Table of Contents

1. Abstract

SupfamMod is a software development project. The aim of the project is to create a protein prediction system, therefore make a contribution to the world of protein prediction and structural biology. Current project brings novelty to the field of bioinformatics by introducing a combination of technologies, techniques and software that haven't been used before.

SupfamMod stands for Superfamily and MODELLER. These are two main software components which are be peered together in order to predict protein structures. Additionally SupfamMod uses a number of supplementary software tools and databases: HMMER3, Superfamily database, ASTRAL database and SCOP database. The combination these tools make SupfamMod a unique system.

SupfamMod was assessed and compared to world's best protein predictors, both automatic and human assisted. The system is takes 67th place out of 79 best protein predictors that participated in CASP9 competition in 2010. Current result is a good achievement for a short summer project and can be improved with further development.

2. Introduction

Molecule of the protein has a unique structure that determines its functionality. Protein folds into three-dimensional structure depending on its amino acid sequence. SupfamMod allows predicting three-dimensional protein structure based on amino acid sequence.

SupfamMod is a protein structure prediction pipelining system. Protein structure prediction is a task for modelling three-dimensional structure of a protein based on its primary structure. The primary structure of a protein is a string of amino-acid molecules. Depending on the arrangement of these molecules protein folds into unique form.

Proteins govern all biological processes a living organism. Determination of a protein structure is essential for understanding its biological functionality. Experimental determination of protein 3D structure is slow and expensive process. While the prediction process can take up to 30 minutes.

Current system is available to use online. Target sequence is accepted in FASTA format. Predicted structure is outputted in PDB format with alignment in PIR format. The protein is previewed directly on web page using Jmol chemical viewer.

Daily self-evaluation logs SupfamMod's performance by making predictions based on latest PDB.org sequences. Then predictions are compared to corresponding original protein sequences using CE software.

SupfamMod implementation step included 3 different prediction techniques, which were tested using 225 protein structures. Usage of multiple templates and Superfamily Model ID for predictions did not improve system's results. SCOP domain ID alignment displayed better results.

Overall SupfamMod showed a good performance in comparison to world's best predictors. Current result is very good for a prediction system, which was created as a summer project. It is important to mention that the competing groups usually include a group of scientists who devote their careers designing protein prediction systems and the groups spend reasonable amounts of money to perform the research.

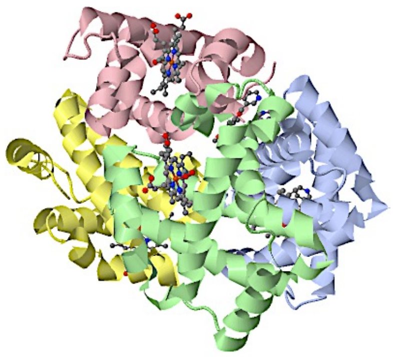
3. Introduction to Protein Structure

In current section is an introduction to the protein structure. Current chapter will explain what are the proteins and what their structures. This chapter should give the reader sufficient knowledge in order to understand what and how this project is trying to achieve.

3.1 Protein

Proteins are nitrogenous organic compounds which molecules are built out of amino acids. These molecules exist in every living organism. Their role in the functionality of biotic processes of any animal or plant is of great significance. They support the growth, development and normal proceeding of all metabolic reactions. In the human body there are expected to be found 90000 of different protein structures, which will have different functionality and purpose. In all living creatures and pants combined there are expected to be found millions of distinctive proteins.

The functionality of the protein is specified by its structure. Proteins are involved in every aspect of organism's life. Proteins may be used in transportation, canalization and signaling functions. Additionally proteins play a very important role of being a structure and shape formation of cells. The importance of proteins for proper functionality of an organism is crucial. The two examples presented below will illustrate current fact. (John W Adamson 1975)


Example 1. Hemoglobin

As an example of proteins it was decided to describe two most well known proteins – insulin and hemoglobin. Insulin is a protein, which is a part of signaling system of the human body. It controls various the metabolism processes in the human body. Without it, human organism will lose its ability to process fat and sugar in the body. Current metabolism malfunction will result in death of a human being. Hemoglobin (presented on the right) is another protein, which has another functionality – transportation. Its primary responsibility is to transport oxygen across the body within red blood cells. Without hemoglobin human being will die of oxygen deficiency, which will cause organism to malfunction and die. These are only two proteins out of thousands, which are involved in the processes of the human body, and their absence will cause organism to fail dramatically. (Michael Schrader 2001) (Cuatrecasas 1981)

In order to talk about the functionality of a protein it is also important to understand its structure and how protein is being produced inside of the organism. The structure of the protein is important due the fact that its functionality is determined by its three-dimensional structure. And the importance of understanding how the structure of the protein is being formed is defined by the topic of current paper.

3.2 DNA to Protein

In order for an organism build a protein inside a cell, there is a special mechanism activated. Its basic responsibilities are to find a gene in the DNA which would correspond to a required protein and then build a protein by its description.

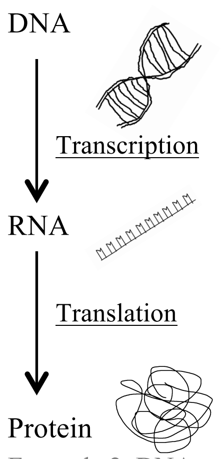
The process of creation of protein in an organism can be divided into two steps:

1. Transcription

In genomes of living organisms there are genes, which encode proteins. Special microbiological machinery that is called RNA polymerase reads the DNA strand. While gene strand is being read, polymerase creates a complimentary messenger RNA (mRNA) strand. Current peace of RNA then is being used, to produce final protein sequence.

2. Translation

The created mRNA strand, which was created in the first step, is being read by ribosome machinery. Ribosome reads every three NT (nucleotides) that represent one amino acid. While the protein is being produced, it folds into three-dimensional structure/molecule.



Example 2. DNA to Protein

Current description of a protein creation process is a bit simplified and as everything in biology there are some exceptions and the process is a bit more complicated. However current chapter is introductory and is written to familiarize the reader with the concept of proteins, their structure and processes that take place in order to produce a protein. For this reason the author has decided not to go into more depth, therefore current description is sufficient to understand the topic and the problem, which is being addressed by current paper. (Nello Cristianini 2009)

3.3 Protein Structure

Proteins consist of chains of amino acids, which are also called residues. There are 20 different amino acids, which construct proteins. Their length varies from 20-40 AAs (amino acids) to several thousand AAs. Although in general proteins are several hundred residues long. Different AAs are combined together into molecular strings, which can be read and represented as amino acid strings. An example of amino acid sequence is presented below:

VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFDLSHGSAQVK
GHGKKVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAA
HLPAEFTPAVHASLDKFLASVSTVLTSKYR

Current AA sequence belongs to hemoglobin and represents a structure that is presented on the above page. Current sequence is 141 residues long and its structure appears to be straightforward. This representation is only the primary structure of the protein, but moreover there are secondary, tertiary and quaternary.

Primary Structure

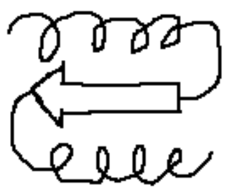
The primary structure as described above is a string of AA letters. The string is linear and finite. Each protein sequence has a beginning and ending site. It contains all the required information in order to build or predict the structure of a protein. Current structure is a basis of the protein and includes all required molecules for protein to fold into higher-level structures.

MPNALSALSD
LHAHKLRVDP
VNFKLLSHMP
NALSALSDLH
AHKLRVDPVN

Ex. 3. Primary structure

Secondary Structure

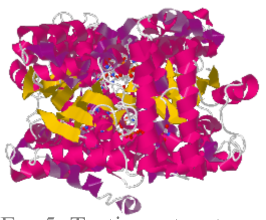
The secondary structure is a sequence of alpha-helices and beta-pleated sheets. Current structure includes some three-dimensional structuring, defined by hydrogen bonds, which represented by iterative forms. α-helix has a form of spring and β-pleated sheets, are similar to plain rectangular leaves.


Ex. 4. Secondary structure

Example 4 on the right shows an approximate representation of secondary structure where alpha-helices (top and bottom) and beta-pleated sheets (in the middle) are established. It is a simplification, which shows how secondary structure might be represented. Those regions correspond to specific primary structure regions.

Tertiary and Quaternary Structure

The tertiary structure packs the secondary structure in one condensed molecule. The process of packing of the molecule is called folding.


Ex. 5. Tertiary structure

An example of such structure is presented on page 3 of current paper under Example 1. This is a precise process which makes every protein molecule unique. Basically by the laws of physics the molecule of a protein with its residues is being condensed into very specific three-dimensional arrangement. The quaternary structure is formed out of several tertiary structures. Which are connected using polypeptide bonds. (Carl-Ivar Brändén 1999)

3.4 3D structure knowledge prospects

Knowing the 3D structure of proteins is the key progress point, which will increase the effectiveness of drug discovery. In the recent years the drug industry suffers from inability to create new effective drugs. The knowledge of protein tertiary and quaternary structures would become a new breakthrough in pharmacology. (Miles Congreve 2005)

This is due the reasons that discovering three-dimensional structures of proteins shows molecule-binding regions and their functionality. Which leads to understanding how specific protein features can be altered or manipulated in order to affect the mechanisms of an (human) organism. Many pharmaceutical companies already use techniques of protein structure determination for drug development. And a number of drugs were already released using current methods. (Horrobin 2000)

The only problem is that the process of tertiary structure determination is extremely slow and expensive. This is due to two technologies, which are being used - x-ray crystallography and nuclear magnetic resonance spectroscopy. These two methods use real proteins in order to build the structural representation of the macromolecule.

4. Protein prediction techniques and algorithms

There are techniques such as x-ray crystallography and NMR spectroscopy, which allow using an experiment to establish 3D structures of proteins. But current techniques might not work with certain proteins because they are too big. Also current methods are relatively slow and expensive. (M S Smyth 2000)

In order to overcome the specified problems, tertiary structure prediction techniques are being created. Current section introduces such prediction techniques and algorithms, which allow simulating tertiary protein structures. Three techniques are described: ab initio, fold recognition and homology modelling. (Zhang 2008)

4.2 Ab initio

Ab initio, also known as new fold protein prediction, is a technique, which uses the laws of physics in order to simulate the folder of the protein. The initial idea of current approach of protein structure prediction was to use molecular dynamics. Current approach was used at the beginning of the development of ab initio techniques. But there are several problems that are associated with current approach.

The first problem is that current approach requires a lot of computational power. For example in order to simulate one microsecond of protein folding of a 100 amino-acid structure approximately 400 hours of regular stationary computer's computational time is required. Unfortunately the folding of a protein of average length takes around 1 second. This increases the time required a million times. And even more time is required for proteins, which are longer than 100 residues. (Richard Bonneau 2001)

To make this task faster to perform large scale super computers are required to be used. But still it takes a reasonable amount of time to fold a single protein.

Another serious problem is that the algorithms required to simulate the protein folding in the water are highly inaccurate. There is a number diverse physical models of water and all of them give inadequate results during the protein fold simulations. Additionally model of how the physical forces are distributed alongside the protein length, makes it extremely difficult to make satisfactory three-dimensional models of proteins.

In order to overcome the mentioned problem a reduced complexity folding models are being developed. The idea is to simplify the algorithms of protein folding in a variety of ways. There is a number of different approaches to ab initio protein emulation – such as:

- Lattice models – the technique is taken from polymer modeling. It algorithm uses the complete search of available space for protein folding. The problem with current approach is that it cannot represent the complex structure of the protein, thus the predicted protein structure is highly inaccurate. On the other hand such method is a lot faster than the traditional one.
- Discrete off-lattice models – current model fixes the problems that occur with simple lattice models by taking in account the complex model of the protein structure, which includes alpha-helices and beta-pleated sheets).

- The Rosetta method – current method uses a technique which minimizes the search space for fold positions using a number of methods, one of them is Monte Carlo search. This method acts better then the previously mentioned once.

- A number of other also exists.

Current method does not produce satisfactory results. But there is several areas where current applied to. Ab initio can be applied when researching the functionality of genes which have unidentified functionality and purpose. This is due the fact that those genes could have different one-dimensional structure, which will make the application of fold recognition and homology impossible. Also current method can be applied to extend and improve other two more successful methodologies (described below) of protein prediction by incorporating the methodology into their prediction algorithms. (Dylan Chivian 2003)

Ab initio is an approach that was thought to be a solution of protein folding problem. But the reality is that it is a very complex method, which requires a lot of additional research and studies to be made. Current approach is promising, but currently other two approaches make far better predictions. And it is important to mention that current approach will not be used in the software implementation and is described to show the reader all possible approaches of protein prediction.

4.3 Fold recognition

Fold recognition, also known, as protein threading, is the technique based on the idea that protein three-dimensional structures share a number of distinct folds. Whereas fold is sub-group of Structural Classification of Proteins which denotes shape similarity group of proteins. Current method is used when the homology modelling method (explained below) cannot be used to some reason. For example the protein sequence, which needs to be modelled does not have any close related known protein structures.

The fold recognition algorithm works in a way that it ‘threads’ the sequence on to possible folds which are available in the database. Then another algorithm is run in order to learn which of the folds corresponds better to the actual sequence.

Current method might be not accurate to get a good 3D structure at the end, but it still can be applied in several cases. Those cases do not require getting a perfect tertiary structure, but rather finding proteins that are evolutionary related to the initial one. Evolutionary relation of proteins is an important aspect of structural biology, for the reason that it might show the functionality of specific proteins.

There is a number of different fold recognition algorithms which differ on how they estimate the physical forces which is required in order to for a protein to fold. Another difference is the algorithms, which are used in order to perform the sequential comparison of several proteins in order to find similar folds. Additionally each algorithm might use or not use homology; or apply different programming techniques, such as dynamic programming or neural networks.

As it was mentioned before current algorithm is not perfect and cannot do perfect predictions. The reason for that the final model will lack some portions of the protein structure. Due to the fact that fold recognition algorithms do not take into account sequence regions which are not both available in the initial protein sequence in the sequence of the fold which it is being compared/aligned to.

Current algorithm’s speed depends on the approximations and simplifications that are being used on different steps of the prediction. For example the problem of finding a matching sequence can be solved in two ways. The first solution is to use algorithms, which will find perfect matches but require a lot of computational power and therefore are slow. Or use algorithms that use some kind of heuristics therefore are fast and find good but not perfect matches. (Godzik 2003) (D. T. Jones 1992)

4.4 Homology modeling

Homology modelling, also known as comparative modelling, is the algorithm, which will be used for the software development part of current project. Therefore it is crucial to understand the specifics of current algorithm, its background, advantages and drawbacks.

Current algorithm is based on the biological understanding of the protein structure and its evolution. It is based on two facts:

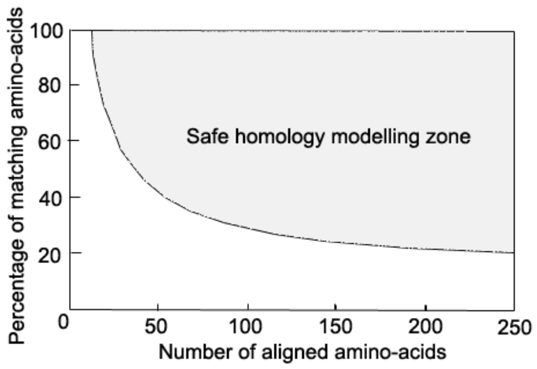
1. Primary structure specifies the tertiary sequence of the protein.
2. Dissimilar primary structures might have similar tertiary structures.

These facts mean that it is possible to take a primary structure and compare it to other structures. When similar structures are found, it is possible to compare their similarities. And based on the similarities create a predicted/target protein model.

Current method gives better results in comparison with fold recognition and ab initio. Additionally it is much more faster than ab initio. Current algorithm is the simplest one to implement. The algorithm is similar to fold recognition, but produces superior results.

There are different implementations of current algorithms, such as 3D-JIGSAW, Geno3D, MODELLER, SwissModel etc. All algorithms are available to use online, free of charge. One of the implementations is available to download and setup on a server – MODELLER. This software will be used in the software implementation of current project. It is described in detail in further sections.

Homology modeling algorithm uses large databases, which include both primary and tertiary structures of the proteins. The structures contained by the databases have been established using x-ray crystallography and nuclear magnetic resonance spectroscopy. Therefore they are accurate representations of real proteins. Two of such databases are PDB and superfamily. Superfamily database will be used in the software development part of current assignment and is described in much detail in further sections.



The databases are used in order to find similar sequences. The sequences, which have a good level of similarity and are long enough, will produce good 3D structures. There is a threshold of how long should be the length of the sequence and what should be the percentage of sequence similarity – current threshold is called safe homology modelling zone. For example if the percentage of similarity is 40 and the number of aligned residues is 150, then the produced protein model will be of good precision. Also it is important to mention that if the percentage of matching AAs is lower than 25% or the number of aligned AAs is lower than 50, then there is no chance to produce a good model. (Elmar Krieger 2003)

Depending on the used implementation protein modelling takes several steps, such as:

1. Aligning sequences and finding good templates;
2. Different type of protein structure modelling (may be divided into a number of sub-categories);
3. Structure refinement using techniques from ab initio and threading;
4. Structure validation.

(B. Contreras-Moreira 2002)

5. Software Development Plan

Current project is primarily based on understanding of protein structure – what is the process bimolecular transformation of gene sequence into protein sequence, and protein sequence into 3D protein structure. Since there is enough knowledge of how this process is done and it is described in the previous section, it is possible to start writing up software developing plan and choosing an appropriate development methodology.

It was an intention of a writer of current project to insure step-by-step modular creation of the software associated with the project. Which states that at first there must be a clear understanding of what has to be done to insure a success of current project and make sure that the project is finished in a timely matter at the specified time.

In order to do this the chosen development methodology must be applied. For current project it was decided to use *Readme Driven Development* methodology. The reason why this methodology was decided to be the best option to use for current project is described in the section below.

5.1 Introduction to Readme Driven Development

Readme Driven Development or RDD is a new approach to software development, which states that in order to start implementing the software a readme document should be written which would include the main features of the software one is developing. After the readme is written it is possible to start developing software based on readme description. Current approach is different from other project management techniques for the reason that it is based on the description of what has to be done. In is a trade of between complex development models such as Waterfall and no documented development plan. The idea is that waterfall methodology includes too much detail for small projects and too time consuming. On the other hand having to documentation is also a bed practice for any project; some specification for the requirement is an essential part of any software development project, which is bigger than making a simple utility like arithmetic calculator. In turn RDD is not overcomplicated and does include a specification that is suitable for defining the project and it's main parts.

It was found that current approach is better suited for current project for the reason that it is relatively small and its implementation will be based on different software components rather that being an entirely standalone product. Also for the reason that before writing the project there was a specification of the project, which states which, software components should be used i.e. current project should be based on MODELLER software and SUPERFAMILY database.

SupfamMod is based on SUPERFAMILY database for the reason that current project is tend to be an extension to the functionality of *supfam.bris.ac.uk* website. And basically current website is based on the mentioned database. The reason for using MODELLER software is that it is freely available to use for educational purposes and its well documented.

Additionally before the project started to be implemented there was an interim report written. During the write-up of the interim report there was an extensive research commenced which stated which software tools will be most appropriate to use with current project. The software products additional to SUPERFAMILY and MODELLER that the author decided to use were HMMER3, ASS3 and CE. The description of current software packages will be presented in the further part of current report.

Another reason why Readme Driven Development was chosen by the author of current project is that current project will be written in Python programming language, which he never used before, so it is impossible to make a precise description of the system, like in Waterfall model before starting the development. The author was learning how to use the language alongside the implementation of current project. Another feature of current project, which was unknown to the author, is the development of server side programs using programming language, which was not primarily created for creating dynamic web pages.

RDD is relatively fast to work with. It does not require using modeling techniques, such as Unified Modeling Language, SysML, FMC etc. Although in order to make sure that the Readme is fully understood, since the language might no be always expressive enough, it was decided to include Flow charts. Flow charts are relatively easy to make and they are highly readable for the people who are not common with modeling techniques and not proficient in programming.

For the reason that current system will also be available online, a decision was made to include a mockup design of the webpages, which will act as a primary input, and output facilities of the system. The design mockup will allow to have a clear understanding of how web page should look like and spend less time arranging HTML elements.

A good feature of Readme Driven Development is that there is a lot less time required to compose a "description" of the software in comparison to other techniques. And due to the fact that the time to submit the project is very limited and that the complete project must be submitted at a strictly specified time margin, it is appropriate to RDD. This insures that there will be more time for the development, research, testing and project paper write up. Additionally there are always risks that something might go wrong, for example illness or partial loss of information, which might affect the timing of the project.

Current methodology is relatively new and has been presented a few years ago. But its application is great for university type summer projects or small business projects. The Readme Driven Development is similar to Document Driven Development, but is limited to only one file. Which is also a good feature, for the reason that the created readme file can be used in to give the description of the software for the end user.

Initially the RDD is similar to information, which was provided, in the interim report of current project, but in a shorter form. Though it will be a lot shorter, for the reason that it will exclude the introduction of the subject where current software might be used. It is assumed that the user will know the prerequisites. Regarding current product it is expected that the user will know a reasonable amount of information about proteins and protein structure prediction. All in all current software product meant to be used by scientists and students, which use bioinformatics.

All in all RDD is a perfect development methodology for current project. It corresponds to the size of the project, its aim and its structure. Additionally author finds this kind of approach most satisfying to work with. It allows switching quickly from planning into the development step and make implementation decisions on the go.

It is important to mention that the RDD excludes testing description, although it will be provided in the end of current project. For the reason that testing will allow to check that the provided software is operational and at the same time it will provide results which will be used for the assessment of the performance of current system. (Preston-Werner 2010)

The next section presents the actual readme of current project, which was used in order to implement the project. Additionally readme can be used as a first step for the reader to have a chance to understand how the system works and how it might be used.

5.1 Readme

SupfamMod is a protein prediction system, which is based on Superfamily database (*http://supfam.cs.bris.ac.uk/SUPERFAMILY/*) and MODELLER software. Additionally it includes such tools and databases as HMMER3, ASS3, Astral and SCOP databases. SupfamMod is based on homology modeling (explained in section 4.4), meaning that in order to produce a three-dimensional protein prediction it requires a target primary protein structure (protein string/one-dimensional protein structure/primary protein structure) and a template protein structure – that includes primary and tertiary protein structures.

The system works as a prediction pipeline. No interference is required with the prediction process. The system is fully automatized and produces a prediction based only on primary protein structure. All components, which are included in the system, are properly interconnected and tested to insure the robustness of the system. But in case if prediction system has failed to produce a proper result, a set of triggers should be implemented in order to inform user about the error.

MODELLER is the main homology prediction mechanism of SupfamMod. It has API, which is used to insure proper prediction settings. MODELLER includes tools for template protein structure search, but they are not used for the reason that HMMER3 and ASS3 are used for template search. Astral and SCOP databases are used with HMMER3 as template databases. SCOP database is used with HMMER3 directly i.e. HMMER3 searches though HMMs in SCOP. Whereas Astral database contains 3D protein structures which correspond to template primary protein structures. SupfamMod will include a number of implementation. ASS3 is an extension to HMMER3 written by Julian Gough, and will be used as an alternative method to HMMER3 to find templates.

Superfamily database is used to get protein primary sequences and 3D protein structure names. This information is found using MySQL database, on which Superfamily database is based. The input query argument is model ID outputted by HMMER3. Then the protein sequence and Astral three-dimensional structure ID are used to feed to the MODELLER to make a prediction.

Current system is an extension to the functionality of Superfamily database web server. Therefore SupfamMod includes a web base frontend, which is available to students and scientists who which to get a protein prediction. As an input the system takes a primary protein structure and as an output it shows the 3D model, which can be viewed directly in the client's browser. Additionally the predicted structure can be downloaded in a PDB format. As an additional functionality the web interface should allow to download target structure and template structure alignments.

The software is based on Python programming language. In comparison to Perl programming language (which also was under consideration), Python language has a number of advantages – it is easily expandable, it is fast to comprehend and learn, the written code is easier to read and understand, it is usually takes less lines to write functions etc. Furthermore API of MODELLER (the main tool which is used for protein prediction) is highly integrated with Python – it is available as a number of classes, which can be effortlessly imported.

The SupfamMod includes self-evaluation system based on new proteins submitted to Protein Data Bank. The new proteins are being automatically downloaded from the pdb.org to SupfamMod's servers and evaluated. The evaluation system is based on CE software. The output is the similarity between predicted protein and the protein from pdb.org database that is determined experimentally. The prediction results are presented on the SupfamMod webpage and can be downloaded by students/scientists for further analysis.

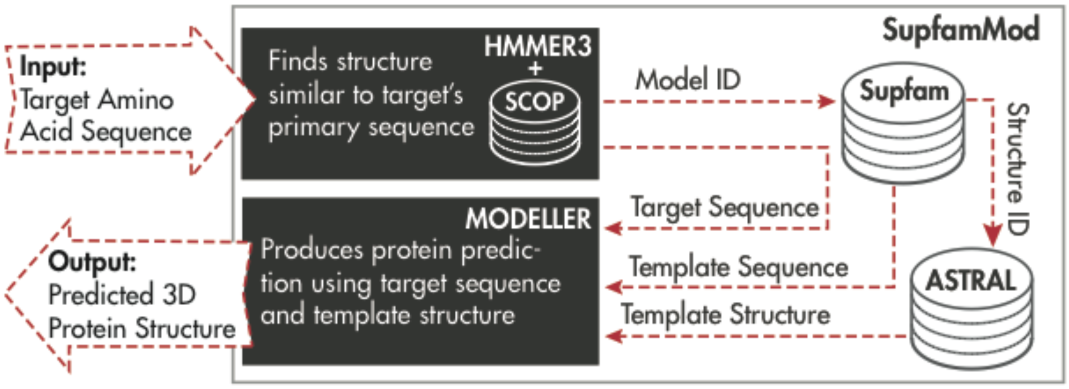
SupfamMod is located at supfam2.bris.ac.uk web server, which is based on Ubuntu 9.10, Apache/2.2.12 and Python 2.6.4. For this reason there is a restriction, which doesn't allow redirecting browser to a "parking" page before the process of prediction is finalized. This means that when the request for a protein prediction will be sent to the server using the web interface, there will be no indication of progress. In order to show the progress or indication of working prediction process AJAX technology is used. It will allow sending request to the server and at the same time showing progress screen in the client's browser. As soon as prediction is ready, the browser is redirected to the results page. Results page is a standalone html file which links to the results files (both alignment and tertiary structure).

SupfamMod is available at: *http://supfam2.cs.bris.ac.uk/SUPERFAMILY_devel/cgi-bin/max/index.cgi*. The page has a brief description of the system and how it should be used. Also it has an example of input. The input field is HTML textbox, which accepts primary protein structures in FASTA format. Below the input box there is a button that submits the prediction request to the server.

5.2 Flowcharts

There are two flowcharts presented in current section. These are additional information in order to get a better understanding of the system and how elements will interact with each other. First presented flowchart will describe how information is being processed, which software components will interact with information etc.

5.2.1 Protein prediction flowchart



Input:
Target Amino Acid Sequence
Finds structure similar to target's primary sequence
HMMER3 + SCOP
SupfamMod
Model ID
Supfam
Structure ID
MODELLER
Produces protein prediction using target sequence and template structure
Target Sequence
Template Sequence
ASTRAL
Template Structure
Output:
Predicted 3D Protein Structure

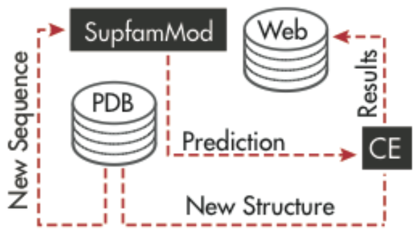
The flowchart that is presented above represents the prediction system. It shows main components, which are used in the system. The arrow on the in the top right corner designates the input. The input is target amino acid sequence. The first step of the prediction is the HMMER3 (described in detail below) software. Its responsibility is to find a primary protein sequence, which is similar to target protein sequence. There will be two variations of HMMER3 system, for the reason that there is a number of varieties of implementations of the SupfamMod. But as a basic description current flowchart shows only one versions of HMMER3. The output of current step is model ID. Model ID is an index, which points at the template protein structure in Superfamily database.

The next step is to use Model ID with Superfamily database. A query is made using model ID in order to get Structure ID of template three-dimensional structure and template primary protein structure. This step also includes three different variations. This is due to three different implementations of SupfamMod system. These will be described in detail in further section of current work.

The structure ID is then used on Astral database, which is a selection of three-dimensional structures which correspond to structure identifications kept in Superfamily database. The structure ID points into some specific directory and file located within Astral database. Current file is the template structure, which is then used for modeling.

In order to make a prediction at least three structures are required – template primary structure, target primary structure and target tertiary structures. These three structures as displayed on the flow chart are then passed to MODELLER in order to make a prediction. The output of the MODELLER is the three-dimensional protein structure. This structure is final prediction and which can be used in studies, experiments etc.

5.2.2 Automatic performance testing flowchart



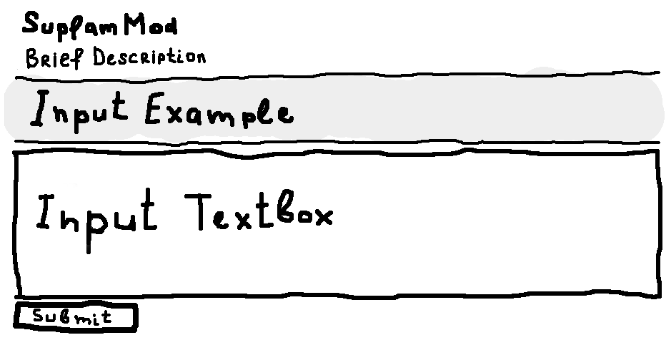
The flowchart presented on the left describes the automatic performance testing system. The idea of the system is to automatically get protein structures from Protein Data Bank and based on their primary sequences predict tertiary structures. Tertiary structures are then compared to experimentally determined structures. Their similarity is outputted as a percentage.

The first step in current flow chart the download of new protein sequence and structure. The new sequence is being inputted into SupfamMod system (the underlying processes of SupfamMod are briefly explained in section 5.2.1). The output of the SupfamMod system is the predicted tertiary structure. At this point there are two structures available – predicted and experimentally determined structure. Both of them are being compared using CE (Combinatorial Extension software). The output of this system is similarity measure – a percentage score. The outputted information is being saved on the server to be preview on the web.

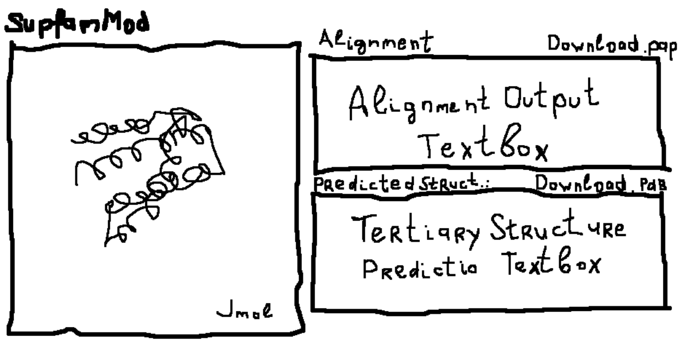
5.3 Website design mock-up

This section includes quick mock-ups of the landing page and results pages. Current section is provided as a part of development methodology. The images presented in current section give better understand of how SupfamMod web pages should look like.

Landing page includes a heading, input example, input textbox and a button for prediction request:



The results page includes heading, presentation of 3D model, alignment and tertiary structure outputs and two download links for these outputs.



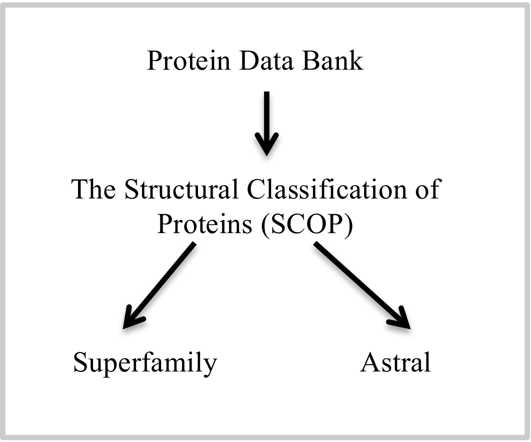
Additionally the web service of SupfamMod will include loading screen, which will notify user about the prediction process. It will be a simple page with SupfamMod heading and a line saying: *"Please wait, 3D protein prediction in progress. This page will automatically reload itself as soon as the prediction is complete. Please do not close the window."*

5.4 Discussion

Readme Driven Development methodology in conjunction with flowchart diagrams and interface design mockups is the initial description description sufficient for understanding how the system act and how different software component should interact together. Although due to complexity of the software components thorough study should take place. A better understanding of the components will allow building a system of high quality. The next section will include information on software and databases, which will be used in SupfamMod.

6. Software and Database Descriptions

Current section primarily focuses on the description of software and databases used in current project. The arrangement of descriptions is based on hierarchy of related studies. E.g. Superfamily database is based on SCOP database and SCOP database derived from on Protein Data Bank (please refer to diagram on the right). Therefore PDB will be described first, SCOP and Superfamily – second and third. Databases will be described first, for the reason that they are the basis and main containers of data required to produce predictions. Current section includes only theoretical information about the software. Technical specifics will be overviewed in implementation section of current project.

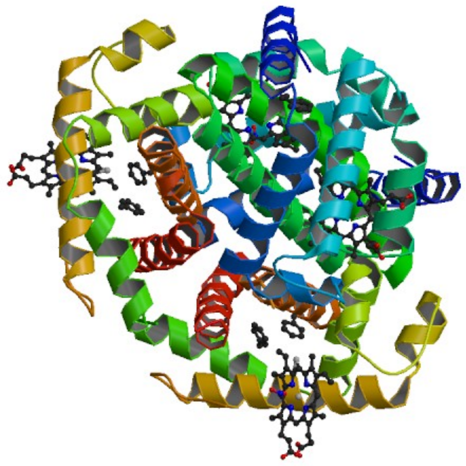


6.1 Protein Data Bank

Protein Data Bank (PDB) is the primary source of proteins structural data. It was established in 1971 as an archive for macromolecular structures i.e. data bank of molecular structures that were determined experimentally. Currently it is the largest protein archive in the world. It consists of 75694 protein structures (Protein Data Bank 2011). When the database was started it included only 7 molecules. Currently over 100 molecules are submitted to the data bank every week and the number of weekly-submitted molecules grows.

Current databank is community driven, which means that various laboratories around the world submit their structures. Most of the submitted proteins are determined experimentally using X-ray crystallography and NMR spectroscopy. Although will developments in theoretical modelling, some proteins are determined using theoretical modelling.

Each structure in Protein Data Bank has a unique identification codes that is called PDBid or PDB ID. The identification code consists of 4 Latin alphanumerical characters in length, whereas the first character is always a digit. The proteins are IDs are automatically generated, but can be adjusted by PDB team to sort proteins into specific groups. For these purposes characters 2 and 3 are used. PDBid is highly important for the reason that most scientific papers will use it to refer to specific protein structures that were used in the study. An example of PDB ID is 3ONZ, which points to 'human tetrameric hemoglobin' structure, which tertiary structure is presented on the right.



Current information is available through Internet using a number of different ways. The most common way how to get information from Protein Data Bank is to use web interface. The website is available at pdb.org. It provides all the information about submitted proteins, such as structure files, publications, molecular description, determination methods etc. SupfamMod project is mainly focused on primary and secondary protein structures. These are available in FASTA format and in PDB format (please refer to section 8).

The intention of Protein Data Bank is to provide accurate information to its users. For this reason all proteins which are submitted to the server are being checked for errors. In case of faulty submission the structure should be revised by the first party and then resubmitted. In some cases proteins may be declined for resubmission.

The data provided by PDB is searchable using a number of search forms and search methods. For example if it is required to find a structure of hemoglobin, then textbox search field is used with 'hemoglobin' as an input. Current query will return a number of available haemoglobin proteins of various species. Or it is possible to search for proteins using direct input of protein IDs.

The same data is also available through FTP, except in a downloadable parsable text files, which is convenient for various to use with external software. Additionally URI can be composed in a special way in order to reach structure or other files directly, using appropriate PDB structure IDs. More advanced methods such as RESTful and SOAP Widgets are available. They include the usage of PDB's APIs and structured queries using XML over HTTP. This approach makes it possible to compose queries in order to return very specific information. For example XML query may ask for the most recent protein structures or for structures that are of specified length.

PDB led to development of other protein resources that provide information about archive's proteins. These resources often display information a different way or summarise data according to some specific parameters. For example SCOP and Superfamily are both based on proteins submitted to Protein Data Bank. Additionally many bioinformatics tools use PDB resources to get missing protein information 'on the fly'. It is an intention of PDB to be flexible and provide information in various forms. This makes PDB the most popular resource for protein structures with 1 protein downloaded every second 24 hours a day and 7 days a week.

Current database will be used directly in the evaluation phase of the software implementation. PDB will be used to fetch a number of proteins primary and tertiary structures, then

SupfamMod will be used to predict the three dimensional protein structures based on downloaded PDB primary structures. The predicted 3D structure will be compared to original/experimentally-determined structure from PDB.org in order to establish similarities. Current procedure will be executed three times with each implementation of SupfamMod, in order to choose an implementation with the best performance. Please refer to evaluation section of current paper for more information.

6.2 SCOP

The Structural Classification of Proteins (SCOP) database includes protein structures classified and ordered as a hierarchy, according to structural specifications of proteins and their biological functionality. SCOP includes proteins of known three-dimensional structures which where determined experimentally using X-ray crystallography and NMR spectroscopy. The database was created in 1994 and was based on approximately 3000 entries. (Murzin, et al. 1995) Currently SCOP is based on 38221 proteins. (Structural Classification of Proteins 2009) The number of proteins has increased over 12 years 10-fold. Current version of SCOP database is out-dated and does not include all proteins available in various protein databases.

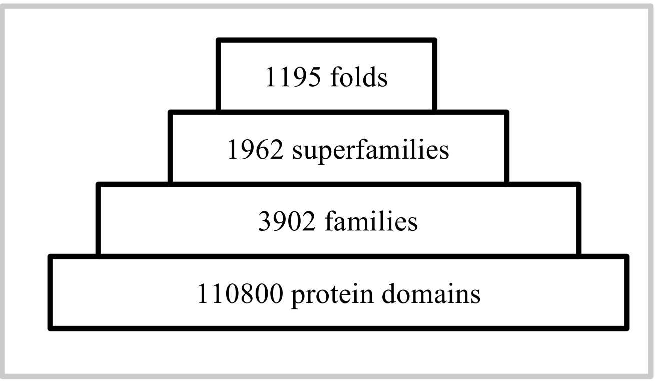
Primarily SCOP uses Protein Data Bank proteins entries for classification. New PDB entries are processed and included in new versions of SCOP databases. The protein entries used for database refinement include primary, secondary and tertiary structures. This is essential in order to analyse and classify them into groups. Classification of the proteins is done using a number of tools, which utilize algorithms and techniques for automatic structural inspection, statistical analysis and similarity establishment between known structures. The tools used in database updates have been modernised over the years in order to cope with growing amount available structures and to facilitate new knowledge about protein structure classification.

Current database is created to provide evolutionary relationships between proteins which structures are known. I.e. database is based on homology of protein structures and the functions. Knowledge of how different proteins are related to each other sheds the light on their functionality and evolution. This knowledge is important for explanation of protein sequences determined using gene discovery in various genome projects. Additionally current database may be used for development of new databases such as superfamily and astral, in order to facilitate more information from the given proteins structures. In case with SupfamMod SCOP will be used in order to find template sequences that are homologous to target sequence.

The Structural Classification of Proteins acts as an interface to Protein Data Bank and provides fold-level, superfamily-level, family-level and domain-level classification. These levels are based on different degrees of similarity between proteins. Currently there are 1195 folds, 1962 superfamilies, 3902 families and 110800 domains. The classification is based on 38221 proteins. (Scop Classification Statistics 2009) The number of domains is smaller than number of proteins for the reason that larger proteins might include a number of domains which are processed separately.

Protein domains are protein structure that is independent and function on its own. Proteins can consist of a number of domains. Therefore a single domain can be a standalone protein.

Family includes proteins with at least 30% protein primary sequence



similarity. But if sequence similarity is under 30%, but function and three-dimensional structure is very similar, the proteins still fall under family-level classification.

Superfamily includes proteins with low sequence similarity. But their common functionality implies that they might have common ancestors.

Folds include proteins of with common secondary structures (please refer to section 3.3) and their arrangement is similar.

(Andreeva, et al. 2008)

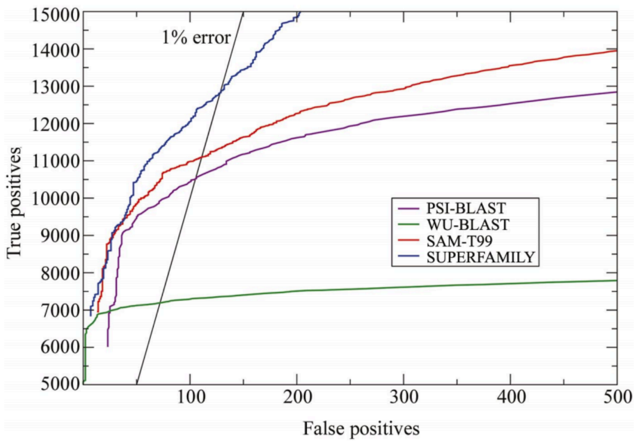
Current database will be used as part of Superfamily database. There will be no direct interaction with SCOP, but rather its variation that is available as a part of Superfamily database, which includes Hidden Markov Models of families and super families presented in SCOP database. For more information please refer to section 6.3.

6.3 Superfamily

6.3.1 General Description

Superfamily is Hidden Markov Model for SCOP superfamilies. It derived directly from The Structural Classification of Proteins database. Superfamily represents all known fully sequences genomes. I.e. genomes consist of a number of genes that encode protein primary structures (please refer to section 3.3), these genes were determined and their tertiary protein structures were experimentally determined. The database contains 60 fully sequences genomes and their sequences, which were analysed and processed in order to derive HMMs.

HMMs of Superfamily derived from using proteins of a superfamily level, aligning them together and running Hidden Markov Model profiling algorithms on them. This technique allows finding proteins of distant evolutionary relationships. The main point of creation of Superfamily database is to apply available data in various projects that require finding homologous proteins against some protein targets. The approach used in Superfamily is superior to other databases or techniques. The figure presented on the right shows performance comparison Superfamily compared to other homology detection software products (J. Gough, The SUPERFAMILY database in structural genomics 2002). The presented information represents the ability of 4 major algorithms to correctly classify superfamilies to domains/proteins from SCOP database. After classification data was obtained, it was compared against actual data from SCOP. The accuracy of Superfamily is at 99% at detecting superfamily assignments (J. Gough, The SUPERFAMILY database in structural genomics 2002). Supefamily performance is perfect with SCOP database, but application to real world superfamily-level sequence classification shows inferior results.



Superfamily database was tested against proteins available in CASP[28] test. Current tests includes proteins with BLAST e-values score greater than 0.1. Current score means that the protein sequences are of high complexity. The results showed that the accuracy of Superfamily was at 23% margin. Which means that if a primary structure of a complex protein is required be assigned to a SCOP level superfamily, then the results might be lower than expected.

Primary designation of Superfamily database in current project is to find homologous sequences of target protein sequences. This is done using provided functionality, which runs protein sequence against available HMM models and returns Model ID of most closely related sequence. The model ID then can be used to fetch corresponding primary and tertiary structures. Current structures then can be applied to homology modeling software (please refer to section 6.5).

The database provides a number of tools to utilize its full functionality and resources. The easiest way to access Superfamily is through usage of Superfamily public server. It is available at *http://supfam.cs.bris.ac.uk/SUPERFAMILY/*. It provides a graphical user interface to run sequence search, keyword search, get genome statistics etc.

The functionality that is required for SupfamMod project is available through downloadable Superfamily tools. These include MySQL database, ASS3 Hidden Markov Model alignment software; SCOP database and HMM files to use with HMMER3 and ASS3. All of these are available at Supefamily website. These tools are required to be installed on the server, which will run SupfamMod.

6.3.2 MySQL data

The MySQL database is divided into 4 groups:

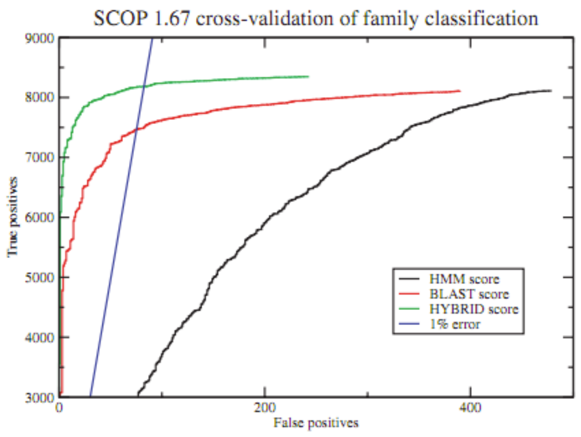
- *Genomes and sequences* – includes 3 tables. It includes information about genomes, such as genome name, domain information, taxonomy, and web links. Also it includes full genome sequences.
- *Domain statistics* – includes 4 tables with various domain assignment statistics. Such as average length, family lengths, quantity of families, number occurrences of domains in genome etc.
- *Protein domains* – includes 4 tables with information about superfamily level classification of available domains.
- *SCOP and model* – includes 5 tables with which contain information about SCOP level assignments, their primary sequences, model IDs and SCOP domain entry IDs.

The most important tables for SupfamMod project are:

- *Des* – contains information about three-dimensional structure of the model or superfamily. It will be used to fetch 3D structures of corresponding model IDs or SCOP domain entry IDs.
- *Model* – it will be used in relation with *des* table in order to find corresponding or SCOP domain entry IDs.
- *Pdb_sequence* – contains protein primary sequences of corresponding model IDs or SCOP domain entry IDs. This data will be used as template primary protein sequences.

(Gough, Karplus, et al., Assignment of homology to genome sequences using a library of hidden Markov models that represent all proteins of known structure 2001)

6.3.3 ASS3



Current tool is used for finding homologues using Superfamily and SCOP databases. Current tool is an extension to HMMER3 software (please refer to section 6.6). The advantage of ASS3, is that it is hybrid method which uses both profile methods (HMMER3) and sequence alignment methods (BLAST). The combination of these two methods allows finding template sequences with better accuracy. The figure presented on the right, shows the performance of three methods. The experiment tested SCOP database sequences in order to classify them according to supefamily level. Moreover current method does not require additional CPU time for finding homologous proteins. Which means that the classification quality increases with no additional cost.

ASS3 hybrid outputs seed ID instead of many model IDs (as in HMMER3). The seed ID (or PX in MySQL database) can be used in order to return template sequence and protein three-dimensional structure. (J. Gough, Genomic scale sub-family assignment of protein domains 2006)

6.3.4 SCOP and HMM database files

A number of database files is required to be downloaded for ASS3 and HMMER3 work:

1. hmmlib_1.75 – contains all Hidden Markov Models for SCOP;
2. self_hits.tab – contains self hits i.e. primary protein sequences which correspond to a number of superfamily level classifications;
3. dir.cla.scop.txt – contains family level classification of seed IDs;
4. model.tab – contains model information, such as tertiary structure IDs and short descriptions;
5. pdbj95d – contains model sequences and their IDs.

File 1 is required for HMMER3 to find model ID for some target sequence; files 2-5 are compulsory for ASS3 to output PX identifications for target proteins. Essentially the presented data is available using provided MySQL database. But in order to improve performance of the system the data provided in parsable text files.

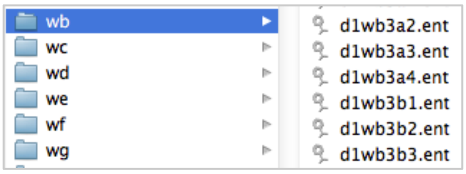
(J. Gough, Download SUPERFAMILY Models, Database Dump and Genome Assignments 2010)

6.4 ASTRAL

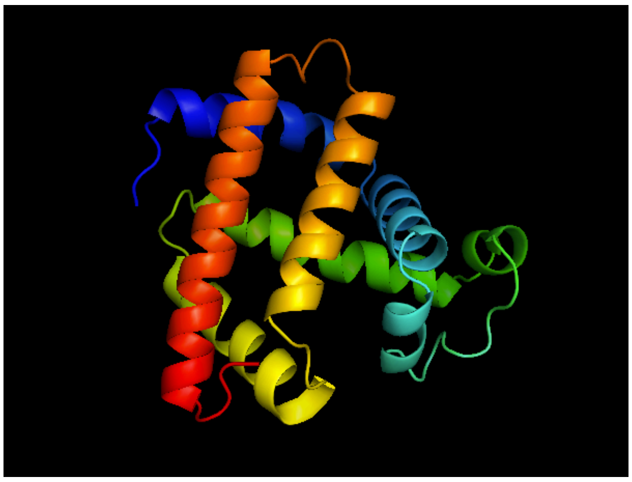
Astral database is based on SCOP. It is an archive of tertiary protein structures that correspond to domains in SCOP database. Current database will the main source for template 3D structures of proteins. Model IDs, which will be outputted by tools provided by Superfamily database, will be linked to ASTRAL structure IDs.

ASTRAL database is a unique collection of three-dimensional structures, which are manually determined and analyzed. It provides protein structures at all levels of SCOP classification, thus rendering it a perfect tool for getting homologous structures which evolutionary relations are close and distant.

Most three-dimensional structures in ASTRAL database correspond to specific regions of sequences. Thus ASTRAL sequence IDs is identical to SCOP domain sequence IDs. Otherwise IDs include additional characters, which point at partial domain sequences. An example of sequence ID is *'d106ma_'* which corresponds to '*a.1.1.2'* SCOP superfamily ID. An example of such folder is presented in the figure on the right. (Chandonia , et al. 2004)



*d106ma_* sequence has an corresponding three-dimensional file, which can be fetched from the database. All the structures are kept as PDB files, which can be used in homology modelling or visualized with protein visualization software, for example with PyMol. The visualization of *d106ma_* is presented on the right.



In order to use ASTRAL database with current project's implementation it has to be downloaded to the SupfamMod server. These files are available at: *http://astral.berkeley.edu/pdbstyle-1.75.html*. After unpacking 118 directories will be created. Each directory will contain a number of structures. The directories and files located in them have a specific style of names. For example if it is required to download *d1****06****ma_* sequence, folder named '06' should be located. I.e. 3rd and 4th characters of protein identifier are related to the folder name. This archive style allows easily finding protein sequences by hand and also increases the performance of computer's filing system. (Chandonia, et al. 2009)

6.5 MODELLER

6.5.1 Introduction to Homology Modelling

Five separate steps are required to be performed in order to produce a protein 3D structure out of a protein primary sequence. Each separate part particulars depend on the implementation of the MODELLER and its settings. Current section will describe a general approach to protein prediction using modeller by describing each one the five steps. The large part of the step description is happening within the modeller system and governed by its settings.

1. Finding protein templates which relate to target protein

This step is involves finding one or a few proteins, which are closely related to the initial protein. Current search is done using tools provided by MODELLER or using external tools. The search is usually done on Protein Data Bank database using Basic Local Alignment Search Tool (or BLAST) or other alignment techniques. In the final implementation of current project, the search will be done using Superfamily database that is located on of the University of Bristol servers. Though current step can be done manually by providing

MODELLER with some specific protein to build a protein tertiary structure upon, but in the case with current project, current task will be fully automated.

2. Selecting Protein Template(s)

Current step involves selecting one protein or a sequence of proteins out of the number of potential proteins that were provided by the previous step. The better prediction of protein structure is achieved in case if a protein or a group of proteins are chosen with the best alignment scores and primary structure similarity. Similarity score indicates the similarity between two proteins. The simple scoring algorithm is based on how many residues are similar in both proteins and how big are gaps in the alignment. Additionally scoring algorithms can be more complicated, depending on what kind of protein structure is predicted, what kind of environment is it in or if it is paired with some chemical element. It is important to mention that MODELLER will make a better prediction in case if a group of proteins if provided for the prediction.

3. Template-Target Alignment

Current step involves the alignment of the target protein and the template proteins. There are two types of alignments that can be made – pair-wise alignment and multiple-alignment. Pair-wise alignment is performed when there is only one template sequence and multiple-alignment is executed when there is more than one template sequence. The quality of final prediction tends to be better using multiple-alignment. This is due the fact that having a number of template sequences gives more information about the target and template sequences, such as their structural relation. The alignment tends to reduce a number of errors, which appear due to poor similarity. Poor similarity is the case when the fraction of similarity is lower than 30%-40%.

The alignment may be performed by the tools, which are provided by the MODELLER software, it can be done by other external tools or manually (in cases of poor target-template similarity). The intention of current project is to make the system automatically make the alignment system without the interference of external tools and users. The MODELLER API allows making the prediction using the default settings. It is also an objective to investigate and use its settings in order to improve the accuracy of predictions.

4. Building 3D Model

Current step performs all the required calculations in order to build the required tertiary structure of the provided protein primary sequence. The success of current step depends on the quality of selected template sequences and their alignment with the target sequence. Building the 3D MODELLER is a process based on two steps. The first step involves the software's algorithm to use the template's 3D model's spatial restraints and relate them to the 1D sequence. Then current information is being applied used by MODELLER's algorithm, which builds tertiary models using molecular dynamics and other structure optimisation methods.

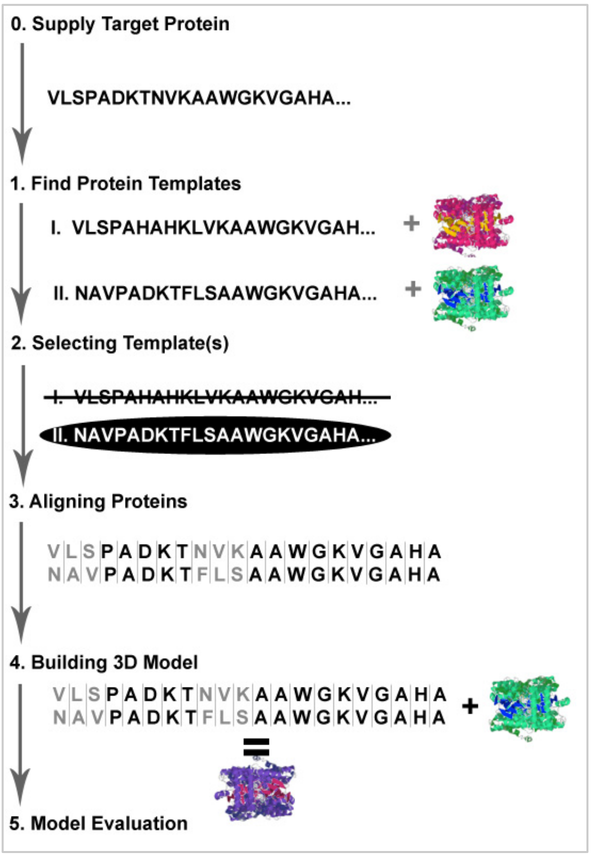


Diagram 2. MODELLER Protein Prediction Process

6.5.2 Software Description

Modeller is the software, which allows performing homology protein predictions. Current software will be the main software component for protein prediction. Modeller is a complex application, which includes a large number of different algorithms and approaches to homology modelling of proteins. The software is highly flexible; it can be used with numerous protein databases, webservers, operating systems and protein sequences. Current features are supported by MODELLER's application programming interface (or API).

The programming interface allows interconnecting different applications, servers and databases with MODELLER. Additionally it allows tuning the software according to final software product specifications and requirements. In order to do that the final product is being paired with predicting software's functions by the means of the mentioned API. It will allows the programmer making an unique software product, which will predict proteins governed by set of specific rules, obtain the requests for protein prediction in specific ways, show prediction results in a particular fashion, automate the prediction process etc.

For example current project requires creating a web application, which will follow a number of specific steps in order to produce the protein. The steps that application will take will depend on what task is being requested by the user (such as a student requesting to predict a protein) or by some external system (such as benchmark servers). In case with benchmark servers the software should include steps, such as:

- Listening for protein prediction request;
- Receiving protein prediction request;
- Resolving the request by performing another set of sub-steps (described in in previous section);
- Sending the prediction;
- Receiving the prediction accuracy/results.

All things considered MODELLER is not just simple 'plug-and-play' software, it is full-blown platform that enables scientists to create prediction servers/software for their specific needs. As for current project, MODELLER provides all the required functionality to satisfy to aims of the project. Additionally it has a lot of potential for the future refinement of current project and will provide a lot of possibilities to improve the prediction by using provided settings.

6.6 HMMER3

HMMER is software based on Hidden Markov Model profiling method for finding homologous structures. It does this by comparing a target sequence to HMMs available in the database. The output of the application is a list of homologous proteins, which are sorted by score. Usually the protein model with the best score is chosen for protein prediction. Current approach will be used in one of the implementations of SupfamMod software.

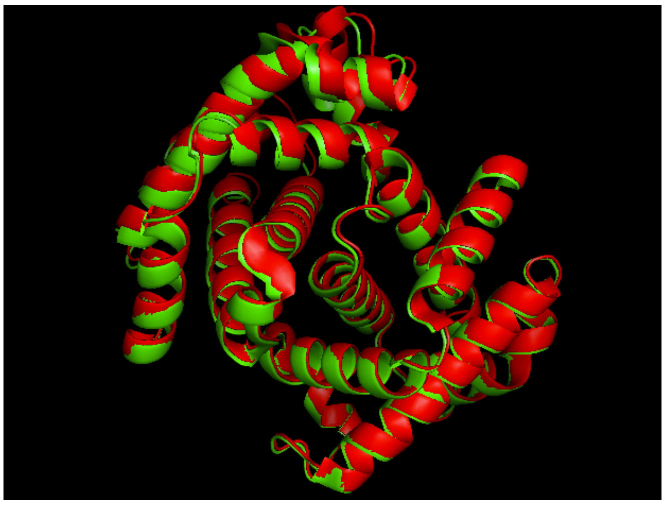
The method used in HMMER3 and was proposed by *Krogh et al.* (1994a). Although HMM models were used in other fields of data mining e.g. speech recognition, Krogh proposed a suitable method to represent amino-acid sequences. This method involves "insert" and "delete" states from residues. I.e. amino-acid characters can be inserted, kept or deleted from the sequence with a specific probability.

Current software has been used in order to build HMM databases, including Superfamily database. Hmmlib file (described in section 6.3.4) was created using current software. For this reason current tool is an ideal instrument for finding homologous structures. The model ID that is outputted by HMMER3 will correspond to information available in both MySQL and plain text databases. This allow straightforwardly interact with available structures and use the data as template proteins for protein prediction.

The ability to of HMMER3 to work with databases allows it to find a number of domains in a single protein structure and output a number of plausible model domain IDs. Current facts allow scientists to manually review a number of possible protein structure models. And in case with SupfamMod a number of template sequences can be used for protein prediction, which in theory should increase the quality of protein prediction.

6.7 Combinatorial Extension (CE)

Current software will be used in the evaluation part of current project and in self-evaluation system of SupfamMod. Current algorithm was chosen for protein comparison due the fact that it provides a number of advantages against other methods. The first advantage is speed that is provided by CE. Due to specifics of the algorithm it reduces the search space and automatically establishes target function.



Visualisation of protein pair-wise protein alignment of Ferric R-state human aquomethemoglobin: *3P5Q* (red) and Human tetrameric hemoglobin: proximal nitrite ligand at beta: 3ONZ (green) with 95% similarity

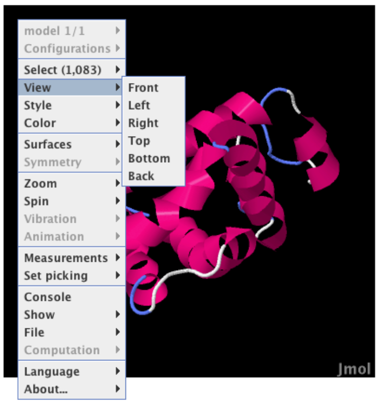
Although other algorithms may provide similar performance and functionality, current software was chosen due to the reason that it is freely available, can be installed on Linux, the command line interface is easy to comprehend and requires only 5 arguments. Furthermore there is no specific solution to similarity measurement/three-dimensional structure alignment problem. Different approaches may produce dissimilar results, for the reason that they use diverse heuristics for alignment. The alignment quality specific method/algorithm to some extend may have subjective interpretation.

Overall results show that specific methods like Dali and VAST produce results similar to CE results. Differences occur when protein structures have low structural similarity. Additionally similarity metrics are dissimilar in different alignment software and are not comparable. Therefore for current project it is important to use only one alignment software for similarity measurement.

The output of CE produces statistical information about the alignment. Such as length of aligned positions, root mean square deviation (or RMSD) (similarity significance value), Z-score (similarity significance value, e.g. scores over 4.5 designate family level similarities, lower than 3.7 low significance similarities), number of not aligned positions, sequence identity percentage for both protein tertiary structures. Current data will be used to build a statistic database based on predictions. (Ilya N. Shindyalov 1998)

6.8 JMol

JMol is protein three-dimensional structure visualisation tools, which will be used on the webpage of SupfamMod. It will display the predicted protein three-dimensional structure directly in Internet browser's window. The advantage of current software is that it is free to use and modify (released under GNU Lesser General Public License). JMol is written in Java, so it can run on any machine, which has an installed Java Virtual Machine.



Current software allows to present visualisation of a very high quality using anti-aliasing algorithm. Additionally JMol software has a 'cartoon' visualisation of proteins, which allows to she helixes and stands.

(Herráez 2006)

7. File Types

Current section describes file formats that are used in bioinformatics for storing one-dimensional and three-dimensional amino-acid structures. File formats, such as FASTA, PIR, PAP and PDB will be used in SupfamMod. It is important to understand the structure of these formats, in order to use them properly. The files of mentioned formats need to be parsed and converted in order to make software components working.

7.1 FASTA

FASTA is the text-based format that is commonly used format for storing one-dimensional amino-acid (protein) and nucleotide (DNA) sequences. SupfamMod uses FASTA only to store protein sequences. An example of FASTA file content:

```
>2RDN:A|PDBID|CHAIN|SEQUENCE
GSHMTNVTGDYTDCTPLLGDRAALDSFYEEHGYLFLRNVLDRDLVKTVAEQMREGLVALGAADPHATLEELTIDSFESVD
EVAMHDYVKYDAFWNNPSTIKVFEQVFGEPVFVFLSTTIRYYPSQAGSEEPSFHYLTPFHQDGFYIGPNQDFRTFWIPLI
RTTRESGGVALADGSHRRGKRDHVLNESFRRFGHPVRGIPPTEVSEDEHLLHSPMEPGDILLFHAHMCHKSIPNLSKDPR
LMRMSMDTRVQPAKSHRGFNAMTPWTESAKDASKGIMAKITGTPTDVE
```

The first line of sequence encoded with FASTA format always starts with "greater than sign" ('>'). The subsequent characters of the same line include a short description of the sequence. In current example 2RDN is the PDB ID of the sequence. 'A' denotes the protein chain (proteins might consist of a number of chains/protein structures). The following |PDBID|CHAIN|SEQUENCE explain the structure of the file, which includes PDB ID, chain ID and actual protein sequence.

The next lines denote amino acids (AA) that construct the protein. Each line is up to 80 characters long. There are 20 different AAs in protein alphabet (please refer to table presented on the left:

A - alanine	P - proline
B - aspartate	Q - glutamine
C - cysteine	R - arginine
D - aspartate	S - serine
E - glutamate	T - threonine
F - phenylalanine	U - selenocysteine
G - glycine	V - valine
H - histidine	W - tryptophan
I - isoleucine	Y - tyrosine
K - lysine	Z - glutamine
L leucine	

(Pearson and Lipman 1988)

7.2 PIR

Current format is similar FASTA, except of a number of slight differences. An example of PIR file contents:

```
>P1;1DK2
sequence::     : :     : :::-1.00:-1.00
SLSAAEADLAGKSWAPVFANKNANGDAFLVALFEKFPDSANFFADFKGKSVADIKASPKLRDHSSTIFTRLNEFV
NNAANAGKMSAMLSQFAKEHVGFGVGSAQFENVRSMFPGFVASVAAPPAGADAAWTKLFGLIIDALKAAGK*
```

Current format also starts with greater than sign. Which is followed by ‘P1’ which is a default code, which encodes ‘Protein1’. This is followed by PDB ID. The next line is for protein description, but might be left with default/null info, as in example. Following lines is the primary protein structure string followed by asterisk (‘*’).
(Sali, alignment.append() -- read sequences and/or their alignment 2011)

7.3 PAP

Current file describes the alignment of primary protein sequences. Every block of protein sequence of 56 characters includes 4 lines. The first line shows position indexes of AAs. The seconds and third lines, displays protein names and aligned sequences. The 4th line shows if every two vertical amino acids are matched, which is labelled with an asterisk or space. Example of PAP file contents (in current example the second protein has a random name):

```
_aln.pos                     10        20        30        40        50
d1mbaa_               SLSAAEADLAGKSWAPVFANKNANGLDFLVALFEKFPDSANFFADFKGKSVADIKA
g354Do8xNhqvorl7c1zh  SLSAAEADLAGKSWAPVFANKNANGDAFLVALFEKFPDSANFFADFKGKSVADIKA
 _consrvd             *************************  *****************************

 _aln.pos               60        70        80        90       100       110
d1mbaa_               SPKLRDVSSRIFTRLNEFVNNAANAGKMSAMLSQFAKEHVGFGVGSAQFENVRSMF
g354Do8xNhqvorl7c1zh  SPKLRDHSSTIFTRLNEFVNNAANAGKMSAMLSQFAKEHVGFGVGSAQFENVRSMF
 _consrvd             ****** ** **********************************************

 _aln.pos                  120       130       140
d1mbaa_               PGFVASVAAPPAGADAAWTKLFGLIIDALKAAGA
g354Do8xNhqvorl7c1zh  PGFVASVAAPPAGADAAWTKLFGLIIDALKAAGK
 _consrvd             *********************************
```

(Sali, alignment.append() -- read sequences and/or their alignment 2011)

7.4 PDB

PDB format describe the three-dimensional structure of macromolecules. The information in the PDB file includes the arrangement of atoms and description of the protein. PDB is a plain is a plain text file with specific structure. The arrangement of atoms is presented with XYZ coordinates. And description of the protein includes many fields such as authors, titles, determination technique, biological origin etc. An example of PDB contents are presented below:

```
ATOM      1  N   SER     1     -74.038 -49.493 -26.590  1.00 62.33           N
ATOM      2  CA  SER     1     -73.717 -48.957 -25.246  1.00 62.33           C
ATOM      3  CB  SER     1     -74.940 -48.216 -24.684  1.00 62.33           C
ATOM      4  OG  SER     1     -74.671 -47.722 -23.382  1.00 62.33           O
ATOM      5  C   SER     1     -73.344 -50.045 -24.296  1.00 62.33           C
ATOM      6  O   SER     1     -73.557 -51.225 -24.564  1.00 62.33           O
ATOM      7  N   LEU     2      -72.751 -49.661 -23.153  1.00 95.97           N
ATOM      8  CA  LEU     2      -72.381 -50.632 -22.174  1.00 95.97           C
ATOM      9  CB  LEU     2      -71.002 -50.392 -21.534  1.00 95.97           C
ATOM     10  CG  LEU     2      -69.837 -50.576 -22.525  1.00 95.97           C
ATOM     11  CD1 LEU     2      -69.930 -49.571 -23.682  1.00 95.97           C
ATOM     12  CD2 LEU     2      -68.479 -50.540 -21.806  1.00 95.97           C
```

Current example includes only a small part of the initial PDB file that is 1000 lines long. Most variety of the lines in the original file represent XYZ coordinates. Current example is a part from a predicted three-dimensional structure created by SupfamMod system.

The first column denotes the line record type, which in current case is atom coordinates (ATOM). Next column represents atom indexes. Following column determines atom name (e.g. N, CA, CB, OG). SER and LEU are residue names (please refer to section 7.1) and identical to AA in the target primary sequence file. 1 and 2 are residue sequence number (i.e. AA in primary protein structure). Next three columns are XYZ coordinates. The last three columns determine occupancy, temperature factor and charge of the atom.

(Protein Data Bank Contents Guide: Atomic Coordinate Entry Format Description Version 3.30 2008)

8. Implementation preparation and overview

Current section is an overview of the implementation phase of the project. Furthermore it includes an overview of the tools used in the development, server configuration and programming paradigm. Current section is an introduction to the implementation process and covers information, which was not included in previous sections.

8.1 Implementation Overview

The implementation phase of current project is divided into a number of steps. Each step will be described in current section. The implementation will be based on Readme (please refer to section 5.1) and Flowcharts (please refer to section 5.2). The information provided in the previous steps was theoretical. The implementation steps will require technical knowledge of the used tools. For this reason technical information will be described. The steps, which will be described in current section, are presented below:

1. Integration with HMMER3;
2. Integration with Superfamily MySQL database;
3. Integration with MODELLER
4. System refinement 1;
5. System refinement 2;
6. HTTP implementation.

8.2 Development tools

Current section gives a brief description of development tools that were used in order to implement the project's software. The author of current project strongly believes that choosing proper development tools lead to completion of software development in timely manner. Additionally the selection of the appropriate tools took a reasonable amount of time, which included the investigation of available solutions and their installation on the workstation.

The development of the system was carried out using Mac OS X system, which imposed restrictions on the range of software, which could be used. Additionally it was decided to carry out the development remotely, without the setup of the server-side software on the workstation. This decision was made due the fact that the setup running on the server and workstation will be slightly different, therefore changes will be required to be made to run the software on the server instead of workstation. Which would require additional testing of all SupfamMod functionality.

The requirements for the development tools are:

- Enable connection between workstation and server;
- Create/edit/delete files;
- Run bash commands.

The connection to the server was realized using two tools:

1. Terminal + SSH protocol;
2. Transmit + SFTP protocol + Textmate.

The first solution was used in order to run bash commands, change permissions, delete files etc. Another solution, which would suit the requirements, would be connecting using remote desktop system (such as Nomachine). But this solution required additional time to setup the appropriate software. Furthermore it requires a fast and stable Internet connection, which is not always available.

The second solution was used to edit files in familiar GUI environment. Transmit is an application which allows to mount SFTP server as a local disk drive. MacFuse (macfuse 2008) was considered as alternative software with similar similarity. Unfortunately it was not stable and slow – it would crash and file upload/download process took considerable amount of time. Textmate text editor was chosen for its ability to highlight the Python and Perl syntax. Additional feature of current software is ability to create a project file, which would automatically link all files of the developed software and show them in a single window using tabulation.

8.3 Server Side Configuration

Server side configuration involved the creation of the account for the SupfamMod system and creating appropriate folders with correct permissions. The first task was done with the help of the administrator of the server, since the author didn't have all the permissions for the server. After permissions were granted, the server could be accessed using SSH and SFTP.

In order to start developing the software, two main folders were created:

- /var/www/html/SUPERFAMILY_devel/max/ - current folder includes HTML files, protein structure files and other files created by SupfamMod. Current files will be accessible using clients' browsers. Current folder was made writable by CGI scripts run from /var/www/html/SUPERFAMILY_devel/cgi-bin/max/ and executable using Apache services (i.e. accessible using browser);
- /var/www/html/SUPERFAMILY_devel/cgi-bin/max/ - current folder contains all executable files, python classes, temp folders and libraries. Current folder was made writable by CGI scripts and executable by Apache services.

Server did not require any additional configuration. This is for the reason that the server was already up and running for the development of Superfamily. The next sections, which describe the implementation of different classes, will include some additional server configuration information. Such as modeller class implementation, which will require installation of MODELLER software on the server.

8.4 OOP/Classes

SupfamMod is implemented using OOP. This is for the reason that OOP has a number of advantages, which will be utilised in current project. For the reason that SupfamMod consists of number third party software solutions, it is logical to write separate classes that will utilise the use of each software component. For example one of the classes will utilise MODELLER software and the other class will connect Superfamily database.

The infrastructure of the classes will be simple: ‘main’ class will control all the classes; class specific data will be encapsulated and controlled by the using specific subroutines. Current approach allows arranging modular implementation of the required functionality. Furthermore current methodology allows at each step of implementation to focus on specific functionality.

Additionally OOP enabled implementation, will allow easily make changes to the implemented software, which is important, as for SupfamMod will include three implementations. Each implementation will be refined, in order to make predictions using different approaches.

No class diagram is presented in current project due to Readme Driven Development model, which is used in current project. Class descriptions will be given in the following sections. Additionally the sections will describe more technical information about the third party software, their inputs and outputs. All in all there will be 5 files:

- ‘predict.cgi’– instantiates and controls all other classes (three different ‘main’ classes will be created, each representing separate implementation);
- ‘hmmclass.py’ class – controls HMMER3, parses its output;
- ‘htmlclass.py’ class – outputs HTML code through CGI;
- ‘pdbclass.py’ class – controls connection to Protein Data Bank;
- ‘supfamclass.py’ class – controls connection to Superfamily;
- ‘modelclass.py’ class – control MODELLER via API.

9. Implementation

Current section describes the process of SupfamMod implementation. Current section describes the input of the author in building SupfamMod system, except of the research (which is described in previous sections). Current section is divided into a number of sections, where each of them describes implementation of specific class or alteration of the class. Each implementation section describes the main points of the implementation.

Additionally to implementation of initial SupfamMod system, the self-evaluation system implementation is described (please refer to section 5.2.2). Self-evaluation system uses new submissions to Protein Data Bank database in order to make 3D protein prediction based on primary protein structures and respectively compare predictions with accurate experimentally determined structures.

It was decided to implement current feature instead of meta-server connection (which allowed evaluate the performance of SupfamMod using external evaluation system). Two main meta-servers existed – BioInfo (http://www.bioinfo.pl) and EVA (http://www.pdg.cnb.uam.es/eva/). Unfortunately these two servers do not accept new protein prediction servers: BioInfo.pl is the process of being shut down and EVA is not operational.

9.1 hmmclass.py

Current section describes the implementation of the 'hmmclass' class, which controls utilizes HMMER3 (please refer to section 6.6) software. Current class is first to be implemented for the reason that it is the first process/black box in the flow chart presented in section 5.2.1. The main function of 'hmmclass.py' is finding homologous protein structures.

In order to facilitate the functionality of HMMER3 it is required to understand how to use it. HMMER3 was successfully installed on the server using binaries provided on the official website. Current software is free to use for educational purposes. (Finn and Clements 2010) Aditionally HMMER3 requires a library with protein Hidden Markov Models, for these purposes Superfamily database - 'hmmlib' is used (please refer to section 6.3.4).

An example of terminal command to run HMMER3:

```
mv0533@supfam2:/var/www/html/SUPERFAMILY_devel/cgi-bin/max$ ../hmmscan ../../db/hmmlib
3ses.fasta
```

, where '`../hmmscan`' is the path to the executable file, '`../db/hmmlib`' is the location of the HMM library and '`3ses.fasta`' is the location of target primary structure in FASTA format. The output example is presented below:

```
# hmmscan :: search sequence(s) against a profile database
# HMMER 3.0 (March 2010); http://hmmer.org/
# Copyright (C) 2010 Howard Hughes Medical Institute.
# Freely distributed under the GNU General Public License (GPLv3).
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# query sequence file:             3ses.fasta
# target HMM database:             ../../db/hmmlib
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Query:       3SES:A|PDBID|CHAIN|SEQUENCE  [L=372]
Scores for complete sequence (score includes all domains):
   --- full sequence ---   --- best 1 domain ---    -#dom-
    E-value  score  bias    E-value  score  bias    exp  N  Model    Description
    ------- ------ -----    ------- ------ -----   ---- --  -------- -----------
      2e-97  326.2  14.0     2.2e-97  326.1    9.7     1.0  1  0045538
Periplasmic binding protein-like II
    7.4e-84  281.5  12.7     8.5e-84  281.3    8.8     1.0  1  0044496
Periplasmic binding protein-like II
    7.4e-79  265.5  13.7     8.5e-79  265.3    9.5     1.0  1  0037070
Periplasmic binding protein-like II
    5.9e-70  236.5  13.8     7.2e-70  236.2    9.5     1.0  1  0046627
Periplasmic binding protein-like II
      2e-68  231.0  10.1     2.5e-68  230.7    7.0     1.0  1  0045575
Periplasmic binding protein-like II
    7.7e-58  196.8  14.2     8.7e-58  196.6    9.9     1.0  1  0047899
Periplasmic binding protein-like II
    8.4e-54  183.4  13.4     9.7e-54  183.2    9.3     1.0  1  0051191
Periplasmic binding protein-like II
```

This is only a small part of the output. The rest of the output includes target and template protein alignments and some other minor statistical information. The remains of the output are not put here as an example, for the reason that they do not contain any essential information for SupfamMod to operate.

The output is available only in plain text format; therefore it needs to be parsed in order to get the necessary information. The primary purpose of HMMER3 in SupfamMod is to provide

Model ID that can is linked to template primary and tertiary structures. The provided example contains 7 protein templates (marked with bold text), which are sorted by the score.

Each line contains specific information about the template, which is divided by whitespace characters. In total there are 10 descriptors (columns) per single template protein:

1. E-value (expected value) – signifies how many similar primary protein structures can be expected to be found by chance in the presented HMM library. The lower score denotes a better match;
2. Score – is the log-odds (logit probability of the event). Current value is not based on the size of the provided HMM library, but only on actual Hidden Markov Models. The output is sorted using current value.
3. Bias – is a correction term used with 'score' field.
4. E-value – same as 1., but for the best scoring domain inside the template, rather then for the whole template sequence. If the score is at magnitude of score, than the template might be not homologous.
5. Score – same as 2., but for best scoring domain.
6. Bias – same as 3., but for best scoring domain.
7. Exp – expected number of similar domains inside the HMM library.
8. N – number of domains selected for the template.
9. Model – the model ID that is linked to primary and tertiary sequences.
10. Description – short description of the template.

(Eddy 2010)

In order to make a prediction, SupfamMod requires the best homologue template, i.e. a template sequence, which has the highest similarity with the target sequence. For the reason that the list is sorted using 'score' value, the first line of the output needs to be parsed, to get the model ID. In order to parse the output a regex was used:

```
[0-9\.e-]\s+[0-9\.e-]+\s+[0-9\.e-]+\s+[0-9\.e-]+\s+[0-9\.e-]+\s+[0-9\.e-]+\s+[0-9\.e-
]+\s+[0-9\.e-]+\s+[0-9]{7}
```

It finds all the lines in the output, which represent the template descriptions (as shown in bold on the previous page). Then the first match is taken and converted into a list of columns. Where 9th column is extracted and can be used with other SupfamMod prediction steps.

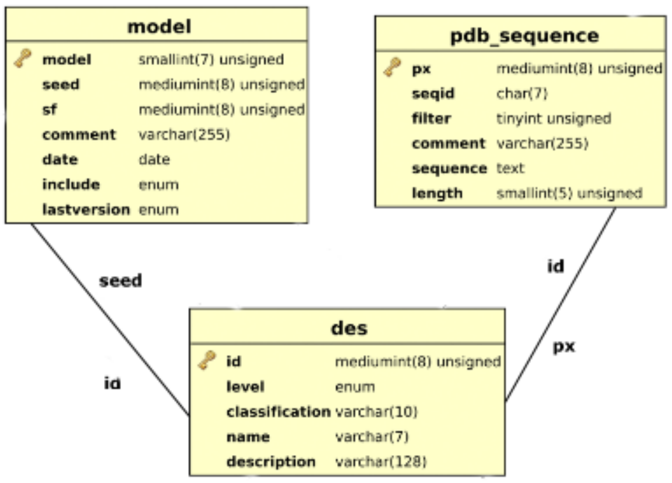
9.2 supfamclass.py

The drawback of 'hmmlib' used in HMMER3 is that it does not contain primary and tertiary protein structures, but Superfamily MySQL database does. Model IDs are unique for Superfamily database and since 'hmmlib' and MySQL database are both based on Superfamily, the model IDs relate to the same information.

There are two pieces of information that has to be gathered using model ID:

1. Template primary sequence;
2. Template 3D structure ID.

ERM diagram presented on the right contains all the required information in order to fetch



Superfamily ERM Diagram
(Gough J. , SUPERFAMILY Description, 2011)

required data. In order to get the primary sequence 'model' and 'pdb_sequence' tables have to be queried. 'Model' table's 'seed' is related to 'pdb_sequence' table's 'px'. 'Pdb_sequence' table's 'sequence' column contains primary protein sequences in FASTA format and 'seqid' contain 3D protein structure ID. Using this information, a SQL query is constructed (where 'model.model' can be any model ID):

```
select pdb_sequence.sequence, pdb_sequence.seqid from model,pdb_sequence
where model.seed=pdb_sequence.px and model.model= '0045538'
```

An example output 3D protein sequence ID is:
```
d3kvta_
```

Current ID is then used to get protein tertiary structure from ASTRAL database (please refer to section 6.3)

An example output sequence is:
```
enrviinvggirhetykatlkkipatrlsrltegmlnydpvlneyffdrhpgvfaqiinyyrsgklhyptdvcgpl
feeelefwgldsnqvepccwmtytahr
```

The outputted sequence in not in proper FASTA format (please refer to section 7.1) and need to be transformed. After the transformation the output will include template name, sequence line length of no more than 80 capitalised characters.

It is important to mention that the server required additional configuration in order to run MySQL with Python. MySQL-Python library was downloaded, compiled and installed on the server machine (Dustman 2011).

9.3 'pdbclass.py'

SupfamMod does not directly use the functionality provided by current class. The primary idea of current class was to connect to Protein Data Bank and download necessary protein structures. After current class was written, it was deprecated, due to redundancy, this is for the reason that all the structural data is available in Superfamily database.

Although 'pdbclass.py' is not used, its certain functions were used in evaluation phase of current project. This is due the fact that specified evaluation technique (described in further sections) requires to download protein structures, both primary and tertiary from Protein Data Bank's website.

Current includes code, which allows downloading tertiary structures and saving them on the local drive allows downloading primary structures. Since primary structures in Protein Data Bank might contain more than one chain, a function was written to return specific chain based on its chain ID (e.g. A, B, C etc.).

Current code will be described in more detail in the evaluation section of current project.

9.4 'modelclass.py'

'modelclass.py' describes the functionality to work with MODELLER. The functionality of current class is based on MODELLER API, which is provided for Python programming language. The description of API, tutorial and manual are provided on MODELLER website in PDF format (Sali, MODELLER: A Program for Protein Structure Modeling 2009). The provided information was used to write appropriate code to do the predictions using the protein structures provided by classes described in previous sections (for theoretical description of MODELLER please refer to section 6.5).

The functionality of modelclass.py includes:

- Conversion of FASTA format to PIR (please refer to section 7.2);
- Template and target primary protein sequence alignment;
- Protein modeling.

MODELLER software was not available on the server and had to be installed. The official website (http://salilab.org/modeller/download_installation.html) provided the required binary files and Installation guide in order to execute a successful installation on Ubuntu Linux (Sali, Modeller 9.9 Release Notes 2011).

9.5.1 Conversion of FASTA to PIR

For the reason that MODELLER supports only primary protein structures only in proprietary PIR format, FASTA files have to be converted. For this reason inbuilt MODELLER functionality is used for format conversion.

MODELLER includes 'alignment' object, which is used for alignment of primary protein structures. One of its features is ability to output structures in a number of formats, regardless of the file input format. It supports input and output of PIR, FASTA, PSS, PAP and a number of other primary structure formats. An example of code is presented below:

```
1. fnFasta =  "".join([self.defaultPath,fileName,".fasta"])
2. fnPir = "".join([self.defaultPath,fileName,".ali"])
3. aln = alignment(self.env)
4. aln.append(file=fnFasta, alignment_format="FASTA")
5. aln.write(file=fnPir, alignment_format="PIR")
```

In first and second lines create variables with FASTA and PIR file paths, which are based on two variables:

- self.defaultPath – the path of the default MODELLER directory;
- fileName – the name of protein structure.

The third line instantiates the alignment object. It uses another object that is called environ() (self.env) in order to instantiate itself. Current object contains most information and functions to execute protein modeling. Environ() class is instantiated once in the 'modelclass.py' and is used in alignment and modeling.

The last to lines are used to input FASTA file and output PIR, where `aln.write()` creates a PIR sequence file in the specified path.

Described sequence format conversion technique is implemented as a subroutine. Current functionality is used on both target and protein primary protein sequence files. Then the outputted sequences are aligned (please refer to section 9.5.2).

9.5.2 Template and target primary protein sequence alignment

Current functionality of 'modelclass.py' is implemented using similar technique described in previous section. The similarity is the fact that alignment() object is used. But instead of appending just one sequence, two protein sequences in PIR are appended, plus template 3D structure from ASTRAL database. An example of code is presented below:

```
1. aln = alignment(self.env)
2. aln.append_model(model(self.env, file=astralPdbPath,
3.     model_segment=('FIRST:@','LAST:')),
4.     align_codes=modelPdbId,
5.     atom_files=self.concatNameFormat(self.defaultPath,modelPdbId,"pdb"))
6. aln.append(file=self.concatNameFormat(self.defaultPath,pdbId,"ali"),
   align_codes="all")
7. aln.align2d()
8. aln.write(file=self.concatNameFormat(self.defaultPath,outputFileName,"ali
   "), alignment_format='pir')
9. aln.write(file=self.concatNameFormat(self.defaultPath,outputFileName,"pap
   "), alignment_format='pap')
```

The first line instantiates the alignment() MODELLER class as described in previous section. Line 2-5 is used to read three-dimensional protein structure template to the alignment and add it to the alignment list. It uses model() object, which holds all information about the three-dimensional structure:

- file=astralPdbPath – loads up the template 3D structure;
- model_segment=('FIRST:@','LAST:')) – tells modeller that the whole 3D structure should be read;
- align_codes=modelPdbId – identification name of current 3D structure;
- atom_files=self.concatNameFormat(self.defaultPath,modelPdbId,"pdb")) – path to 3D model;

Target sequence is read using line 6. Line 7 is used to start the alignment process. There is a number of align functions, such as malig() which is used to do simple align of 1D structures, when 3D structure is not available. In our case align2d is used, for the reason that there is a 3D structure. Lines 8 and 9 output the alignment in PIR and PAP formats, both of them will be used in the modelling procedure.

9.5.3 Protein Modelling

Protein modelling is the last step in protein prediction. It creates a final 3D protein structure file in PDB format. Current functionality is implemented using automodel() class. It uses the created alignment and 3D protein structure. After the prediction is completed, an assessment is made using DOPE and GA341 methods. The assessment is required to give an estimated quality measurement of the prediction. An example of code is presented below:

```
1. am = automodel(self.env,
      alnfile=self.concatNameFormat(self.defaultPath,alignmentFileName,"ali"
   ),
      knowns=modelPdbId,
      sequence=pdbId,
      assess_methods=(assess.DOPE, assess.GA341))
2. am.starting_model = 1
3. am.ending_model = 1
4. am.make()
5. finalModels = filter(lambda x: x['failure'] is None, am.outputs)
6. key = 'DOPE score'
7. finalModels.sort(lambda a,b: cmp(a[key], b[key]))
8. resultPdb = finalModels[0]['name']
9. shutil.copy(resultPdb,"".join([self.defaultPath,"/",pdbId,".pdb"]))
```

The first takes the alignment files as it argument (`alnfile=…`), template identification (`knowns=…`), target sequence id (`sequence=…`) and assessment methods (`assess_methods=…`).

Target sequence ID and template ID are used to look for structure files in current directory. E.g. by setting `knowns=1plc` MODELLER will look for '1plc.pdb' in the directory.

Lines 2 and 3 are used to set how many predictions should MODELLER make. Modelling of a single protein can take up to 30 minutes, for this reason it was decided to make only one prediction per target sequence. In practice, if there are a number of predictions, assessment scores can be used to choose the best structure. Current functionality is implemented using lines 5-8. After the prediction is completed using am.make(), am() object will contain a list (array) of predicted structure names with scores. Current list is being sorted using the score in descending order. Then the first protein name is being selected. Line 9 is used to rename and copy the predicted file to desirable directory.

9.6 'htmlclass.py'

Current class is created to present functionality of SupfamMod as a webpage. There are two ways in which 'htmlclass.py' can be used:

- Outputting HTML directly to CGI;
- Saving HTML files on the server.

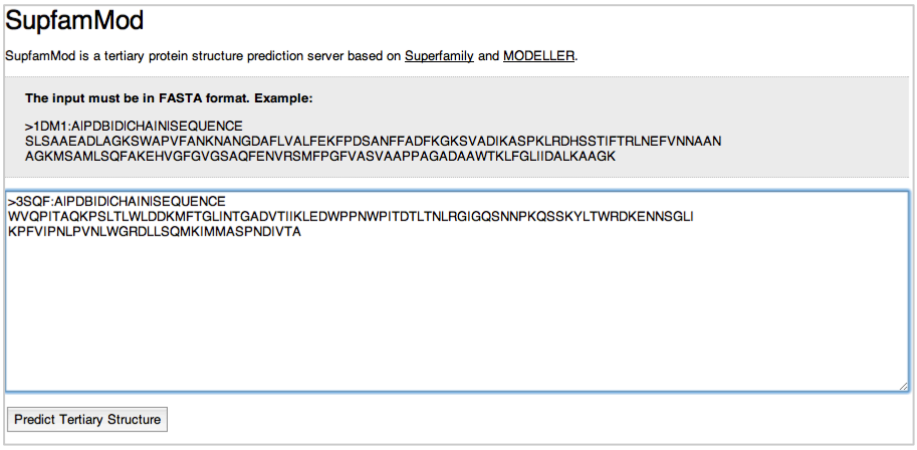
Additionally the outputted webpages include JavaScript and JAVA code. JavaScript is used to implement AJAX functionality and JAVA used to render 3D predictions.

The implementation of 'htmlclass.py' is divided into two parts:

1. Homepage implementation and input handling;
2. Prediction presentation.

9.6.1 Homepage implementation and input handling

SupfamMod is available at *http://supfam2.cs.bris.ac.uk/SUPERFAMILY_devel/cgi-bin/max/index.cgi.* The input is implemented as text box; it accepts primary protein structures in FASTA format. After the sequence is submitted an AJAX request is being sent to the webserver containing submitted FASTA sequence. The reason for sending AJAX request, instead of simply loading the page with prediction, is the fact that the prediction of a protein might take up to 30 minutes and there is should be some indication of progress. AJAX allows sending a request and displaying a loading page. The initial idea was to implement a landing page, which would appear after a sequence is submitted and would reload itself until a prediction was made. Unfortunately due to restrictions of Python it is not possible to output/flush text via CGI before the scripts finish running. An example of a homepage and loading page is presented below:



SupfamMod

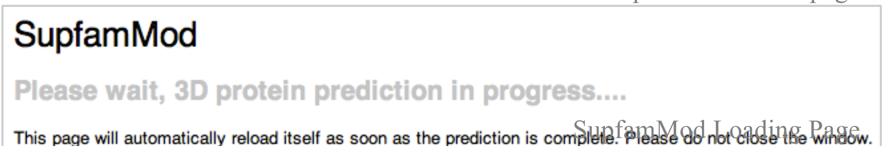
SupfamMod is a tertiary protein structure prediction server based on Superfamily and MODELLER.

The input must be in FASTA format. Example:

```
>1DM1:A|PDBID|CHAIN|SEQUENCE
SLSAAEADLAGKSWAPVFANKNANGDAFLVALFEKFPDSANFFADFKGKSVADIKASPKLRDHSSTIFTRLNEFVNNAAN
AGKMSAMLSQFAKEHVGFGVGSAQFENVRSMFPGFVASVAAPPAGADAAWTKLFGLIIDALKAAGK
```

```
>3SQF:A|PDBID|CHAIN|SEQUENCE
WVQPITAQKPSLTLWLDDKMFTGLINTGADVTIIKLEDWPPNWPITDTLTNLRGIGQSNNPKQSSKYLTWRDKENNSGLI
KPFVIPNLPVNLWGRDLLSQMKIMMASPNDIVTA
```

SupfamMod Homepage



SupfamMod

Please wait, 3D protein prediction in progress....

This page will automatically reload itself as soon as the prediction is complete. Please do not close the window.

SupfamMod Loading Page

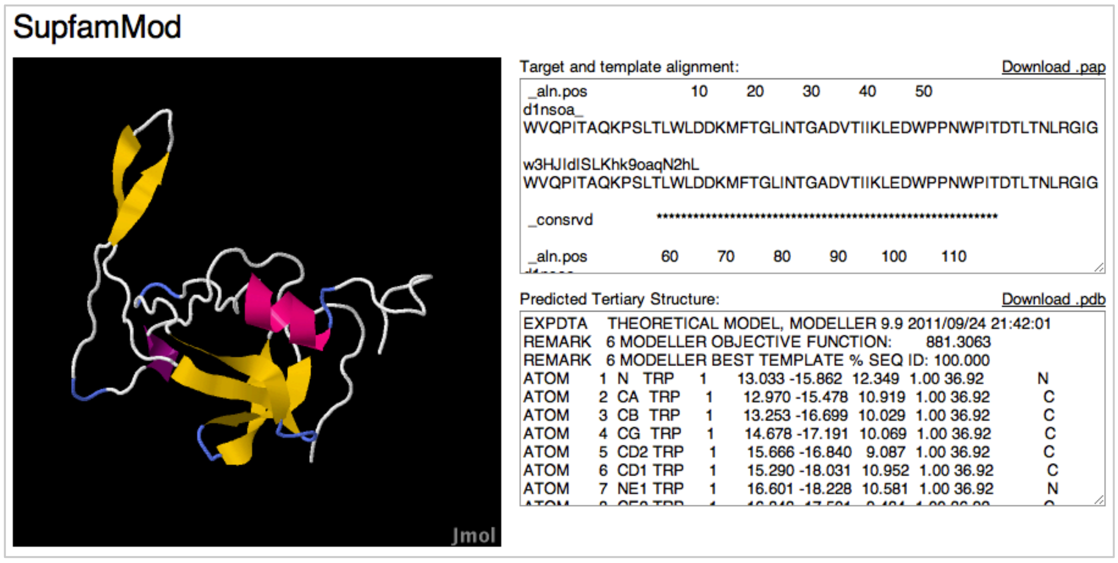
After input is received by ‘predict.cgi’, it can be used to start the prediction process (please refer to section 9.7). An example of code:

```
sequence = sfam.makeFasta(html.getPOSTValue("seq"),target)
```

Current line of code is used to save the sequence to the server for further processing and prediction. Sfam.makefasta() takes in two arguments – plaintext sequence and sequence name. The plaintext sequence is being normalized and set to a proper FASTA format (please refer to section 7.1).

9.6.2 Prediction presentation

Prediction presentation is implemented using similar functions as described in previous section. But instead of flushing the HTML via CGI, the webpage is being saved on the server. Subsequently client’s browser can be redirected to the page using AJAX. An example of the page is presented below:



Target and template alignment:

```
 _aln.pos              10     20     30     40     50
d1nsoa_
WVQPITAQKPSLTLWLDDKMFTGLINTGADVTIIKLEDWPPNWPITDTLTNLRGIG

w3HJIdISLKhk9oaqN2hL
WVQPITAQKPSLTLWLDDKMFTGLINTGADVTIIKLEDWPPNWPITDTLTNLRGIG

 _consrvd      ********************************************************

 _aln.pos      60     70     80     90     100    110
```

Predicted Tertiary Structure:

```
EXPDTA    THEORETICAL MODEL, MODELLER 9.9 2011/09/24 21:42:01
REMARK   6 MODELLER OBJECTIVE FUNCTION:       881.3063
REMARK   6 MODELLER BEST TEMPLATE % SEQ ID: 100.000
ATOM      1  N   TRP     1      13.033 -15.862  12.349  1.00 36.92           N
ATOM      2  CA  TRP     1      12.970 -15.478  10.919  1.00 36.92           C
ATOM      3  CB  TRP     1      13.253 -16.699  10.029  1.00 36.92           C
ATOM      4  CG  TRP     1      14.678 -17.191  10.069  1.00 36.92           C
ATOM      5  CD2 TRP     1      15.666 -16.840   9.087  1.00 36.92           C
ATOM      6  CD1 TRP     1      15.290 -18.031  10.952  1.00 36.92           C
ATOM      7  NE1 TRP     1      16.601 -18.228  10.581  1.00 36.92           N
```

The prediction in PDB can be downloaded from presented page, with corresponding target and template alignment. Additionally the prediction rendered using Jmol software. Current software was installed on the server and adjusted to preview a high quality render. An example of the code is presented below:

```
jmolApplet(400, 'load w3HJIdISLKhk9oaqN2hL.pdb; cpk off; wireframe
off; cartoon; color cartoon structure; set antialiasDisplay true');
```

Current line of code uses the JavaScript file embedded into the results page, that activates Jmol software. The arguments of current function include the location of the three-dimensional structure, type of rendering and rendering quality.

Additionally an error page can be generated instead of a results page, in case there was an error while the prediction process was commences. Please refer to section 9.7 for more information.

9.7 ‘predict.cgi’

Predict.cgi instantiates and controls all the classes described in previous sections. Furthermore it contains all the constants that are required to run SupfamMod, these constants include paths to “working” directories, software locations and database paths. An example of the code, which includes key part of the predict.cgi, is presented below:

```
1.  hmm.run()
2.  hmm.parseResults()
3.  hmmModel = hmm.getModel(0)
4.  sfam.setModel(hmmModel)
5.  sfam.queryStructures()
6.  sfam.closeConnection()
7.  mdl.makeFile(sfam.getAstralSequence(),sfam.getAstralId(),"fasta")
8.  mdl.convertFastaToPir(sfam.getAstralId())
9.  mdl.convertFastaToPir(target)
10. mdl.alignProteins(target, sfam.getAstralId(), sfam.getAstralFileName(),
    "".join([sfam.getAstralId(),"_",target]))
11. mdl.modelProtein(target, sfam.getAstralId(),
    "".join([sfam.getAstralId(),"_",target]),CGI_PATH)
12. html.makeResultPage(target,DEFAULT_PATH,sfam.getAstralId())
```

Description of the presented lines of code:

1. Runs HMMER3, the output is saved in the ‘hmm’ object;
2. The output is being parsed;
3. The template ID of the highest score is returned;
4. Pass template ID to the ‘sfam’ object;
5. Query MySQL database with template ID for 1D sequence and 3D structure ASTRAL ID;
6. Closes MySQL connection;
7. Saves 1D template sequence in FASTA format;
8. Converts template to PIR;
9. Converts target to PIR;
10. Aligns template and target;
11. Makes a protein prediction;
12. Creates a page with results.

9.8 Additional Implementations

There are two additional implementations of SupfamMod, which are created in order to try to improve the prediction quality. After all three implementations are available; they will be evaluated using a random selection of protein sequences from Protein Data Bank. The best implementation will be used with SupfamMod web service.

The SupfamMod prediction implementation changes will be made to template selection system, for the reason that alterations to MODELLER are to complex for a summer project, whereas template selection is manageable and might improve predictions.

9.8.1 Multiple Templates Prediction

Current method uses multiple templates in order make a prediction. Current method was used in other homology modelling system and was able to improve prediction quality (Larsson, et al. 2009). The papers state that increasing the number of templates increase the quality of prediction.

Multiple templates prediction method extends the functionality described in section 9.1. As we can see in the HMMER3 output example provided in section 9.1, there are a number of templates with good scores.

A number of additional subroutines and functions were appended into ‘hmmclass.py’ and ‘modelclass.py’, plus a new main Python script file (‘multiplePredict.py’) was created in order facilitate current method. Examples of the changes are given and explained in section below. ‘multiplePredict.py’ is not described in current section, for the reason that it basically uses the new functions presented by ‘hmmclass.py’ and ‘modelclass.py’.

9.8.1.1 ‘hmmclass.py’ multiple template prediction changes

```
def inclusionCount(self):
        hmmCountResultsRe = re.compile("-{11}.*-{6}\sinclusion",re.DOTALL)
        hmmCountResultsOutput = hmmCountResultsRe.findall(self.hmmOutput)[0]
        hmmCountResults = len(re.findall("\n",hmmCountResultsOutput))
        #allow 4 or less models to be inluded in prediction
        if hmmCountResults > 5:
                self.hmmOutput = hmmCountResultsOutput
                hmmCountResults=4
        elif hmmCountResults > 1:
                self.hmmOutput = hmmCountResultsOutput
        return hmmCountResults-1
```

Current function outputs the number of templates available for prediction. The number of maximum models that can be used in prediction is set to 4. Current method is based two facts:

- Multiple template models increase the time required to produce the protein and since the SupfamMod is a pipeline protein prediction system, time designated per single prediction should be limited. Additionally for the reason that the system should be tested using 250 proteins, the processor time is also of an essence. Although if current method will give an improvement, more experiments might be done with increased or decreased number of templates.
- The quality of protein templates in the HMMER3 output drops over a number of presented entries. And since the prediction quality depends on the characteristics of the used templates, it is not feasible to use templates with low scores.

9.8.1.2 ‘modelclass.py’ multiple template prediction changes

A new function was appended to current class that allows aligning multiple proteins sequences. Current code is based on MODELLER API and is similar to code presented in section 9.5.2. It uses the same API functions altered in order to accept multiple sequences. Essentially an additional ‘for’ loop was appended, which uses a list of protein supplied by `’modelPdbIds’` argument. An example of additional loop is presented below:

```
for modelPdbId in modelPdbIds:
     aln.append_model(model(self.env, file=astralPdbPath, model_segment=('FIRST:@','LAST:')),
     align_codes=modelPdbId, atom_files=self.concatNameFormat(self.defaultPath,modelPdbId,"pdb"))
```

Presented piece of code substitute lines 2-5 presented in section 9.5.2.

9.8.2 SCOP domain ID of closest structure

Current implementation is based on theory presented in sections 6.3.3 and 6.3.4. The basic idea of current method is to use is to functionality of ASS3 software, which implements a new hybrid method for finding homologous protein sequences (J. Gough, Genomic scale sub-family assignment of protein domains 2006).

A number of changes were made to 'hmmclass.py' and 'supfamclass.py', plus a new main Python script file ('pxPredict.py') was created in order facilitate current method. Examples of the changes are given and explained in section below. 'pxPredict.py' is not described in current section, for the reason that it basically uses the new functions presented by 'hmmclass.py' and 'supfamclass.py'.

9.8.2.1 'hmmclass.py' SCOP domain ID changes

In order to use current methodology additional software was installed on the server. The software is implemented using PERL and uses HMMER3 software and SCOP database files (please refer to section 6.3.4). Current file uses a number of arguments: SCOP database files paths, target sequence and output folder directories. An output example is presented below:

```
1. 2NhcnXhf7rSRrIwrGnvX
2. 0040873
3. 2-144
4. 7.05e-34
5. 2
6. LSAAEADLAGKSWAPVFANKNANGDAFLVALFEKFPDSANFFADFKGKSVADIKASPKLRDHSSTIFTRLNEF
   VNNAANAGKMSAMLSQFAKEHVGFGVGSAQFENVRSMFPGFVASVAAPPAGADAAWTKLFGLIIDALKA
7. 5.32e-07
8. 15149
9. 46463
```

Description of the output:

1. Sequence ID
2. SUPERFAMILY model ID
3. Match region
4. Evalue score
5. Model match start position
6. Alignment to model
7. Family evalue
8. SCOP Family ID
9. SCOP domain ID of closest structure (px value)

(Gough, Karplus, et al., Assignment of homology to genome sequences using a library of hidden Markov models that represent all proteins of known structure. 2001)

Line 9 is of interest for the purposes of current project. It is the ID of the structure, which will be used for the protein prediction. Please refer to the next section for more information.

Two additional functions (subroutines) were appended to 'hmmclass.py':

- runAssignments() – runs ASS3 and saves the output;
- getPx() – parses the saved output and returns SCOP domain ID of closest structure.

9.8.2.1 'supfamclass.py' SCOP domain ID changes

Additional procedures implemented in current section are based on section 9.2. Two supplementary subroutines were implemented:

- queryPx() – queries the Supefamily MySQL database in order to return the template primary protein sequence and protein tertiary structure ID;
- setPx() – assigns SCOP domain ID of closest structure to the ‘supfamclass.py’.

9.9 Self-evaluation system

Self-evaluation (please refer to section 5.2.2) system is implemented using two Python script files:

- ‘selfEvaluate.py’ – controls the connection between SupfamMod and Protein Data Bank and runs predictions on downloaded structures.
- ‘massCe.py’ – runs CE software (pleaser refer to section 6.7) and prints results to plaintext files and HTML files, which are automatically made available on SupfamMod website.

The implementation of self-evaluation system does not use OOP. Two presented files are common Python scripts, which run other applications (like CE and SupfamMod) using natural Bash shell commands. For a number of reasons the self-evaluation system wasn’t implemented as a part of SupfamMod classes:

1. Self-evaluation system is based on the code used in evaluation of SupfamMod (please refer to section 10).
2. SupfamMod’s system primary objective is to make protein predictions, whereas self-evaluation system is an external entity.
3. Self-evaluation will be run on the server using `’crontab’`. (Reznick 1993). Current application is a part of installation of the server on which SupfamMod is running. It allows scheduling execution of applications. Current line of code will be appended to `‘crontab’` configuration: `’0 1 * * 3 /var/www/html/SUPERFAMILY_devel/max/evaluation/selfEvaluate.py’.` Current setup will run the code every Thursday at 1:00.

9.9.1 ‘selfEvaluate.py’

Current method uses the The RCSB PDB RESTful (REpresentational State Transfer) Web Service (Protein Data Bank 2011) interface in order to download new protein structures. Current method is a part of PDB API that allows fetching protein PDB IDs using XML queries. The example Python code:

```
url= 'http://www.rcsb.org/pdb/rest/search'
query =   """
<orgPdbQuery>
<queryType>org.pdb.query.simple.ReleaseDateQuery</queryType>
<database_PDB_rev.date.comparator>between</database_PDB_rev.date.comparat
or>
<database_PDB_rev.date.min>""" + day + """</database_PDB_rev.date.min>
<database_PDB_rev.date.max>""" + day + """</database_PDB_rev.date.max>
<database_PDB_rev.mod_type.value>1</database_PDB_rev.mod_type.value>
</orgPdbQuery>
"""
req= urllib2.Request(url, data = query)
conn = urllib2.urlopen(req)
result = conn.read()
conn.close
```

HTTP connection is made between SupfamMod server and PDB (at `'http://www.rcsb.org/pdb/rest/search'`) using the presented lines. It contains XML message container with a query. In current example lines 3-9 requests a list of PDB IDs, which were submitted on present day. Afterwards all the new PDB IDs are used to download corresponding primary structure sequences (using `http://www.pdb.org/pdb/files/fasta.txt?PDB_ID`, where PDB_ID is changed to required protein sequence identification). It is important to check if the downloaded sequences include only a single protein chain, for the reason that SupfamMod predicts only tertiary protein sequences. Protein sequences with a number of chains other than 1 should be predicted using quaternary protein prediction systems. After the chain quantity is checked the protein is saved and corresponding tertiary protein sequence is downloaded (using `http://www.pdb.org/pdb/files/PDB_ID.pdb`, where PDB_ID is changed to required protein sequence identification).

Then it is possible to make a prediction based on the downloaded protein sequences. In the example presented below ‘pxPredict.py’ (SCOP domain ID of closest structure) SupfamMod implementation is used. But depending on the evaluation results which will be determined and presented in the next section, SupfamMod implementation will be exchanged to superior one.

```
commands.getoutput("".join(["python /var/www/html/SUPERFAMILY_devel/cgi-bin/max/pxPredict.py
/var/www/html/SUPERFAMILY_devel/max/evaluation/",day,"/",resultItem,".fasta
/var/www/html/SUPERFAMILY_devel/max/evaluation/",day,"/","results/"]))
```

‘pxPredict.py accepts two arguments: target primary sequence in FASTA format and result output folder.

After the required protein sequences are downloaded, ‘massCe.py’ can be executed. This is done using the presented line of code:

```
 commands.getoutput("".join(["python
/var/www/html/SUPERFAMILY_devel/max/evaluation/massCe.py
/var/www/html/SUPERFAMILY_devel/max/evaluation/",day,"/ ",
"/var/www/html/SUPERFAMILY_devel/max/evaluation/",day,"/results/ ", day]))
```

Three arguments are passed to ‘massCe.py’: path to sequence, path to results output directory and submission date.

9.9.2 ‘massCe.py’

Current script reads the primary protein sequence names from the specified directory and passes them to CE one by one. The example presented below shows the execution of CE.

```
output = commands.getoutput("".join(["bash
/var/www/html/SUPERFAMILY_devel/max/evaluation/jfatcat/runCE.sh -file1 ",
targetFolder , resultItem, " -file2 ", destFolder, resultItem, " -
printCE"]))
```

CE accepts two arguments – paths to tertiary protein sequences that are required to be compared. It outputs the 3D structure alignment information in plain text. This includes sequence identity percentile, primary sequence alignment and 3D sequences alignment coordinates. An example of the output is presented below:

```
Alignment length = 197 Rmsd = 3.38A Z-Score = 5.5 Gaps = 96(48.7%) CPU =
4264 ms. Sequence identities = 14.2%
Chain 1:   39 EESDTFNPAATIYDGKIVVMYRAEDNSAQGIGSRTSRLGYATSTDGIHFERDTKPA--
FYPAKDNQAENE
```

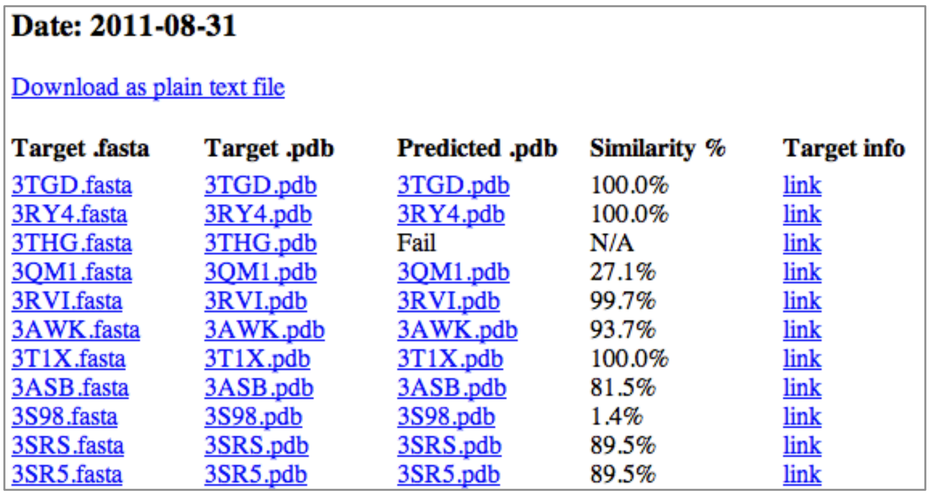
```
Chain 2:  115 PGGTEDPRIAMTE--DGTYVLLYTQWNR----KVPRLAVATSK--
DLKHWTKFGPAFEKAYN--------

Chain 1:  107 CPGGTE---DPRIAMTEDGTYVLLYTQWNR---KVP--RLAVATSKDLKHWT----
KFGPAFEKAYNGKF
Chain 2:  169
GKFKDEATKSASLVTTLKGDKQVIAKVNGKYFMYWGEKNVYAATSDNLIDWDPLLDENGELLK------L
Chain 1:  165 K--------DEATKSASLVTTLKGDKQVIAKVNGKYFMYWGE---------------
KNVYAATSD--N
Chain 2:  233 FSPRSGYFDSQLTECGPPAI--------LTKD--
GIVLLYNGKNEPGEKGDTAYPANSYCAGQALFDVNN
Chain 1:  209 LIDWDPLLDENGELLKLFSPRSGYFDS-----
QLTECGPPAILTKDGIVLLYNGKNEPGEKGDTAYPANS
Chain 2:  293 PTKLIGRLD-----KPFLQPTDD-FEKSGQYPAGTVFVEGLVYYR-NKWYLYYGCA------
------DS
Chain 1:  274 YCAGQALFDVNNP
Chain 2:  344 FVAVAVSDKQLNF
     X1 = ( 0.649783)*Xorig + (-0.190233)*Yorig + (-0.735930)*Zorig + (
68.
953789)
     Y1 = (-0.547010)*Xorig + (-0.789281)*Yorig + (-0.278954)*Zorig + (
138.680016)
     Z1 = (-0.527789)*Xorig + ( 0.583821)*Yorig + (-0.616921)*Zorig + (
181.466088)
```

Current file needs to be parsed for information to be extracted and previewed on the SupfamMod webpage. The main area of interest is the first line of the output. In order to parse it a regular expression is used:

```
([0-9]+)\sRmsd\s=\s([0-9\.]+)A\sZ\-Score\s=\s([0-9\.]+)\sGaps\s=\s([0-
9]+)\(([0-9\.]+)\%\).*identities\s=\s([0-9\.]+)\%
```

It would've been possible just to take the first line of the output and parse it, but due to possible errors in comparison method or absence of predicted protein structure – CE might show an unexpected output.

After all the mentioned procedures are complete, the parsed info is being saved on the server. This information is available on the landing page of SupfamMod via link on the bottom of the page. An example of self-evaluation task:


Date: 2011-08-31

Download as plain text file

Target .fasta	Target .pdb	Predicted .pdb	Similarity %	Target info
3TGD.fasta	3TGD.pdb	3TGD.pdb	100.0%	link
3RY4.fasta	3RY4.pdb	3RY4.pdb	100.0%	link
3THG.fasta	3THG.pdb	Fail	N/A	link
3QM1.fasta	3QM1.pdb	3QM1.pdb	27.1%	link
3RVI.fasta	3RVI.pdb	3RVI.pdb	99.7%	link
3AWK.fasta	3AWK.pdb	3AWK.pdb	93.7%	link
3T1X.fasta	3T1X.pdb	3T1X.pdb	100.0%	link
3ASB.fasta	3ASB.pdb	3ASB.pdb	81.5%	link
3S98.fasta	3S98.pdb	3S98.pdb	1.4%	link
3SRS.fasta	3SRS.pdb	3SRS.pdb	89.5%	link
3SR5.fasta	3SR5.pdb	3SR5.pdb	89.5%	link

An additional function was appended to 'htmlclass.py' to allow displaying links to the results on the landing page. It checks for available folders signed with an appropriate date and therefore displays links to related html files.

10. Evaluation

The evaluation of SupfamMod includes two main steps: determination of best SupfamMod protein prediction implementation and SupfamMod prediction quality comparison to world's best prediction systems. East step will describe testing methodology, implementation details and annotated results.

10.1 Best implementation evaluation method

Each implementation will use a number of randomly selected protein sequences from Protein Data Bank. The selected sequences will correspond to a number of features:

1. Protein release dates should be after July 2009, for the reason that HMM database is built on the entries from PDB, which were submitted before July 2009. Which consequently will result in perfect model matches and 100% prediction accuracy, which contradict the purpose of the test.
2. The downloaded proteins sequences should include only one chain, for the reason that SupfamMod is a tertiary protein prediction system and does not predict quaternary systems (multiple chains of protein 1D and 3D structures).

Two scripts were written in order to download the sequences:

- `'getFASTAFromPDB.org.py'` – downloads a set number of FASTA sequences from Protein Data Bank, check if they correspond to features described above and saves to the file system.
- `'getPDBforFASTAfromPDB.org.py'` - downloads PDB sequences for related FASTA sequences.

Overall 4000 protein sequences were downloaded. But after reconsideration the number of sequences was reduced to 225, due to the fact that a single protein prediction takes approximately 5 minutes. Which means that each test will require a 20 hours of server's processor time per SupfamMod implementation.

'massPredict.py' runs the testing process, which is script for pipelining protein prediction. Its implementation is similar to one described in 9.9.1. The mechanism of current script goes through a specified folder; checks for FASTA sequences and runs specified SupfamMod implementation iteratively on all of the sequences. The script takes three arguments – target folder, output folder and path to SupfamMod executable script.

After all three tests are completed; it is possible to run next script – 'massCe.py'. The described version of the script is similar to one described in section 9.9.2, except it doesn't output the information as HTML files, but only as plain text files. Additionally the output includes only the similarity percentile between predicted and experimentally determined protein structures. Output fragment of one of the output files is presented below. Line 6 of the example shows a failed prediction, it does not contain any data, its scores are considered as 0:

```
1. #PDBID    Len    Rmsd   Z-sc   Gap Cnt      Gap %  Identity %
2. T0560.pdb 60     2.71   4.2    8      13.3   65.0
3. T0567.pdb 133    2.25   6.5    2      1.5    94.0
4. T0569.pdb 56     4.54   3.1    12     21.4   46.4
5. T0515.pdb 117    2.50   4.9    36     30.8   41.9
6. T0602.pdb
7. T0605.pdb 49     3.58   4.4    2      4.1    6.1
8. T0630.pdb 31     1.75   3.3    1      3.2    9.7
9. T0527.pdb 39     3.46   2.6    11     28.2   38.512
```

The next step is parsing current information and displaying it with GnuPlot (William and Kelley 2010). For these purposes '`preparePredictionResultsForGnuplot.py`' was written. It parses the prediction results files in order to produce '.data' files, which can be used by GnuPlot script. Two files are outputted by '`preparePredictionResultsForGnuplot.py`' – data for histogram and data for distance plot. Histogram data describes how many how many predictions were made in every 5% range. Distance plot shows the mean quality over all predictions.

'`ceResults1.txtpercentageHist.data`' includes a number of subroutines:

- parseFile() – parses the file with RegEx, which converts the columns and rows into arrays of data.
- prepareHistTemp() – prepares the array for the parsed data.
- prepareHist() – allocates parsed scores to array().
- prepareDistancePlot() – gradually calculates the mean of prediction score(%) and allocates it to array();
- writeHistData() – flushes histogram and distance plot data into a plaintext file, which can be used with gnuplot script.

An example of histogram data:

```
5      11
10     15
15     9
20     2
25     4
30     6
35     6
40     3
45     1
50     1
55     4
60     5
65     7
70     8
75     4
80     5
85     10
90     16
95     14
100    94
```

First column is the percentage range (0-5, 5-10 etc.) and second column is the number of predictions in the range.

An example of distance plot data:

```
1      84.4
2      92.2
3      94.8
4      94.875
5      95.9
6      96.5833333333
7      92.0428571429
8      81.425
9      76.3666666667
10     71.57
11     73.5636363636
12     72.1666666667
13     68.8307692308
14     71.0571428571
15     71.5666666667
16     72.28125
17     72.6058823529
18     70.2722222222
19     71.8105263158
20     73.22
```

First column is the percentage is the number of predictions and the second column is the mean percentage over the number of predictions.

After running '`preparePredictionResultsForGnuplot.py`' on the available data (three CE results files, one per SupfamMod implementation), the output is 6 files: 3 histogram data files and 3 distance plot data files. Now this data can be used to build plots using GnuPlot scripts. An example of such script (gnuPlotHist.gp) is presented below:

```
set term postscript
set output "figure.eps
set style fill solid 1.0 noborder
set style data histogram
set boxwidth 1
set xtics scale 0
set style histogram cluster gap 1
set ylabel "Number of predictions"
set xlabel "Similarity to target protein (%)"
```

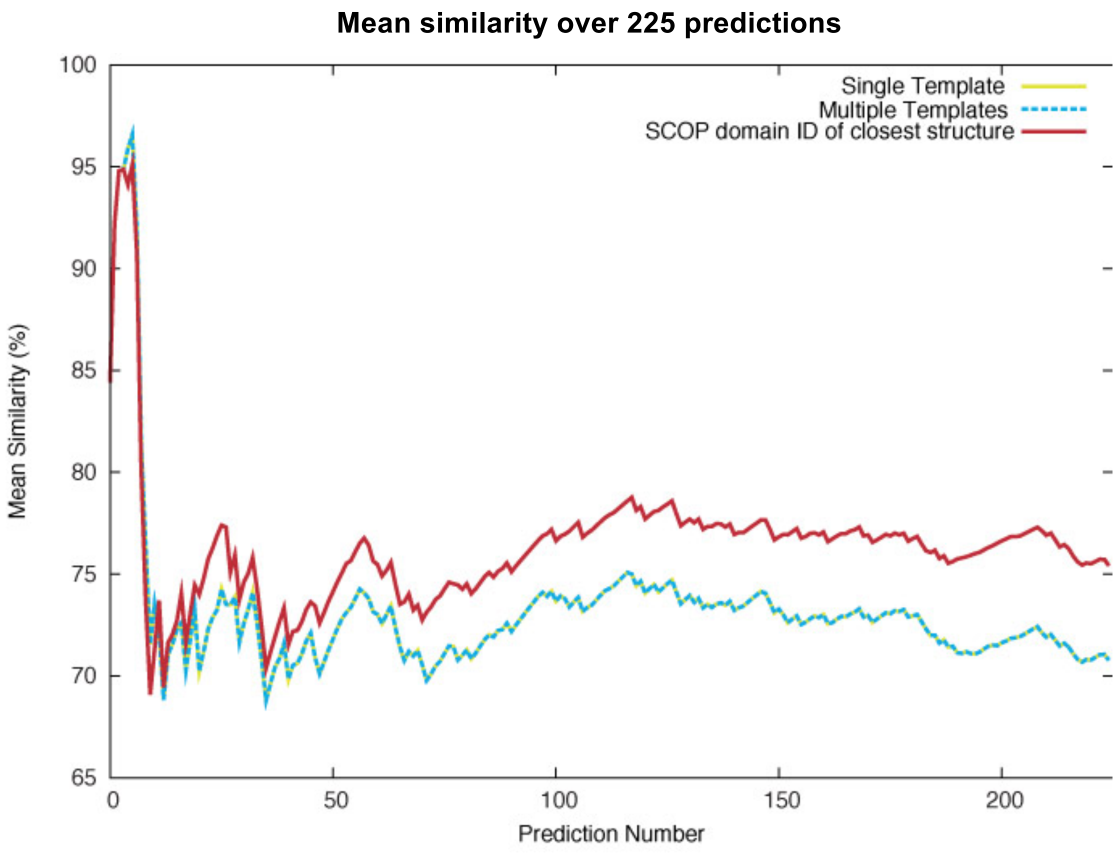
```
10. plot 'ceResults1.txtpercentageHist.data' using 2:xticlabel(1) t "Superfamily Model
    ID (Single Template)" lt rgb "#0075aa", 'ceResults2.txtpercentageHist.data' using
    2 t "Superfamily Model ID (Multiple Templates)" lt rgb "#ffa438",
    'ceResults4.txtpercentageHist.data' using 2 t "SCOP domain ID of closest
    structure" lt rgb "#53b300"
```

Lines 1-2 set the output file format to eps. Lines 3-7 set general preview settings of the histogram; 8-9 – set the story of the chart; 10 – points at files with data, set style, colour of the line and label the line.

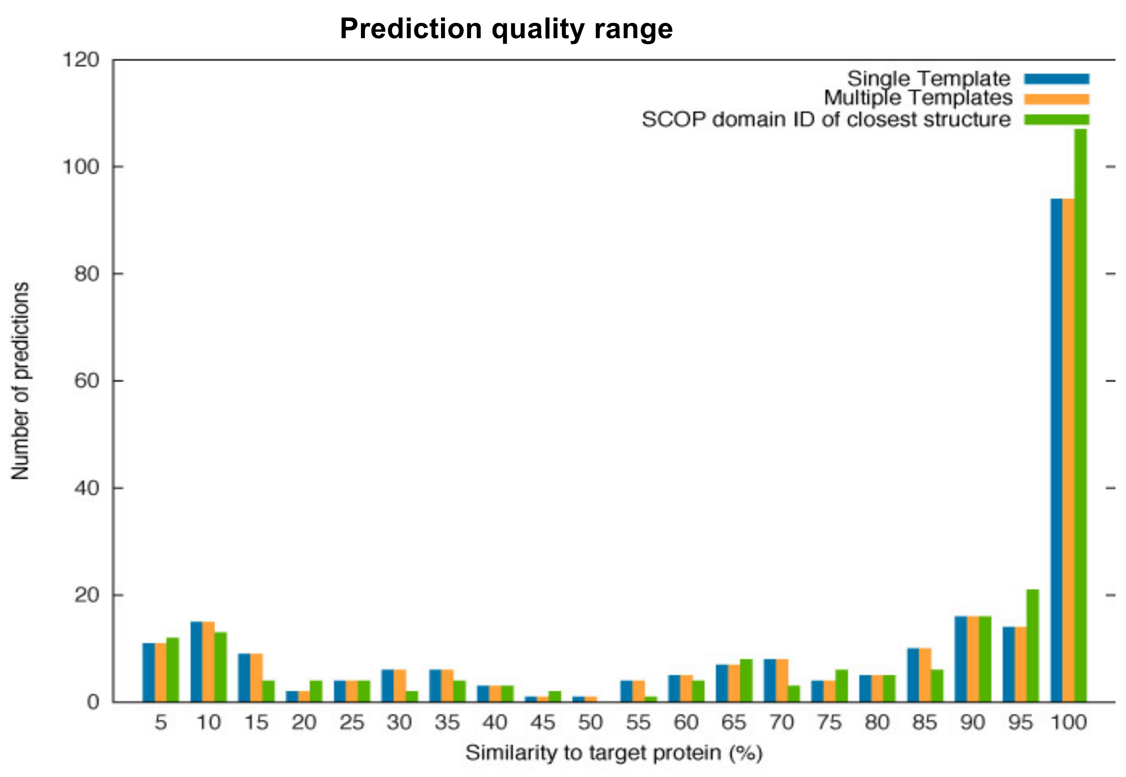
In order to generate the distance plot - gnuPlotDist.gp script is used. It is similar to previous example and follows the same logic:

```
1.  set term postscript
2.  set output "figure.eps
3.  set style fill solid 1.0 noborder
4.  set ylabel "Mean Similarity (%)"
5.  set xlabel "Prediction Number"
6.  set xrange [0:225]
7.  plot 'ceResults1.txtdistancePlot.data' using 2 t "Superfamily Model ID (Single
    Template)" w line ls 1 lt rgb "#E2E23C" lw 5,'ceResults2.txtdistancePlot.data'
    using 2 t "SCOP domain ID of closes structure (Multiple Templates)" w line ls 1 lt
    rgb "#15ADE4" lw 5,'ceResults4.txtdistancePlot.data' using 2 t "SCOP domain ID of
    closes structure" w line ls 1 lt rgb "#BF2F38" lw 5
```

10.2 Best implementation evaluation results



Current graphs displayed above shows that the mean prediction quality of SCOP domain ID of closest structure is approximately 6% better after 225 tests. Single and multiple template prediction have identical prediction quality.

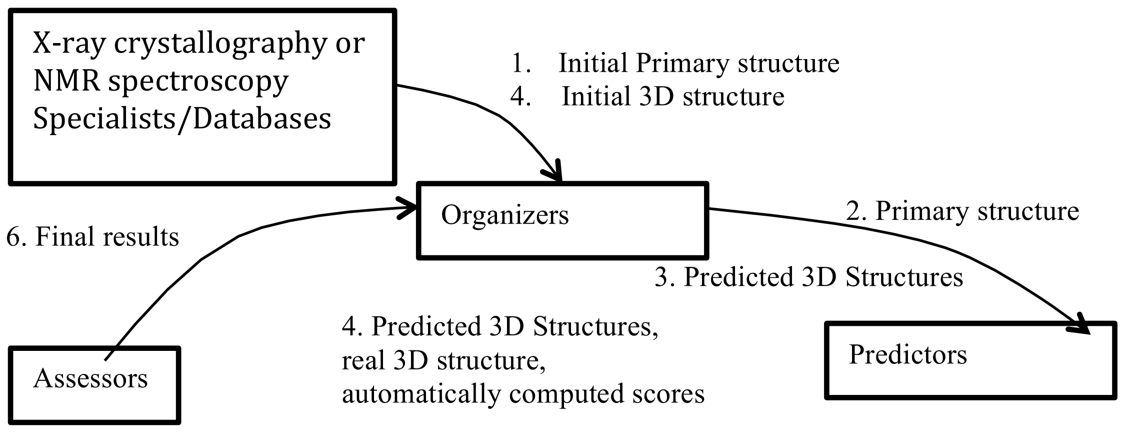


Current histogram shows that there are more predictions of better quality in comparison to single and multiple template methods. Furthermore the best increase in prediction quality is observed in 95-100 ranges.

Current results show that SCOP domain ID of closest structure is the best implementation of SupfamMod. For this reason the final evaluation will be done using the best mentioned method and SupfamMod web service would also run this method.

10.3 Real word performance evaluation description

The performance of the final implementation will be assessed by the data provided by CASP9 competition. CASP is a protein prediction competition that is run every two years. It shows which prediction techniques and scientists make best predictions and show the level of the progress, which was made in the field. The below diagram shows the data flow of CASP9 competition:


X-ray crystallography or NMR spectroscopy Specialists/Databases
1. Initial Primary structure
4. Initial 3D structure
Organizers
2. Primary structure
3. Predicted 3D Structures
6. Final results
4. Predicted 3D Structures, real 3D structure, automatically computed scores
Assessors
Predictors

The CASP9 competition organisers receive protein structures, which were experimentally determined and not released to Protein Data Bank (not available publically). Then predictors get the primary structures of these structures in order to make predictions, which are sent back to CASP9. The predictions are then compared to exact models and scores are being assigned to the group results. All these results are made available online for analysis.

The plan for current evaluation is to download the experimentally determined protein sequences, which were used in CASP9 competition and download predictions made by all the groups. Then predict proteins based on primary protein sequences of experimentally determined protein sequences. Subsequently CE software will be used in order to compare predicted tertiary structures of all groups and SupfamMod to experimentally determined protein structures. Finally the data will be visualised using distance plot of mean similarity over all protein structures and all prediction groups.

It is important to mention that there was a possibility to use information from BioInfo.org meta-server and related CAFASP competition. But unfortunately bioinfo.org doesn’t have all predicted sequences properly stored.

10.4 Real word performance evaluation implementation

First step is to download and unpack all target (experimentally determined) tertiary structures; they are available at: http://predictioncenter.org/download_area/CASP9/targets/casp9.targ_unsplit.tgz. Related FASTA structures are not in a single compressed package, therefore `‘getFastaFromCASP9.py’` Python script was implemented to download 1D structures according. A fragment of the scrip is presented below:

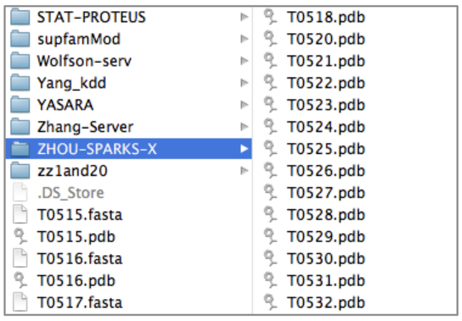
```
resultList = os.listdir(".")
for resultItem in resultList:
    fn = re.sub("\.pdb", "", resultItem)
    casp9URL =
"".join(["http://predictioncenter.org/casp9/target.cgi?view=sequence&target=", fn])
    caspResult = urllib2.urlopen(urllib2.Request(casp9URL)).read()
```

The fetches all PDB files names, removes the ‘.pdb’ file extension and uses it in a link `http://predictioncenter.org/casp9/target.cgi?view=sequence&target=TARGET` which relates to target primary sequence.

`‘getPdbCASP9results.py’` Python script was written to download and sort predictions of participating prediction groups into folders. A fragment of the scrip is presented below:

```
print commands.getoutput("".join(["wget
http://predictioncenter.org/download_area/CASP9/server_predictions/", fn,
".3D.srv.tar.gz"]))
print commands.getoutput("".join(["mv ", fn, ".3D.srv.tar.gz", " ", fn,
".tar.gz"]))
print commands.getoutput("".join(["tar
-xvzf ", fn, ".tar.gz"]))
```

Current fragments shows how the links for the prediction archive are constructed and unpacked. After unpacking prediction files are moved to appropriate prediction group folder and renamed to relate to target protein sequence. An example of such folder structure is presented on the right –



STAT-PROTEUS
supfamMod
Wolfson-serv
Yang_kdd
YASARA
Zhang-Server
ZHOU-SPARKS-X
zz1and20
.DS_Store
T0515.fasta
T0515.pdb
T0516.fasta
T0516.pdb
T0517.fasta

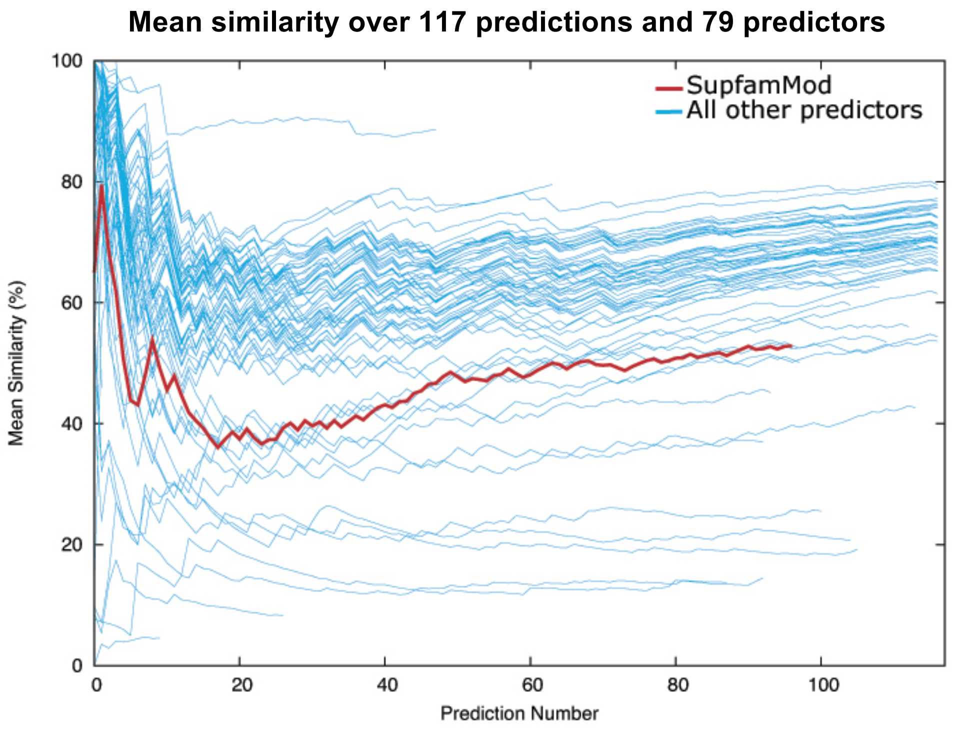
T0518.pdb
T0520.pdb
T0521.pdb
T0522.pdb
T0523.pdb
T0524.pdb
T0525.pdb
T0526.pdb
T0527.pdb
T0528.pdb
T0529.pdb
T0530.pdb
T0531.pdb
T0532.pdb

subfolders contain predicted protein structures and are named respectively to the group name. The root folder contains target protein data – primary and tertiary structures.

Then SupfamMod makes its own predictions using ‘massPredict.py’ as described in section 10.1. The predicted structures are save in SupfamMod subfolder. At this stage all the data is collected and is ready to be processed using CE and corresponding scripts. Overall there are 117 target structures and 80 groups including SupfamMod that prediction results have to be compared.

At this point ‘`massCe.py`’ can be used on all of the groups in order to get target-prediction comparison results. And plot result ‘`plotResults.py`’ script can be used to prepare data for the GnuPlot. These methods are identical to the ones that are described in section 10.1. The difference is that will be no histogram data plotted, for the reason there is too much data to fit; distance plot will be sufficient to visually compare SupfamMod performance to other predictors. Another distinction is that the distance plot will not include the failed predictions (which scores are set to 0% similarity). This will allow showing mean similarity of successful predictions. At the same time it will be visible how many predictions have failed by SupfamMod in comparison to other predictors.

10.5 Real word performance evaluation results



Current graph shows that SupfamMod prediction system has a comparable prediction quality to other predictors – both automatic and human assisted. The quality of protein predictions is at 45% similarity, which places SupfamMod at 67 place amongst CASP9 participants. The number of failed predictions I quite high, but still not the lowest one. The high failed prediction rate high for the reason that CASP9 target sequences were of high complexity and included new folds (protein structures which are not found in HMM library).

Overall SupfamMod showed a good performance and it definitely wasn’t the worst prediction system. Current result is very good for a prediction system, which was created as a summer project. It is important to mention that the competing groups usually include a group of scientists who devote their careers designing protein prediction systems and the groups spend reasonable amounts of money to perform the research.

11. Discussion

Current project allowed building a SupfamMod protein structure prediction system that proved itself to be successful in the evaluation step of the project. The tasks proposed by current project were fully fulfilled: a report was written; SupfamMod system was implemented and evaluated.

The project plan had been changed – instead of becoming a part of meta-server; SupfamMod includes self-evaluation system, which solves a task of automatic system evaluation, but using different approach. This decision was made due to current unavailability of meta-servers on the Internet.

Current system was built in order to be a supplementary feature of Superfamily database. For this reason current application includes a web interface. Since SupfamMod is running on one of the Superfamily servers, the system can be done visible to Superfamily users by adding an appropriate link on the website.

Works Cited

Zhang, Yang. “Progress and challenges in protein structure prediction.” *Curr Opin Struct Biol.*, June 2008: 342–348.
William, Thomas, and Colin Kelley. “gnuplot 4.4.” *gnuplot.* 5 March 2010. http://www.gnuplot.info/docs_4.4/gnuplot.pdf (accessed June 27, 2011).
Volker A. Eyrich, Marc A. Mart-Renom, Dariusz Przybylski, Mallur S. Madhusudhan, András Fiser, Florencio Pazos, Alfonso Valencia, Andrej Sali, Burkhard Rost. “EVA: continuous automatic evalution of protein structure prediction servers.” *Bioinformatics* 17, no. 12 (2001): 1242-1243.
Andras Fiser, Andrej Sali. “Modeller: Generation and Refinement of Homology-Based Protein Structure Models.” *METHODS IN ENZYMOLOGY* 374 (2003): 461-491.
Andreeva, Antonina, et al. “Data growth and its impact on the SCOP database: new developments.” *Nucleic Acids Research* 36 (2008): D419-D425.
Andriy Kryshtafovych, Oleh Krysko, Pawel Daniluk, Zinovii Dmytriv, Krzysztof Fidelis. “Protein structure prediction center in CASP8.” *Proteins*, no. 77 (2009): 5-9.
B. Contreras-Moreira, P.A. Bates. “Domain Fishing: a first step in protein comparative modelling.” *Bioinfomatics* 18, no. 8 (2002): 1141-1142.
Bourne, Philip E. “CASP and CAFASP Experiments and Their Findings.” *Structural Bioinformatics*, 2003: 501-507.
Cuatrecasas, Steven Jacobs And Pedro. “Insulin Receptor: Structure and Function.” *Endocrine Reviews* 2 (1981).
CASP. 12 May 2011. http://predictioncenter.org/ (accessed May 12, 2011).
Carl-Ivar Brändén, John Tooze. *Introduction to protein structure.* Garland Pub, 1999.
Chandonia , John-Mar, et al. “The ASTRAL Compendium in 2004.” *Nucleic Acids Research* 32 (2004): D189±D192.
Chandonia, John-Marc, et al. *The ASTRAL Compendium for Sequence and Structure Analysis.* 5 June 2009. http://astral.berkeley.edu/ (accessed June 6, 2011).
Eddy , Sean R. *HMMER User’s Guide.* March 2010.
Elmar Krieger, Sander B. Nabuurs, Gert Vriend. “Homology Modeling.” *Structural Bioinformatics*, 2003: 509-523.
Dustman, John A. 8 September 2011. http://sourceforge.net/projects/mysql-python/ (accessed July 15, 2011).
D. T. Jones, W. R. Taylor, J. M. Thorton. “A new approach to protein fold recognition.” *Nature* 358 (July 1992).
Dylan Chivian, Tymothy Robertson, Richard Bonneau, David Baker. “Ab Initio Methods.” *Structural Bioinformatics*, 2003: 547-557.
Finn, Rob, and Jody Clements. “The current version of HMMER.” *HMMER.* 28 June 2010. http://hmmer.janelia.org/software (accessed June 28, 2011).
Gough, Julian. *Download SUPERFAMILY Models, Database Dump and Genome Assignments.* 2010. http://supfam.cs.bris.ac.uk/SUPERFAMILY/downloads.html (accessed July 20, 2011).
Gough, Julian. “Genomic scale sub-family assignment of protein domains.” *Nucleic Acids Research* 34, no. 13 (July 2006): 3625–3633.
—. *SUPERFAMILY Description.* 3 May 2011. http://supfam.cs.bris.ac.uk/SUPERFAMILY/ (accessed May 3, 2011).
Gough, Julian. “The SUPERFAMILY database in structural genomics.” *Biological Crystallography*, 2002: 1897-1900.
Gough, J, K Karplus, R Hughey, and C Chothia. “Assignment of homology to genome sequences using a library of hidden Markov models that represent all proteins of known structure.” *J Mol Biol.* 313, no. 4 (November 2001): 903-19.

Gough, J, K Karplus, R Hughey, and C Chothia. “Assignment of homology to genome sequences using a library of hidden Markov models that represent all proteins of known structure.” 313, no. 4 (November 2001): 903-919.

Godzik, Adam. “Fold Recognition Methods.” *Structural Bioinformatics*, 2003: 525-546.

Ilya N. Shindyalov, Philip E. Bourne. *Protein Structure Alignment by Incremental Combinatorial Extension (CE) of the Optimal Path.* San Diego: San Diego Supercomputer Center, 1998.

Helen M. Berman, John Westbrook, Zukang Feng, Gary Gillil, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov and Philip E. Bourne. “The Protein Data Bank.” *Nucl. Acids Res* 28, no. 1 (2000): 235-242.

Herráez, Angel. “Biomolecules in the Computer: Jmol TO THE RESCUE.” *BIOCHEMISTRY AND MOLECULAR BIOLOGY EDUCATION* 34, no. 4 (November 2006): 255–261.

Horrobin, David F. “Innovation in the pharmaceutical industry.” *Journal of the Royal Society of Medicine* 93 (July 2000): 341-345.

Julian Gough, Kevin Karplus, Richard Hughey, Cyrus Chothia. “Assignment of Homology to Genome Sequences using a Library of Hidden Markov Models that Represent all Proteins of Known Structure.” *Journal of Molecular Biology*, 2001: 903-919.

John W Adamson, Clement A.Finch. “HEMOGLOBIN FUNCTION, OXYGEN AFFINITY, AND ERYTHROPOIETIN.” *Annu. Rev. Physiol.*, 1975: 351-369.

Larsson, Per, Björn Wallner, Erik Lindahl, and Arne Elofsson. “Using multiple templates to improve quality of homology models in automated homology modeling.” *Protein Science* 17, no. 6 (January 2009).

Leszek Rychlewski, Daniel Fische. “LiveBench-8: The large-scale, continuous assessment of automated protein structure prediction.” *Protein Science*, September 2005: 240-245.

Nello Cristianini, Matthew William Hahn. *Introduction to computational genomics: a case studies approach.* Cambridge: Cambridge University Press, 2009.

Murzin, Alexey G., E. Brenner Brenner, Tim Hubbard, and Cyrus Chothia. “SCOP: A structural Classification of Proteins Database for the Inverstigation of Sequences and Structures.” *J. Mol. Biol.*, 1995: 536-540.

M S Smyth, J H J Martin. “x Ray crystallography.” *Clin Pathol: Mol Pathol*, 2000: 53:8-14.

“macfuse.” *macfuse.* 6 December 2008. http://code.google.com/p/macfuse/ (accessed June 30, 2011).

Michael Schrader, Peter Schulz-Knappe. “Peptidomics technologies for human body fluids.” *TRENDS in Biotechnology* 19, no. 10 (October 2001): 55-59.

Miles Congreve, Christopher W. Murray, Tom L. Blundell. “Structural biology and drug discovery.” *DDT* 10, no. 13 (July 2005): 895-907.

Morris F. Whites, C. Ronald Kahn. “The Insulin Signaling System.” *The Journal of Biological Chemestry* 269 (January 1994): 1-4.

Pearson, W R, and D J Lipman. “Improved tools for biological sequence comparison.” *PNAS* 85, no. 8 (April 1988): 2444-2448.

Preston-Werner, Tom. *Readme Driven Development.* 23 8 2010. http://tom.preston-werner.com/2010/08/23/readme-driven-development.html (accessed 7 20, 2011).

Protein Data Bank. 8 September 2011. http://www.pdb.org/ (accessed September 8, 2011).

Protein Data Bank. 2011. http://www.pdb.org/pdb/software/rest.do (accessed 08 10, 2011).

“Protein Data Bank Contents Guide: Atomic Coordinate Entry Format Description Version 3.30.” *Worldwide Protein Data Bank.* 2008. ftp://ftp.wwpdb.org/pub/pdb/doc/format_descriptions/Format_v33_A4.pdf (accessed August 5, 2011).

Sali, Andrej. *alignment.append() -- read sequences and/or their alignment.* 29 March 2011. http://salilab.org/modeller/manual/node274.html (accessed July 15, 2011).

—. *Modeller 9.9 Release Notes.* 4 April 2011. http://salilab.org/modeller/9.9/release.html#rpm (accessed June 20, 2011).

Sali, Andrej. *MODELLER: A Program for Protein Structure Modeling.* 12 June 2009.

Schulz, Roland. “Protein Structure Prediction.” 2007, 1-12.

Scop Classification Statistics. 23 February 2009. http://scop.mrc-lmb.cam.ac.uk/scop/count.html#scop-1.75 (accessed August 10, 2011).
Shindyalov, Ilya N. "Databases and Tools for 3-D Protein Structure Comparison and Alignment." *Databases and Tools for 3-D Protein Structure Comparison and Alignment.* 2001. ftp://ftp.sdsc.edu/pub/sdsc/biology/CE/src/ (accessed May 3, 2011).
Structural Classification of Proteins. June 2009. http://scop.mrc-lmb.cam.ac.uk/scop/ (accessed August 10, 2011).
Reznick, Larry. "Using cron and crontab." *Sys Admin* 2, no. 4 (1993).
Richard Bonneau, David Baker. "AB INITIO PROTEIN STRUCTURE PREDICTION: Progress and Prospects." *Annu. Rev. Biophys. Biomol. Struct.*, 2001: 30:173-89.
"The current version of HMMER." *HMMER.* 28 March 2010. http://hmmer.janelia.org/software (accessed June 29, 2011).
The RCSB PDB RESTful Web Service interface. 7 May 2011. http://www.pdb.org/pdb/software/rest.do (accessed May 7, 2011).