Results from FFAS3 method

Results from Central limit method:

SEQUENCE PAIR	IDENTITY	NUM. OF HITS	LENGTH	SCORE	ACCURACY
1AZCA-1RKRA.align 123	69.77	126	129	2.326546	97.6744186
1AZCA-1JZGA.align 121	63.28	125	128	2.430701	97.65625
1AZCA-1DYZA.align 119	89.06	125	128	2.488038	97.65625
1A53A-1PII0.align 1	30.12	202	259	2.638757	77.992278
1CF3A-1JU2A.align 135	20.61	214	393	2.79841	54.4529262
1CF3A-1KDGA.align 136	16.54	178	405	2.827991	43.9506173
1AK20-1S3GA.align 116	33.51	110	191	2.944811	57.591623
1LUGA-1ZNC0.align 192	31.23	132	269	2.966648	49.070632
1LUGA-1ZNCA.align 194	32.22	224	270	2.966648	82.962963
1AK20-1ZIN0.align 117	36.13	110	191	2.996919	57.591623
1LUGA-1RJ6A.align 189	35.98	242	264	3.045907	91.6666667
1AC50-1IVYA.align 16	21.17	330	504	3.077281	65.4761905
1H45A-1LGBC.align 173	98.11	158	159	3.129039	99.3710692
1TON0-1UCY0.align 1126	26.44	86	261	3.14	32.9501916
1AC50-1WHT0.align 17	20.17	224	463	3.143895	48.3801296
1OOYA-1POIA.align 1116	18.39	210	261	3.166227	80.4597701
1BX1A-1H45A.align 128	74.21	316	318	3.18	99.3710692
1I4NA-1PII0.align 182	31.1	229	254	3.244877	90.1574803
1E3JA-1JVBA.align 148	25.41	115	181	3.245112	63.5359116
1CE2A-1H45A.align 132	66.67	319	321	3.261909	99.376947
1KEQA-1LUGA.align 187	53.59	235	237	3.272355	99.1561181
1LUGA-1V9EA.align 191	80.16	257	257	3.308144	100
1H45A-1NKXA.align 174	38.3	278	342	3.332507	81.2865497
1AK20-1AKY0.align 115	46.35	112	192	3.345782	58.3333333
1CF3A-1GPEA.align 134	65.71	381	385	3.361984	98.961039
1OS8A-1TON0.align 1117	25.63	195	238	3.414002	81.9327731

1HRDA-1HWXA.align 181	24.74	147	194	3.418304	75.7731959
1AZCA-1JOI0.align 110	60.16	128	128	3.449973	100
1EA5A-1THG0.align 150	23.41	402	576	3.468597	69.7916667
1EA5A-1GZ7A.align 154	24.56	426	570	3.55158	74.7368421
1CPT0-1LFKA.align 130	23.02	135	404	3.595433	33.4158416
1A8I0-1YGPA.align 14	46.81	235	831	3.612093	28.2791817
1H45A-1JNFA.align 171	60.99	213	323	3.619388	65.9442724
1PII0-1VC4A.align 1119	32.68	217	257	3.665152	84.4357977
1CPT0-1JFBA.align 139	25.25	166	408	3.787099	40.6862745
1F8UA-1BCE0.align 165	30.16	220	547	3.792794	40.2193784
1MX1A-1BCE0.align 103	30.66	447	548	3.820093	81.5693431
1CPT0-1ODOA.align 141	20.78	120	409	3.822938	29.3398533
1AC50-1CPY0.align 15	24.31	381	473	3.825584	80.5496829
1H45A-1JW1A.align 172	68.22	237	321	3.832626	73.8317757
1K4YA-1BCE0.align 185	28.15	300	547	3.898908	54.8446069
1DX4A-1EA5A.align 147	37.68	104	544	3.902861	19.1176471
1CPT0-1DZ4A.align 142	21.58	131	417	3.91	31.4148681
1EUZA-1HRDA.align 161	35.2	157	179	3.972296	87.7094972
1EA5A-1F6WA.align 152	29.35	478	552	3.976952	86.5942029
1N5MA-1BCE0.align 1111	31.56	214	545	4.002013	39.266055
1EA5A-1MX1A.align 156	31.68	458	546	4.016177	83.8827839
1EA5A-1QE3A.align 159	27.19	452	526	4.033488	85.9315589
1DLC0-1I5PA.align 145	21.59	148	227	4.05053	65.1982379
1BVUA-1HRDA.align 127	33.52	160	179	4.052246	89.3854749
1EA5A-1K4YA.align 155	29.85	431	546	4.073736	78.9377289
1EA5A-1BCE0.align 140	30.4	477	546	4.099	87.3626374
1TON0-1RP2A.align 1124	28.94	83	235	4.100905	35.3191489
1MRJ0-1QI7A.align 100	18.01	185	261	4.189095	70.8812261
1DIQA-1QLTA.align 143	38.46	197	260	4.197452	75.7692308
1H45A-1H76A.align 169	60.25	320	322	4.256189	99.378882
1A8I0-1L5SA.align 12	81.38	257	795	4.312136	32.327044
1H45A-1IEJA.align 160	51.38	284	327	4.350891	86.8501529
1K6DA-1POIA.align 186	19.43	120	247	4.358905	48.582996
1AW1A-1N55A.align 118	38.89	241	252	4.366476	95.6349206
1B26A-1HRDA.align 124	35.23	152	176	4.445739	86.3636364
1EA5A-1N5MA.align 157	58.83	526	532	4.468538	98.8721805
1EA5A-1F8UA.align 153	57.97	527	533	4.494265	98.8742964
1F6WA-1BCE0-align 164	78.42	518	533	4.503138	97.1857411
1MRJ0-1UQ5A.align 102	35.55	225	256	4.562148	87.890625
1MRJ0-1TFMA.align 101	31.33	212	249	4.598989	85.1405622
1CPT0-1DLC0.align 137	34.06	200	229	4.616483	87.3362445
1IDK0-1QCXA.align 183	66.3	347	359	4.679016	96.6573816
1M6JA-1N55A.align 198	50.58	254	257	4.707082	98.8326848
1A8I0-1L5WA.align 13	42.45	231	808	4.707212	28.5891089
1LEHA-1C1DA.align 120	37.44	187	203	4.734829	92.1182266
1MRJ0-1ONKA.align 199	32.28	222	254	4.754607	87.4015748

1N55A-1O5XA.align 106	44.58	237	249	4.770565	95.1807229
1N55A-1R2RA.align 107	49.4	240	249	4.852942	96.3855422
1H45A-1RYOA.align 175	60.87	303	322	4.873815	94.0993789
1N55A-1BTMA.align 104	44	238	250	4.892353	95.2
1B9BA-1N55A.align 125	42.4	244	250	4.902055	97.6
1AZCA-1NWPA.align 122	60.94	128	128	4.961099	100
1DLC0-1JI6A.align 146	75.77	224	227	4.982348	98.6784141
1KV5A-1N55A.align 188	69.88	248	249	4.993571	99.5983936
1TON0-1TRNA.align 1125	42.54	219	228	5.018021	96.0526316
1N55A-1NEYA.align 105	45.38	242	249	5.066887	97.188755
1F6DA-1V4VA.align 163	35.37	320	376	5.114064	85.106383
1AD3A-1BXSA.align 19	26.06	390	449	5.3306	86.8596882
1BI9A-1UZBA.align 126	30.84	419	509	5.344547	82.3182711
1O9JA-1UZBA.align 1113	31.03	447	506	5.352206	88.3399209
1UZBA-1BXSA.align 129	30.83	447	506	5.358535	88.3399209
1O04A-1UZBA.align 1112	30.83	446	506	5.380837	88.1422925
1AD3A-1BI9A.align 18	23.5	362	451	5.439382	80.2660754
1AD3A-1O9JA.align 112	25.66	379	452	5.442263	83.8495575
1AD3A-1O04A.align 111	25.44	391	452	5.448973	86.5044248
1N55A-1TCDA.align 109	68.95	247	248	5.490807	99.5967742
1F6DA-1O6CA.align 162	47.27	354	366	5.55541	96.7213115
1E3JA-1PL7A.align 149	62.36	139	178	5.744487	78.0898876
1AD3A-1UXNA.align 113	22.08	377	453	5.779145	83.2229581
1AD3A-1UZBA.align 114	21.83	393	449	5.856967	87.5278396
1LVL0-1ONFA.align 197	27.73	118	220	5.895121	53.6363636
1AD3A-1EUHA.align 10	23.81	387	441	5.90488	87.755102
1IYXA-1ONEA.align 184	44.11	272	297	5.976877	91.5824916
1OEPA-1ONEA.align 1114	55.29	119	293	6.106505	40.6143345
1P5HA-1Q7EA.align 1118	56.47	394	425	6.130155	92.7058824
1LVL0-1GRS0.align 195	27.68	139	224	6.320241	62.0535714
1ONEA-1PDZ0.align 1115	63.95	289	294	6.453656	98.2993197
1GESA-1LVL0.align 166	29.09	142	220	6.485401	64.5454545
1PZEA-1T2DA.align 1110	47.74	78	155	6.978652	50.3225806
1GV1A-1PZEA.align 167	37.93	134	145	7.196792	92.4137931
1LVL0-1LADA.align 196	42.42	145	231	7.252491	62.7705628
1PZEA-1UXJA.align 1121	33.79	76	145	7.506439	52.4137931
1CDOA-1HETA.align 131	52.26	164	199	7.842339	82.4120603
1HETA-1M6HA.align 179	58.38	158	197	8.297164	80.2030457
1DITA-1HETA.align 144	72.73	163	198	8.395378	82.3232323
1HETA-1HSOA.align 177	85.86	166	198	8.491598	83.8383838

Table 5: Results from Central Limit method

Results from Information measure:

SEQUENCE PAIR	IDENTITY	NUM. OF HITS	LENGTH	SCORE	ACCURACY
1AZCA-1RKRA.align_123	69.77	120	129	3.148005	93.02325581
1AZCA-1JZGA.align_121	63.28	114	128	3.221488	89.0625
1AZCA-1DYZA.align_119	89.06	117	128	3.268343	91.40625
1E3JA-1JVBA.align_148	25.41	110	181	3.788823	60.77348066
1A53A-1PII0.align_1	30.12	206	259	3.790522	79.53667954
1AZCA-1JOI0.align_110	60.16	121	128	4.03987	94.53125
1OOYA-1POIA.align_1116	18.39	211	261	4.298754	80.84291188
1AK20-1S3GA.align_116	33.51	110	191	4.573377	57.59162304
1I4NA-1PII0.align_182	31.1	227	254	4.633372	89.37007874
1BX1A-1H45A.align_128	74.21	314	318	4.64	98.74213836
1OS8A-1TON0.align_1117	25.63	191	238	4.678401	80.25210084
1AK20-1ZIN0.align_117	36.13	110	191	4.715858	57.59162304
1TON0-1UCY0.align_1126	26.44	86	261	4.73	32.95019157
1CE2A-1H45A.align_132	66.67	318	321	4.784122	99.06542056
1H45A-1NKXA.align_174	38.3	342	342	4.892506	100
1H45A-1LGBC.align_173	98.11	158	159	4.988107	99.37106918
1AK20-1AKY0.align_115	46.35	110	192	4.993077	57.29166667
1AZCA-1NWPA.align_122	60.94	128	128	5.060215	100
1PII0-1VC4A.align_1119	32.68	218	257	5.116611	84.82490272
1HRDA-1HWXA.align_181	24.74	194	194	5.119852	100
1CF3A-1JU2A.align_135	20.61	209	393	5.13648	53.18066158
1CF3A-1KDGA.align_136	16.54	181	405	5.206509	44.69135802
1K6DA-1POIA.align_186	19.43	121	247	5.405477	48.98785425
1H45A-1JNFA.align_171	60.99	214	323	5.465528	66.25386997
1CPT0-1LFKA.align_130	23.02	135	404	5.479936	33.41584158
1TON0-1RP2A.align_1124	28.94	235	235	5.510303	100
1EUZA-1HRDA.align_161	35.2	179	179	5.553004	100
1BVUA-1HRDA.align_127	33.52	179	179	5.586709	100
1LUGA-1ZNC0.align_192	31.23	269	269	5.590944	100
1LUGA-1ZNCA.align_194	32.22	224	270	5.590944	82.96296296
1H45A-1JW1A.align_172	68.22	237	321	5.670341	73.8317757
1CPT0-1JFBA.align_139	25.25	166	408	5.709054	40.68627451
1AC50-1IVYA.align_16	21.17	336	504	5.719475	66.66666667
1CPT0-1DZ4A.align_142	21.58	130	417	5.82	31.17505995
1LUGA-1RJ6A.align_189	35.98	264	264	5.835612	100
1CPT0-1ODOA.align_141	20.78	120	409	5.848619	29.3398533
1AC50-1WHT0.align_17	20.17	463	463	5.911321	100
1A8I0-1YGPA.align_14	46.81	831	831	5.926558	100
1CF3A-1GPEA.align_134	65.71	317	385	5.998557	82.33766234
1BI9A-1UZBA.align_126	30.84	421	509	6.025383	82.71119843
1O04A-1UZBA.align_1112	30.83	446	506	6.055684	88.14229249
1O9JA-1UZBA.align_1113	31.03	448	506	6.065021	88.53754941

1UZBA-1BXSA.align_129	30.83	444	506	6.097239	87.74703557
1B26A-1HRDA.align_124	35.23	176	176	6.125555	100
1DLC0-1I5PA.align_145	21.59	154	227	6.127994	67.84140969
1AD3A-1BXSA.align_19	26.06	444	449	6.132996	98.88641425
1AD3A-1O04A.align_111	25.44	391	452	6.155402	86.50442478
1AD3A-1BI9A.align_18	23.5	360	451	6.182964	79.82261641
1AD3A-1O9JA.align_112	25.66	381	452	6.213703	84.2920354
1LUGA-1V9EA.align_191	80.16	257	257	6.227231	100
1LEHA-1C1DA.align_120	37.44	186	203	6.271126	91.62561576
1F6DA-1V4VA.align_163	35.37	319	376	6.287119	84.84042553
1H45A-1IEJA.align_160	51.38	285	327	6.340487	87.1559633
1H45A-1H76A.align_169	60.25	318	322	6.343656	98.75776398
1KEQA-1LUGA.align_187	53.59	235	237	6.375556	99.15611814
1MRJ0-1QI7A.align_100	18.01	187	261	6.450029	71.64750958
1CPT0-1DLC0.align_137	34.06	200	229	6.649626	87.33624454
1AD3A-1UXNA.align_113	22.08	453	453	6.714322	100
1MRJ0-1UQ5A.align_102	35.55	229	256	6.72296	89.453125
1DIQA-1QLTA.align_143	38.46	260	260	6.723626	100
1AD3A-1EUHA.align_10	23.81	388	441	6.793389	87.98185941
1MRJ0-1TFMA.align_101	31.33	249	249	6.810171	100
1TON0-1TRNA.align_1125	42.54	220	228	6.832828	96.49122807
1AW1A-1N55A.align_118	38.89	252	252	6.835295	100
1A8I0-1L5SA.align_12	81.38	795	795	6.835542	100
1F6DA-1O6CA.align_162	47.27	352	366	6.928352	96.17486339
1AD3A-1UZBA.align_114	21.83	390	449	6.940936	86.8596882
1MRJ0-1ONKA.align_199	32.28	222	254	6.947872	87.4015748
1GESA-1LVL0.align_166	29.09	142	220	6.959899	64.54545455
1LVL0-1GRS0.align_195	27.68	224	224	7.010285	100
1IDK0-1QCXA.align_183	66.3	347	359	7.013639	96.65738162
1H45A-1RYOA.align_175	60.87	303	322	7.03577	94.09937888
1E3JA-1PL7A.align_149	62.36	178	178	7.054646	100
1DLC0-1JI6A.align_146	75.77	224	227	7.247263	98.6784141
1A8I0-1L5WA.align_13	42.45	808	808	7.272206	100
1LVL0-1ONFA.align_197	27.73	118	220	7.459781	53.63636364
1EA5A-1THG0.align_150	23.41	400	576	7.483016	69.44444444
1EA5A-1GZ7A.align_154	24.56	429	570	7.586422	75.26315789
1AC50-1CPY0.align_15	24.31	358	473	7.818661	75.68710359
1N55A-1R2RA.align_107	49.4	240	249	7.852081	96.38554217
1IYXA-1ONEA.align_184	44.11	273	297	7.940152	91.91919192
1OEPA-1ONEA.align_1114	55.29	119	293	8.078508	40.61433447
1EA5A-1QE3A.align_159	27.19	452	526	8.143128	85.93155894
1N55A-1O5XA.align_106	44.58	241	249	8.181449	96.78714859
1MX1A-1BCE0.align_103	30.66	447	548	8.241111	81.56934307
1M6JA-1N55A.align_198	50.58	254	257	8.282576	98.83268482
1EA5A-1F6WA.align_152	29.35	475	552	8.343328	86.05072464
1F8UA-1BCE0.align_165	30.16	220	547	8.355074	40.21937843

1N55A-1BTMA.align 104	44	237	250	8.441402	94.8
1CDOA-1HETA.align 131	52.26	199	199	8.516372	100
1K4YA-1BCE0.align 185	28.15	286	547	8.524375	52.28519196
1DX4A-1EA5A.align 147	37.68	104	544	8.526212	19.11764706
1KV5A-1N55A.align 188	69.88	248	249	8.554265	99.59839357
1EA5A-1MX1A.align 156	31.68	456	546	8.554892	83.51648352
1N5MA-1BCE0.align 1111	31.56	215	545	8.613395	39.44954128
1ONEA-1PDZ0.align 1115	63.95	288	294	8.628617	97.95918367
1EA5A-1BCE0.align 140	30.4	477	546	8.641654	87.36263736
1B9BA-1N55A.align 125	42.4	244	250	8.685403	97.6
1N55A-1NEYA.align 105	45.38	242	249	8.720511	97.18875502
1EA5A-1K4YA.align 155	29.85	432	546	8.736659	79.12087912
1LVL0-1LADA.align 196	42.42	231	231	9.095095	100
1GV1A-1PZEA.align 167	37.93	145	145	9.277181	100
1PZEA-1T2DA.align 1110	47.74	155	155	9.421396	100
1PZEA-1UXJA.align 1121	33.79	145	145	9.594587	100
1N55A-1TCDA.align 109	68.95	247	248	9.616022	99.59677419
1EA5A-1N5MA.align 157	58.83	526	532	9.689771	98.87218045
1EA5A-1F8UA.align 153	57.97	526	533	9.765296	98.68667917
1HETA-1M6HA.align 179	58.38	197	197	9.785246	100
1P5HA-1Q7EA.align 1118	56.47	396	425	9.957903	93.17647059
1F6WA-1BCE0-align 164	78.42	518	533	9.961377	97.18574109
1DITA-1HETA.align 144	72.73	198	198	10.40201	100
1HETA-1HSOA.align 177	85.86	198	198	10.68392	100

Table 6: Results from Information Measure