HotRegion v2.0: A new method to predict hot regions in protein-protein interfaces

by

Damla Övek

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Computational Sciences and Engineering

August 10, 2020

HotRegion v2.0: A new method to predict hot regions in protein-protein interfaces

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Damla Övek

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Prof. Attila Gürsoy (Advisor)

_____

Assoc. Prof. Mehmet Sayar

_____

Assoc. Prof. Nurcan Tunçbağ

Date: August 10, 2020

To my family…

ABSTRACT

HotRegion v2.0: A new method to predict hot regions in protein-protein interfaces

Damla Övek

Master of Science in Computational Sciences and Engineering

August 10, 2020

Proteins interact with each other through their interface to fulfil essential functions in the cell. The study of protein interactions will have a profound effect on understanding various biological pathways. Binding free energies are not uniformly distributed among the residues found in protein-protein interfaces (PPI). Hot regions are tightly packed residue clusters in PPIs that account for the majority of the binding free energy of proteins, and hence, are crucial for the stability of complexes. Providing specificity to binding sites, these regions are of great importance for drug discovery in pharmaceutical research. Experimental discovery of hot regions is time-consuming and requires high effort. Hence, there is a need for computational methods. The existing hot region prediction algorithms perform clustering on computationally predicted hot spots and ignore the potential existence of non-hot spot residues in hot regions. However, experimental studies have demonstrated that non-hot spot residues can also be found in hot regions. In this thesis, using unsupervised learning approaches, we propose a novel method to predict hot regions which may contain both hot spot and non-hot spot residues. We combine affinity propagation (AP) and density-based spatial clustering of applications with noise (DBSCAN) to cluster interface residues and develop a web-based tool to predict and visualize hot regions. Furthermore, we have developed a database that contains hot region information for more than 600.000 protein complexes. In our web server, users can query data from our database or submit a new run to obtain hot regions of a complex that is not found in the database. Our tool demonstrates a 3D structural visualization of the complex with colored hot regions and highlighted hot spot residues. Structural features of interface residues, i.e., accessible surface area values and knowledge-based pair potentials, are also indicated. In order to evaluate our method, we have compared our results with the experimental studies. The precision of our algorithm is 0.68 and the accuracy is 0.62. Additionally, we have conducted a case study to test the significance of our predicted regions for clinical studies. We have investigated the complex of human programmed death-1 (PD-1) and its ligand PD-L1. Our algorithm correctly identified the residues which are known to be significant for PD-L1-antibodies and small-inhibitors. Lastly, we have compared our algorithm with our previous hot region prediction method. The results have shown that our tool outperforms the previous version of HotRegion and may be a leveraging step to novel pharmaceutical studies.

ÖZETÇE

HotRegion v2.0: Protein-protein etkileşim arayüzlerindeki sıcak bölgeleri tahmin etmek için yeni bir yöntem

Damla Övek

Hesaplamalı Bilimler ve Mühendislik, Yüksek Lisans

10 Ağustos 2020

Proteinler hücre içerisindeki fonksiyonlarını yerine getirebilmek için birbirleriyle etkileşim halindedirler. Bu etkileşimleri ara yüzleri aracılığıyla gerçekleştirirler. Proteinlerin etkileşimlerini incelemek biyolojik yolları anlamak için oldukça önemlidir. Serbest bağlanma enerjileri protein-protein etkileşim ara yüzlerinde bulunan amino asitler arasında eşit olarak dağılmamıştır. Serbest bağlanma enerjisinin büyük çoğunluğuna sahip olan sıkıca paketlenmiş amino asit kümeleri sıcak bölgeler olarak adlandırılır. Bu bölgeler, serbest bağlanma enerjisinin büyük bir kısmına sahip olduklarından protein komplekslerinin sabitlik derecesi açısından çok büyük önem arz ederler. Bağlanma bölgelerine özgüllük sağlayan bu bölgeler, farmasötik araştırmalarda ilaç keşfi için büyük önem taşımaktadır. Sıcak bölgelerin deneysel keşfi zaman alıcıdır ve yüksek çaba gerektirir. Dolayısıyla, hesaplamalı tahmine dayalı yöntemlere ihtiyaç vardır. Mevcut sıcak bölge tahmin algoritmaları, hesaplamalı olarak tahmin edilen sıcak noktalar üzerinde kümeleme gerçekleştirir ve sıcak bölgelerde sıcak olmayan noktaların potansiyel varlığını göz ardı eder. Öte yandan, deneysel çalışmalar sıcak bölgelerde sıcak olmayan noktaların da var olduğunu göstermektedir. Bu tezde, gözetimsiz öğrenme yaklaşımlarını kullanarak hem sıcak nokta olan hem de sıcak nokta olmayan amino asitleri içerebilecek sıcak bölgeleri tahmin etmek için yeni bir yöntem öneriyoruz. Önerdiğimiz yöntemde, amino asitleri kümelemek için afinite yayılımı (AP) ve gürültülü uygulamaların yoğunluk tabanlı mekânsal kümelenmesi (DBSCAN) yöntemlerini birleştirdik ve sıcak bölgeleri tahmin eden ve görüntüleyen web tabanlı bir araç geliştirdik. Ayrıca, 600.000'den fazla protein kompleksi için sıcak bölge bilgilerini içeren bir veri tabanı geliştirdik. Web sunucumuzda, kullanıcılar veri tabanımızdan veri sorgulayabilir veya veri tabanında bulunmayan bir kompleksin sıcak bölgelerinin tahmin edilmesi için algoritmayı çalıştırabilir. Geliştirdiğimiz web tabanlı araç, proteinlerin üç boyutlu yapısını görselleştirir, sıcak bölgeleri renklendirir ve sıcak noktaları vurgular. Ayrıca, ara yüz amino asitlerinin yapısal özelliklerini -göreceli erişilebilir yüzey alanı ve bilgiye dayalı eş potansiyel- de gösterir. Yöntemimizi değerlendirmek için, sonuçlarımızı deneysel çalışmalarla karşılaştırdık. Algoritmamızın doğruluk derecesi 0.62 ve hassasiyeti 0.62. Ayrıca, tahmin edilen sıcak bölgelerin klinik çalışmalar açısından önemini test etmek için bir vaka çalışması yaptık. İnsan programlanmış ölüm 1 ve ligand kompleksini inceledik. Metodumuz PD-L1 antikorları ve küçük inhibitörler için önemli olduğu bilinen amino asitleri doğru bir şekilde saptadı. Son olarak, algoritmamızı önceki sıcak bölge tahmin algoritmamızla karşılaştırdık. Sonuçlara göre, aracımız önceki yöntemimizden daha iyi performans gösterdi ve yeni farmasötik çalışmalar için manivela gücü sağlayabilir.

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to my advisors Prof. Attila Gürsoy and Prof. Özlem Keskin for their guidance. They were always friendly. Their continuous support helped me through my research and writing my thesis. I will always be grateful for being one of their students.

I owe my deepest gratitude to my family for always being there and for making me who I am today. Without their love, emotional support, understanding and endless encouragement I would not be able to pursue this master's program. I have always felt so lucky and proud that I had such wonderful and loving parents.

I would like to thank Batuhan Baydar for being always there for me. His support, motivation, encouragement, and patience are invaluable. He was the one who has most closely shared my greatest moments and also my tears during my master's journey. I feel so lucky to have him.

I could not image a better office environment. Every result described in this thesis was accomplished with the help and support of my officemates. I could never forget their endless support and motivation. Thus, I would like to thank to former and current COSBI lab members for their help, all the amazing times we have shared and all the fun we have had.

I am proud to be a member of Koç University. I am grateful for the financial support and countless facilities that Koç University provided to me. I would also like to express my gratitude to all the professors whom I have taken courses from. The knowledge that I gained from them is priceless and I will always be grateful for that.

Finally, I would like to thank dear committee members Assoc. Prof. Nurcan Tunçbağ and Assoc. Prof. Mehmet Sayar for their valuable time.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

DBSCAN	Density-based Spatial Clustering of Applications with Noise
AP	Affinity Propagation
HS	Hot Spot
NH	Non-hot Spot
ASA	Accessible Surface Area
PDB	Protein Data Bank
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative

Chapter 1:
INTRODUCTION

Proteins have crucial roles in various molecular and cellular processes. They interact with each other through their interaction sites, called the protein-protein interface, to fulfill their function in the cell. The study of the interactions of proteins involved in essential biological pathways will have a profound effect on understanding these biological pathways.

The strength of the binding interaction between a protein and its binding partner, i.e., binding affinity is defined in terms of the binding free energies of individual residues. Binding free energies are not uniformly distributed among the residues found in protein-protein interfaces. Hot regions are tightly packed residue clusters in PPIs that account for the majority of the binding free energy of proteins, and hence, are crucial for the stability of complexes [1]. Providing specificity to binding sites, these regions are of great importance for drug discovery in pharmaceutical research. Investigation of hot spot residues is significant to understand the principles of the interaction of two proteins [2].

Experimentally, hot regions are investigated by alanine scanning mutagenesis combined with X-ray crystallography [3-5]. However, this experimental procedure is time-consuming and requires high effort. Therefore, computational methods are required. In the literature, various algorithms can be found for hot region prediction [1, 6-8]. The existing hot region prediction algorithms perform clustering on computationally predicted hot spots and ignore the potential existence of non-hot spot residues in hot regions. However, experimental studies have demonstrated that non-hot spot residues can also be found in hot regions. Here, we propose a novel approach to fill this gap. The main objective of this study is to develop an algorithm to computationally predict hot regions using unsupervised machine learning approaches. We combine two well-known clustering algorithms: affinity propagation (AP)[9] and density-based spatial clustering of applications with noise (DBSCAN)[10].

Our algorithm, first, extracts interface residues of a protein complex. A protein interface is defined as a region that is composed of a set of residues that binds two protein molecules by non-covalent interactions. For interface interaction, our algorithm

uses the interface definition of Tuncbag et al. According to this definition, two residues, belonging to different binding partners of the complex, are interacting if the distance between any of their atoms is less than the sum of their van der Waals radii plus a 0.5 Å tolerance [11].

The second step of the algorithm is the feature calculation. There are plenty of researches on computational hot spot prediction. According to these studies, knowledge-based pair potentials and relative accessible surface area of residues have the strongest effects on hot spot prediction. Hence, we use these features in our algorithm. We obtain the knowledge-based pair potentials from Keskin et al. [12] and calculate relative accessible surface area using Naccess [13].

Finally, the algorithm clusters residues based on calculated features plus their 3D coordinates. First, clustering is performed by using affinity propagation. Then, from the resulting clusters, the algorithm calculates average values for density parameters required by DBSCAN. After the parameters are calculated, DBSCAN is run and final clusters are obtained.

We prefer to combine two different clustering algorithms for clustering in order to achieve best results. Affinity Propagation is parameter free, and effective to find exemplars in a dataset but not suitable with arbitrary shaped clusters. On the other hand, DBSCAN is suitable for clustering datasets with arbitrary structures. However, it needs density parameters and is not suitable for datasets with varying density clusters. Combining these two algorithms achieve better results than using one of them alone [14].

We compared our predictions with experimentally found hot regions [3-5]. Results have demonstrated that our predictions are highly correlated with experimental findings. The precision of our algorithm is 0.68 and the accuracy is 0.62.

Additionally, we introduce a web-based tool which demonstrates predicted hot regions and provides detailed structural information about the protein interface in question with a friendly user interface. Furthermore, we have developed a database which contains hot region information for 641.061 protein complexes. In our web server, users can query data from our database or submit a new run to obtain hot regions of a complex not found in the database. Our tool demonstrates a 3D structural visualization of the complex with colored hot regions and highlighted hot spot residues.

Furthermore, we have conducted a case study to test the significance of our predicted regions for clinical researches. We have shown that the hot regions predicted by our algorithm contains residues which are important for designing small inhibitor molecules to interfere PD-1/PD-L1 complex [15].

Lastly, we have compared our algorithm with our previous hot region prediction method [1]. The results have shown that our tool outperforms the previous algorithm and may be a leveraging step to novel pharmaceutical studies.

The remaining part of the thesis is formed from the following chapters: Chapter 2 presents a comprehensive review of studies focusing on experimental and computational investigation of hot regions, machine learning based hot spot prediction, as well as unsupervised learning-based clustering methods. This chapter provides background information on unsupervised learning, clustering algorithms, AP, DBSCAN, and protein interfaces.

In chapter 3, methodology of our computational hot region prediction algorithm is explained in detail. In a given protein complex, first, interface residues are extracted. Then, features are calculated. After feature calculation, a distance matrix is formed. We use 3D coordinates, knowledge-based pair potentials and accessible surface area values of residues as our features. Once the features are calculated, affinity propagation algorithm is applied for finding clusters' exemplars. Then, we form a density list and use these densities to calculate optimal parameters for DBSCAN. Finally, we predict hot regions.

Chapter 4 explains the details of an experiment conducted on nine protein complexes (PDB IDs: 1JTG, 1A0O, 1A22, 1BRS, 1EMV, 1IBR, 2C5L, 2WPT, 4G0N). For these complexes, we have ground truths from previous experimental studies [3-5]. Thus, we evaluate our clusters using ground truths. In this chapter, we show our evaluation metrices. In addition, we perform a case study to demonstrate the significance of hot regions in medical studies [15]. Lastly, we carry out a comparison study of our tool with our existing hot region prediction method. Results are followed by a short discussion.

Finally, the thesis is concluded with the major conclusions and the future directions of the study.

Chapter 2:
LITERATURE REVIEW

This chapter presents an extensive literature review of related work in the field. First, we give general information about protein interfaces. Then, we explain the most current computational hot spots and hot region prediction methods. We continue by presenting clustering algorithms and exploring unsupervised learning. This chapter concludes with a comprehensive summary about previous studies related to affinity propagation (AP) and density based spatial clustering of applications with noise (DBSCAN).

2.1 Protein-Protein Interaction

Protein-protein interactions (PPI) are crucial for biological processes in all living cells. Identifying protein interfaces provides a better insight of infection mechanisms and, hence, is potentially significant for pharmaceutical uses and drug development. Here, we explain state of the art PPI prediction techniques, and we discuss related challenges and current issues on this topic.

Experimental techniques identify the physiochemical interactions of proteins in order to predict the functional relationships between them. These techniques are yeast two-hybrid based methods [16], mass spectrometry [17], Tandem Affinity Purification [18], protein chips [19], and hybrid approaches [20]. However, they are expensive, time consuming, labor intensive, and unable to cover the whole PPI network. Therefore, there is a need for computational techniques. Computational prediction of PPIs is an extensively studied topic, and many approaches have been proposed. We can categorize these approaches, based on the protein features which they use, into: Sequence-based techniques and structure-based techniques.

In sequence-based techniques, the features of amino acids (aa) are utilized. These techniques can be further classified as statistical methods and machine learning (ML)-based methods. Table 1 and Table 2 display summaries of statistical and ML-based methods respectively.

Table 1:Statistical Sequence-based PPI prediction approaches [21].

Approach	Extracted Features	Technique/Tool	Datasets
Mirror Tree (Pazos and Valencia 2001) [22]	Similarity of phylogenetic trees	Evolutionary distance, McLachlan AA homology matrix	*Escherichia coli* protein (Dandekar *et al.* 1998) [23]
PIPE (Pitre *et al.* 2006) [24], PIPE2 (Pitre *et al.* 2008) [25]	Short AA polypeptides	Similarity measure	Yeast protein (DIP and MIPS)
Co-evolutionary Divergence (Liu *et al.* 2013) [26]	Co-evolutionary information,	Pairwise alignment, ClustalW2	Human protein (Matsuya *et al.* 2008 [27], Prasad *et al.* 2009 [28])

Table 2: Machine Learning Sequence-based PPI prediction approaches [21].

Approach	Extracted Features	Technique/Tool	Datasets
Auto Covariance (Guo *et al.* 2008) [29]	AA physicochemical properties	Auto covariance, SVM	Yeast protein (DIP and MIPS)
Pairwise Similarity (Zaki et al 2009) [30]	Pairwise similarity	SVM	Yeast protein
AA Composition (Roy *et al.* 2009) [31]	AAC	Logistic regression, SVM, Naive Bayes	Yeast protein (GRID), worm protein (Li *et al.* 2004) [32], fly protein (Biogrid)
AA Triad (Yu *et al.* 2010) [33]	AA triad information	RVKDE	Human protein (HPRD)
UNISPPI (Valente *et al.* 2013) [34]	Frequency and composition of AA physiochemical properties	Decision tree	Twenty different eukaryotic species
ETB-Viterbi (Kern *et al.* 2013) [35]	AA residues	Hidden Markov models, Early Traceback Viterbi	3DID database

Alternatively, in structure-based methods, three-dimensional structural features are used [36]. Structure-based methods include template-based, statistical, and ML-based methods. The structure-based methods are summarized in Table 3.

Table 3: Structure-based PPI prediction approaches [21].

Approach	Extracted Features	Technique/Tool	Datasets
PRISM (Tuncbag *et al.* 2011 [37])	Interaction surface of crystalline complex structures	Naccess, MultiProt, Fiber-Dock	Human Protein (Ozbabacan et al. 2012 [38], Tuncbag *et al.* 2009 [39])
PrePPI (Zhang *et al.* 2012) [40]	Secondary structure	Bayesian networks, Naive Bayes	Yeast protein, Human protein
PID Matrix Score (Kim *et al.* 2002) [41]	Potentially interacting domain pairs	PID matrix	DIP, InterPro, TrEMBL/SwissProt
PreSPI (Han *et al.* 2003 [42], 2004 [43])	Domain combination interaction probability	Interacting probability equation	Yeast protein (DIP), PDB
DCC (Jang *et al.* 2012) [44]	Intra-protein and inter-protein domain interactions	Interaction significance matrix	S. cerevisiae protein, Pfam
MEGADOCK (Ohue *et al.* 2013a)	Shape complementarities, electrostatics, and hydrophobic interactions	rPSC, ZRANK	Bacterial protein (Ohue *et al.* 2012, Matsuzaki *et al.* 2013)
Meta Approach (Ohue *et al.* 2013b)	Interaction surface of crystalline complex structures, shape complementarities, electrostatics, and hydrophobic interactions	PRISM, MEGADOCK	Human protein
Random Forest (Chen and Liu 2005)	Existence of similar domains	Random Forest	DIP, Deng *et al.*, Schwikowski *et al.*, Pfam
Struct2Net (Singh *et al.* 2006, 2010)	Homology with known protein complexes in PDB	Logistic regression	Yeast, Fly, and Human protein

2.2 Hot Spot Residues

Some residues in protein interfaces account for the majority of the binding energy even though they comprise only a small fraction of the interface. These residues

are called “hot spots”. Hot spot residues are essential for understanding protein interaction principles. [2]

Previous studies have demonstrated that the amino acid composition of hot spots is not random. The amino acids which have the highest probabilities of occurrence are tryptophan (21%), arginine (13.3%), and tyrosine (12.3%). [45]

Experimentally, hot spot residues are found by alanine scanning mutagenesis. First, residues are mutated to alanine which is the most basic amino acid. Then, the change in binding free energy (ΔΔG) is calculated. If the residue in question is a hot spot, then ≥ 2.0 kcal energy change occurs. However, alanine scanning mutagenesis requires intensive time and effort. Therefore, various computational methods have been developed for hot spot prediction. For the computational prediction of hot spots, researchers exploit the physicochemical properties of amino acids (hydrophobicity, hydrophilicity, polarity, and average accessible surface area), evolutionary information in terms of evolutionary conservation score, the position-specific scoring matrix (PSSM), and other sequence descriptors.

In 2007, Kortemme *et al.* introduced a server, Robetta, for the prediction and analysis of protein structures. Robetta is able to provide ΔΔG value for the mutation of each residue to alanine in a given protein structure. [46] Darnell *et al.* proposed a method (KFC) to predict hot spots by a machine learning approach in 2008. The KFC server characterizes the local structural environment for each residue in the interface and compares the environments with the experimentally found hot spots’ environments to make the prediction. [47] In 2009, Tuncbag *et al.* proposed an empirical formula using the relative accessible surface area of residues in the complex state and knowledge-based pair potentials [2] and developed a web server, called HotPoint [11].

Over the last ten years, numerous ML-based methods have been proposed. These methods have been using structure-based, sequence-based, and/or energy-based features. In order to find the strongest features to achieve best accuracy, feature selection algorithms, such as F-score, random forest, support vector machines–recursive feature elimination (SVM-RFE), minimum redundancy maximum relevance (mRMR), and maximum relevance maximum distance (MRMD) have been used. For future extraction, linear discriminant analysis (LDA) and principal component analysis (PDA) are the most commonly used methods. In the prediction model, frequently used machine learning approaches are nearest neighbor, support vector machines (SVM), decision

trees, Bayesian networks, neural networks, and ensemble learning. Studies have shown that best predictive results are achieved by using one of the ensemble learning algorithms, gradient tree boosting (GTB), and by combining ASA related features and PSSM (ASA+PSSM) [48].

2.3 Hot Regions

Research on protein interfaces and hot spots has reported that hot spots are not randomly located on the interfaces; instead, they are found as clusters. These energetically important clusters are known as hot regions or hot segments. Taking this knowledge into consideration, for understanding protein interactions, hot regions provide better information than hot spots solely. Therefore, hot regions have potential significance for discovering treatments for diseases. As in the case of hot spots, experimental discovery of hot regions is time as well as labor intensive. Thus, computational techniques are required.

In 2011, Cukuroglu *et al.* proposed a technique for hot region prediction. HotRegion is a web-based server and database which predicts and shows hot spots and hot regions. For hot spot prediction, it uses the same formula as the HotPoint server does [1]. HotRegion predicts hot regions by a simple graph algorithm. It constructs a graph where hot spots are nodes, and there is an edge between two nodes if two hot spots are close enough to interact. After graph formation, the algorithm finds subgraphs containing at least three nodes and labels these subgraphs as hot regions. A weakness of this algorithm is that it merely considers hot spots, but it does not note the other residues which may highly contribute to the binding free energy.

In 2016, a machine learning-based hot region prediction method was developed by Lin et al. [7] Their algorithm uses support vector machines (SVM) based on an ensemble learning system to predict hot spots. Precisely, they combined SVM and AdaBoost. They utilized minimum-redundancy-maximum-relevance (mRMR) algorithm so as to select features for their classification. They stated that evolutionary conservation score is one of the most important features of residues for hot spot prediction. After classifying hot spot and non-spot residues, they found hot regions by using the local community structure detecting algorithm based on the identification of boundary nodes (LCSDA).

Jing Hu and Xiaolong Zhang proposed density-based clustering for hot region prediction [49]. First of all, they tried clustering residues and then removing non hotspots from clusters. However, they faced the struggle of finding optimal density parameters. Therefore, they could not achieve good accuracy. They tried to address this issue with various approaches. In 2015, they combined neighbor residues with interface residues to achieve improve accuracy [49]. However, their results were still not satisfactory. Later that year, they proposed using incremental clustering with parameter selection. In this approach, they, first, fixed density and enumerate radius of clusters. Then, they reversed their technique and enumerated density by fixing radius. By using parameter tuning, they could achieve slight improvement in accuracy [6].

In 2018, they have analyzed hot region predictions with different amino acid mutations [7]. They suggested that distinct mutations would result in various energy changes and hence, different feature selections would be required for predictive models in the presence of mutations. Results have demonstrated that RctASA, hydrophobicity, BpASA and BminCX are the most significant features for distinguishing hot spots and non-hot spots. Additionally, they concluded that their model performs better in the presence of Leu, Gly and Pro mutations.

2.4 Clustering Algorithms

Clustering is defined as the process of grouping similar objects together. The objective of clustering algorithms in unsupervised learning is to find data points with similar features and group them. There are a wide variety of algorithms developed to cluster unlabeled data points.

2.4.1 K-means Clustering

K-means clustering is one of the oldest but most frequently used clustering algorithms. Once developed, it was a computationally expensive procedure. However, it has been improved throughout the years. Input for k-means clustering algorithm is “k” whose definition is the number of clusters desired to be found. This algorithm first places k centroids in random locations on a given data set. Secondly, it assigns data points to the closest clusters using Euclidean distance between data points and

centroids. Then, it redefines each cluster's center as a mean of data points belonging to that cluster. This procedure is repeated until no change occurs.

Mean of each cluster is called the centroid of that cluster. The centroids, usually, do not correspond to any point in the given dataset but live in the same space. The algorithm aims to select centroids so that it minimizes a criterion called inertia or within-cluster sum-of-squares. Inertia is a measure of how internally coherent the clusters are. However, there are two major disadvantages of this approach:

1. Inertia assumes that clusters are isotropic and convex which not always the case. Thus, it could not perform good when the clusters are elongated or have irregular shapes.
2. Inertia is not a normalized metric. The smaller it is, the better the results are. If it is zero, then the results are optimal. However, in high dimensional spaces, Euclidean distances are inflated. In this case, a dimensionality reduction algorithm such as Principal Component Analysis (PCA) can be required for better performance.

Besides these disadvantages, K-means Clustering is not useful for the data sets with unknown number of clusters [50].

2.4.2 Affinity Propagation

Affinity propagation (AP) is another clustering algorithm which was first introduced by Frey et al. In this algorithm, there is no need to know how many clusters there are in a data set. The algorithm is based on messaging among data points. [19]

AP's input is the similarities between the data points. Based on certain criteria, this algorithm finds representatives for each cluster, named "exemplars". Then, until a consensus is obtained, the data points exchange messages among themselves.

AP is useful since it does not require number of clusters in a given dataset. However, it is a computationally expensive algorithm. Time complexity is O(NT) where N is the number of data points and T is number of iterations until convergence is achieved. The memory complexity of the algorithm is also high, which is $O(N^2)$. Therefore, AP is useful for small and medium size data sets whereas it is not useful for large data sets [9].

2.4.3 Mean Shift

The objective of Mean Shift clustering is to identify blobs in a smooth density of samples. Similar to previously explained clustering algorithms, mean shift clustering is based on centroids. It updates centroids until it converges to the mean of the points within a given region. The algorithm has a post-processing step to filter near-duplicates and form the final set of centroids.

Unlike k-means clustering but like affinity propagation, mean shift automatically finds the number of clusters. On the other hand, it suffers from the drawback of not being highly scalable. During the execution of the algorithm, nearest neighbor search is performed multiple times [51].

2.4.4 Spectral Clustering

In Spectral Clustering, the first step is dimensionality reduction using the eigenvalues of the given dataset's similarity matrix. The second step is clustering the data in lower dimensional space by means of K-means clustering approach.

Spectral clustering is a popular approach in graph theory which enables identification of communities of nodes in a graph according to the edges connecting them. It is a flexible algorithm and can be used for clustering non graph data as well.

The recent version of the algorithm needs the number of clusters as input since it uses k-means for clustering. Its performance is satisfactory for a data set having small number of clusters. However, it is not advised for dataset with large number of clusters [52].

2.4.5 Hierarchical Clustering

Hierarchical clustering is the name of a clustering algorithms family which forms nested clusters by merging or splitting them successively. In hierarchical clustering, a dendrogram represents the hierarchy of clusters. The leaves of the dendrogram corresponds to clusters with one data point whereas the root corresponds to the cluster with all data points. Clustering can be performed by a top-down or bottom-up approach.

Agglomerative Clustering is an example of hierarchical clustering using a bottom up approach. The final clusters are formed by successive merges. There are multiple

criteria which can be used as merging strategy: ward, maximum (complete) linkage, average linkage, and single linkage.

Ward aims to minimize variance similar to k-means. It minimizes the sum of squared distances within all clusters. Maximum or complete linkage works to minimize the maximum difference between clusters. As the name suggests, the average distance among clusters are minimized in average linkage approach. Finally, in single linkage, the closest pair of data points between clusters are minimized.

Agglomerative Clustering is scalable and thus, can be used for large number of samples. However, it becomes computationally expensive when no connectivity constraints exist between samples and hence, the algorithm requires to consider all possible merges in each iteration [53].

2.4.6 Density-based Spatial Clustering of Applications with Noise

Recently, various unsupervised clustering methods have emerged from the framework of k-means clustering including density-based spatial clustering of applications with noise (DBSCAN) [10]. Density-based Spatial Clustering of Applications with Noise is an efficient clustering algorithm for spatial data sets. This algorithm has two inputs: epsilon (eps) and a minimum number of points (minPoints). Epsilon is a radius which is used to draw an imaginary circle around a data point and find nearby points. Using epsilon value, there has to be at least minPoints data points close to each other in order to define a cluster. By default, DBSCAN algorithm uses Euclidean distance. However, it also works for data sets with predefined distance matrices.

MinPoints parameter highly depends on the data set. A general approach to finding minPoints is that it should be at least equal to n+1 where n is the number of dimensions of data points. However, optimally, different minPoints parameters should be utilized for varying density clusters. Otherwise, algorithm may misidentify some small clusters as noise. In order to prevent misidentification, either the algorithm must be modified or the most representative minPoints parameter must be used.

Finding optimal epsilon is crucial for forming adequate clusters. DM/DBSCAN is an algorithm which finds an optimal epsilon for a dataset and clusters points by DBSCAN with found eps [54]. In order to find epsilon, the algorithm first finds kth

nearest neighbor of each data point. Then, it sorts distances in ascending order. In the graph of kth nearest neighbor vs. data point, critical point gives the optimal value.

One of the advantages of DBSCAN is the ability to differentiate noise points which do not belong to any of the clusters. On the other hand, one epsilon and one minPoints may not be representative enough for clusters with varying densities. Therefore, DBSCAN does not work efficiently on datasets with various densities. Furthermore, DBSCAN algorithm struggles to separate clusters which are close to each other [14].

2.4.7 Hierarchical Density-based Spatial Clustering of Applications with Noise (HDBSCAN)

Hierarchical DBSCAN is an extended version of DBSCAN to hierarchical clustering. It is developed by Campello *et al*. [55] HDBSCAN runs DBSCAN with varying epsilon. Then, the results are integrated to find a clustering that is the most stable one over the epsilon. HDSCAN is developed to address the issue of DBSCAN on datasets with varying density clusters and it is more robust to parameter selection.

2.4.8 OPTICS

The OPTICS algorithm is very similar to DBSCAN algorithm. The main difference is that the OPTICS takes the epsilon parameter as a range instead of a single value. It builds a reachability graph. It assigns two attributes to each sample which are reachability distance and the cluster ordering. These attributes are used for determining cluster membership.

The reachability distance allows OPTICS to cluster data sets with varying density clusters. However, DBSCAN has a shorter execution time as OPTICS runs repetitively for varying eps values. On the other hand, a single OPTICS run is shorter than the cumulative run time of DBSCAN multiple times.

In OPTICS, spatial indexing trees are used, which eliminates the need for calculating the full matrix distance. Therefore, OPTICS is more efficient on large data sets in terms of memory usage when compared to DBSCAN.

OPTICS and HDBSCAN produce similar results on large datasets. Although OPTICS is more efficient in terms of memory complexity, HDBSCAN is better in terms of time complexity since it is multithreaded [56].

2.4.9 Birch

Birch clusters datasets using a graph-based approach. It forms a tree called Clustering Feature Tree (CFT). The nodes of the tree (CF Nodes) carry lossy compressed information of data points. The CF Nodes contain sub clusters called Clustering Feature sub clusters (CF Sub clusters).

CF sub clusters contain the following information and allow elimination of unnecessary data:

- Number of samples in a subcluster.
- Linear Sum - A n-dimensional vector holding the sum of all samples
- Squared Sum - Sum of the squared L2 norm of all samples.
- Centroids - To avoid recalculation linear sum / n_samples.
- Squared norm of the centroids.

Birch can be considered as a data reduction algorithm. If the number of clusters is given as an input parameter, the compressed data is fed into a global clusterer. If the number of clusters is not specified, the sub clusters from the leaves are directly obtained as cluster labels and data points are assigned to nearest clusters [57].

2.4.10 APSCAN

Chen *et al.* developed a new clustering algorithm combining AP and DBSCAN in 2011[14]. As stated above, DBSCAN is insufficient for datasets having varied density clusters. APSCAN algorithm aims to solve this issue.

The APSCAN algorithm introduces a modified version of DBSCAN called double-density-based SCAN (DDBSCAN). The DDBSCAN algorithm takes two set of density parameters as input; in other words, it takes two sets of epsilon-minPoints pair. The purpose of the DDBSCAN is to find clusters with each set of parameters independently and come to a better conclusion of the correct clusters.

In the APSCAN algorithm, local densities are detected first, and a normalized density list is created by affinity propagation (AP). Then, each density parameter pair is combined with another from the normalized density list and given to DDBSCAN to produce clusters. In the end, various clusters with varying parameters are obtained. Lastly, these clusters are synthesized into a final result.

In summary, the two major advantages of the APSCAN algorithm are (i) working without predefined input parameters and (ii) having the ability to cluster data with varying densities while preserving the nonlinear data structure for such sets. Therefore, APSCAN algorithm works better for the data sets with varying density clusters which are very common in computational biology and other real-world applications.

In this thesis, we have used a simpler version of APSCAN. First, we form clusters using AP. Then, we generate a density list. Instead of running DBSCAN multiple times, we run it only once with the average of epsilon values and minimum of minPoint values in the density list. We use affinity propagation to determine DBSCAN parameters. We prefer to use DBSCAN over OPTICS or HDBCAN as they are optimized for large data sets. The nature of our problem involves small datasets, i.e., protein interfaces. Therefore, with a wise selection of parameters, DBSCAN produces satisfactory results in shorter execution time. Details of the algorithm is explained in the following section.

Chapter 3:
METHODOLOGY

In this chapter, the materials and methods for design and implementation of our new hot region prediction web server are explained. The section starts with the preview and a brief commentary on main functionalities that user can find at our web server, and this is followed by step-by-step explanation of the inputs, processing steps and output features in the methodology workflow. This is followed by software architecture of the system, database management, and web-based properties related to interface visualization.

3.1 HotRegion v2.0 Web Server

HotRegion v2.0 is an automated pipeline tool to predict hotspots and hot regions on a protein-protein interface and visualize the 3D structure of the complex. The web page is connected to a database which contains hot region information of 641.061 protein complexes and is continuously growing as users submit new jobs to the server.

Figure 1: HotRegion v2.0 home page

The main functions that users can take advantage of by using our web-based tool can be listed and summarized as follows:

- *Users can retrieve hot region information from the database.* For each protein complex, the database contains PDB ID, keywords, gene ontology and molecular function data. Using PDB ID or with a keyword-based search, users can obtain hot region information of the protein complex in question. The database stores hot region information as a json object for each protein complex. The json object contain interface residues, their structural properties, hot spot information and the hot region cluster identifier if a residue resides in a hot region or -1 otherwise.
- *Users can obtain hot region information for the complexes that are not found in the database.* When the protein complex of interest does not exist in the database, users can run the hot region prediction algorithm on the complex. The algorithm requires structural information of the complex in PDB format. In this case, either the user provides the structure file in PDB format or specifies a PDB ID and two chain identifiers to obtain the structure file from Protein Data Bank. The algorithm, first, extracts interface residues. Secondly, it calculates structural properties of residues such as accessible surface area and knowledge-based pair energies and then, clusters residues. Once hot regions are obtained, the database updates itself.
- *Users can retrieve structural properties of interface residues.* The algorithm calculates accessible surface area of residues in the monomer state and in the complex state. Relative accessible surface area values are also calculated for both states. In order to calculate accessible surface area, the algorithm uses an external algorithm called Naccess [61]. Knowledge-based pairwise energies [12] for the residues are also calculated by the algorithm. These structural properties are stored in the database with the hot region information. Therefore, in addition to the hot region information, users can access structural properties of residues by using our tool.
- *Users can see a list of hot spot residues.* Hot spots have great importance for understanding interface stability. They are found in clusters at protein-protein interfaces. Therefore, hot regions are rich in hot spots. Researchers who study protein interactions tend to investigate hot spots in hot regions. For this reason, we predict hot spots using HotSpot algorithm [11], store hot

spot information in our database and highlight hot spots in our hot regions during 3D visualization.

- *Users can retrieve gene ontology annotations for each protein.* In the previous version of HotRegion database, users can query hot region information based on PDB ID. However, this approach is not sufficient for every case. For example, a researcher may be studying a pathway which includes various proteins. To obtain hot region information for all complexes which involve in the pathway, the user has to find PDB IDs of all complexes and enter them to the system one by one. This is time consuming. In order to improve the capability, we come up with a more user-friendly approach for the HotRegion v2.0. In addition to PDB ID, we store GO annotations obtained from UniProt. Thus, users can query complexes with a keyword-based search. For the given example, when the user enters the name of the biological pathway, all proteins involving in the pathway are shown as a list and the user can obtain hot region information for all proteins by only clicking the corresponding entry from that list.
- *Users can view 3D structural models of protein complexes.* For selected PDB-PDB interaction, users can visualize the complex structure, can identify the interface and non-interface residues, view hot regions and hot spot residues.
- *Users can download the results.* The tool generates a file which shows all calculated features, hot spot and hot region information. Users can download this file and use it with other tools afterwards.
- *Users can retrieve jobs.* When a user submits a job to predict hot regions, the results are stored in the database with a unique job identifier. The returning users can query the results of their previously submitted jobs by entering their email addresses and job ids.

3.2 Workflow of Methodology

Our methodology stages have been tailored to the needs of researchers working on protein-protein interactions. As the experimental findings are available for a very limited number of complexes and experimental procedures are time consuming as well

as labor intensive, many researchers are going towards computational methods and tools to analyze protein complexes. Proteins tend to interact with other proteins to fulfill their functions. These interactions occur throughout interfaces. The binding specificity and affinity of protein interfaces are determined by the properties of residues which form the interface. For protein complex formation, correct orientations of residues are required.

Analyzing the distribution of the residues across the interface and the interactions among them help us understand how proteins recognize their binding partners. The interface residues have tendency to cooperate during the interactions and they form modules in the interface. These modules provide specificity and binding affinity to the proteins and their combinations yield a powerful mechanism for a protein to bind its multiple partners through unique interfaces.

Hot regions are the clusters of interface residues. As shown in previous studies, they are useful for interpretation of interface features. In this thesis, we present a web-based tool to illustrate the residue cooperativity information at protein-protein interfaces. We have a database which stores protein–protein interfaces which are extracted from Protein Data Bank (PDB). This database dynamically updates itself according to the search queries submitted by users. When a user searches for a protein complex which does not exist in the database, the structure file is downloaded from PDB, hot regions are found by the algorithm and the results are saved to database. With our tool, we aim to help researchers detect the functionally important interface residues and cooperativity between them, investigate the stability of the complexes, and understand the specificity of the interfaces.

In a protein complex, the interface is defined as a region where the residues of two proteins are in contact. Two residues from each protein are contacting if the distance between any two atoms, belonging to these residues is less than the sum of their van der Waals radii plus 0.5 Å. Protein-protein interfaces are composed of hot spot and non-hot spot residues. Hot spot residues are the residues whose relative accessible surface area (ASA) are less than or equal to 20 and knowledge-based pair potentials in absolute value is higher than 18. In order to define hot regions, interface residues are clustered using density-based clustering of applications with noise (DBSCAN) algorithm. This algorithm requires two parameters to define the densities of clusters. Optimal parameters for DBSCAN are found with the help of another clustering algorithm, called affinity propagation (AP). Affinity propagation, without extra parameters, forms

clusters and finds exemplars of these clusters. The optimal density parameters for DBSCAN are calculated from the clusters found by AP. Although affinity propagation can form clusters without parameters, it cannot produce satisfactory results for clusters with arbitrary shapes. Therefore, DBSCAN is required to find hot regions correctly.

The subsequent sections follow the workflow of the methodology in Figure 2 and explains the input data which comes from the user, the processing steps of HotRegion v2.0, and the output features in detail.

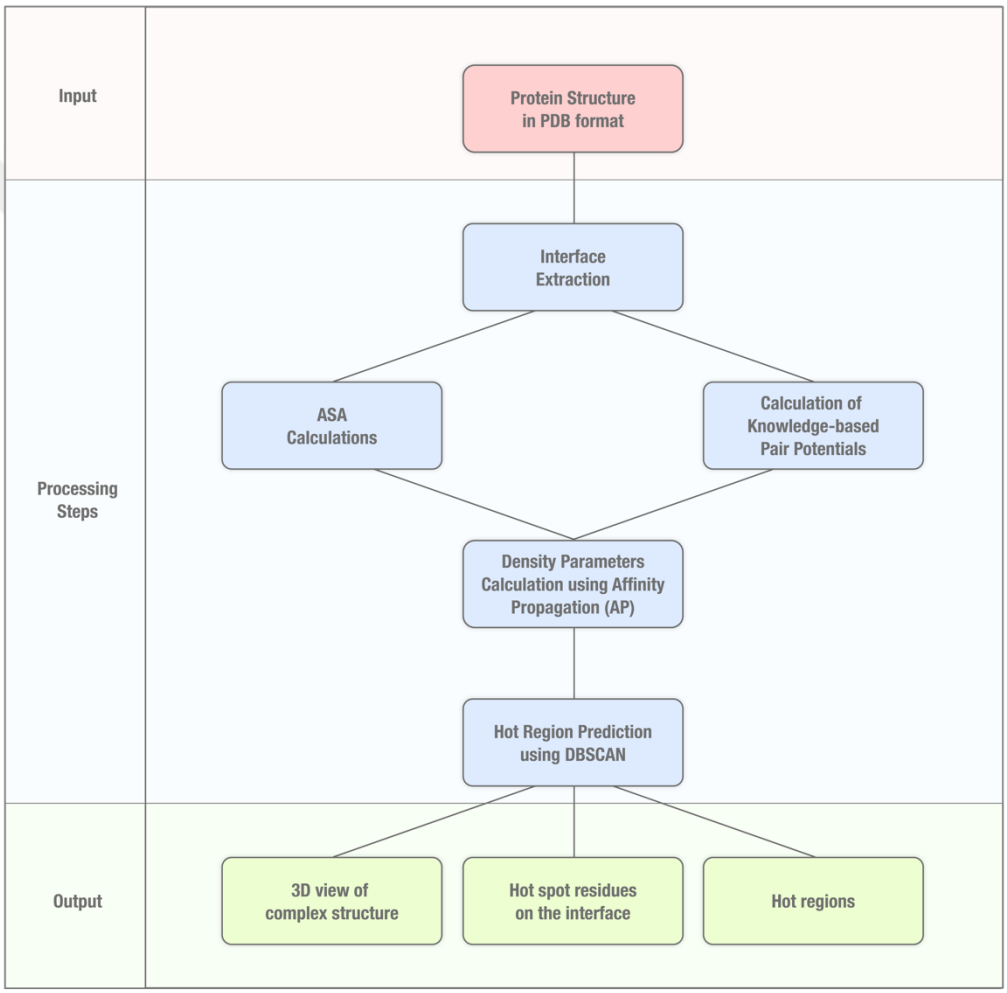

Figure 2: HotRegion v2.0 Framework

3.2.1 Inputs

The algorithm takes two inputs: Interface identifier and PDB file. Interface ID is a 6-letter identifier which is obtained by merging 4 letter PDB ID and two chain identifiers. The interface identifier of the barnase-barstar complex is, for instance,

1BRSAD where 1BRS is the PDB ID of the complex and it contains two chains; chain A and chain D.

In order to calculate features of residues and cluster them, the algorithm requires structural information of the complex. The Protein Data Bank stores structural data of macromolecules which are obtained from X-ray diffraction and NMR studies. The data is available in PDB format which is a standard representation for macromolecule structures. Since its creation in 1970, PDB format has been widely used by a vast amount of software [58].

```
         1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
ATOM     32  N   AARG A  -3      11.281  86.699  94.383  0.50 35.88           N
ATOM     33  N   BARG A  -3      11.296  86.721  94.521  0.50 35.60           N
ATOM     34  CA  AARG A  -3      12.353  85.696  94.456  0.50 36.67           C
ATOM     35  CA  BARG A  -3      12.333  85.862  95.041  0.50 36.42           C
ATOM     36  C   AARG A  -3      13.559  86.257  95.222  0.50 37.37           C
ATOM     37  C   BARG A  -3      12.759  86.530  96.365  0.50 36.39           C
ATOM     38  O   AARG A  -3      13.753  87.471  95.270  0.50 37.74           O
ATOM     39  O   BARG A  -3      12.924  87.757  96.420  0.50 37.26           O
ATOM     40  CB  AARG A  -3      12.774  85.306  93.039  0.50 37.25           C
ATOM     41  CB  BARG A  -3      13.428  85.746  93.980  0.50 36.60           C
ATOM     42  CG  AARG A  -3      11.754  84.432  92.321  0.50 38.44           C
ATOM     43  CG  BARG A  -3      12.866  85.172  92.651  0.50 37.31           C
ATOM     44  CD  AARG A  -3      11.698  84.678  90.815  0.50 38.51           C
ATOM     45  CD  BARG A  -3      13.374  85.886  91.406  0.50 37.66           C
ATOM     46  NE  AARG A  -3      12.984  84.447  90.163  0.50 39.94           N
ATOM     47  NE  BARG A  -3      12.644  85.487  90.195  0.50 38.24           N
ATOM     48  CZ  AARG A  -3      13.202  84.534  88.850  0.50 40.03           C
ATOM     49  CZ  BARG A  -3      13.114  85.582  88.947  0.50 39.55           C
ATOM     50  NH1AARG A  -3      12.218  84.840  88.007  0.50 40.76           N
ATOM     51  NH1BARG A  -3      14.338  86.056  88.706  0.50 40.23           N
ATOM     52  NH2AARG A  -3      14.421  84.308  88.373  0.50 40.45           N
```

Figure 3: An example snapshot from a PDB file

The PDB file of a protein contains various information including the source study, primary and secondary structures of the protein, and atom coordinates. Figure 3: An example snapshot from a PDB file demonstrates atom coordinates section of a PDB file, which our hot region prediction algorithm particularly uses. The meanings of each column are explained in Table 4.

Table 4: PDB file atom coordinates section format explanation

COLUMNS	DEFINITION
7-11	Atom serial number
13-16	Atom name
17	Alternate location indicator
18-20	Residue name
22	Chain identifier
23-16	Residue sequence number
27	Insertion code
77-78	Element symbol
79-80	Charge on the atom

In case the alternative locations are exist for atoms in the PDB file, the algorithm considers only the first location. The first model is utilized for NMR structures. The algorithm only works for protein complexes. Therefore, it returns no information for DNA and RNA structures.

3.2.2 Interface Extraction

A protein interface is defined as a region which is composed of a set of residues that binds two protein molecules by non-covalent interactions. For interface interaction, our algorithm uses the interface definition of Tuncbag et al. According to this definition, two residues, belonging to different binding partners of the complex, are interacting if the distance between any of their atoms is less than the sum of their van der Waals radii plus a 0.5 Å tolerance [11]. The tolerance value can be manipulated by users through our web-based tool.

As shown in Figure 3, the coordinates of atoms are given in the PDB file. The algorithm parses PDB file using PBD Parser module of Biopython library [59]. Using this module, the residues of a chain (a partner in the complex) can be extracted easily. The code spinet below (**Error! Reference source not found.**) demonstrates this procedure.

Algorithm 1: Extracting residues by chain id using Biopython

```
import Bio.PDB
parser = PDBParser()
structure = parser.get_structure('PDBStructure', "PDBFile.pdb")
model = structure[0]
chain_1 = model[chain_id_1]
chain_2 = model[chain_id_2]
```

After the extraction of residues by chain id, the algorithm loops through the residues of first chain and checks if one of the atoms of this residue is interacting with an atom of another residue from the second chain. When an interacting pair is found, both residues are added to a list and loop continues with the next residue.

Algorithm 2: Algorithm to check whether two atoms are interacting

```
def is_atoms_interacting(self, atom_1, atom_2):
    r1 = dictionaries.VDW_RADII[atom_1.get_parent().get_resname()][atom_1.get_id()]
    r2 = dictionaries.VDW_RADII[atom_2.get_parent().get_resname()][atom_2.get_id()]
    distance = abs(atom_1 - atom_2)
    return True if distance < (r1+r2+0.5) else False
```

Algorithm 2 indicates how the algorithm finds interacting atoms coming from two different residues belonging to different chains. VDW_RADII is a dictionary which contains radius of all atoms which may be found in residues. Full code of interface extraction module can be found in the Appendix section.Inputs

3.2.3 Feature Calculation

The interface extracted by interface extraction module is sent to feature calculation module. Feature calculation module has two sub modules which are ASA calculator and knowledge-based pair potential calculator.

One of the core concepts in machine learning is feature selection. Feature selection has great impact on the performance of learning model. It aims to eliminate redundant and/or irrelevant features to decrease complexity and increase model performance. In recent decades, protein-protein interaction and hot spot prediction studies depend highly on machine learning approaches. Therefore, feature selection is essential for these studies to achieve good results. Feature selection studies on protein interfaces have

demonstrated that using the combination of ASA related features and pairwise energy scores reveals best predictive results in hot spot prediction. Thus, these features are significant for predicting the stability of interactions between residue pairs.

Solvent accessible surface area (SASA) or accessible surface area (ASA) is the area on the surface of a biomolecule which is accessible to a solvent. The term, ASA, was defined by Lee and Richards in 1971 and it is also known as Lee-Richard molecular surface [60]. In our algorithm, ASA related features are calculated using Naccess [61]. Naccess finds the atomic accessible surface by using 'rolling a ball' approach. In this approach, a probe with a known radius is rolled around the surface of the macromolecule and the accessible surface is defined as the path traced out by its center. Water molecule, that has 1.4 Å radius, is used as probe. Hence, the surface is called the solvent accessible surface. Naccess approximates the accessible surface of each atom by making successive thin slices through the 3D molecular volume. In our pipeline, we calculate four ASA related features using Naccess. These features are ASA in monomer state, relative ASA in monomer state, ASA in complex state, and relative ASA in complex state. For a given complex, if a residue has lower ASA related features, then this residue is not on the surface and hence, they may contact other residues to contribute to the interaction of two partners in the complex.

Knowledge-based potential is defined as an energy function that is derived from the analysis of known protein structures. There are various methods to determine potentials. In the scope of this thesis, we use knowledge-based solvent mediated inter-residue potentials calculated by Keskin *et al*. [12] They followed a statistical approach to calculate potentials and form a 20x20 matrix which shows 210 distinct contact potentials between all possible pairs of 20 amino acids. The contact potentials are in RT unit where R is the universal gas constant, and T is the absolute temperature.

Pair potential calculator submodule calculates a potential value for each interface residue. For each residue, submodule extracts the neighbors of that residue. A residue is defined as a neighbor of another residue if the distance between their side chain center of mass is less than 7.0 Å, and there are at least four other residues in between, in other words, they should not be close neighbors in the sequence. The threshold value (7.0 Å) for neighbor detection can be changed by users through our web-based software. We adopt the total contact potential definition from Tuncbag *et al*. [11] According to this

definition, the absolute value of the sum of the contact potentials between a residue and its neighbors is equal to the total contact potential of that residue.

Appendix section includes full code of feature calculation module.Inputs

3.2.4 Density Calculation using Affinity Propagation

In our pipeline, feature extraction module is followed by the clustering module. Clustering module has two submodules. First submodule finds exemplars of the clusters and calculates density parameters using affinity propagation method. The density parameters are fed into the second submodule which runs density-based clustering of applications with noise algorithm to obtain final results.

Affinity propagation [9] is a clustering algorithm which clusters data points based on a message passing approach. It differs from k-means and k-medoids clustering algorithms since it does not require the number of clusters to be determined or estimated before running the algorithm. On the other hand, it is similar to k-medoids as it founds a subset of data points which is composed of the representatives of the clusters.

Figure 4 is an illustration of how affinity propagation works on two-dimensional data points. AP constructs a network on which each node represents a data point. It simultaneously considers all nodes as potential representatives of clusters. Then, it recursively sends messages along the edges of the network until good clusters are formed. Similar to k-medoids, it measures similarity between data points using negative Euclidean distance or squared error. In (A), colored points represent cluster exemplars according to the current evidence. A directed arrow from a node to an exemplar demonstrates the message that the point belongs to the cluster of that exemplar and the darkness of the arrow represents the strength of the message. The strength of the message from a data point to an exemplar is defined as responsibility. Responsibility is calculated from (1)

$$r(i,k) \leftarrow s(i,k) - \max_{k' \; such \; that \; k' \neq k} \{a(i,k') + s(i,k')\} \qquad (1)$$

where i is the data point, k is the exemplar, $s(i,k)$ is similarity, and $a(i,k)$ is availability. Availability is the message sent from exemplar to a data point to indicate

the degree of the availability of that candidate exemplar for the data point. Availability is calculated from (2)

$$a(i,k) \leftarrow \min\{0, r(k,k) + \sum_{i' such\ that\ i' \notin \{i,k\}} \max(0, r(i',k))\} \qquad (2)$$

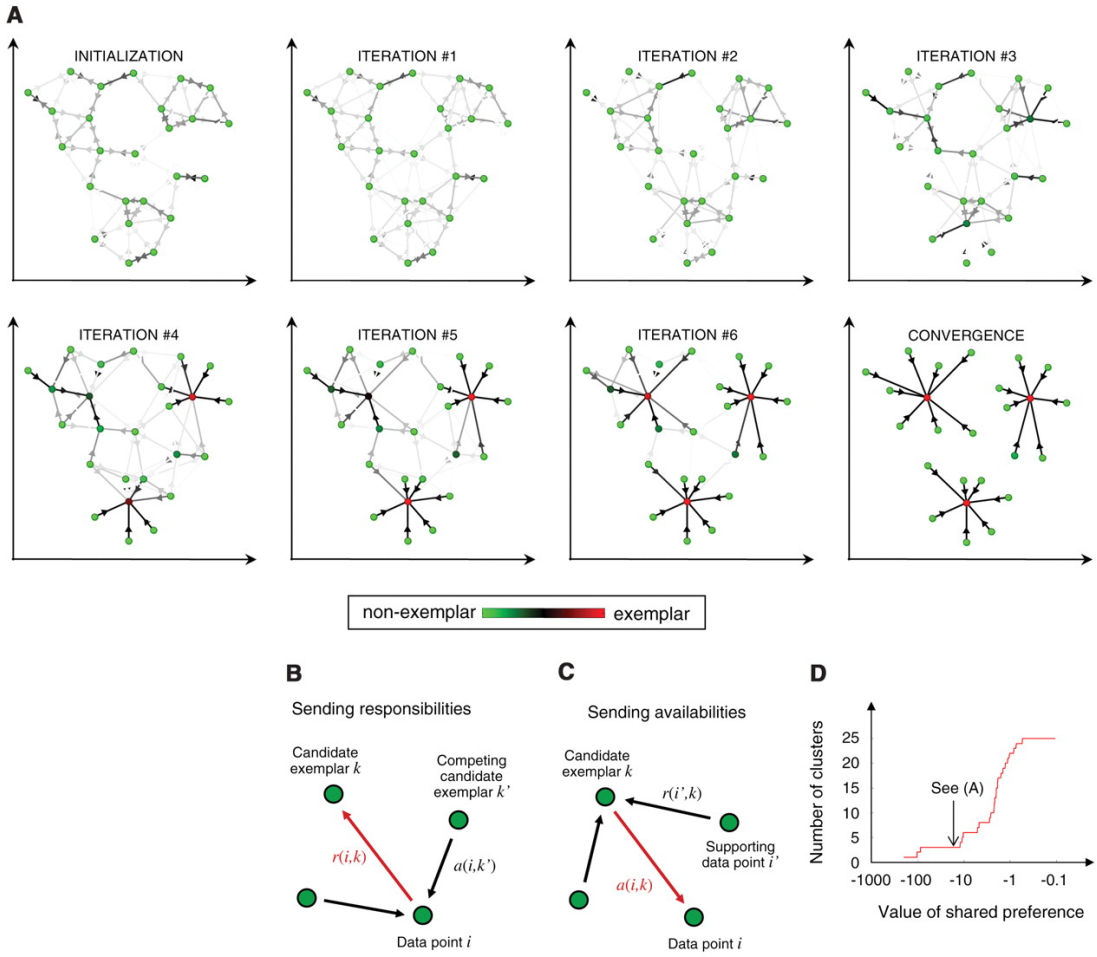

Figure 4: Working principle of affinity propagation [9]

Cluster module of scikit-learn library has an implementation of affinity propagation. In our clustering module, we call affinity propagation algorithm from scikit-learn. It has various configuration parameters including *'affinity'* to use precomputed similarities instead of negative squared Euclidean distance, and *'random_state'* to obtain reproducible results. Here, AP is called with default parameters.

AP returns cluster centers and cluster labels for each data point. Using cluster representatives and labels, our algorithm finds a pair of optimal density parameters. In order to find optimal parameters, the algorithm forms a density list using the same

approach with APSCAN [14]. The density of a cluster is represented with two parameters which are the number of points in the cluster and the radius of it. Given a dataset of N points, AP divides the dataset into K fragments. The density of each fragment is defined below:

$$\rho_i = \frac{number_i}{radius_i} \quad where\ i = 1,2,3, \dots, K \tag{3}$$

First of all, our algorithm calculates density for each fragment. Secondly, it finds the average value of all densities. Then, it constructs a list of fragments whose densities are higher than the average density while eliminating the ones whose density is less than or equal to the average. The constructed list contains (number, radius) pairs. From that list, minimum of the numbers and average of the radii are selected as clustering parameters for the next submodule. Source code of the submodule can be found in Appendix section.

3.2.5 Hot Region Prediction using Density-based Clustering of Applications with Noise

Our algorithm takes advantage of density-based clustering of applications with noise (DBSCAN) approach to predict hot regions on protein-protein interfaces (PPIs). DBSCAN is a non-parametric clustering algorithm [10]. It clusters closely packed points, which has many nearby-neighbors, and marks outliers, which lie on low-density regions, as noise. In summary, the main concept of the DBSCAN algorithm is to locate high density regions which are separated by low density regions. Density is defined as follows:

- Density at a point P: The number of points within a circle of radius ($\in$) from point P.
- Dense region: For each point in the cluster, the circle with radius $\in$ contains at least minimum number of points.

The DBSCAN algorithm has two mandatory inputs which are *'minPoints'* and *'eps'*. The closeness of points to each other to be considered a part of a cluster is determined by epsilon (eps). If the distance between two points is less than or equal to eps, then these points are considered as neighbors. MinPoints is defined as the minimum number of points required to mark a region as a cluster. For instance, each cluster

should contain at least 5 data points if minPoints parameter is set to 5. In DBSCAN, there are three types of data points: core point, border point, and noise point.

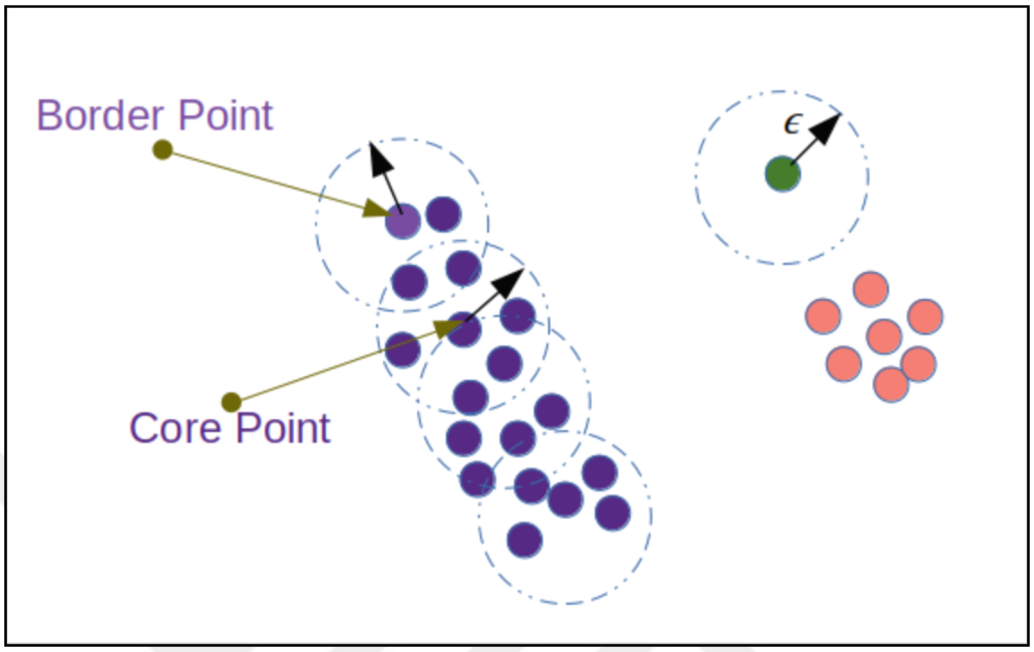

Figure 5: Data points types and epsilon neighborhood in DBSCAN [62]

- Core point is a point which has at least minPoints points within its epsilon neighborhood.
- Border point is a point which has less than minPoints points within its epsilon neighborhood but is a neighbor of a core point.
- Noise point (outlier) is a point which is neither a border point nor a core point.

Between any two different points, there can be one of the three different relationships. These relationships are density reachable, directly density reachable, and density connected.

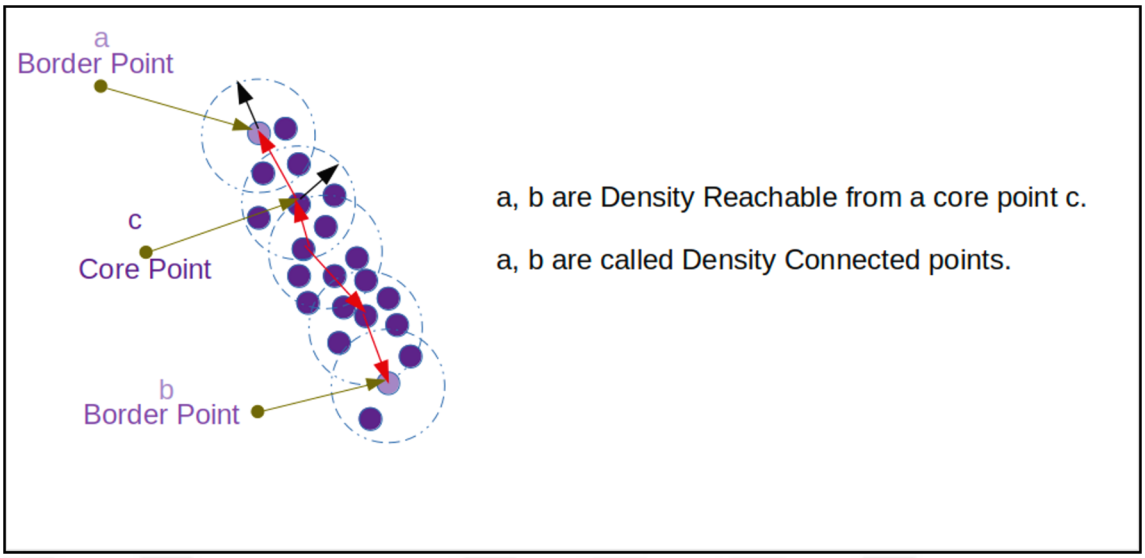

Figure 6: Relationships between data points in DBSCAN. [62]

Figure 6 is an illustration of three kinds of relationships between data points. As can be seen in the figures, border points have significantly fewer neighbor points than that of core points. Setting minPoints parameter to a low value to include border points in clusters can cause the algorithm to fail detecting and excluding outliers from the clusters. At this point, the concepts of density reachable and density connected points appear. A data point, a is directly density reachable from another point b if b is a core point and a is in b's epsilon neighborhood. The points a is density reachable from another point b if there is a chain of points $b_1,b_2,b_3,\ldots b_n$ where b_n=a such that b_{i+1} is directly density reachable from b_i for $\exists\ i \in [1, n-1]$. In some cases, two points a and b are both border points and they belong to the same cluster. However, they do not share a common specific core point. If there exists a core point from which points a and b are both density reachable, then these points are called density connected.

The DBSCAN algorithm starts running from a random point. The epsilon neighborhood of the point is determined based on the parameter eps. If there exists at least minPoints neighbors in epsilon neighborhood, then the point is marked as core point and the algorithm starts constructing a cluster. Otherwise, the point is marked as noise. When a cluster construction starts, the algorithm adds the points which reside in the epsilon neighborhood of the core points to the cluster. If these points are also core points, their neighbors are also added to the cluster. Then, the algorithm picks another point at random and repeats the procedure until no data points remain unvisited. During execution, a point which is marked as noise may become a part of a cluster later.

Algorithm 3: Pseudo-code of DBSCAN

```
DBSCAN(D, eps, minPoints):
      C = 0
      for each unvisited point P in dataset D:
            mark P as visited
            NeighborPoints = regionQuery(P, eps)
            if sizeof(NeighborPoints) < minPoints:
                  mark P as noise
            else:
                  C = next cluster
                  expandCluster(P, NeighborPoints, C, eps,
                  minPoints)

expandCluster(P, NeighborPoints, C, eps, minPoints):
      add P to cluster C
      for each point P' in NeighborPoints:
            if P' is unvisited:
                  mark P' as visited
                  NeighborPoints' = regionQuery(P', eps)
                  if sizeof(NeighborPoints') > minPoints:
                        NeighborPoints += NeighborPoints'
                  if P' is not yet a member of any cluster:
                        add P' to the cluster C

regionQuery(P, eps):
      return all points that are in P's eps neighborhood including P
```

Scikit-learn library has an implementation of DBSCAN algorithm in its cluster module, from which our algorithm calls. The source code of our clustering with DBSCAN submodule is attached to Appendix section.

3.2.6 Retrieval of Hot Region Information using HotRegion v2.0 Webserver

We have developed a web-based software in addition to the hot region prediction algorithm based on machine learning approaches. The web-based software provides a user-friendly interface to users and allow them to obtain hot region information easily.

To eliminate running the hot region prediction algorithm for the same complex multiple times, we store the results in a database. Therefore, users can query hot region information from the database very fast. The database contains all the complexes found in Protein Data Bank as of April 2020. The database has two tables and the data can be queried via simple or advanced search. First table stores protein-protein interface information. The data stored in this table offers researchers to investigate hot regions of protein complexes and obtain structural properties of interface residues including pair potentials, ASA and relative ASA values of monomer and complex forms. PDB title, Gene Ontology (GO) annotation molecular function, and biological process information is also stored in the table. Second table deposits the results of jobs submitted by users with customized parameters and/or user uploaded PDB files. Using their email addresses and job identifiers, users can retrieve the results any time. The database is publicly available for free without any authentication requirements.

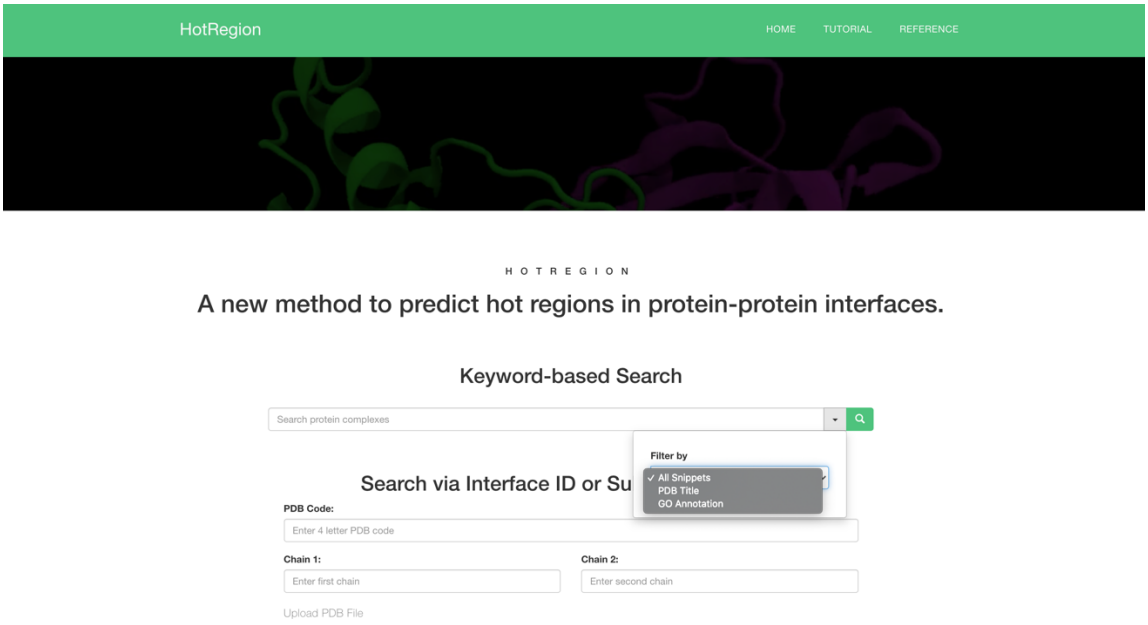

Figure 7: Keyword-based search

Figure 7 is an illustration of keyword-based search from the HotRegion v2.0 web page. Users retrieve a list of complexes by entering keywords. From that list, they obtain hot region information by clicking the corresponding entry in the complex list. The results can be filtered by PDB title and GO annotation. If search keyword is not found in the database, the system notifies user with an alert box.

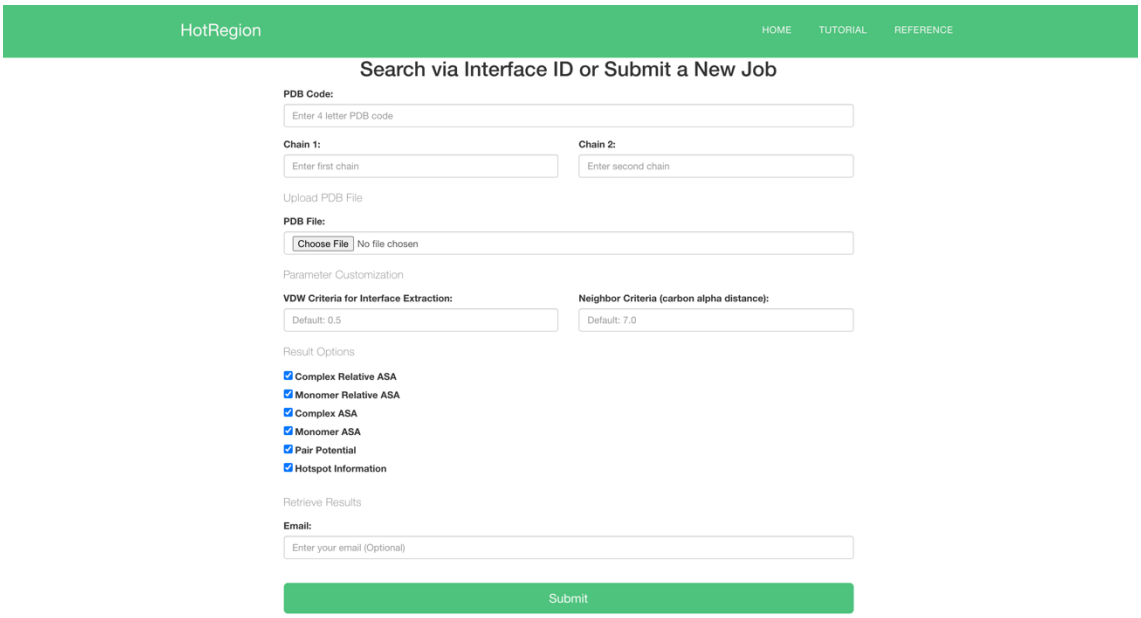

Figure 8: Search via interface ID or submit a new job with advanced options

In addition to the keyword-based search, users can get hot region information by typing PDB ID and two chain identifiers. If the results for the complex exists in the database, the tool directly shows the results. Otherwise, our automated pipeline

downloads the file from PDB, predicts hot regions, and shows the results. In this case, the database updates itself and the results are saved to the database. The system allows users to upload their complex structures in PDB format. Users can change threshold values used in the algorithm such as VDW criteria for interface extraction or carbon alpha distance for neighbor extraction. When the users upload their custom structure and/or change parameters, the system predicts hot regions based on the inputs, shows the results, and store these results in the database. The system displays a unique job identifier to the user. The user can query and download the results later using the job id (Figure 8).

3.2.7 Outputs

The HotRegion v2.0 software shows outputs as a table. In this table, interface residues are indicated with their names, sequence numbers, and chain identifiers. Additionally, total pair potentials, ASA and relative ASA in monomer and complex states are demonstrated in the table. Last two columns of the table contain hot spot and hot region information. Hot spot residues are marked as ‘HS’ while others are marked as ‘NS’. Hot region information is shown as cluster identifiers. The residues which do not belong to any cluster are marked as -1, others are marked as the ids of the clusters which they belong to. For instance, a reside is marked as 1 if this residue belongs to the cluster #1. The prediction results are stored in a text file. This result file is downloadable in addition to the PDB file which includes interface residue coordinates. Users can download these files to visualize results later using any visualization tool.

R E S U L T S

Interface ID	Residue ID	Residue Name	Chain ID	Complex ASA	Relative Complex ASA	Monomer ASA	Relative Monomer ASA	Pair Potential	Hot Spot Status	Hot Region ID
1A0OAB	211	V	B	4.24	2.92	40.18	26.53	18.40	HS	0
1A0OAB	214	F	B	21.03	10.54	170.71	85.58	26.81	HS	0
1A0OAB	182	L	B	5.83	3.01	52.16	29.20	40.79	HS	0
1A0OAB	217	E	B	80.94	46.99	107.32	62.30	5.58	NH	1
1A0OAB	122	K	A	40.47	20.15	100.95	50.27	7.06	NH	1
1A0OAB	213	C	B	36.16	26.93	54.57	40.64	17.74	NH	1
1A0OAB	207	D	B	33.54	23.89	77.27	55.04	6.61	NH	2
1A0OAB	100	Q	A	97.07	54.38	148.39	83.13	9.95	NH	2

Figure 9: HotRegion v2.0 results for 1A0OAB

Figure 9 is an example result table for 1A0OAB. As seen in the figure, first column corresponds to interface ID, following two columns show the residue number in the

protein sequence and the one letter residue name. It is followed by the chain identifier which the residue is located. Next four columns indicate ASA and relative ASA values for complex and the monomer form. Pair potential follows the ASA columns, and the last two columns indicate if the residue is a hot spot and if it resides in a hot region. Hot spot residues are marked as HS whereas non-hot spots are shown as NH. Hot regions are represented with a region id. The residues which belong to the same cluster are marked with the same cluster id and colored accordingly. Hot region ID for the residues that do not belong to any of the clusters is -1 and the rows corresponding to these residues are colored white.

3.2.8 Visualization

HotRegion v2.0 web tool provides three-dimensional visualization of protein-protein interfaces using LiteMol [63].

LiteMol is used to interactively visualize 3D structural molecular data in web-based platforms. By itself, LiteMol is a web-based tool used for visualization. LiteMol.js, on the other hand, is a library written in TypeScript and compiled to JavaScript. It has various advantages. It is interactive and user-friendly. Furthermore, it works very fast even for huge protein structures.

Algorithm 4: Loading a PDB structure to visualize with LiteMol

```
var plugin = LiteMol.Plugin.create({ target: '#litemol' });
plugin.loadMolecule({
  id: '1a0o',
  url: 'https://www.ebi.ac.uk/pdbe/entry-files/download/pdb1a0o.ent',
  format: 'pdb'
});
```

In the visualization, each partner of the complex is represented in cartoon form with a different color by default. The residues in hot regions are shown in ball representation. Each cluster is colored differently but none of the clusters is colored red. Hot spot residues are also represented as balls and colored red.

LiteMol allows users to rotate the structure, zoom in or out, highlight residues, open visualization full screen, and take a snapshot. Additionally, users are able to

change representation of the complex. Available representations are cartoon, C-α trace, balls and sticks, VDW balls, and surface representation. Coloring can also be changed.

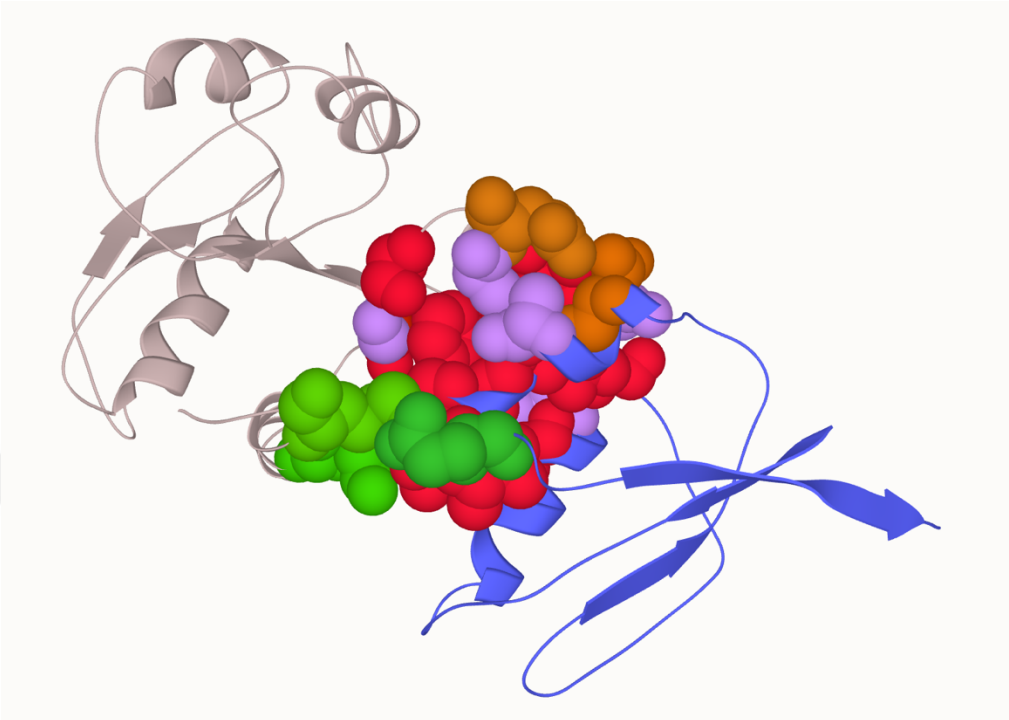

Figure 10: 3D visualization of CheA-CheY complex using LiteMol

3.3 Software Architecture

In this study, overall system has two major parts that are a standalone hot region prediction program and a web application. Standalone program is developed in Python3. As shown in Figure 11, various Python libraries are used for development. Python is an easy to code, human readable programming language. Having a lot of modules and libraries, it is very powerful to perform each task in a machine learning project. Moreover, it is open source and supported by huge number of resources and high-quality documentation.

Biopython package has a PDB parser module to safely parse protein structure files. NumPy is very efficient for scientific computing and scikit-learn encapsulates almost every functionality required for data mining and data analysis. In this study, we use clustering module of the scikit-learn library. In sci-kit-learn, both of the clustering algorithms (AP & DBSCAN) which we use are already implemented and easily usable.

As explained in Section 3.2.1, input to our algorithm is a PDB file. However, the program is also capable of automatically downloading a PDB file from Protein Data

Bank through a Rest API when a PDB ID is specified. REST stands for Representational State Transfer. It is a software architectural style which is used to gather data from a server without any knowledge of what is happening on the server-side. With a simple HTTP request, our program downloads the PDB file and perform processing tasks on it.

Using the same approach, we collect data from UniProt as well. Specifically, we collect keywords associated with the protein, GO annotation, and molecular function data. With the information coming from UniProt, users can search for proteins in our web interface and query data from our database.

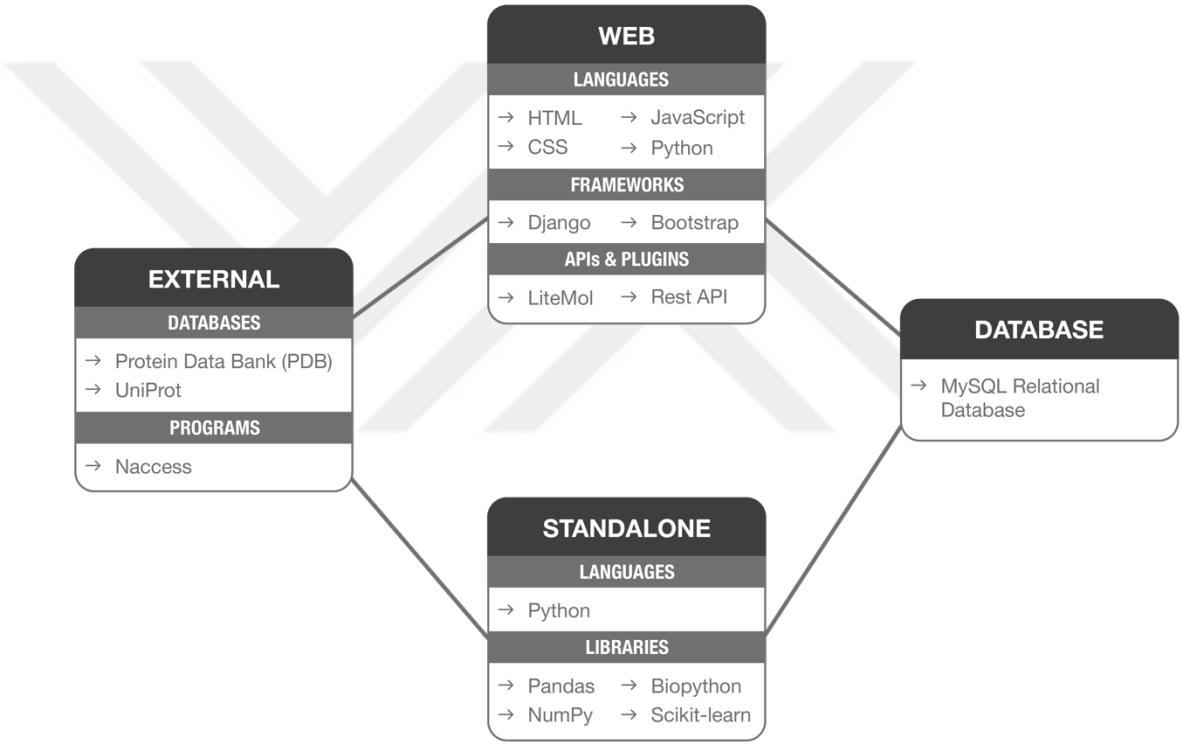

Figure 11:Software architecture

As explained in Section 3.2.3, we calculate the accessible surface area for interface residues using Naccess. Naccess is a standalone program which calculates the accessible area of a molecule from a file in PDB format. It is implemented in Fortran and free for academic and non-profit uses. It runs on most Linux/Unix systems including Cygwin under Windows.

The second major part of our software is the web interface. We have developed the interface using the Django framework. Django is a high-level Python framework used for web development. It is free and open source. It has the advantages of fast and clean

development, scalability, and security assurance. It is able to easily connect to MySQL databases and execute select, insert, and update queries.

Front-end of the webserver is implemented in HTML, CSS, and JavaScript. Bootstrap framework is also used. It is a front-end framework for faster and easier web development. It is composed of HTML and CSS based design templates. It supports the latest stable versions of all browsers and platforms.

In our web server, results are visualized using the LiteMol plugin. LiteMol is utilized to visualize 3D structural molecular data. It is implemented in TypeScript and uses WebGL for 3D visualization.

We use a relational database, MySQL to connect our standalone program to our web server. MySQL is one of the most popular database management systems and it provides concurrent multiuser access to its databases. Detailed information about our database can be found in Section 3.4.

3.4 Database Management

A relational database is one of the major components of HotRegion v2.0. The objectives of having a database are to quickly retrieve hot region information for known complexes and to eliminate unnecessary re-runs of prediction algorithm on the same complexes. As explained before, HotRegion v2.0 uses a MySQL database to deposit data. The database has two tables: one for storing results obtained by using default parameters, and the other one for saving results of user uploaded structures and/or the results obtained by using custom parameters. The first table has seven columns and as of now, it contains 641.061 interfaces. Each entry has a unique id. Second column stores PDB ID of the complex. The third and the fourth columns contain chain ids of two partners in the complex respectively. Fifth column contains PDB title of the complex and the sixth one contains a string of comma-separated GO annotation keywords. The final column stores the results of the algorithm as a json string. Instead of preserving an entry for each residue of a complex in the table, we compress the results in a json string and store it as an attribute of the complex. By this way, we aim to decrease memory usage and fasten the query execution time for a complex. The description of the table can be found in Table 5.

Table 5: Description of the main table

Field	Type	Null	Key	Default
ID	varchar(10)	NO	PK	NULL
PDB_ID	char(4)	NO		NULL
chain_1	char(1)	NO		NULL
chain_2	char(1)	NO		NULL
PDB_title	varchar(200)	NO		NULL
go_keywords	longtext	NO		NULL
results	json	NO		NULL

The second table has three columns which are a unique job identifier, email of the user and the results. The overview of the table is shown in Table 6

Table 6: Description of the custom jobs table

Field	Type	Null	Key	Default
job_id	varchar(10)	NO	PK	NULL
email	varchar(50)	NO		NULL
results	json	NO		NULL

Chapter 4:
RESULTS AND DISCUSSION

This chapter includes the results of performance analysis of our hot region prediction algorithm, and the tutorial of our web-based tool. The section starts with the evaluation of predicted clusters, continues with a case study demonstrating the significance of hot regions for drug binding, and this is followed by a comparative analysis of our method with the previous version of HotRegion. Finally, the chapter is concluded with the discussion of the results and future works of the study.

4.1 Evaluation of the Algorithm

To evaluate our method, we first validate the consistency within the predicted hot region clusters. For this purpose, we calculate the silhouette constant. Silhouette constant measures the similarity of a data point to the other points in the same cluster compared to the data points in other clusters. Silhouette constant has value in range [-1, 1]. If it is equal to 1, that means the data points are clustered correctly and the clusters are well-separated from each other. However, if the value is negative, then some points are put into wrong clusters. When there are overlapping clusters, the silhouette score becomes close to 0. Here, we calculate silhouette constants for the hot region results of nine different complexes. The results are shown in Table 7.

Table 7: Silhouette Constants

Complex	PDB ID	Silhouette Score
TEM-1 β-Lactamase / β-Lactamase inhibitor protein complex	1JTG	0.142
CheY binding domain of CheA in complex with CheY	1A0O	0.158
Human growth hormone bound to single receptor	1A22	0.186
Barnase-Barstar complex	1BRS	0.123
Complex of Ran with Importin-β	1IBR	0.238
Colicin E9 DNase domain with its cognate immunity protein Im9	1EMV	0.080
PLC epsilon Ras association domain with hRas	2C5L	-0.032
Im2 in complex with colicin E9 DNase	2WPT	0.034
H-Ras-GppNHp bound to the RBD of Raf Kinase	4G0N	0.194

We assess the prediction performance of our method by using the results of experimental findings as ground truths. We calculate evaluation metrices such as accuracy, precision, F-measure, and recall. For the performance assessment, we have

used nine complexes mentioned above (PDB IDs: 1JTG, 1A0O, 1A22, 1BRS, 1EMV, 1IBR, 2C5L, 2WPT, 4G0N). The experimental results for 1JTG, 1BRS, 1A0O, 1A22, and 1IBR are taken from Reichmann *et al*. [4], experimental hot regions of 1EMV and 2WPT are got from [64], and the hot regions of 4G0N and 2C5L are collected from [3].

Table 8 shows the results of TEM-1 β-Lactamase / β-Lactamase inhibitor protein complex. The residues which do not belong to any cluster are marked as -1. Experimental results are indicated in Figure 12. As seen in the figure, five residue clusters are identified on the interface by X-ray crystallography. Black lines in the figure represents interactions between residues.

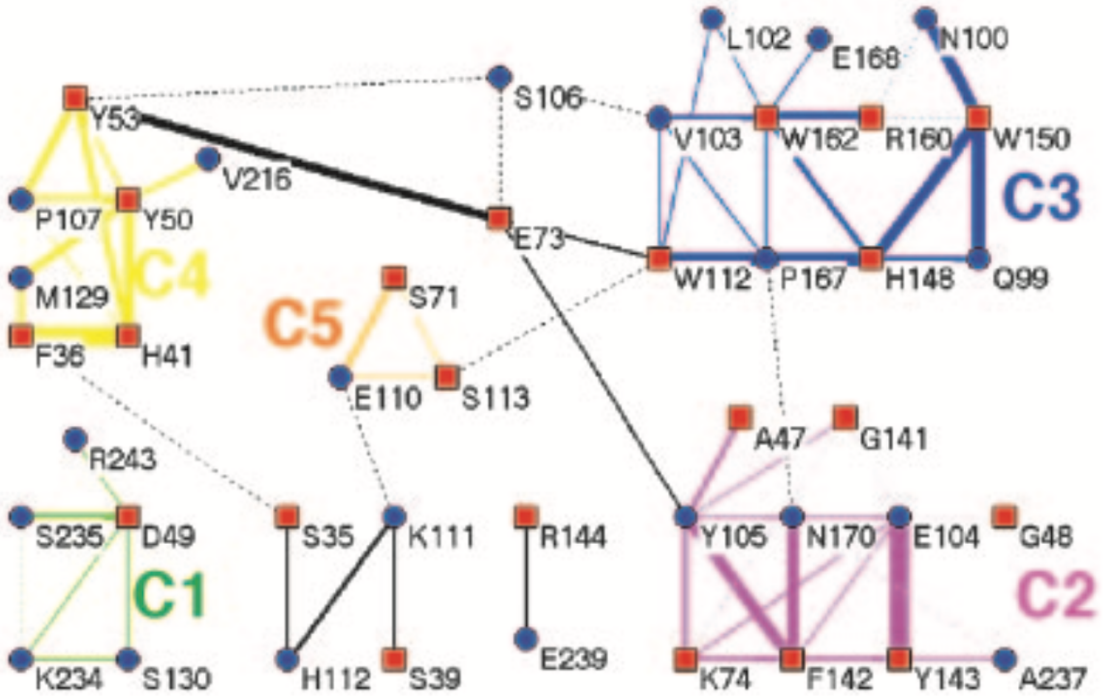

Figure 12: Experimental clusters of TEM-1 β-Lactamase / β-Lactamase inhibitor protein complex [4]

Table 8 demonstrates that our algorithm succeeds to identify clusters 1, 4 and 5. However, it fails to separate clusters 3 and 4 from the cluster 4. On the other hand, Figure 12 shows that Tyr-53 in the cluster 4 is interacting with Glu-73 which also interacts with Trp-112 from the cluster 3 and Tyr-105 from the cluster 2. Therefore, we could say that it is actually logical to merge clusters 2, 3, and 4 into one single cluster.

Table 8: HotRegion v2.0 results for TEM-1 β-Lactamase / β-Lactamase inhibitor protein complex

Residue ID	Residue Name	Chain ID	Complex ASA	Relative Complex ASA	Monomer ASA	Relative Monomer ASA	Total Contact Potential	Hot Spot Status	Observed Cluster	Predicted Cluster
53	Y	B	0.70	0.33	39.44	18.54	10.32	NH	4	4
107	P	A	0.53	0.39	71.16	52.27	18.39	HS	4	4
50	Y	B	10.08	4.74	96.09	45.16	20.57	HS	4	4
216	V	A	13.74	9.07	66.33	43.80	25.61	HS	4	1
129	M	A	1.91	0.98	59.43	30.61	42.36	HS	4	4
36	F	B	0.16	0.08	64.05	32.11	28.84	HS	4	4
41	H	B	0.00	0.00	2.74	1.50	23.53	HS	4	4
243	R	A	7.96	3.33	21.36	8.95	37.17	HS	1	1
49	D	B	11.55	8.23	149.63	106.58	26.48	HS	1	1
235	S	A	1.12	0.96	5.33	4.58	20.29	HS	1	1
234	K	A	3.50	1.74	3.50	1.74	11.41	NH	1	1
130	S	A	4.63	3.97	25.97	22.29	15.07	NH	1	1
71	S	B	3.39	2.91	23.39	20.08	6.93	NH	5	5
110	E	A	15.59	9.05	101.89	59.15	18.24	HS	5	5
113	S	B	17.81	15.29	32.95	28.28	11.21	NH	5	5
102	L	A	3.61	2.02	53.55	29.98	45.23	HS	3	4
168	E	A	56.30	32.69	98.23	57.03	10.24	NH	3	4
100	N	A	76.43	53.10	141.41	98.24	5.98	NH	3	-1
103	V	A	0.02	0.01	39.78	26.27	19.22	HS	3	4
162	W	B	21.00	8.42	96.41	38.66	15.36	NH	3	4
160	R	B	71.75	30.05	143.79	60.22	13.09	NH	3	4
150	W	B	28.63	11.48	89.95	36.97	17.67	NH	3	4
112	W	B	13.35	5.35	78.11	31.32	27.16	HS	3	4
167	P	A	16.67	12.25	45.89	33.71	11.35	NH	3	4
148	H	B	1.01	0.55	36.40	19.90	12.47	NH	3	4
99	Q	A	39.72	22.25	127.73	71.56	20.17	NH	3	-1
47	A	B	2.29	2.12	9.20	8.52	16.49	NH	2	4
141	G	B	18.48	23.07	23.98	29.94	11.83	NH	2	4
105	Y	A	7.34	3.45	161.97	76.13	25.37	HS	2	4
170	N	A	3.09	2.15	23.96	16.65	16.75	NH	2	4
104	E	A	11.82	6.86	110.41	64.10	16.40	NH	2	4
143	Y	B	32.13	15.10	61.01	28.68	23.22	HS	2	4
237	A	A	1.47	1.36	41.82	38.74	27.26	HS	2	4
35	S	B	35.46	30.44	115.85	99.44	15.04	NH	-1	4
111	K	A	51.25	25.52	136.10	67.78	9.49	NH	-1	6
39	S	B	2.98	2.56	22.84	19.61	12.04	NH	-1	6

112	H	A	36.14	19.76	54.86	30.00	14.23	NH	-1	6
144	R	B	129.86	54.39	175.66	73.57	11.95	NH	-1	-1
239	E	A	30.16	17.51	103.63	60.16	12.30	NH	-1	-1
106	S	A	0.00	0.00	13.47	11.56	16.24	NH	-1	4
73	E	B	5.74	3.33	45.17	26.22	18.97	HS	-1	4

Furthermore, as seen in the figure, there are connections among all clusters although they are represented as dashed lines, implying the interactions are weak. This means that all hot regions are somehow close to each other which explains why the silhouette constant has a value close to 0.

Table 9: HotRegion v2.0 results for CheY binding domain of CheA in complex with CheY

Residue ID	Residue Name	Chain ID	Complex ASA	Relative Complex ASA	Monomer ASA	Relative Monomer ASA	Total Contact Potential	Hot Spot Status	Observed Cluster	Predicted Cluster
211	V	B	4.42	2.92	40.18	26.53	18.40	HS	0	0
214	F	B	21.03	10.54	170.71	85.58	26.81	HS	0	0
217	E	B	80.94	46.99	107.32	62.30	5.85	NH	1	1
122	K	A	40.47	20.15	100.95	50.27	7.06	NH	1	1
207	D	B	33.54	23.89	77.27	55.04	6.61	NH	2	2
100	Q	A	97.07	54.38	148.39	83.13	9.95	NH	2	2
213	C	B	36.16	26.93	54.57	40.64	17.74	NH	1	-1
182	L	B	5.38	3.01	52.16	29.20	40.79	HS	0	0
210	A	B	25.17	23.32	63.39	58.72	11.59	NH	0	0
103	A	A	10.41	9.64	22.22	20.58	28.01	HS	0	0
216	I	B	1.67	0.95	2.46	1.40	37.30	HS	1	-1
106	Y	A	13.99	6.58	69.11	32.48	24.72	HS	0	0
96	I	A	22.38	12.78	86.53	49.41	24.29	HS	2	2
178	E	B	19.56	11.36	51.87	30.11	22.34	HS	0	0
99	A	A	12.87	11.92	66.90	61.97	15.31	NH	0	0
181	H	B	60.71	33.20	129.66	70.90	14.77	NH	0	0
92	K	A	58.11	28.94	131.18	65.33	13.11	NH	0	0
105	G	A	5.21	6.50	24.04	30.01	14.61	NH	0	0
215	V	B	19.82	13.09	59.20	39.09	19.64	HS	1	1
95	I	A	1.95	1.11	49.53	28.28	25.05	HS	0	0
126	K	A	65.06	32.40	151.61	75.50	8.32	NH	1	1

Table 10: HotRegion v2.0 results for Human growth hormone bound to single receptor

Residue ID	Residue Name	Chain ID	Complex ASA	Relative Complex ASA	Monomer ASA	Relative Monomer ASA	Total Contact Potential	Hot Spot Status	Observed Cluster	Predicted Cluster
304	W	B	6.73	2.70	161.60	64.81	31.44	HS	0	0
276	W	B	34.77	13.94	91.65	36.75	39.47	HS	2	2
45	L	A	1.71	0.96	70.38	39.40	43.91	HS	2	2
418	N	B	22.22	15.44	151.87	105.51	17.92	NH	1	0
371	V	B	35.51	23.45	72.35	47.77	20.20	NH	1	0
175	T	A	6.32	4.54	54.57	39.18	15.41	NH	0	0
42	Y	A	35.49	16.68	114.00	53.58	13.04	NH	2	2
48	P	A	6.59	4.84	64.10	47.09	11.84	NH	2	-1
275	E	B	86.89	50.44	108.87	63.20	5.15	NH	2	-1
322	C	B	2.70	2.01	73.30	54.59	27.14	HS	2	2
41	K	A	9.96	4.96	36.62	18.24	20.84	HS	1	1
168	K	A	6.79	3.38	57.03	28.40	15.76	NH	0	1
306	P	B	0.33	0.24	36.59	26.88	16.68	NH	2	2
25	F	A	32.38	16.23	80.27	40.24	24.88	HS	1	-1
417	R	B	59.75	25.03	107.19	44.89	20.05	NH	1	0
56	E	A	44.35	25.75	65.33	37.93	17.08	NH	2	2
167	R	A	26.44	11.07	61.18	25.62	25.74	HS	1	1
171	D	A	9.49	6.76	67.53	48.10	17.88	NH	0	0
327	E	B	39.04	22.66	102.44	59.47	21.32	NH	1	1
271	R	B	11.46	4.80	87.05	36.46	23.35	HS	2	2
367	K	B	90.51	45.07	156.80	78.08	4.67	NH	0	0
320	E	B	53.90	31.29	93.63	54.36	12.92	NH	2	2
364	D	B	5.14	3.66	24.16	17.21	11.26	NH	0	0
179	I	A	0.22	0.13	22.52	12.86	32.12	HS	0	0
68	Q	A	43.23	24.22	81.06	45.41	22.38	NH	0	0
243	R	B	3.87	1.62	34.73	14.55	30.33	HS	0	0
419	S	B	9.54	8.19	34.91	29.97	17.54	NH	1	0
46	Q	A	72.91	40.85	149.68	83.85	19.62	NH	2	2
369	W	B	0.74	0.30	93.67	37.56	31.79	HS	0	0
64	R	A	58.58	24.54	180.59	75.64	14.21	NH	0	0
21	H	A	2.73	1.49	33.84	18.50	16.07	NH	1	0
172	K	A	11.62	5.79	41.85	20.84	13.76	NH	0	-1
67	T	A	0.00	0.00	7.82	5.61	21.99	HS	0	0
18	H	A	28.00	15.31	99.67	54.50	13.64	NH	1	0
178	R	A	37.17	15.57	98.46	41.24	22.14	HS	1	0

Table 11: HotRegion v2.0 results for Barnase-Barstar complex

Residue ID	Residue Name	Chain ID	Complex ASA	Relative Complex ASA	Monomer ASA	Relative Monomer ASA	Total Contact Potential	Hot Spot Status	Observed Cluster	Predicted Cluster
29	Y	D	65.79	30.92	162.57	76.41	16.65	NH	0	0
76	E	D	58.28	33.83	82.37	47.82	13.91	NH	1	1
82	F	A	111.40	55.85	140.05	70.21	9.48	NH	0	-1
42	T	D	30.90	22.19	70.81	50.84	9.93	NH	1	1
60	E	A	67.01	38.90	136.40	79.19	9.13	NH	0	0
38	W	D	22.54	9.04	77.92	31.25	31.04	HS	1	1
35	D	D	4.16	2.96	119.91	85.41	23.75	HS	0	0
85	S	A	17.93	15.39	41.51	35.63	8.77	NH	0	-1
35	W	A	15.06	6.04	26.00	10.43	33.34	HS	1	1
44	W	D	50.98	20.44	131.13	52.59	19.46	NH	0	-1
103	Y	A	0.37	0.17	52.43	24.64	36.63	HS	0	-1
87	R	A	0.07	0.03	3.91	1.64	27.75	HS	0	0
56	F	A	9.35	4.69	27.79	13.93	32.70	HS	0	0
34	L	D	29.59	16.56	77.71	43.50	20.53	HS	1	1
43	G	D	1.79	2.23	58.02	72.43	11.88	NH	0	0
31	G	D	1.07	1.34	38.34	47.87	7.73	NH	0	0
33	N	D	10.42	7.24	70.08	48.69	13.25	NH	0	0
59	R	A	51.12	21.41	210.25	88.06	16.21	NH	1	1
84	N	A	38.48	26.73	50.50	35.08	4.49	NH	0	-1
102	H	A	1.10	0.60	108.84	59.51	30.75	HS	0	0
58	N	A	28.23	19.61	30.08	20.90	12.39	NH	0	-1
30	Y	D	8.51	4.00	12.72	5.98	32.49	HS	0	0
27	K	A	19.32	9.62	76.01	37.85	15.96	NH	1	1
36	A	D	0.00	0.00	19.23	17.81	16.96	NH	0	0
83	R	A	1.01	0.55	36.40	19.90	12.47	NH	0	0
39	D	D	39.72	22.25	127.73	71.56	20.17	NH	0	0

Table 12: HotRegion v2.0 results for Complex of Ran with Importin-β

Residue ID	Residue Name	Chain ID	Complex ASA	Relative Complex ASA	Monomer ASA	Relative Monomer ASA	Total Contact Potential	Hot Spot Status	Observed Cluster	Predicted Cluster
144	L	A	13.74	7.69	26.77	14.99	32.00	HS	4	-1
113	E	A	88.02	51.10	171.64	99.65	9.80	NH	0	0
288	D	B	42.57	30.32	64.18	45.72	18.52	NH	4	5
10	T	B	3.65	2.62	56.34	40.45	25.22	HS	2	1
56	A	B	3.20	2.96	14.36	13.30	17.95	NH	2	1
107	D	A	18.42	13.12	33.95	24.18	13.54	NH	3	1
166	R	A	32.58	13.65	72.93	30.55	16.90	NH	3	3
45	V	A	15.99	10.56	36.62	24.18	18.14	HS	2	2
62	K	B	25.44	12.67	41.27	20.55	9.14	NH	1	1
13	P	B	89.03	65.40	134.72	98.96	11.15	NH	2	2
107	D	A	18.42	13.12	33.95	24.18	13.42	NH	3	1
106	P	B	21.22	15.59	99.94	73.42	5.83	NH	0	-1
110	R	A	17.36	7.27	192.91	80.80	26.41	HS	1	1
285	N	B	4.66	3.24	14.98	10.41	14.43	NH	4	5
22	Q	B	37.73	21.14	102.23	57.27	19.60	NH	2	1
147	Y	A	10.52	4.94	33.33	15.67	26.70	HS	4	3
68	K	B	97.89	48.75	210.99	105.07	10.96	NH	3	1
12	S	B	23.00	19.74	25.25	21.67	6.69	NH	2	2
59	L	B	3.24	1.81	65.64	36.75	40.10	HS	1	1
239	Q	B	23.38	13.10	66.32	37.15	22.92	HS	4	6
340	D	B	41.12	29.29	108.23	77.09	11.75	NH	4	3
105	R	B	57.29	23.99	92.17	38.60	18.06	NH	0	0
47	V	A	27.74	18.32	69.46	45.87	13.16	NH	2	2
278	Q	B	0.73	0.41	27.83	15.59	16.29	NH	4	6
75	L	A	8.89	4.98	125.86	70.46	40.01	HS	2	1
111	Q	B	10.95	6.13	65.37	36.62	24.24	HS	1	1
140	R	A	15.90	6.66	155.54	65.14	19.46	HS	4	5
342	W	B	34.16	13.70	151.07	60.58	20.26	HS	4	6
52	V	B	13.65	9.01	52.48	34.65	24.39	HS	2	1
139	H	A	15.49	8.47	42.37	23.17	13.13	NH	4	6
281	E	B	4.80	2.79	60.75	35.27	18.06	HS	4	5

18	L	B	4.50	2.52	67.59	37.84	39.16	HS	2	2
143	N	A	37.97	26.38	141.98	98.64	12.70	NH	4	6
81	I	A	2.30	1.31	106.88	61.03	35.97	HS	2	1
145	Q	A	3.12	1.75	49.73	27.86	15.46	NH	4	3
55	V	B	10.10	6.67	40.97	27.05	27.98	HS	2	1
274	E	B	77.57	45.03	88.14	51.17	5.54	NH	4	-1
11	V	B	42.86	28.30	107.88	71.24	21.35	NH	2	1
338	D	B	46.42	33.07	86.14	61.36	10.43	NH	3	3
156	Y	B	13.55	6.37	54.57	25.65	26.85	HS	1	1
111	V	A	19.72	13.02	64.73	42.74	33.51	HS	1	1
64	W	A	6.78	2.72	66.37	26.62	32.70	HS	2	2
77	D	A	6.10	4.35	64.20	45.73	18.03	HS	1	1
159	Q	B	69.11	38.72	108.70	60.90	11.26	NH	1	1
141	K	A	46.65	23.23	128.14	63.81	13.22	NH	4	5
63	N	B	38.94	27.05	81.05	56.31	10.74	NH	1	1

Table 13: HotRegion v2.0 results for PLC epsilon Ras association domain with HRas

Residue ID	Residue Name	Chain ID	Complex ASA	Relative Complex ASA	Monomer ASA	Relative Monomer ASA	Total Contact Potential	Hot Spot Status	Observed Cluster	Predicted Cluster
38	D	A	3.97	2.83	62.34	44.40	12.84	NH	0	0
41	R	A	63.02	26.39	126.18	52.85	21.50	NH	0	0
37	E	A	29.63	17.20	67.23	39.03	18.38	HS	0	0
64	Y	A	24.63	11.58	56.71	26.65	17.32	NH	1	1
36	I	A	19.43	11.10	96.24	54.96	16.81	NH	1	1

Table 14: HotRegion v2.0 results for H-Ras-GppNHp bound to the RBD of Raf Kinase

Residue ID	Residue Name	Chain ID	Complex ASA	Relative Complex ASA	Monomer ASA	Relative Monomer ASA	Total Contact Potential	Hot Spot Status	Observed Cluster	Predicted Cluster
38	D	A	19.81	14.11	57.19	40.74	13.88	NH	0	0
41	R	A	48.42	20.28	143.06	59.92	15.09	NH	0	0
84	K	B	74.47	37.08	159.00	79.18	12.56	HS	0	0
59	R	B	93.08	38.98	105.21	44.07	11.80	NH	1	1
37	E	A	33.06	19.19	136.93	79.49	15.65	NH	1	1
36	I	A	85.69	48.93	149.78	85.53	16.79	NH	1	-1

Table 15: HotRegion v2.0 results for Im2 in complex with colicin E9 DNase

Residue ID	Residue Name	Chain ID	Complex ASA	Relative Complex ASA	Monomer ASA	Relative Monomer ASA	Total Contact Potential	Hot Spot Status	Observed Cluster	Predicted Cluster
37	V	A	0.00	0.00	22.16	14.63	33.12	HS	0	0
50	S	A	0.37	0.32	33.41	28.68	22.06	HS	0	0
53	I	A	0.02	0.01	12.15	6.94	32.14	HS	0	0
54	Y	A	5.31	2.50	110.71	52.04	26.44	HS	0	0

Table 16: HotRegion v2.0 results for Colicin E9 DNase domain with its cognate immunity protein Im9

Residue ID	Residue Name	Chain ID	Complex ASA	Relative Complex ASA	Monomer ASA	Relative Monomer ASA	Total Contact Potential	Hot Spot Status	Observed Cluster	Predicted Cluster
33	L	A	1.21	0.68	35.41	19.82	44.11	HS	0	0
34	V	A	27.68	18.28	86.41	57.06	21.22	HS	0	0
37	V	A	0.00	0.00	17.98	11.87	34.86	HS	0	0
50	S	A	0.59	0.51	33.89	29.09	20.15	HS	1	1
53	I	A	0.00	0.00	13.84	7.90	37.67	HS	1	1
54	Y	A	2.52	1.18	101.48	47.70	32.11	HS	1	1

Table 17: Number of TP, TN, FP, and FN for each complex and in total.

	TP	TN	FP	FN
1JTG	14	2	23	2
1A0O	18	0	0	2
1A22	23	0	8	4
1BRS	20	0	0	6
1IBR	19	0	24	3
2C5L	5	0	0	0
4G0N	5	0	0	1
2WPT	4	0	0	0
1EMV	6	0	0	0
Total	**114**	**2**	**55**	**18**

In total, we have investigated 189 interface residues from 9 protein-protein complexes. 81 of these residues are predicted as hot spots. 114 of them are put into correct hot regions, 2 of them are correctly omitted from hot regions, 55 of them are located into wrong hot regions, and 18 of them are incorrectly excluded from the clusters which they belong to. Therefore, the accuracy of our hot region prediction algorithm is:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{116}{189} = 0.62 \quad (4)$$

The recall is:

$$Recall = \frac{TP}{TP + FN} = \frac{114}{132} = 0.86 \quad (5)$$

The precision of the method is:

$$Precision = \frac{TP}{TP + FP} = \frac{114}{169} = 0.68 \quad (6)$$

Finally, we have checked the balance between precision and recall by calculating F-measure as follows:

$$F - measure = \frac{2 \times Recall \times Precision}{Recall + Precision} = \frac{1.17}{1.54} = 0.76 \quad (7)$$

We have discussed our results in Section 4.5

4.2 Case Study

Programmed cell death protein 1 (PD-1) is a type 1 transmembrane receptor. It is a member of CD28 family [65] which is expressed in B cells, activated T cells and myeloid cells [66]. Expression of PD-1 on the surface of activated T cells induces various tissues to express programmed death ligand-1 (PD-L1). Binding of PD-1 to its ligand causes the activation of inhibitory kinases which have role on proliferation and adhesion of T cells. Studies have shown that PD-1 ligands are upregulated in many tumor types and they evade immune responses. Therefore, blocking the PD-1/PD-L1 interactions may normalize the antitumor response. Although several inhibitors have been reported to target PD-1/PD-L1, developing small molecular inhibitors remains unsolved. Understanding the PPI of PD-1/PD-L1 may help developing such small inhibitors [15].

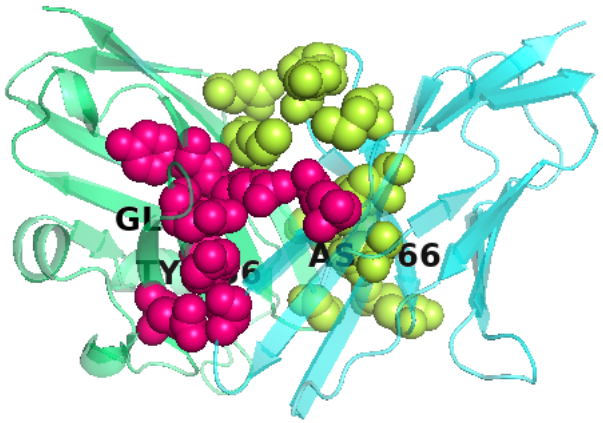

Figure 13: Hot regions of the complex of human programmed death-1 (PD-1) and its ligand PD-L1

A recent study which investigates the complex of human programmed death-1 (PD-1) and its ligand PD-L1 in quantum mechanical level and was conducted by Lim *et al.* has shown that residues Tyr-56, Glu-58, (from chain A) and Asn-66 (from chain B) are significant for PD-L1-antibodies and small-inhibitors [15]. As shown in the Figure 13, these residues are found on hot regions, predicted by our HotRegion v2.0 algorithm, which is also overlapping with the findings of a X-ray structure based study [15].

4.3 Comparison with Existing HotRegion Method

Our previous method predicts hot regions with a graph-based approach. This method generates a network of hot spot residues and finds the hot spots that are close to each other and locate them in the same cluster. Therefore, the hot regions only contain hot spot residues. On the other hand, experimental studies have proven that hot regions are not only composed of hot spot residues but also non-hot spot residues. To observe the improvements in predictions, we compare results of approximately 25.000 PDB complexes obtained by using two methods separately. The results, shown in Table 18, have proven that our new algorithm is better than our previous algorithm in two ways:

- Our new method finds non hotspot residues on the hot regions whereas the previous one cannot.
- Our new algorithm is able to expand clusters and place residues, that are not directly in contact but share a contact residue, on the same hot region.

Table 18: Comparison of HotRegion v1.0 and v2.0

	HotRegion v1.0	**HotRegion v2.0**
Average number of residues in a region	5	9
Average number of hot spots in a cluster	5	5
Average number of non-hot spots in a cluster	0	4
Average monomer ASA	64.21	57.56
Average relative monomer ASA	37.07	44.87
Average complex ASA	7.43	26.61
Average relative complex ASA	4.07	9.37
Average pair potential	26.10	19.75
Average distance between residues in the same region	7.89	7.37

Furthermore, we have evaluated our previous method on the same experimental dataset explained in Section 4.1. The results are demonstrated in the Table 19: Evaluation of previous HotRegion method

Table 19: Evaluation of previous HotRegion method

	TP	TN	FP	FN
1JTG	6	7	11	17
1A0O	4	0	5	11
1A22	5	0	8	22
1BRS	5	0	4	17
1IBR	9	0	11	26
2C5L	1	0	0	4
4G0N	0	0	0	6
2WPT	4	0	0	0
1EMV	6	0	0	0
Total	**40**	**7**	**39**	**103**

As a result, the accuracy of the previous HotRegion method is 0.25, recall is 0.28, precision is 0.51, and F-measure is 0.36. Therefore, we conclude that our new approach outperforms our previous approach.

4.4 HotRegion v2.0 Web-based Tool

We associate our standalone hot region prediction software with a web interface to provide researchers a user-friendly interface to easily investigate and visualize hot regions of protein-protein complexes. Researchers can use our web page to analyze

structural properties of protein interfaces, predict interface hot spots and hot regions, and visualize the 3D structure of the complexes. Following figures demonstrates the main functionalities of our web-based tool and overview of the web pages.

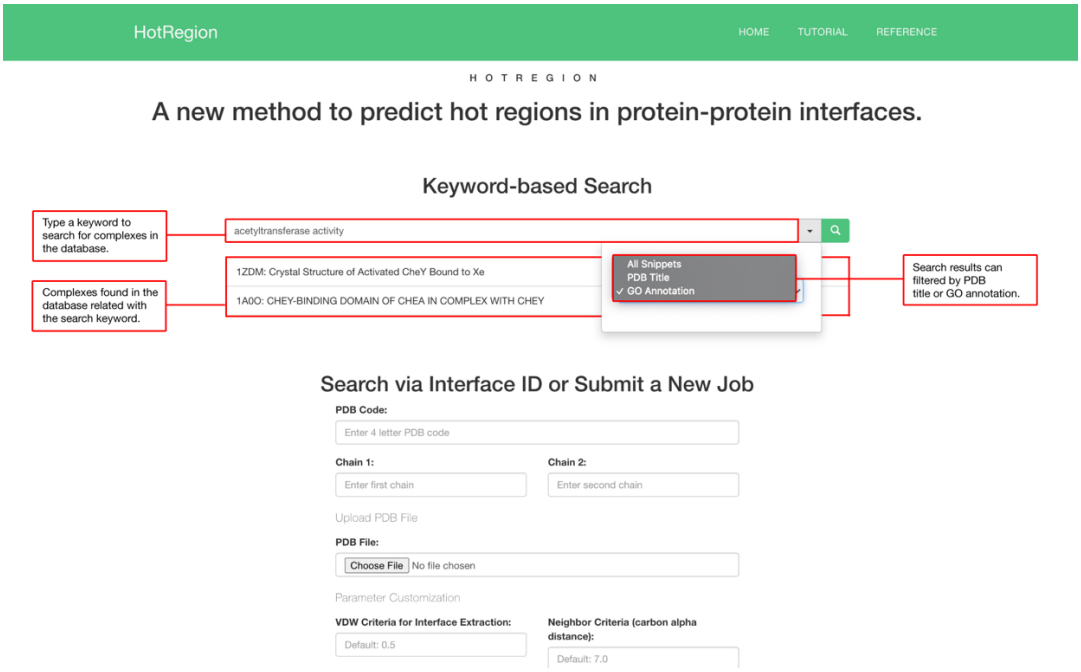

Figure 14: HotRegion v2.0 keyword-based search

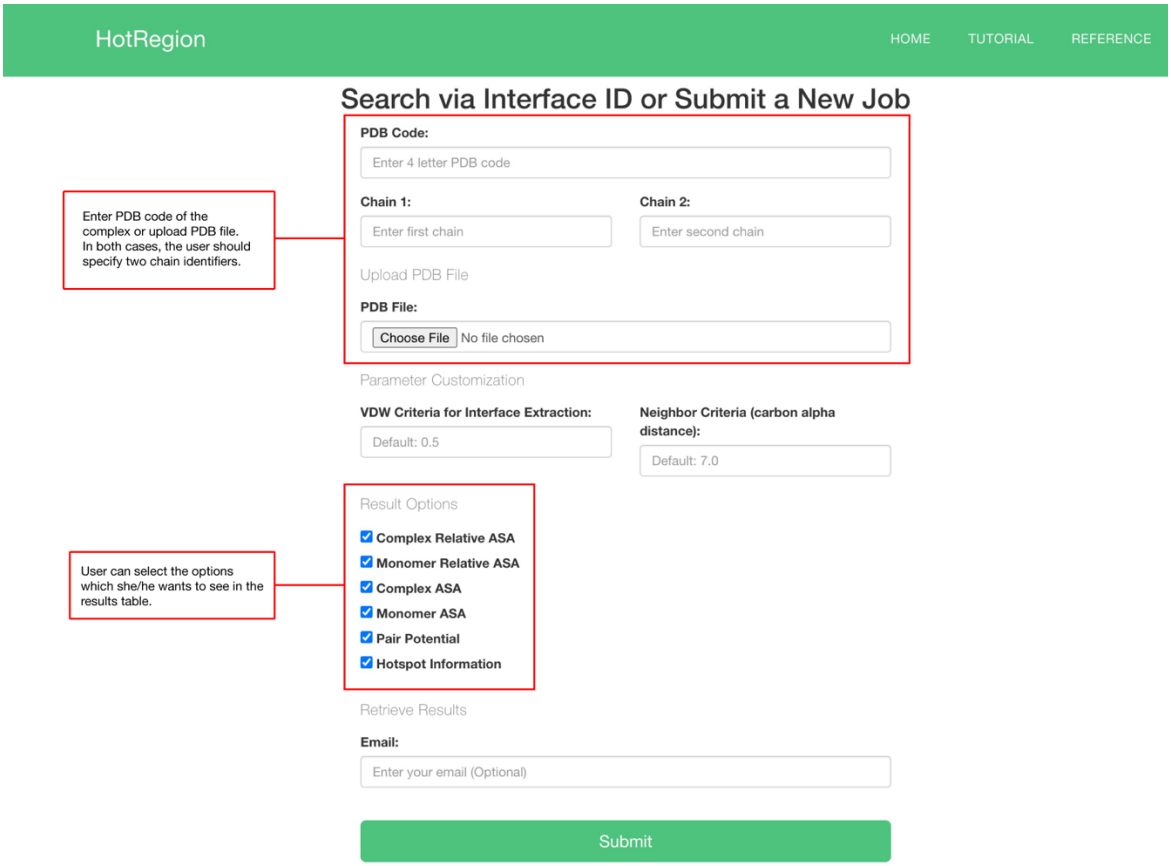

Figure 15: HotRegion v2.0 search and run jobs with advanced options

4.5 Discussion & Future Work

In this study, we have developed a novel method to predict hot regions on protein-protein interfaces using two clustering algorithms in combination. We have prepared a data set of 9 complexes with 189 residues to assess our method. We have measured the inter- and intra-distances of our clusters by silhouette constant. Results have demonstrated that silhouette constant values for 8 out of 9 complexes are positive but close to 0. This means that our algorithm successfully packs residues into clusters, however, can be improved to separate overlapping clusters better. Then, we have investigated each complex individually. The results of the complexes 1JTGAB, and 1A22AB have shown that the algorithm fails to separate two clusters from each other since there are residues which interact with residues from both clusters. This can be expected since the silhouette constant is close to 0. As a future work, the algorithm can be improved to separate neighboring hot regions from each other.

On the other hand, the results have proven that the algorithm successfully places the majority of the residues into correct hot regions with 0.68 precision, 0.62 accuracy, 0.86 recall, and 0.76 F-measure. Even though these metrices can be improved in the future, we claim that our algorithm has succeeded to produce satisfactory results.

Additionally, we have designed and implemented a web interface, and constructed a relational database to store hot region results of more than 600.000 protein-protein complexes. We have developed a friendly user interface using Bootstrap. We have implemented the web interface in Python 3 to achieve best performance in terms of time efficiency. For memory efficiency, we have chosen to store results as a json in the database.

Finally, we have conducted a case study on PD-1/PD-L1 complex. Our results have demonstrated that the investigation of protein-protein interactions is essential for medical studies and hot regions are of great importance in protein-protein interactions. For PD-1/PD-L1 complex, the residues found in the hot regions are crucial for small inhibitor molecule designs. Therefore, researchers can use our prediction tool for such studies.

Chapter 5:
CONCLUSION

In this thesis, we develop a hot region prediction tool which has three major components. These components are a standalone prediction software, a web interface and a relational database.

The prediction software analyzes interfaces and predicts hot regions on PPIs. The algorithm, first, extracts interface residues from structure file in PDB format. Then, it calculates structural features of interface residues which are ASA and relative ASA values in monomer and complex state, and total contact potentials. ASA related features are calculated using an external software Naccess, and the knowledge-based pair potentials from Keskin et al [4]. The algorithm combines two well-known unsupervised learning algorithms AP and DBSCAN. AP is run with the 3D coordinates of the residues and calculated features. From the results of AP, density parameters are calculated and fed to DBSCAN. Finally, hot regions are found using DBSCAN.

Researchers can easily obtain hot region information and visualize 3D structure of the complexes using the web interface. The database stores more than 600.000 complexes as of now and it updates itself automatically. The web interface provides advanced search capabilities and filtering options and allows users to customize algorithm parameters.

We have evaluated HotRegion v2.0 algorithm in light of experimental findings. The evaluation metrices have demonstrated that our results are highly correlated with the results of experimental studies. Furthermore, we have conducted a case study and proven that our tool is very beneficial for researchers to investigate protein-protein interactions for drug and inhibitor molecule design studies.

HotRegion v2.0 is a novel approach as it forms hot regions from both hot spot and non-hot spot residues whereas the existing approaches cluster only hot spot residues. Furthermore, comparing with the experimental results, we have proven that our approach is correct which means hot regions contains non-hot spot residues as well. Additionally, the web interface of HotRegion v2.0 provides users a keyword-based search opportunity which existing software tools do not offer. In this way, users do not necessarily have to know PDB ID of the complexes to obtain hot region information.

In conclusion, we have developed a novel, successful prediction software to predict hot regions on PPIs and offered a user friendly, easy to use web interface for the investigation and the visualization of hot regions. We have carefully chosen and used external tools, libraries, plugins, databases, and frameworks. As a result, we have generated a software product which is novel, computationally efficient, and satisfactory in terms of prediction performance.

BIBLIOGRAPHY

[1] E. Cukuroglu, A. Gursoy, and O. Keskin, “HotRegion: a database of predicted hot spot clusters,” *Nucleic acids research,* vol. 40, no. D1, pp. D829-D833, 2012.

[2] N. Tuncbag, A. Gursoy, and O. Keskin, “Identification of computational hot spots in protein interfaces: combining solvent accessibility and inter-residue potentials improves the accuracy,” *Bioinformatics,* vol. 25, no. 12, pp. 1513-1520, 2009.

[3] G. Buhrman, O. Casey, B. Zerbe, B. M. Kearney, R. Napoleon, E. A. Kovrigina, S. Vajda, D. Kozakov, E. L. Kovrigin, and C. Mattos, “Analysis of binding site hot spots on the surface of Ras GTPase,” *Journal of molecular biology,* vol. 413, no. 4, pp. 773-789, 2011.

[4] D. Reichmann, O. Rahat, S. Albeck, R. Meged, O. Dym, and G. Schreiber, “The modular architecture of protein–protein binding interfaces,” *Proceedings of the National Academy of Sciences,* vol. 102, no. 1, pp. 57-62, 2005.

[5] H. Nakhaeizadeh, E. Amin, S. Nakhaei-Rad, R. Dvorsky, and M. R. Ahmadian, “The RAS-effector interface: isoform-specific differences in the effector binding regions,” *PloS one,* vol. 11, no. 12, pp. e0167145, 2016.

[6] J. Hu, and X. Zhang, “Improving hot region prediction by parameter optimization of density clustering in PPI,” *Methods,* vol. 110, pp. 35-43, 2016.

[7] L. Xiaoli, and Z. Xiaolong, “Prediction of Hot Regions in PPIs Based on Improved Local Community Structure Detecting,” *IEEE/ACM Trans Comput Biol Bioinform,* vol. 15, no. 5, pp. 1470-1479, Sep-Oct, 2018.

[8] D. Nan, and X. Zhang, "Prediction of hot regions in protein-protein interactions based on complex network and community detection." pp. 17-23.

[9] B. J. Frey, and D. Dueck, “Clustering by passing messages between data points,” *science,* vol. 315, no. 5814, pp. 972-976, 2007.

[10] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise." pp. 226-231.

[11] N. Tuncbag, O. Keskin, and A. Gursoy, “HotPoint: hot spot prediction server for protein interfaces,” *Nucleic acids research,* vol. 38, no. suppl_2, pp. W402-W406, 2010.

[12] O. Keskin, I. Bahar, R. Jernigan, A. Badretdinov, and O. Ptitsyn, “Empirical solvent-mediated potentials hold for both intra-molecular and inter-molecular inter-residue interactions,” *Protein Science,* vol. 7, no. 12, pp. 2578-2586, 1998.

[13] S. Hubbard, and J. Thornton, “Naccess: Department of biochemistry and molecular biology, university college london,” *Software available at http://www. bioinf. manchester. ac. uk/naccess/nacdownload. html*, 1993.

[14] X. Chen, W. Liu, H. Qiu, and J. Lai, “APSCAN: A parameter free algorithm for clustering,” *Pattern Recognition Letters,* vol. 32, no. 7, pp. 973-986, 2011.

[15] H. Lim, J. Chun, X. Jin, J. Kim, J. Yoon, and K. T. No, “Investigation of protein-protein interactions and hot spot region between PD-1 and PD-L1 by fragment molecular orbital method,” *Sci Rep,* vol. 9, no. 1, pp. 16727, Nov 13, 2019.

[16] P. L. Bartel, and S. Fields, *The yeast two-hybrid system*: Oxford University Press, USA, 1997.

[17] A.-C. Gavin, M. Bösche, R. Krause, P. Grandi, M. Marzioch, A. Bauer, J. Schultz, J. M. Rick, A.-M. Michon, and C.-M. Cruciat, “Functional organization

of the yeast proteome by systematic analysis of protein complexes," *Nature,* vol. 415, no. 6868, pp. 141-147, 2002.

[18] G. Rigaut, A. Shevchenko, B. Rutz, M. Wilm, M. Mann, and B. Séraphin, "A generic protein purification method for protein complex characterization and proteome exploration," *Nature biotechnology,* vol. 17, no. 10, pp. 1030-1032, 1999.

[19] H. Zhu, M. Bilgin, R. Bangham, D. Hall, A. Casamayor, P. Bertone, N. Lan, R. Jansen, S. Bidlingmaier, and T. Houfek, "Global analysis of protein activities using proteome chips," *science,* vol. 293, no. 5537, pp. 2101-2105, 2001.

[20] A. H. Y. Tong, B. Drees, G. Nardelli, G. D. Bader, B. Brannetti, L. Castagnoli, M. Evangelista, S. Ferracuti, B. Nelson, and S. Paoluzi, "A combined experimental and computational strategy to define protein interaction networks for peptide recognition modules," *Science,* vol. 295, no. 5553, pp. 321-324, 2002.

[21] M. Shatnawi, "Review of Recent Protein-Protein Interaction Techniques," *Emerging Trends in Computational Biology, Bioinformatics, and Systems Biology*, pp. 99-121, 2015.

[22] F. Pazos, and A. Valencia, "Similarity of phylogenetic trees as indicator of protein–protein interaction," *Protein engineering,* vol. 14, no. 9, pp. 609-614, 2001.

[23] T. Dandekar, B. Snel, M. Huynen, and P. Bork, "Conservation of gene order: a fingerprint of proteins that physically interact," *Trends in biochemical sciences,* vol. 23, no. 9, pp. 324-328, 1998.

[24] S. Pitre, F. Dehne, A. Chan, J. Cheetham, A. Duong, A. Emili, M. Gebbia, J. Greenblatt, M. Jessulat, and N. Krogan, "PIPE: a protein-protein interaction prediction engine based on the re-occurring short polypeptide sequences between known interacting protein pairs," *BMC bioinformatics,* vol. 7, no. 1, pp. 365, 2006.

[25] S. Pitre, C. North, M. Alamgir, M. Jessulat, A. Chan, X. Luo, J. R. Green, M. Dumontier, F. Dehne, and A. Golshani, "Global investigation of protein–protein interactions in yeast Saccharomyces cerevisiae using re-occurring short polypeptide sequences," *Nucleic acids research,* vol. 36, no. 13, pp. 4286-4294, 2008.

[26] C. Hsin Liu, K.-C. Li, and S. Yuan, "Human protein–protein interaction prediction by a novel sequence-based co-evolution method: co-evolutionary divergence," *Bioinformatics,* vol. 29, no. 1, pp. 92-98, 2013.

[27] A. Matsuya, R. Sakate, Y. Kawahara, K. O. Koyanagi, Y. Sato, Y. Fujii, C. Yamasaki, T. Habara, H. Nakaoka, and F. Todokoro, "Evola: Ortholog database of all human genes in H-InvDB with manual curation of phylogenetic trees," *Nucleic acids research,* vol. 36, no. suppl_1, pp. D787-D792, 2007.

[28] T. Keshava Prasad, R. Goel, K. Kandasamy, S. Keerthikumar, S. Kumar, S. Mathivanan, D. Telikicherla, R. Raju, B. Shafreen, and A. Venugopal, "Human protein reference database—2009 update," *Nucleic acids research,* vol. 37, no. suppl_1, pp. D767-D772, 2009.

[29] Y. Guo, L. Yu, Z. Wen, and M. Li, "Using support vector machine combined with auto covariance to predict protein–protein interactions from protein sequences," *Nucleic acids research,* vol. 36, no. 9, pp. 3025-3030, 2008.

[30] N. Zaki, S. Lazarova-Molnar, W. El-Hajj, and P. Campbell, "Protein-protein interaction based on pairwise similarity," *BMC bioinformatics,* vol. 10, no. 1, pp. 150, 2009.

[31] S. Roy, D. Martinez, H. Platero, T. Lane, and M. Werner-Washburne, "Exploiting amino acid composition for predicting protein-protein interactions," *PloS one,* vol. 4, no. 11, pp. e7813, 2009.

[32] S. Li, C. M. Armstrong, N. Bertin, H. Ge, S. Milstein, M. Boxem, P.-O. Vidalain, J.-D. J. Han, A. Chesneau, and T. Hao, "A map of the interactome network of the metazoan C. elegans," *Science,* vol. 303, no. 5657, pp. 540-543, 2004.

[33] C.-Y. Yu, L.-C. Chou, and D. T.-H. Chang, "Predicting protein-protein interactions in unbalanced data using the primary structure of proteins," *BMC bioinformatics,* vol. 11, no. 1, pp. 167, 2010.

[34] G. T. Valente, M. L. Acencio, C. Martins, and N. Lemke, "The development of a universal in silico predictor of protein-protein interactions," *PloS one,* vol. 8, no. 5, pp. e65587, 2013.

[35] C. Kern, A. J. González, L. Liao, and K. Vijay-Shanker, "Predicting interacting residues using long-distance information and novel decoding in hidden markov models," *IEEE Transactions on NanoBioscience,* vol. 12, no. 3, pp. 158-164, 2013.

[36] A. Porollo, J. Meller, W. Cai, and H. Hong, "Computational methods for prediction of protein-protein interaction sites," *Protein-Protein Interactions-Computational and Experimental Tools,* vol. 472, pp. 3-26, 2012.

[37] N. Tuncbag, A. Gursoy, R. Nussinov, and O. Keskin, "Predicting protein-protein interactions on a proteome scale by matching evolutionary and structural similarities at interfaces using PRISM," *Nature protocols,* vol. 6, no. 9, pp. 1341, 2011.

[38] S. E. A. Ozbabacan, O. Keskin, R. Nussinov, and A. Gursoy, "Enriching the human apoptosis pathway by predicting the structures of protein–protein complexes," *Journal of structural biology,* vol. 179, no. 3, pp. 338-346, 2012.

[39] N. Tuncbag, G. Kar, A. Gursoy, O. Keskin, and R. Nussinov, "Towards inferring time dimensionality in protein–protein interaction networks by integrating structures: the p53 example," *Molecular Biosystems,* vol. 5, no. 12, pp. 1770-1778, 2009.

[40] Q. C. Zhang, D. Petrey, L. Deng, L. Qiang, Y. Shi, C. A. Thu, B. Bisikirska, C. Lefebvre, D. Accili, and T. Hunter, "Structure-based prediction of protein–protein interactions on a genome-wide scale," *Nature,* vol. 490, no. 7421, pp. 556-560, 2012.

[41] K. K. Wan, J. Park, and J. K. Suh, "Large scale statistical prediction of protein-protein interaction by potentially interacting domain (PID) pair," *Genome Informatics,* vol. 13, pp. 42-50, 2002.

[42] D. Han, H.-S. Kim, J. Seo, and W. Jang, "A domain combination based probabilistic framework for protein-protein interaction prediction," *Genome Informatics,* vol. 14, pp. 250-259, 2003.

[43] D.-S. Han, H.-S. Kim, W.-H. Jang, S.-D. Lee, and J.-K. Suh, "PreSPI: a domain combination based prediction system for protein–protein interaction," *Nucleic acids research,* vol. 32, no. 21, pp. 6312-6320, 2004.

[44] W.-H. Jang, S.-H. Jung, and D.-S. Han, "A computational model for predicting protein interactions based on multidomain collaboration," *IEEE/ACM*

transactions on computational biology and bioinformatics, vol. 9, no. 4, pp. 1081-1090, 2012.

[45] I. S. Moreira, P. A. Fernandes, and M. J. Ramos, "Hot spots—A review of the protein–protein interface determinant amino-acid residues," *Proteins: Structure, Function, and Bioinformatics,* vol. 68, no. 4, pp. 803-812, 2007.

[46] D. E. Kim, D. Chivian, and D. Baker, "Protein structure prediction and analysis using the Robetta server," *Nucleic acids research,* vol. 32, no. suppl_2, pp. W526-W531, 2004.

[47] S. J. Darnell, L. LeGault, and J. C. Mitchell, "KFC Server: interactive forecasting of protein interaction hot spots," *Nucleic acids research,* vol. 36, no. suppl_2, pp. W265-W269, 2008.

[48] S. Liu, C. Liu, and L. Deng, "Machine learning approaches for protein–protein interaction hot spot prediction: Progress and comparative assessment," *Molecules,* vol. 23, no. 10, pp. 2535, 2018.

[49] J. Hu, X. Zhang, X. Liu, and J. Tang, "Prediction of hot regions in protein–protein interaction by combining density-based incremental clustering with feature-based classification," *Computers in biology and medicine,* vol. 61, pp. 127-137, 2015.

[50] S. C. Venkateswarlu, M. A. Bhanu, and Y. Katta, "Implementation of K-Means Clustering Algorithm Using Java," *Global Journal of Computer Science and Technology*, 1969.

[51] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE transactions on pattern analysis and machine intelligence,* vol. 17, no. 8, pp. 790-799, 1995.

[52] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing,* vol. 17, no. 4, pp. 395-416, 2007.

[53] W. H. Day, and H. Edelsbrunner, "Efficient algorithms for agglomerative hierarchical clustering methods," *Journal of classification,* vol. 1, no. 1, pp. 7-24, 1984.

[54] N. Rahmah, and I. S. Sitanggang, "Determination of optimal epsilon (eps) value on dbscan algorithm to clustering data on peatland hotspots in sumatra." p. 012012.

[55] R. J. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates." pp. 160-172.

[56] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: ordering points to identify the clustering structure," *ACM Sigmod record,* vol. 28, no. 2, pp. 49-60, 1999.

[57] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: an efficient data clustering databases method for very large." pp. 103-114.

[58] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, "The protein data bank," *Nucleic acids research,* vol. 28, no. 1, pp. 235-242, 2000.

[59] B. Chapman, and J. Chang, "Biopython: Python tools for computational biology," *ACM Sigbio Newsletter,* vol. 20, no. 2, pp. 15-19, 2000.

[60] B. Lee, and F. M. Richards, "The interpretation of protein structures: estimation of static accessibility," *Journal of molecular biology,* vol. 55, no. 3, pp. 379-IN4, 1971.

[61] S. Hubbard, "NACCESS: program for calculating accessibilities," *Department of Biochemistry and Molecular Biology, University College of London*, 1992.

[62] S. Bhattacharyya, "DBSCAN Algorithm: Complete Guide and Application with Python Scikit-Learn," 9th June, 2019.
[63] M. J. Conroy, D. Sehnal, M. Deshpande, R. Svobodova, S. Mir, K. Berka, A. Midlik, S. Velankar, and J. Koc, "LiteMol: web-based 3D visualization of macromolecular structure data." pp. C669-C669.
[64] N. A. Meenan, A. Sharma, S. J. Fleishman, C. J. MacDonald, B. Morel, R. Boetzel, G. R. Moore, D. Baker, and C. Kleanthous, "The structural and energetic basis for high selectivity in a high-affinity protein-protein interaction," *Proceedings of the National Academy of Sciences,* vol. 107, no. 22, pp. 10080-10085, 2010.
[65] D. M. Pardoll, "The blockade of immune checkpoints in cancer immunotherapy," *Nature Reviews Cancer,* vol. 12, no. 4, pp. 252-264, 2012.
[66] Y. Ishida, Y. Agata, K. Shibahara, and T. Honjo, "Induced expression of PD-1, a novel member of the immunoglobulin gene superfamily, upon programmed cell death," *The EMBO journal,* vol. 11, no. 11, pp. 3887-3895, 1992.

Appendix A: Hot Region Prediction Algorithm using AP & DBSCAN

Main.py

```
from Bio import BiopythonWarning
from Bio.PDB import *
import warnings
import sys, os
from interface_extraction.extract import InterfaceExtroctor
from clustering.dbscan import DoubleDBSCAN
import pandas as pd
from clustering.apscan import run_apscan
import urllib.request
from helpers import write2file, get_com
from feature_calculation.asa import ASACalculator
from feature_calculation.pp import PPCalculator
from sklearn import metrics

def parse_pdb(structure, pdb_id, chain_id_1, chain_id_2):
    os.system("chmod -R 777 PDBs")

    class Chain1Select(Select):
        def accept_chain(self, chain):
            if chain.id==chain_id_1:
                return 1
            else:
                return 0

    class Chain2Select(Select):
        def accept_chain(self, chain):
            if chain.id==chain_id_2:
                return 1
            else:
                return 0
```

```
        class Chain12Select(Select):
            def accept_chain(self, chain):
                if (chain.id==chain_id_1) or (chain.id==chain_id_2):
                    return 1
                else:
                    return 0
        io = PDBIO()
        io.set_structure(structure)
        io.save("PDBs/%s%s.pdb" %(pdb_id.lower(), chain_id_1.lower()),
Chain1Select())
        io.save("PDBs/%s%s.pdb" %(pdb_id.lower(), chain_id_2.lower()),
Chain2Select())
        io.save("PDBs/%s%s.pdb" %(pdb_id.lower(),
chain_id_1.lower()+chain_id_2.lower()), Chain12Select())
        return


    def hotregion(interface_id):
        '''
            HotRegion v.2: A new method to predict hot regions in protein-protein
interfaces
            Usage: python main.py <interface id>

            Parameters:
            ----------
            interfaceID: String
                    4 letter pdb code + chain 1 identifier + chain 2 identifier.

            Returns:
            -------
            hotRegions: CSV file
                    Output file showing which residues belong to which regions and which
residues are identified as noise.
        '''
```

```
        pdb_id = interface_id[0:4].upper()
        chain_id_1 = interface_id[4].upper()
        chain_id_2 = interface_id[5].upper()

        path = "PDBs/%s.pdb" %(pdb_id)
        url = "http://files.rcsb.org/download/%s.pdb" %(pdb_id)

        if os.path.exists(path) == False:
           urllib.request.urlretrieve(url, path)

         # Set and parse protein structure
        parser = PDBParser()
        structure = parser.get_structure('myPDBStructure', "PDBs/%s.pdb" %(pdb_id))
        parse_pdb(structure, pdb_id, chain_id_1, chain_id_2)
        structure = parser.get_structure('myPDBStructure', "PDBs/%s%s.pdb"
%(pdb_id.lower(), chain_id_1.lower()+chain_id_2.lower()))
        model = structure[0]
        chain_1 = model[chain_id_1]
        chain_2 = model[chain_id_2]
        residue_list = Selection.unfold_entities(structure, 'R')

        # Interface Extraction
        interface_extractor = InterfaceExtroctor(structure, chain_1, chain_2)
        interface_extractor.extract_interface()
        interface_extractor.non_interface = set(residue_list) -
interface_extractor.interface_list
        #interface_extractor.extract_neighbors()
        #interface_extractor.interface_list.update(interface_extractor.neighbor_list)

        # Feature Calculation
        asa_calculator = ASACalculator("PDBs", interface_id, chain_1, chain_2)
        asa_calculator.run_naccess()
```

```
    pp_calculator = PPCalculator(residue_list)
    pp_calculator.calculate_pp()

    # DataFrame Generation
    data_dict = {'X':[],'Y':[], 'Z':[], 'relCompASA':[], 'relMonASA':[], 'compASA':[],
'monASA':[], 'PP':[]}
    for residue in list(interface_extractor.interface_list):
        try:
            com = get_com(residue)
            data_dict['X'].append(com[0])
            data_dict['Y'].append(com[1])
            data_dict['Z'].append(com[2])
            data_dict['relCompASA'].append(asa_calculator.relative_complex_asa[resid
            ue])
            data_dict['relMonASA'].append(asa_calculator.relative_monomer_asa[resid
            ue])
            data_dict['compASA'].append(asa_calculator.complex_asa[residue])
            data_dict['monASA'].append(asa_calculator.monomer_asa[residue])
            data_dict['PP'].append(pp_calculator.pair_potentials[residue])
        except:
            interface_extractor.interface_list.remove(residue)

    df = pd.DataFrame(data_dict)
    labels = run_apscan(df)
    silhouette = metrics.silhouette_score(df, labels, metric='euclidean')
    write2file(interface_id,
        interface_extractor.interface_list,
        labels,
        asa_calculator.complex_asa,
        asa_calculator.relative_complex_asa,
        asa_calculator.monomer_asa,
        asa_calculator.relative_monomer_asa,
        pp_calculator.pair_potentials
```

```
        )
        print(f"Silhouette score of {interface_id}: {silhouette}")
    if __name__ == '__main__':
        warnings.simplefilter('ignore', BiopythonWarning)
        if len(sys.argv) == 2:
            interface_id = sys.argv[1].upper()
            hotregion(interface_id)
        else:
            print("Usage: python main.py <interface id>")
```

Helpers.py

```
    import pandas as pd
    from dictionaries import DICT_ATOM
    import math

    def three2one(residue):
        if residue ==   "ALA": return "A"
        elif residue == "CYS": return "C"
        elif residue == "ASP": return "D"
        elif residue == "GLU": return "E"
        elif residue == "PHE": return "F"
        elif residue == "GLY": return "G"
        elif residue == "HIS": return "H"
        elif residue == "ILE": return "I"
        elif residue == "LYS": return "K"
        elif residue == "LEU": return "L"
        elif residue == "MET": return "M"
        elif residue == "ASN": return "N"
        elif residue == "PRO": return "P"
        elif residue == "GLN": return "Q"
        elif residue == "ARG": return "R"
        elif residue == "SER": return "S"
        elif residue == "THR": return "T"
        elif residue == "VAL": return "V"
```

```
    elif residue == "TRP": return "W"
    elif residue == "TYR": return "Y"
    else: return "X"

def write2file(interfaceID, residue_list, labels, compASAs, relCompASAs,
monASAs, relMonASAs, PPs):
    res_ids = []
    res_names = []
    chain_ids= []
    comp_ASAs = []
    rel_comp_ASAs = []
    mon_ASAs = []
    rel_mon_ASAs = []
    pair_potentials = []
    info = []
    for residue in residue_list:
      res_ids.append( str(residue.id[1]) )
      res_names.append( three2one(residue.get_resname()) )
      chain_ids.append( residue.get_parent().id )
      comp_asa = "%6.2f"%(compASAs[residue])
      comp_ASAs.append(comp_asa)
      rel_comp_asa = "%6.2f"%(relCompASAs[residue])
      rel_comp_ASAs.append(rel_comp_asa)
      mon_asa = "%6.2f"%(monASAs[residue])
      mon_ASAs.append(mon_asa)
      rel_mon_asa = "%6.2f"%(relMonASAs[residue])
      rel_mon_ASAs.append(rel_mon_asa)
      pp = "%6.2f"%(PPs[residue])
      pair_potentials.append(pp)
      if (float(relCompASAs[residue]) <=20 and float(PPs[residue])>=18.0):
        info.append('HS')
      else:
        info.append('NS')
```

```
        d = {'RES_ID': res_ids, 'RES_NAME': res_names, 'CHAIN': chain_ids,
'CLUSTER': labels, 'Complex ASA':comp_ASAs, 'Relative Complex ASA':
rel_comp_ASAs, 'Monomer ASA':mon_ASAs, 'Relative Monomer ASA':
rel_mon_ASAs, 'PP': pair_potentials, 'Status': info}
        df = pd.DataFrame(data=d)
        df.to_csv(interfaceID+'_data.csv', sep=',', encoding='utf-8')

    def distance_calculation(vector_1, vector_2):
        x1 = vector_1[0]
        y1 = vector_1[1]
        z1 = vector_1[2]

        x2 = vector_2[0]
        y2 = vector_2[1]
        z2 = vector_2[2]

        distance = math.sqrt(((x2-x1)**2)+((y2-y1)**2)+((z2-z1)**2))
        return distance

    def get_com(residue):
        x_total = 0
        y_total = 0
        z_total = 0
        if(len(residue.get_resname()) >3):
            resname = residue.get_resname()[1:4]
        else:
            resname = residue.get_resname()
        num_atoms = len(DICT_ATOM[resname])
        for atom_id in DICT_ATOM[resname]:
            atom = residue[atom_id]
            x_total += atom.get_coord()[0]
            y_total += atom.get_coord()[1]
            z_total += atom.get_coord()[2]
```

```
        return [x_total/num_atoms,y_total/num_atoms,z_total/num_atoms]
```

Interface_extraction.extract.py

```
    import pandas as pd
    import dictionaries

    class InterfaceExtroctor:
        def __init__(self, structure, chain_1, chain_2):
            self.structure = structure
            self.chain_1 = chain_1
            self.chain_2 = chain_2
            self.interface_list = set()
            self.non_interface = set()
            self.neighbor_list = set()

        def is_atoms_interacting(self, atom_1, atom_2):
            r1 =
dictionaries.VDW_RADII_EXTENDED[atom_1.get_parent().get_resname()][atom_1.g
et_id()]
            r2 =
dictionaries.VDW_RADII_EXTENDED[atom_2.get_parent().get_resname()][atom_2.g
et_id()]
            distance = abs(atom_1 - atom_2)
            return True if distance < (r1+r2+0.5) else False


        def is_residues_interacting(self, residue_1, residue_2):
            for atom_1 in residue_1:
                for atom_2 in residue_2:
                    try:
                        if self.is_atoms_interacting(atom_1, atom_2):
                            return True
                    except:
                        pass
```

```
        return False

    def is_interface_residue(self, residue_1):
        for residue_2 in self.chain_2:
            if self.is_residues_interacting(residue_1, residue_2):
                self.interface_list.add(residue_1)
                self.interface_list.add(residue_2)
        return

    def extract_interface(self):
        for residue_1 in self.chain_1:
            self.is_interface_residue(residue_1)

    def find_neighbors(self, residue):
        for neighbor in self.non_interface:
            if neighbor.get_resname() != "HOH":
                if neighbor.get_parent() == residue.get_parent():
                    try:
                        if (abs(residue['CA'] - neighbor['CA']) <= 6) and
(abs(residue.get_id()[1]-neighbor.get_id()[1]) >= 4):
                            self.neighbor_list.add(neighbor)
                    except:
                        pass
                else:
                 try:
                        if abs(residue['CA'] - neighbor['CA']) <= 6:
                                self.neighbor_list.add(neighbor)
                 except:
                        pass

    def extract_neighbors(self):
        for residue in self.interface_list:
            self.find_neighbors(residue)
```

Feature_calculation.asa.py

```
import os
import re
from dictionaries import MAX_ASA

class ASACalculator:

    def __init__(self, pdb_path, interfaceID, chain_1, chain_2):
            self.pdb_path = pdb_path
            self.interfaceID = interfaceID
            self.chain_1 = chain_1
            self.chain_2 = chain_2
            self.complex_asa = dict()
            self.monomer_asa = dict()
            self.relative_complex_asa = dict()
            self.relative_monomer_asa = dict()

    ###################### Calculate RelASA ########################
    def parse_naccess_file(self):
            pdb_id = self.interfaceID[0:4]
            file = self.interfaceID
            nfile = open(self.pdb_path+ "/" +file.lower()+".naccess","r")
            line = nfile.readline()
            while line:
                    if line.startswith("RES"):
                            resPosition = line[9:14].replace(" ","")
                            chain = line[8]
                            if (chain == self.chain_1.id) :
                                    residue = self.chain_1[int(re.sub("\D", "",
resPosition))]
                            else:
                                    residue = self.chain_2[int(re.sub("\D", "",
resPosition))]
```

```
                        temp = line[14:].strip().split()
                        ASA = float(temp[0])
                        self.complex_asa[residue] = ASA
                        self.relative_complex_asa[residue] = ASA /
MAX_ASA[residue.get_resname()] * 100
                    line = nfile.readline()
                nfile.close()
                file = pdb_id+self.interfaceID[4]
                nfile = open(self.pdb_path+ "/" +file.lower()+".naccess","r")
                line = nfile.readline()
                while line:
                    if line.startswith("RES"):
                        resPosition = line[9:14].replace(" ","")
                        chain = line[8]
                        residue = self.chain_1[int(re.sub("\D", "", resPosition))]
                        temp = line[14:].strip().split()
                        ASA = float(temp[0])
                        self.monomer_asa[residue] = ASA
                        self.relative_monomer_asa[residue] = ASA /
MAX_ASA[residue.get_resname()] * 100
                    line = nfile.readline()
                nfile.close()
                file = pdb_id+self.interfaceID[5]
                nfile = open(self.pdb_path+ "/" +file.lower()+".naccess","r")
                line = nfile.readline()
                while line:
                    if line.startswith("RES"):
                        resPosition = line[9:14].replace(" ","")
                        chain = line[8]
                        residue = self.chain_2[int(re.sub("\D", "", resPosition))]
                        temp = line[14:].strip().split()
                        ASA = float(temp[0])
                        self.monomer_asa[residue] = ASA
```

```
                                self.relative_monomer_asa[residue] = ASA /
MAX_ASA[residue.get_resname()] * 100
                        line = nfile.readline()
                nfile.close()
                return

        def calc_sasa(self, structure):
                pdb_file = structure.lower()
                pdb_ID = pdb_file
                pdb_file = "%s.pdb" %(pdb_file)
                cmd = "%s/naccess %s > out.txt 2>error.txt" %("naccess",pdb_file)
                os.system("cp %s/%s.pdb naccess/%s.pdb " %(self.pdb_path, pdb_ID,
pdb_ID))
                os.chmod("naccess/%s.pdb" %(pdb_ID), 777)
                os.chdir("naccess")
                os.system(cmd)
                os.chdir("..")
                os.system("cp naccess/%s.rsa %s/%s.naccess" %(pdb_ID, self.pdb_path,
pdb_ID))
                os.system("chmod -R 777 %s" %(self.pdb_path))
                os.system("chmod -R 777 naccess")
                os.system("rm naccess/*.log naccess/*.asa naccess/*.pdb naccess/*.rsa")
                return

        def run_naccess(self):
                pdb_id = self.interfaceID[0:4]
                for structure in [pdb_id+self.interfaceID[4], pdb_id+self.interfaceID[5],
self.interfaceID]:
                        self.calc_sasa(structure)
                self.parse_naccess_file()
```

Feature_calculation.pp.py

```
from dictionaries import CONTACT_POTENTIALS, PP_indices
```

```
from helpers import get_com, distance_calculation

class PPCalculator:

    def __init__(self, residue_list):
            self.residue_list = residue_list
            self.pair_potentials = dict()


    def extract_neighbors(self, residue):
            com_residue = get_com(residue)
            neighbors = []
            for neighbor in self.residue_list:
                try:
                        com_neighbor = get_com(neighbor)
                        if (residue != neighbor):
                                distance =
distance_calculation(com_residue,com_neighbor)
                                if (residue.get_parent().id ==
neighbor.get_parent().id): #they are in the same chain
                                        if(abs(residue.get_id()[1]-
neighbor.get_id()[1]) >= 4):
                                                if (distance <= 7.0):
                                                        neighbors.append(neighbor)
                                else:
                                        if (distance <= 7.0):
                                                neighbors.append(neighbor)
                except:
                        pass
            return neighbors

    def contact_potentials(self, residue):
            neighbors = self.extract_neighbors(residue)
```

```
            sum_pp = 0.0
            for neighbor in neighbors:
                indexI = PP_indices[residue.get_resname()]
                indexJ = PP_indices[neighbor.get_resname()]
                if(CONTACT_POTENTIALS[indexI][indexJ] != 0):
                    sum_pp+=CONTACT_POTENTIALS[indexI][indexJ]
                else:
                    sum_pp+=CONTACT_POTENTIALS[indexJ][indexI]
            return sum_pp

        def calculate_pp(self):
            for residue in self.residue_list:
                try:
                    pp = self.contact_potentials(residue)
                    self.pair_potentials[residue] = abs(pp)
                except:
                    pass
```

Clustering.apscan.py:

```
from sklearn.preprocessing import normalize
from sklearn.cluster import AffinityPropagation
import pandas as pd
import math
import numpy as np
from clustering.dbscan import DoubleDBSCAN

# Step 3: DBSCAN
def run_dbscan(data, density_list):
    min_points_new = min(density_list, key = lambda t: t[0])[0]
    eps2 = sum(n for _, n, _ in density_list) / len(density_list)
    db = DoubleDBSCAN(eps2, min_points_new)
    results = db.cluster(data)
    return results
```

```
# Step 2: Density List Generation
def calc_distance(vec1, vec2):
    x1 = vec1[0]
    y1 = vec1[1]
    z1 = vec1[2]

    x2 = vec2[0]
    y2 = vec2[1]
    z2 = vec2[2]

    return math.sqrt(((x2-x1)**2)+((y2-y1)**2)+((z2-z1)**2))


def generate_density_list(data, cluster_centers_indices, labels):
    n_clusters_ = len(cluster_centers_indices)
    density_list = []
    for k in range(n_clusters_):
        class_members = labels == k
        cluster_center = cluster_centers_indices[k]
        radius = 0
        x_center = data.loc[cluster_center]['X']
        y_center = data.loc[cluster_center]['Y']
        z_center = data.loc[cluster_center]['Z']
        distances = []
        num_members = 0
        for member in range(len(class_members)):
            if(class_members[member]):
                num_members += 1
                x_member = data.loc[member]['X']
                y_member = data.loc[member]['Y']
                z_member = data.loc[member]['Z']
                distance = calc_distance([x_center, y_center, z_center], [x_member,
y_member, z_member])
```

```
                distances.append(distance)
                radius += distance
        radius = radius / num_members
        df = pd.DataFrame(distances, columns=['d'])
        df = df[df.d <= radius]
        if df.shape[0] > 1:
            number = df.shape[0]
            try:
                density = number/math.pow(radius, 2)
                density_list.append((number, radius, density))
            except OverflowError:
                pass
    density_list.sort(reverse=True, key=lambda tup: tup[2])
    return density_list

# Step 1: Run Affinity Propagation
def affinity_propagation(data):
    clusterer = AffinityPropagation()
    af = clusterer.fit(data)
    return af

def run_apscan(data):
    af = affinity_propagation(data)
    cluster_centers_indices = af.cluster_centers_indices_
    labels = af.labels_
    density_list = generate_density_list(data, cluster_centers_indices, labels)
    final_result = run_dbscan(data, density_list)
    return final_result
```

Clustering.dbscan.py

```
from sklearn.cluster import DBSCAN
class DoubleDBSCAN:
    def __init__(self, epsilon, min_points):
        self.epsilon = epsilon
```

```
        self.min_points = min_points

    def cluster(self, data):
        db = DBSCAN(eps=self.epsilon, min_samples=self.min_points).fit(data)
        labels = db.labels_
        return labels
```