Stony Brook University

CLASSIFICATION WITH PARTIAL INFORMATION FOR BIOINFORMATICS AND TEXT SEGMENTATION

A DISSERTATION PRESENTED

BY

CHANG ZHAO

TO

THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

COMPUTER SCIENCE

STONY BROOK UNIVERSITY

DECEMBER 2009

Stony Brook University

The Graduate School

Chang Zhao

We, the dissertation committee for the above candidate for the degree of Doctor of Philosophy, hereby recommend acceptance of this dissertation.

I.V. Ramakrishnan - Dissertation Advisor
Professor of Computer Science

Michael Kifer - Dissertation Advisor
Professor of Computer Science

Steve Skiena - Chairperson of Defense
Professor of Computer Science

C.R. Ramakrishnan
Associate Professor of Computer Science

Enrico Pontelli
Professor of Computer Science
New Mexico State University

This dissertation is accepted by the Graduate School.

Lawrence Martin
Dean of the Graduate School

Abstract of the Dissertation

Classification with Partial Information for Bioinformatics and Text Segmentation

by

Chang Zhao

Doctor of Philosophy

in

Computer Science

Stony Brook University

2009

Bioinformatics and text segmentation have attracted enormous research efforts in recent years. Classification techniques, especially profile hidden Markov model (PHMM) and conditional random field (CRF), are established computational vehicles in these two fields for extracting useful information from the vast amount of data resulted from rapid progress in molecular biology and ever increasing World Wide Web activities.

In this dissertation, we have developed techniques that exploit partial information to overcome a number of significant limitations of extant PHMM and CRF techniques in bioinformatics and text segmentation. Our work has advanced classification techniques in these two fields along the PHMM and CRF directions.

Our research on classification in bioinformatics has been conducted in the context of Toxin Knowledge Base (TKB), a comprehensive bioinformatics resource to detect potential virulent proteins. One of the most important research problems in TKB is to improve the accuracy of predicting whether a protein is potentially virulent based on sequence homology and active site similarity.

PHMM is recognized as the state-of-the-art for detecting sequence homology. Extant PHMM training approaches either use completely unaligned or completely aligned sequences. The PHMMs resulted from these two training approaches present contrasting trade-offs w.r.t. alignment information and the accuracy of the search outcome. Producing the complete alignment information is a labor intensive process involving expensive structural analysis of entire sequences. We have developed a PHMM training technique that is parameterized w.r.t. alignment information. Our technique can improve the accuracy of PHMMs when training sequences are only partially aligned.

Current techniques for profiling 3-D biological structures with PHMM are restricted in that they only deal with entire protein structures and cannot be applied to important functional substructures such as active sites. We have expanded PHMM to profile protein active sites for their similarity search. The core of our technique is a novel serialization that captures certain conserved physico-chemical and structural features of active sites. Although our sequential representation of active sites captures only partial information about them, experiments show that our technique is practical.

In the field of text segmentation, one of the biggest limitations of existing CRF approaches is the need for manual labeling of training data, which is generally labor intensive and time consuming. We have developed a CRF training technique that can eliminate the manual work needed for labeling examples and automatically learn CRF from partial training information in structured reference data.

Our experiences show that our partial information exploiting techniques can improve PHMM classification accuracy when completely aligned sequences are not available, expand PHMM applicability on 3-D structures, or eliminate manual work in labeling training sequences for CRF.

Contents

List of Tables viii

List of Figures ix

1 Introduction 1

1.1 Classification in Bioinformatics and Text Segmentation 1

1.2 Our approach 4

1.2.1 Classification in Bioinformatics with PHMM 4

1.2.2 Automatic Text Segmentation with CRF 7

1.3 Research Contributions 8

1.4 Dissertation Organization 10

2 Preliminaries 11

2.1 Protein Structure 11

2.2 Profile Hidden Markov Model (PHMM) 13

2.3 Conditional Random Field (CRF) 18

3 Toxin Knowledge Base 23

3.1 Introduction 23

3.2 TKB System Architecture 24

3.3 MuToxin Process 28

3.4 Research Issues 33

4 Protein Profiling with Partially Aligned Sequences 34

4.1 Introduction 34

4.2 Techniques 36

4.2.1 Partially Aligned Sequences 36

4.2.2 Model Decomposition 38

4.2.3 Parameter Composition 40

4.3 Evaluation 43

4.3.1 Experimental Setup 44

4.3.2 Recall and Precision 44

4.3.3 Effect of Varying Training Set 45

4.3.4 Effect of Varying Alignment Information 46

4.4 Related Work 50

5 Protein Active Site Profiling 52

5.1 Introduction 52

5.2 Techniques 54

5.2.1 Serializing Active Sites 55

5.2.2 PHMM for Active Sites 57

5.3 Evaluation 61

5.3.1 Experimental Setup 61

5.3.2 Performance Metrics 63

5.3.3 Experimental Results 63

5.3.4 Discussions 65

5.4 Related Work 65

6 Automatic Text Segmentation with CRF **68**

6.1 Introduction 68

6.2 Techniques 71

6.2.1 Reference Table 71

6.2.2 Attribute CRF 74

6.2.3 Unsupervised Text Segmentation with Attribute CRF 78

6.3 Evaluation 83

6.3.1 Experimental Setup 83

6.3.2 Brute Force Segmentation 85

6.3.3 Experimental Results 86

6.4 Related Work 92

7 Conclusion **95**

Bibliography **97**

List of Tables

2.1 Example CRF with Two Labels 21
2.2 Possible Ways of Labeling [Huntington, NY] 22
3.1 TKB Statistics 28
3.2 Botulinum E and Thermolysin Alignment Statistics 30
3.3 Chitinase-3 like protein-1 and Endoglucanase E Alignment Statistics 31
4.1 Transition Expectations 42
5.1 Example Active Site Atoms 56
5.2 Data Statistics for Different Protein Families 61
5.3 Data Statistics for Subfamilies of Nu:-His-Elec Catalytic Triad 61
6.1 A Segment of a Reference Table 70
6.2 Negatively Labeled Examples 77
6.3 Estimated Starting Positions 82
6.4 Datasets Used for Experiments 84
6.5 Bibliography Reference Table Fragment 85
6.6 Performance of UnsupervisedSeg with Noisy Data 89

List of Figures

2.1 (a) Schematic Diagram of an Amino Acid. (b) Polypeptide Chain. 12
2.2 Sequence of Bovine Ribonuclease 12
2.3 Formation of an Active Site 13
2.4 Active Site of Botulinum Neurotoxin Serotype E(PDB ID: 1T3A) 14
2.5 A Segment from the Multiple Alignment of 7 Globin Protein Sequences . . 15
2.6 PHMM Structure 16
2.7 An Example PHMM 17
2.8 Vertex and Edge Features 19
2.9 A Small Set of Training Examples 21
3.1 TKB System Architecture 26
3.2 TKB Process for Discovery of Potentially Virulent Proteins 28
3.3 Sequence Alignment of Botulinum E and Thermolysin 30
3.4 Similarity between active sites of Botulinum and Thermolysin 31
3.5 Sequence Alignment of Chitinase-3 like protein-1 and Endoglucanase E . . 32
3.6 Similarity between active sites of Chitinase and Endoglucanase E 32
4.1 Partial alignment information for ten ig sequences 37
4.2 Decomposition of a 74 length PHMM structure using the partially aligned sequences in Figure 4.1 39
4.3 Experimental data with 15% training set on the 5 Prosite families 43
4.4 P_PHMM precision against SAM and metaMEME 45

4.5 P_PHMM recall against U_PHMM and TCoffee 45
4.6 Comparing recall of P_PHMM with U_PHMM and metaMEME for (a) ps00012, (b) ps00475, (c) ps00622, (d) ps00675, and (e) ps01330 46
4.7 Impact on P_PHMM F-measure of varying alignment information for the 5 PROSITE families 47
4.8 Impact on P_PHMM F-measure of varying alignment information for ig family 48
4.9 The 5 SCOP families 48
4.10 Impact on P_PHMM performance of varying alignment information for the SCOP families (a) a1.1.2, (b) b1.1.2, (c) b.34.2.1, (d) c.47.1.5, and (e) d.169.1.1 49
5.1 A Segment of a Multiple Alignment 58
5.2 Precision Performance of (a) 5 Protein Families and (b) Subfamilies of Nu:-His-Elec 63
5.3 Recall Performance of (a) 5 Protein Families and (b) Subfamilies of Nu:-His-Elec 64
5.4 F-Measure Performance of (a) 5 Protein Families and (b) Subfamilies of Nu:-His-Elec 64
6.1 Restaurant Addresses Presented in the Same Attribute Order 73
6.2 Example Test Token Sequences 73
6.3 Example Training Data Derived from Reference Table 74
6.4 Performance for Address Dataset 87
6.5 Performance for Product Dataset 87
6.6 Performance for Bibliography Dataset 87
6.7 Accuracy of UnsupervisedSeg with Noisy Data 88
6.8 Varying Training Set Size of Address Dataset (a) Running Time (b) F-Measure 90

6.9 Varying Training Set Size of Product Dataset (a) Running Time (b) F-Measure 90
6.10 Varying Training Set Size of Bibliography Dataset (a) Running Time (b) F-Measure 90
6.11 Accuracy of InferOrder 91
6.12 Comparative Performance (Precision) 92
6.13 Comparative Performance (Recall) 92
6.14 Comparative Performance (F-Measure) 92

Chapter 1

Introduction

1.1 Classification in Bioinformatics and Text Segmentation

Recent advances in molecular biology have resulted in an enormous multitude of biological data, such as DNA sequences, RNA sequences, protein sequences, and protein 3-D structures. This has created the rich research area of bioinformatics centered around applying information technologies to analyze these biological data. One of the most important problems in bioinformatics is homology search, which identifies similarities between nucleic or amino acid sequences and structures owing to shared ancestry. Another key bioinformatics problem is gene finding, which identifies stretches of genomic DNA sequences that are biologically functional. This usually refers to protein-coding genes and may also include other functional elements such as RNA genes and regulatory regions.

Text segmentation is the process of partitioning plain text strings into meaningful units. An example is to divide a plain address text string into street number, street, zip code, and state. Text segmentation is crucial to natural language processing and information extraction from the World Wide Web.

Bioinformatics and text segmentation are two fields where classification is widely used. The task of classification is to classify examples into given set of categories based on past observations. For instance, homology search in bioinformatics is a classification problem in that it predicts whether an input biological molecule (DNA, RNA, or protein) is a family member given known members of that family. In the case of text segmentation, partitioning an input text string into meaningful units amounts to classifying each word in the string into predefined categories corresponding to what is meaningful.

Classification with human-crafted rules is labor-intensive and error prone. Researchers have resorted to machine learning based classification approaches which automatically learn classifiers from training examples and use them to accurately predict which category an input example belongs to. Examples of machine learning based classification techniques are decision tree [56], naive Bayesian classifier [69], maximum entropy classifier [54, 47], hidden Markov model [53], and conditional random field [34].

In bioinformatics and text segmentation, classification usually involves sequence analysis tasks that assign a label from a predefined label set to each token in an input sequence, where what constitutes a token is application-specific. For example, in bioinformatics applications a token is usually a letter denoting a nucleic or amino acid whereas in text segmentation tokens can be words that are delimited by white spaces. We use the term "labeling" for such sequence analysis tasks in this dissertation. Usually in the labeling process, the choice of one label affects and is affected by the choices of other labels.

Hidden Markov model (HMM) [53] and conditional random field (CRF) [34] are best-known statistical models for labeling and widely used in bioinformatics and text segmentation. Both are able to model the impact of previously chosen labels on the choices of successive labels and are thus suitable for the aforementioned sequence analysis tasks. A special form of HMM called profile hidden Markov model (PHMM) [18] has been introduced for bioinformatics, whose structure is specialized to capture mutations and conserved regions in biological molecules during evolution.

Both HMM and CRF automatically learn classifiers from training examples and use them to classify new examples. An HMM can be trained from a set of unlabeled sequences by Baum-Welch or from a set of labeled sequences by smoothed maximum likelihood frequency counting [53]. In contrast, CRF is usually trained only from labeled sequences [12, 16, 59]. In bioinformatics, labeled sequences can be automatically derived from a sequence alignment and used to train a PHMM [18].

Although PHMM and CRF have been the preferred statistical models in bioinformatics and text segmentation, existing approaches have several significant limitations.

Extant PHMM training approaches either use completely unaligned sequences [28, 30] or completely aligned sequences [20]. PHMMs trained from completely aligned sequences have been shown to have a much higher degree of classification accuracy than those trained from unaligned sequences. However, producing the complete alignment information is a labor-intensive process involving expensive structural analysis of entire sequences. Partial alignment information that could result in more accurate PHMMs when used in training is ignored by these extant approaches.

Structure (and functional substructure) similarity search is a very important problem in bioinformatics. However, it is less explored compared to sequence homology search. Structures, like sequences, can be grouped into families based on their similarity. Most existing techniques for detecting similarity in structures are based on pair-wise comparison, which can fail to detect remote family members. Even though a profile based approach like PHMM does not suffer from this remote member detection issue, its current use for 3-D biological structures [3] is restricted in that it only deals with entire protein structures and can not be applied to important functional substructures.

CRF is a dominant statistical model used in text segmentation. However, the training of CRF needs manually labeled data [34, 59, 15, 39]. Because manual labeling of training data is generally labor-intensive and time-consuming, the use of CRF in text segmentation is considerably constrained.

1.2 Our approach

In this dissertation, we focus on techniques that exploit partial information for training PHMMs and CRFs in bioinformatics and text segmentation. Our techniques overcome the limitations of existing approaches discussed in Section 1.1. They can significantly improve PHMM classification accuracy when complete alignment information is not available, expand PHMM applicability on 3-D structures, or reduce manual work in labeling training sequences for CRF.

1.2.1 Classification in Bioinformatics with PHMM

Our research on classification in bioinformatics has been conducted in the context of the Toxin Knowledge Base (TKB). We have developed PHMM-based techniques to profile protein sequences from partially aligned sequences and to profile protein active sites, which are 3-D substructures in proteins that determine the functional properties of proteins.

1.2.1.1 Toxin Knowledge Base

Toxin Knowledge Base (TKB) [31] is a comprehensive bioinformatics resource to identify homologues of toxins, structural motifs of toxins and virulent factors in non-toxic proteins. It includes processes for acquiring and structuring toxin data from a number of sources including the web and prediction algorithms for detecting potential virulent proteins based on the aggregated toxin data.

Just as any protein, toxin sequence homology usually indicates common ancestry and thus similarity in functionality. For this reason, one important way of detecting potential virulent proteins adopted by TKB is to search for homologues of toxins. PHMM is recognized as a powerful technique for this task because it can probabilistically model protein sequence families, capture the conserved features among family members, and can thus detect more remote homologues. Improving accuracy of PHMM is an important research

problem in TKB.

Advances in recombinant DNA technology have opened up possibilities for hiding the virulent domain of a toxin in an otherwise non-toxic protein. Such virulent chimeric molecules can not be detected by sequence homology. TKB integrates several bioinformatics and structural biology resources into a specially designed workflow to detect virulent chimeric molecules. Specifically, TKB computes the putative active sites of any given protein and then predicts whether it is a virulent chimeric protein depending on whether the putative active sites are similar to toxin active sites. Active sites, like sequences, can be grouped into families based on similarity. The success of profiles in detecting remote sequence homology inspired us to explore the possibility of profiling active sites in proteins.

1.2.1.2 Profiling Protein Sequences with PHMM

Protein sequence homology search is an effective means of understanding the characteristics of a new protein through comparison to other sequences with known rich biological information. Because organisms from the same ancestor (i.e. belonging to the same family) may have changed during evolution, profile based homology search [6, 26, 5, 20, 30] is more favorable than pair-wise comparison [46, 62, 4, 50] because the profile of a family captures the common features shared by family members and can thus detect more remote homologues of the family members.

Profile hidden Markov model (PHMM) is recognized as a powerful technique for statistically profiling families of protein sequences. A PHMM is a specialized HMM with insert, delete, and match states corresponding to insertion mutations, deletion mutations, and conserved regions in proteins during evolution. The training of a PHMM for a protein sequence family needs example sequences belonging to the family.

Most PHMM training approaches either use completely unaligned [28, 30] or completely aligned sequences [20]. PHMMs trained from completely aligned sequences have been shown to identify remote homologues with a much higher degree of accuracy than

those trained from completely unaligned sequences. However, producing the alignments is a labor intensive process involving expensive structural analysis of proteins.

We have developed a PHMM based technique for profiling protein families from partially aligned sequences [44]. Compared with completely aligned sequences where alignment information is available for each and every amino acid in training sequences, alignment information is only available for some amino acids in partially aligned sequences. By exploiting the observation that partially aligned sequences give rise to independent subsequences, the PHMM for the entire sequences is decomposed into sub-PHMMs corresponding to those subsequences and then the parameters of the PHMM are composed from parameters of the sub-PHMMs.

An interesting aspect of our technique is that it gives rise to a family of PHMMs which are parameterized with regard to the alignment information and thus allows for learning PHMMs that can trade the accuracy of remote homologue identification for labor needed to produce alignment information.

1.2.1.3 Profiling Protein Active Sites with PHMM

Protein active site similarity search finds 3-D substructures in a protein that are similar to the active sites of another proteins. It is a complementary means to sequence homology search in understanding new proteins by comparing them with other known proteins.

State-of-the-art techniques for determining active site similarity [32, 55, 64] are based on pair-wise comparison of active sites. Just like sequences, active sites can be grouped into families whose members are related by similarity of their functions. Similar sites exhibit variability in their physico-chemical and structural features. Pair-wise comparison based techniques may use features that may not be common to all family members and hence can fail to identify remote family members. In contrast, a profile based approach can catch shared features among family members and can thus detect remote members.

The success of PHMM in profiling sequences has inspired people to develop 3-D structural PHMMs. However, high-dimensional PHMMs are computationally intractable, as can be derived by the intractability of high-dimensional hidden Markov models (HMMs) [37]. Researchers have resorted to serialized representations of 3-D structures. Current technique for profiling 3-D protein structures [3] with PHMM serializes amino acids according to their order in 1-D sequence. It can not be applied to 3-D active site profiling because amino acids in similar active sites may not be co-linear in their sequences.

We have developed a PHMM-based technique for profiling protein active sites [71]. To be more specific, we have developed a novel serialization of 3-D active sites which does not depend on order of amino acids in sequences. Sequences resulted from this serialization capture certain physico-chemical and 3-D geometric features of active sites. PHMM parameters are then estimated using these sequences. While our sequential representation of active sites captures only partial information about them, experimental results with our technique suggest that it is effective in practice.

1.2.2 Automatic Text Segmentation with CRF

Text segmentation partitions input text strings into meaningful units. In the World Wide Web, data (such as product, bibliographic and address data) exists as unstructured text strings. They have to be segmented into structured records to facilitate efficient query processing and analysis.

Conditional random field (CRF) is a discriminative probabilistic model that is gaining acceptance as an effective computing machinery for text segmentation. Compared to HMM, CRF eliminates the need to make any independence assumptions about the elements in the input sequence.

A CRF model is characterized by a set of weighted vertex feature functions and edge feature functions, whose weights are usually learned from labeled training data. Labeling

can be a labor-intensive process. One can avoid the labeling step by using structured reference tables whose data domains and that of the input text data given for segmentation, coincide. The reference table contains partial information for training the CRF in the sense that edge features corresponding to the transitions between labels are missing.

Inspired by a recent work in [1] on their use for training HMMs, we have developed a novel technique for automatic text segmentation with CRF [70]. Assuming sequences to be segmented come in batches and sequences in a batch conform to the same attribute ordering, we build CRF models for each attribute in the reference table, use them to decide the attribute ordering of a batch of input sequences, derive labeled training data from the reference table according to that ordering, train a global CRF model, and use the global CRF model to segment the batch of input sequences.

1.3 Research Contributions

In this dissertation research, we have made the following contributions:

- We have developed Toxin Knowledge Base (TKB), the first comprehensive bioinformatics resource to identify homologues of toxins, structural motifs of toxins and virulent factors in other proteins.
  - We have developed processes for acquiring and structuring toxin data from a number of sources including the web.
  - We have developed prediction algorithms for detecting potential virulent proteins based on the aggregated toxin data.
- We have improved PHMM-based approaches by leveraging partially aligned sequences to achieve higher classification accuracy when completely aligned sequences are not available.

  - We have developed effective algorithms to decompose a PHMM into sub-PHMMs according to partial alignment information and to compose the parameters of sub-PHMMs into the parameter of the PHMM.
  - An important aspect of our technique is that it gives rise to a family of PHMMs which are parameterized with regard to the alignment information.

- We have developed an original approach for profiling protein active sites with PHMM.
  - We have developed a novel serialization of three dimensional active sites in proteins.
  - Based on this serialization we have expanded traditional PHMMs designed for profiling one dimensional sequences of residues to accommodate both physico-chemical and three dimensional geometric features.
  - An important aspect of our method is that it is able to detect remote members of active site families by exploiting the commonality amongst the members captured by profiling. In contrast non-profiling based methods (such as those that rely on pair-wise comparisons as in SPASM [32]) are unable to do so.

- We have pioneered unsupervised text segmentation with CRF by exploiting structured reference data.
  - Our technique is fully unsupervised in that, when a reference table whose data domain coincides with that of input sequences is available, it eliminates the need for manually labeled data, which are required by traditional CRF approaches.
  - We have developed simple yet effective heuristics to construct labeled examples from the reference table for training classifiers of reference table attributes.

- We have developed practical algorithms to infer the attribute order shared by a batch of input sequences using the attribute classifiers and to derive labeled sequences from the reference table for training text segmentation CRF.

1.4 Dissertation Organization

The rest of this dissertation is organized as follows. Chapter 2 presents an introduction to protein structure, profile hidden Markov models and conditional random fields to set the context for understanding this dissertation. Chapter 3 introduces the Toxin Knowledge Base system and motivates the work in profiling proteins families from partially aligned sequences as well as profiling protein active sites. Chapter 4 provides the technique details and experimental results of profiling protein families from partially aligned sequences. Chapter 5 describes our serialized representation of protein active sites, the adaptation of PHMM for profiling active sites, and our experimental results. Chapter 6 discusses our solution to automatic text segmentation by exploiting reference tables as well as the experimental results. Chapter 7 concludes this dissertation by summarizing our work on exploiting partial information for classification in bioinformatics and text segmentation applications.

Chapter 2

Preliminaries

In this chapter, we give an overview of protein structure, Profile Hidden Markov Models (PHMMs) and Conditional Random Fields (CRFs) to set the context for understanding the rest of the dissertation. Specifically, knowledge about protein sequences in Section 2.1 and PHMM in Section 2.2 is needed for understanding Chapter 3 and 4. Knowledge about protein active sites in Section 2.1 and PHMM in Section 2.2 is for Chapter 5. An introduction to CRFs in Section 2.3 is needed for Chapter 6.

2.1 Protein Structure

The building blocks of proteins are twenty amino acids. Examples of these include Alanine, Valine, Histidine, Glycine, etc. They are usually referred to by their symbolic (3-letter and 1-letter) abbreviations e.g., the 3-letter ALA or the 1 letter A for Alanine, VAL or V for Valine and so on.

All of the twenty amino acids have in common a central carbon atom (C_α) to which are attached a hydrogen atom, an amino group (NH_2), and a carboxyl group ($COOH$). The rest of an amino acid, which is called the *side chain*, is different for different amino acids. These terms are illustrated in Figure 2.1(a). Amino acids are joined end-to-end to

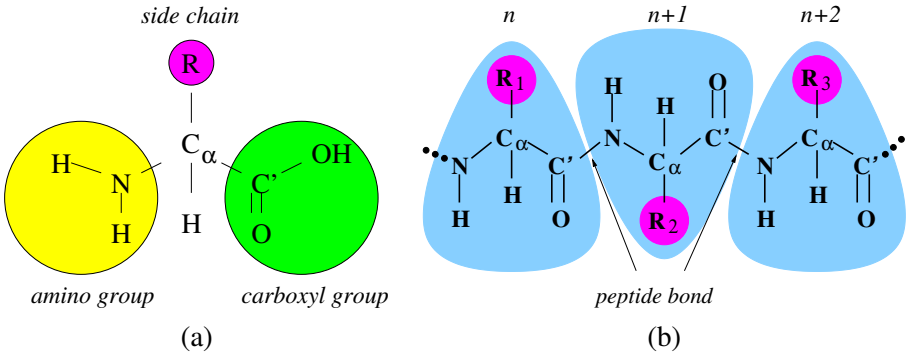


Figure 2.1: (a) Schematic Diagram of an Amino Acid. (b) Polypeptide Chain.

```
KETAAAKFERQHMDSSTSAASSSNYCNQMMKSRNLTKDRCKPVNTFVHES
LADVQAVCSQKNVACKNGQTNCYQSYSTMSITDCRETGSSKYPNCAYKTT
QANKHIIVACEGNPYVPVHFDASV
```

Figure 2.2: Sequence of Bovine Ribonuclease

form a polypeptide chain when the carboxyl group of one amino acid condenses with the amino group of the next to eliminate water and a peptide bond is formed, as shown in Figure 2.1(b). The *N-terminus* of a polypeptide is the end with its amino group NOT involved in a peptide bond. The *C-terminus* is the end with its carboxyl group NOT involved in a peptide bond. The list of all constituting amino acids in the chain of a protein in order starting at the N-terminus and proceeding to the C-terminus is called the *sequence* of the protein. Note that a protein may consist of multiple polypeptide chains. When denoting amino acids by their 1-letter codes, a protein sequence is simply a string of letters taken from the alphabet {A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y}. For example, the sequence of the protein BOVINE RIBONUCLEASE which we mention in Section 5.1 is denoted by the string in Figure 2.2.

The repeating units in a polypeptide chain are called *residues*. In Figure 2.1(b) the elements within each "shaded triangular" area correspond to a residue. A residue is usually referred to by its name or abbreviation followed by its position in the chain. For example, H233 refers to the 233rd residue in a chain, which is a histidine.

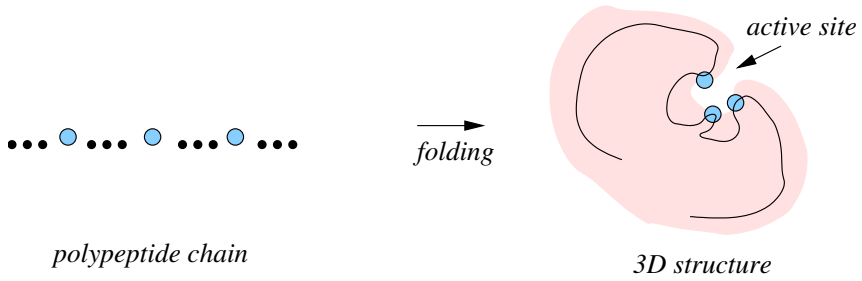


Figure 2.3: Formation of an Active Site

The polypeptide chain of a protein folds in space to form the three-dimensional structure of the protein. The folding of the polypeptide chain typically creates a crevice or cavity on the protein surface. This crevice, called an *active site*, contains a set of residue side chains which might be far apart in the polypeptide chain. They are brought together in the 3-D structure and are disposed in such a way that they can make noncovalent bonds only with certain partners, which can be a protein, DNA, metal ion, etc. The 3-D structure of a protein, especially the localized structure of its active site, determines the functional properties of the protein. Figure 2.3 sketches the formation of an active site. Note that a protein can have several active sites. By examining the interaction of a protein and its binding partner, the protein's active site can be identified. Alternatively, active sites can be inferred by computational tools such as MOE Active Site Finder [82] and Q-SiteFinder [36].

Figure 2.4 shows the active site of butolinum neurotoxin serotype E. It contains three residues: H211, E212, and H215, represented by the sticks in the figure. It determines that this protein has the function of binding a zinc ion which is represented by the ball in the figure.

2.2 Profile Hidden Markov Model (PHMM)

A PHMM is a statistical learning-based technique for modeling DNA and protein sequences families. The underlying principles of PHMMs are based upon the mathematics

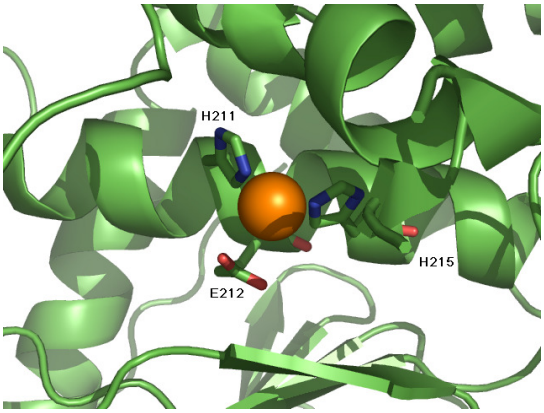


Figure 2.4: Active Site of Botulinum Neurotoxin Serotype E(PDB ID: 1T3A)

of Hidden Markov Models [53] which have found wide applicability in bioinformatics and text segmentation. An HMM is a probabilistic finite state automaton defined by a set of states, a set of state transitions with probabilities assigned to them and a set of observation symbols that are emitted in a state with certain probabilities. The sequence of states corresponding to a visible observation sequence is "hidden" and hence has to be estimated. PHMMs extend the traditional notion of HMMs to model biological sequence families. In this section, we briefly review PHMMs and their application in modeling biological sequence families. A more detailed discussion can be found in [18]. We remark that PHMMs for modeling DNA and protein sequences mainly differ in the domain of emission symbols used. So without loss of generality our review will describe PHMMs for protein sequences only.

Protein sequences typically come in families. Members of a family have a common ancestor and normally maintain the same or related function. Although they have diverged during evolution through insertions and deletions, their functional residues are usually conserved. A multiple alignment of family members reveals the relationship among them. For example, in Figure 2.5 which is a segment of the multiple alignment of seven globin protein sequences taken from [18], it is obvious that residues in some columns are more conserved than in others. A simple rule to decide whether a column is conserved is that if more than

```
HBA_HUMAN        ...VGA--HAGEY...
HBB_HUMAN        ...V----NVDEV...
MYG_PHYCA        ...VEA--DVAGH...
GLB3_CHITP       ...VKG------D...
GLB5_PETMA       ...VYS--TYETS...
LGB2_LUPLU       ...FNA--NIPKH...
GLB1_GLYDI       ...IAGADNGAGV...
                      ***  *****
```

Figure 2.5: A Segment from the Multiple Alignment of 7 Globin Protein Sequences

half of the sequences have a residue instead of a dash present in the column, then that column is conserved. In Figure 2.5, the columns marked with stars are conserved. The two non-starred residues in GLB1_GLYDI correspond to insertions. If a sequence has a dash in a conserved column, then it has undergone a deletion.

PHMMs are HMMs whose structures are specialized to capture such conserved residues as well as insertions and deletions in sequence families. Figure 2.6 shows the structure of a PHMM. The structure has a *Begin* state and an *End* state, denoted B and E respectively in the figure, and a sequence of columns of states between B and E. Each column, from 1 to n, has three states - a *match*, *insert*, and *delete* state. These are denoted by M_i, I_i, and D_i respectively for the i^{th} column. Intuitively, match states correspond to conserved residues among sequences while insert and delete states correspond to divergence in sequences from a common ancestor due to insertions and deletions respectively. The insert state I_0 corresponds to insertions before the first matching residue in sequences. Observe from Figure 2.6 that the structure of the model is parameterized only by the length of the model, i.e., the number of columns of states.

The transitions in the model structure are fixed and corresponds to the underlying semantics of matches, insertions, and deletions. In particular, a match state M_i can make a transition to M_{i+1}, I_i and D_{i+1} respectively. An insert state I_i has transitions to M_{i+1}, D_{i+1}, and to I_i itself. A delete state D_i has transitions to M_{i+1}, D_{i+1}, and I_i. These state transitions are also shown in Figure 2.6.

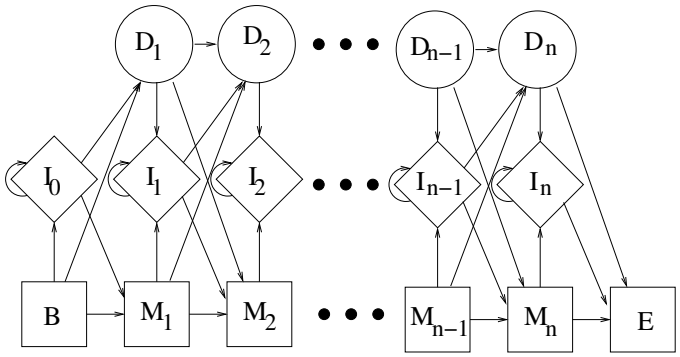


Figure 2.6: PHMM Structure

For protein sequences, the emission symbols are the twenty amino acids. Match and insert states emit residues while delete states are non-emitting silent states. The non-emission of residues from delete states conforms to the semantics of these states – a residue in the representation of the family is not observed in an individual sequence. The begin and end states define the start and end markers of the model and consequently they do not emit residues.

The parameters of a PHMM are usually learned from a set of sequences known as members of a family. When the alignment of the sequences is given, computing the model probabilities reduces to smoothed maximum-likelihood parameter estimation using the frequency counts of transition and emission events. Figure 2.7 (taken from [18]) is a PHMM of length 8. Emission probabilities are shown as bars opposite the different amino acids for each match state, and the values of transition probabilities are indicated by the thickness of the lines. The self looping transitions on the insert states are probability values given as percentages. The emission probabilities are uniformly distributed among 20 amino acids for all the insert states except I_3 where the emission probabilities are 0.09 for amino acid A and D and 0.045 for all the others.

When the alignment of the training sequences is unknown, learning PHMM parameters is done with Baum-Welch's [9] iterative algorithm which is a special case of the more general Expectation-Maximization (EM) algorithm [17]. Starting from initial parameter

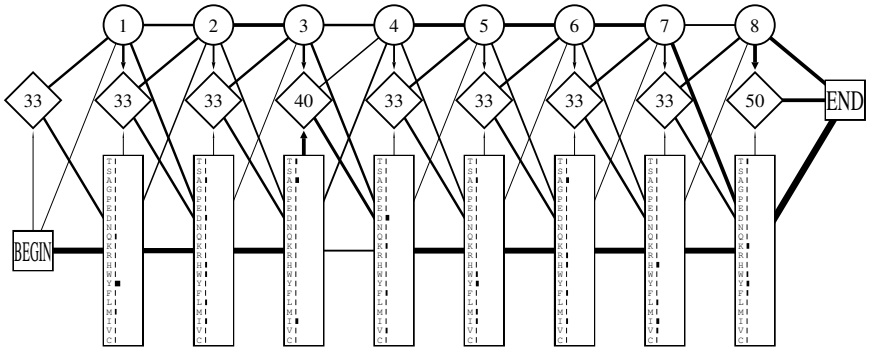


Figure 2.7: An Example PHMM

values, the algorithm terminates after a fixed number of iterations or after a local maximum has been reached. In each iteration, the current parameter values are used to first compute transition and emission expectations (E step) which are then subsequently used to generate the best possible parameter values for the next iteration (M step).

Given a PHMM model M profiling a family S and an input protein sequence x the model determines if x is a member of S, i.e., is it similar to the members of S profiled by M? At a high level this is done as follows: First, the best path (i.e., state sequence) is computed using the well known Viterbi algorithm [53]. In particular, the Viterbi algorithm efficiently computes a state sequence y' that maximize the conditional joint probability $P(x, y|M)$, i.e., $y' = arg\max_y P(x, y|M)$. For example, the best path for the sequence "VGAHAGEY" and the model in Figure 2.7 is found to be $Start \rightarrow M_1 \rightarrow M_2 \rightarrow M_3 \rightarrow M_4 \rightarrow M_5 \rightarrow M_6 \rightarrow M_7 \rightarrow M_8 \rightarrow End$. Next, we compute $p(x, y'|M')$ where M' is a random model that is identical to M in length and transition probabilities. However, the emission probability for each emission symbol a is independent of the states, i.e., for all match and insert states a always occurs with the same frequency q_a. A choice for q_a is the frequency of the amino acid a occurring in a standard sequence database such as SWISS-PROT [7]. Finally we compute the base 2 log-odds ratio $log(\frac{P(x,y|M)}{P(x,y|M')})$ called the *bit score*. If this score falls above a threshold then x is said to be a member of S. The threshold is a global value and details on how it is determined appears in [79].

2.3 Conditional Random Field (CRF)

Conditional Random Field (CRF) [34] is a probabilistic framework that can be used to segment and label sequence data.

The input to the segmentation task consists of a sequence of tokens where what constitutes a token is application-specific. For example, in bioinformatics application a token is usually a letter whereas in the address segmentation problem tokens are words that are delimited by white-spaces. We will therefore assume that there is an application-specific tok function that maps any input to a *token sequence*. For example, tok("1 2 3 convenience store 144 Hempstead Tpke W Hempstead NY") is the sequence of 11 tokens: [1, 2, 3, convenience, store, 144, Hempstead, Tpke, W, Hempstead, NY]. The *length* of a token sequence is the number of tokens in the sequence.

For a token sequence $[t_1, \ldots, t_n]$, a *token sub-sequence* $sub(i, j)$ is defined to be $[t_i, t_{i+1}, \ldots, t_j]$ for any $1 \leq i \leq j \leq n$. Note the tokens in a token sub-sequence are contiguous. Therefore, for a token sequence of length n, the number of token sub-sequences is $n + (n-1) + \cdots + 1 = n(n+1)/2$.

Segmenting a token sequence $[t_1, \ldots, t_n]$ with CRFs amounts to assigning a label sequence $[l_1, \ldots, l_n]$ where each l_i $(1 \leq i \leq n)$ is assigned to token t_i from a predefined label set. These labels correspond to attribute names. Therefore token sub-sequences might correspond to instances of attributes. For example, the token sub-sequence [144, Hepmstead, Tpke] corresponds to an instance of the attribute STREET.

In this dissertation, we use bold fonts to denote vectors, such as $\mathbf{x}$ for an input token sequence, and $\mathbf{y}$ for a label sequence. Normal fonts denote scalars, such as x and y for a single token and label respectively.

CRF is a discriminative model in the sense that it directly computes the conditional probability distribution of label sequences $\mathbf{y}$ given a particular input token sequence $\mathbf{x}$. In contrast generative models such as HMMs compute a joint distribution over both label and

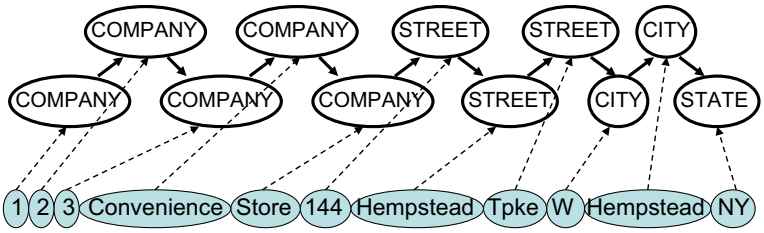


Figure 2.8: Vertex and Edge Features

token sequences. For example, for a given token sequence [1, 2, 3, convenience, store, 144, Hempstead, Tpke, W, Hempstead, NY] shown in Figure 2.8, the conditional probability of the label sequence [COMPANY, COMPANY, COMPANY, COMPANY, COMPANY, STREET, STREET, STREET, CITY, CITY, STATE] will be computed by a CRF model.

The conditional probability distribution of label sequences given an input token sequence is defined by a set of features capturing transitions between labels as well as relations between a label and the corresponding token it is assigned to, the other tokens in the neighborhood of this corresponding token, or even the entire token sequence.

Features capturing transitions between labels are called *edge features*. Examples of such features are shown in Figure 2.8 using solid arrows, namely, the four transitions: from COMPANY to COMPANY, one from COMPANY to STREET, two from STREET to STREET, one from STREET to CITY, one from CITY to CITY, and one from CITY to STATE.

Features capturing relations between the i-th label in a label sequence and the token sequence are called *vertex features*. Examples of such features are shown in Figure 2.8 using dashed arrows. The vertex feature between the token NY and the label STATE denotes a relationship. It could simply mean that a token NY is assigned the label STATE. It may also mean that the label STATE is assigned to a token that consists of two upper case letter.

Note that vertex features are not mere mappings between a label and the token it is assigned to, as is the case with emission symbols and states in HMMs. For instance, we can have a dashed arrow between the token 144 and the second STREET label in the label

sequence in Figure 2.8 denoting a feature capturing the relation that the previous token is a number and the current label is STREET. Such relations are difficult to capture in HMMs.

We follow the notation in [39] in the rest of this section. CRF features are boolean functions. Both edge and vertex features can be written as $f(y_{i-1}, y_i, \mathbf{x}, i) \mapsto \{0, 1\}$ where y_{i-1} is the $(i-1)$-th label, y_i is the i-th label, and $\mathbf{x}$ is the token sequence. The value of a vertex feature function f does not depend on y_{i-1}; however that of an edge feature does.

Let $[[c]]$ be the indicator function whose value is 1 when the condition c is satisfied and 0 otherwise. For the example shown in Figure 2.8, the edge feature for the transition between STREET and CITY and the vertex feature for the relation between the word "NY" and the label STATE can be written as follows:

$$f(y_{i-1}, y_i, \mathbf{x}, i) = [[y_i = CITY \; and \; y_{i-1} = STREET]]. \tag{2.1}$$

$$f(y_{i-1}, y_i, \mathbf{x}, i) = [[x_i \; is \; NY \; and \; y_i = STATE]]. \tag{2.2}$$

Let us denote the vector of all feature functions by $\mathbf{f}$. A vector λ of real numbers with the same length as $\mathbf{f}$ defines the weights of the feature functions.

The feature function vector $\mathbf{f}$ and the corresponding weight vector λ are used to define the conditional probability distribution over label sequences $\mathbf{y}$ given an input token sequence $\mathbf{x}$ as follows:

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} e^{\sum_{i=1}^{|\mathbf{x}|} \sum_{j=1}^{|\mathbf{f}|} \lambda_j f_j(y_{i-1}, y_i, \mathbf{x}, i)}. \tag{2.3}$$

where $|\mathbf{x}|$ is the length of $\mathbf{x}$, $|\mathbf{f}|$ is the number of feature functions, and $Z(\mathbf{x})$ is a normalizing factor equal to:

$$\sum_{\mathbf{y}'} e^{\sum_{i=1}^{|\mathbf{x}|} \sum_{j=1}^{|\mathbf{f}|} \lambda_j f_j(y'_{i-1}, y'_i, \mathbf{x}, i)}.$$

Therefore,

$$\sum_{\mathbf{y}'} P(\mathbf{y}'|\mathbf{x}) = 1$$

```
{
    <[Stony, Brook, NY],    [CITY, CITY, STATE]>,
    <[Port, Jefferson, NY], [CITY, CITY, STATE]>
}
```

Figure 2.9: A Small Set of Training Examples

Feature Function	Weight
$[[x_i$ is Stony and $y_i = CITY]]$	0.44
$[[x_i$ is Brook and $y_i = CITY]]$	0.73
$[[x_i$ is Port and $y_i = CITY]]$	0.44
$[[x_i$ is Jefferson and $y_i = CITY]]$	0.73
$[[x_i$ is NY and $y_i = STATE]]$	2.16
$[[y_{i-1}$=CITY and $y_i = CITY]]$	0.29
$[[y_{i-1}$=CITY and $y_i = STATE]]$	2.06
$[[start = CITY]]$	0.87
$[[end = STATE]]$	2.16

Table 2.1: Example CRF with Two Labels

for all label sequence $\mathbf{y}'$ of token sequence $\mathbf{x}$ which means the conditional probabilities of all possible label sequences for a given token sequence sum up to 1.

For a given token sequence $\mathbf{x}$, CRFs compute the label sequence $\mathbf{y}$ with the highest conditional probability $P(\mathbf{y}|\mathbf{x})$ as the best label sequence, i.e., $arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x})$. An important aspect of CRFs is to learn a CRF model from training data. Usually feature functions are assumed to be given and therefore learning corresponds to estimating the weights of feature functions. Details of CRF inference and learning algorithms can be found in the seminal work of [34] as well as in [59, 51].

For an exposition of how CRFs segment token sequences, let us look at a simplified example. CRFs are trained from labeled token sequences, i.e. the set of paired token and label sequences $\{\langle \mathbf{x}_1, \mathbf{y}_1 \rangle, \ldots, \langle \mathbf{x}_m, \mathbf{y}_m \rangle\}$ (example see Figure 2.9).

Suppose we have trained a CRF model from the training data in Figure 2.9. The feature functions and their weights are listed in Table 2.1. For this CRF, the length of the feature function vector is 9, i.e., $|\mathbf{f}| = 9$.

Huntington	NY	F	e^F	$P(\mathbf{y}\|\mathbf{x})$
CITY	CITY	1.16	3.2	0.2%
CITY	STATE	7.25	1408.1	94.7%
STATE	CITY	0	0	0%
STATE	STATE	4.32	75.2	5.1%

Table 2.2: Possible Ways of Labeling [Huntington, NY]

There are four possible label sequences (see the first two columns in Table 2.2 for the token sequence $\mathbf{x}$= [Huntington, NY]. The sum of their weighted feature functions, $F = \sum_{i=1}^{2}\sum_{j=1}^{9}\lambda_j f_j(y_{i-1}, y_i, \mathbf{x}, i)$ and conditional probability $P(\mathbf{y}|\mathbf{x})$ computed from Equation 2.3 are also listed in the third and fifth columns respectively of the table. For example, the conditional probability of the label sequence [CITY, CITY] is 3.2/(3.2+1408.1+0+75.2) = 0.2%. From the table we see that the best way of labeling [Huntington NY] is to assign the label CITY to token Huntington and STATE to NY.

Chapter 3

Toxin Knowledge Base

3.1 Introduction

Proteins form the most diverse biological macro-molecules in nature. Toxins are a class of proteins that are of prime importance because of their virulent nature – even a small quantity can wreak havoc amongst large populations. Advances in recombinant DNA technology have opened up possibilities for the production of bio-engineered pathogens on scales that could make them into formidable weapons of bio-terrorism. Yet another risk lurks in the form of chimeric molecules, in which the virulent domain of a toxin can be hidden in an otherwise non-toxic protein and could thus thwart the existing techniques for detecting toxins.

Hence, the study of toxins becomes crucial in order to build an effective defense against such potent molecules. It is essential to recognize their virulence factors, understand their reaction mechanisms, understand their inhibition mechanisms and also investigate whether a non-toxic protein has a virulent domain hidden in it. Apart from these characteristics, it is also important to understand the structure-function-and-genetic relationships between various toxins and between toxins and proteins in general.

Performing physical wet-lab experiments to understand this class of proteins, given that

a large number of toxins exist in nature, would be time consuming and expensive. Hence, computational methods for building this knowledge have to be explored.

Public databases of biological macro-molecules have become very popular with the biological community. For example the UniProtKB [85] , RCSB-PDB [78], NCBI [76] have become valuable sources of knowledge for biologists in general. Apart from this, there are several tools available, which can be used to analyze this vast biological data, and obtain new information which may further our understanding about these diverse macro-molecules. However easy-to-use systems that analyze this extensive knowledge to deduce more knowledge are almost non existent.

In the past several years, we have been developing the Toxin Knowledge Base (TKB), a bioinformatics resource to identify homologues of toxins, structural motifs of toxins and virulent factors in other proteins. TKB includes processes for acquiring and structuring toxin data from a number of sources including the web and prediction algorithms for detecting potential virulent proteins based on the aggregated toxin data. The availability of TKB helps speed-up research into adequate bio-defenses against potential biological warfare agents.

In this chapter, we will first introduce the architecture of TKB as well as its main components in Section 3.2. Then we will describe the processes we have for discovery of potentially virulent proteins in Section 3.3. After that we will summarize with further research needed in sequence homology search and active site similarity search.

3.2 TKB System Architecture

The Toxin Knowledge Base (TKB) is a highly curated bioinformatics resource that allows for classification, assimilation, synthesis, analysis and dissemination of knowledge about toxins based on their structural and genomic information. It is envisioned as a large repository of molecular information regarding toxins and other virulent factors. TKB's

salient features are listed below:

- It integrates several disparate off-the-shelf tools and public Web sites and incorporates them into a single workflow by plumbing together their inputs and outputs using data extraction and mediation tools.

- It includes a flexible data acquisition system that incorporates algorithms and processes that make it scalable in view of the rapidly growing information about proteins.

- It includes engineered workflows for a number of common and uncommon tasks, such as homology search and search for otherwise benign proteins that have the virulent domain of a known toxin hidden in them and thus have the potential to be morphed into a toxin.

TKB is comprised of three major components: (1) A powerful data-acquisition/ administration system for direct deposition of data related to toxins, (2) a friendly curation system for inserting and changing data of individual toxins as well as logging the changes, and (3) an ad-hoc query and reasoning system to access and to analyze information. Figure 3.1 shows the system architecture of TKB showing the querying and reasoning subsystem, the curation subsystem, and the data acquisition/administration subsystem. It also shows the architecture of the system, from the user's perspective. The toxin knowledge base essentially is a data source which provides three kinds of interfaces to the user: one used to query the knowledge base, one used to deposit/change information of individual toxins, and the other used to update the toxin and homologue information in the knowledge base to reflect changes in data sources.

The Query and Reasoning Interfaces facilitates the following:

- Toxin Search - Selective retrieval of toxin information.

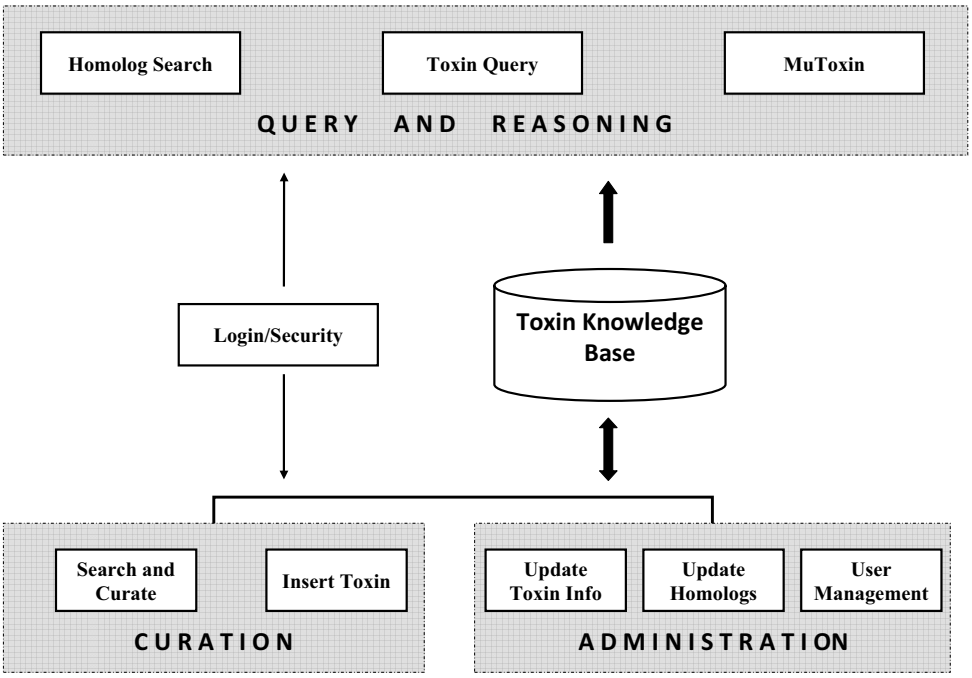


Figure 3.1: TKB System Architecture

- Homology Search - Finding toxins that are homologous to a given protein sequence.
- MuToxin - Determining whether a protein can be transformed into a toxin.

The Curator Interfaces is accessible only to users with administrative rights. It facilitates the following:

- Search and Curate: This involves assisting domain experts to curate toxin information crawled from publicly available databases.
- Insert Toxin: This involves assisting domain experts to input information about newly identified toxins.

The Administrative Interfaces is accessible only to users with administrative rights. It facilitates the following:

- User-initiated: This involves updating the knowledge base with batches of newly identified toxins as well as synchronizing the knowledge base with data sources from which toxin information has been harvested.

- Automated: This involves updating the knowledge base with new homologues and their models for the toxins on a periodic basis, so as to keep the toxin knowledge base up-to-date.

- User Approval: This allows a new user's identity to be verified and approved for use of the TKB.

The system has been developed entirely using Java, Java Server Pages (JSP), HTML and XML/ XSLT technologies. Essentially organized into three layers (based on the Model View Controller design pattern), the front end (view) of the system consists of interacting JSP which are kept extremely functional. All the aspects of user views and definitions are made using XSLT, which allows for a very flexible front end to be developed.

The controller objects are developed as Java Servlets, with ability to handle multiple sessions, control opening and closing of new windows as and when required, pass session control to JSP and retain information for further processing of user commands. The controller objects do not generate any HTML artifacts except for some administrative logs that are stored at the server end for monitoring the status of the system.

The model (back end) is implemented using Oracle 11G as the primary database, with extensive support using XML. The database schema is very flexible in order to accommodate periodic changes that may be necessary because of the ever expanding knowledge within the field of toxicology. We also provide here in Table 3.1 the current status of the database and the various statistics as an estimate of the size of the database tables.

Total Number of Toxins	1459
Number of Toxins with Structures	563
Total number of Homologues	138,982 (95 Homologues / Toxin)

Table 3.1: TKB Statistics

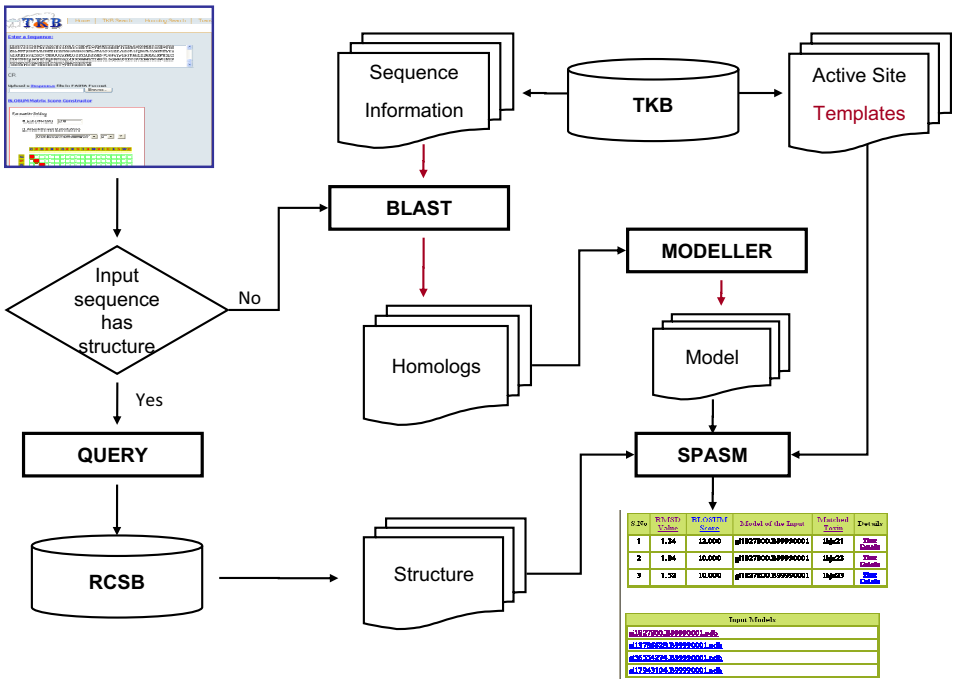


Figure 3.2: TKB Process for Discovery of Potentially Virulent Proteins

3.3 MuToxin Process

An important and interesting aspect of TKB is its ability to detect potential virulent chimeric proteins. Specifically, TKB determines whether a given protein (1) resembles a toxin at its active site and (2) whether residue substitutions at specific locations on the protein, can modify the protein into a toxin. To answer such queries, TKB integrates three separate off-the-shelf bioinformatics and structural biology resources into a specially designed workflow, which is depicted in Figure 3.2.

When the user provides an input protein sequence through the user interface (the top-left

rectangle in the Figure 3.2), the structure of the input is retrieved from RCSB-PDB if it has one and fed to the SPASM program [32]; otherwise the homologues of the input sequence are collected using the BLAST tool [4, 5]. If one or more structures of the homologues exist within the RCSB PDB structure database [78], models are built using the Modeler program [57] and fed to the SPASM program. Based on the toxin active site information available, the SPASM program superposes the structure/model against a database of active site templates and compares them to find a possible match using a customized substitution matrix score. If a match is found, it is evidence that the input protein resembles a toxin in some fashion. Another output from the workflow is a table of substitution scores and positions at which possible residue substitutions need to be made in order to achieve the match. This provides information on whether the protein can be a potential chimera and can hide a potentially toxic active site into a benign protein.

As an example, suppose the sequence of Endoglucanase E (Swissprot: P10477) is given as the input sequence, the RMSD cutoff value is set to 2A, and the allowed substitutions are set to Blosum-45 matrix with cutoff 2. This means that if a pair of residues (X,Y) has a score greater than or equal to 2 in the Blosum-45 matrix then X can by substituted by Y and vice versa. For such inputs, the process in Figure 1 above outputs three matches: the first match is for Chitinase-3 like protein-1 (PDB: 1HJX) active site (GLY 181, LYS 182, THR 184) with RMSD 1.34; the other two matches are both for Chitinase-3 like protein-1 active site (ARG 144, LYS 147, GLN 148) with RMSD 1.84 and 1.52, respectively. Details about the matches are also available. For example, the user can see that if ARG 181 of Endoglucanase E is mutated to LYS and SER 183 to THR, then Endoglucanase E will possibly have an active site similar to the (GLY 181, LYS 182, THR 184) site of Chitinase-3 like protein-1.

We have successfully used TKB to discover potentially mutable proteins. We briefly discuss two such case studies, which is indicative of TKB's capability to discover bio-engineered proteins. The first one was to ascertain the similarity of the reaction mechanism

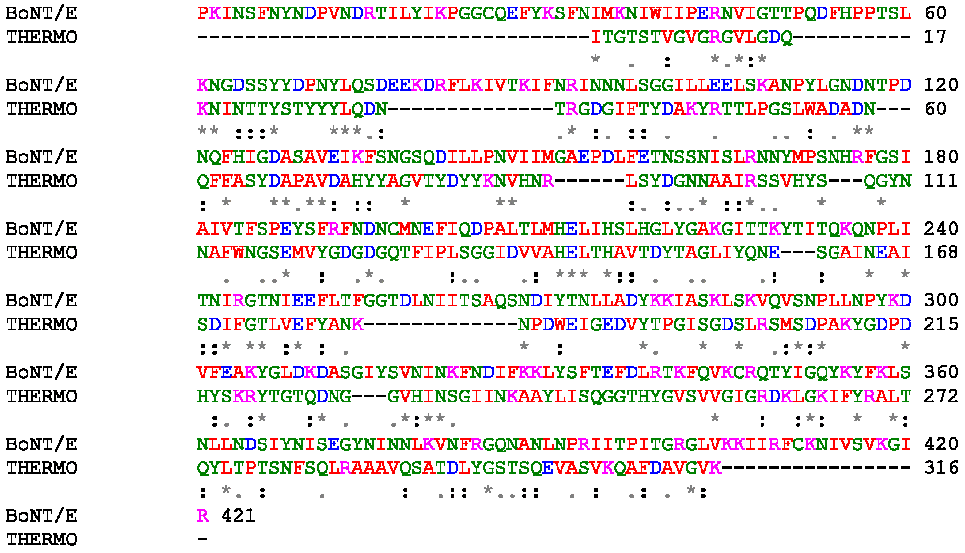


Figure 3.3: Sequence Alignment of Botulinum E and Thermolysin

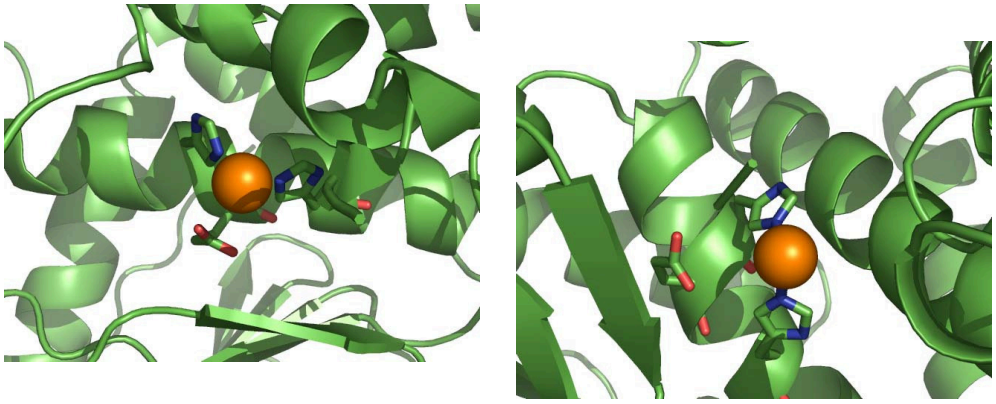
Length	421
Number of identical matches	56
Number of positive matches	88

Table 3.2: Botulinum E and Thermolysin Alignment Statistics

in Thermolysin and Botulinum neurotoxin Type E.

Thermolysin and Botulinum share a same motif HEXXH + E and it is speculated that they have similar reaction mechanism. The sequence alignment between Thermolysin and Botulinum neurotoxin serotyp E and statistics of the alignment is shown in Figure 3.3 and Table 3.2, respectively.

Because the alignment between Thermolysin and Botulinum does not indicate that they are closely related, one may conclude that they do not share any significant structural or functional similarity. Structural alignment of the full structures of Thermolysin and Botulinum also fails to reveal the functional similarity between them. By concentrating on the active sites, TKB is able to find that the active sites of Thermolysin and Botulinum are



(a) Botulinum Active Site (b) Thermolysin Active Site

Figure 3.4: Similarity between active sites of Botulinum and Thermolysin

Length	696
Number of identical matches	74
Number of positive matches	94

Table 3.3: Chitinase-3 like protein-1 and Endoglucanase E Alignment Statistics

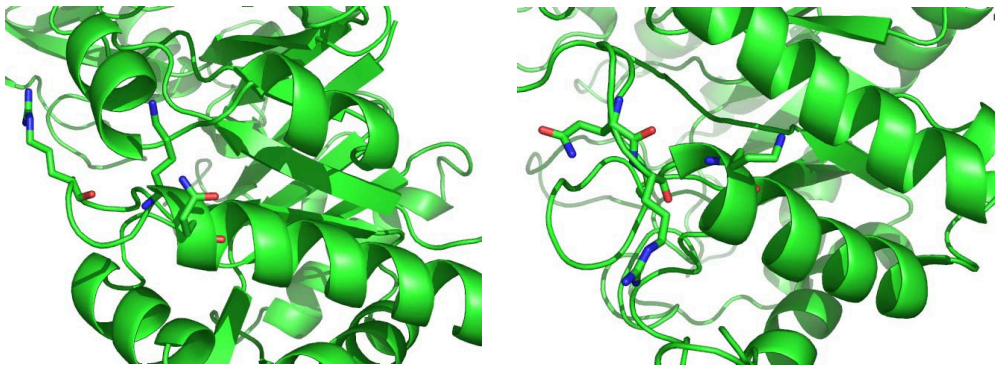
similar, as shown in Figure 3.4, and therefore predicts that Thermolysin can potentially be mutated to function like Botulinum neurotoxin.

The second case study is the interesting discovery of the similarity of the reactive mechanisms of Endoglucanase E and Chitinase-3 like protein-1. (A literature search that we conducted seems to indicate that this is not yet known.) A sequence comparison between Endoglucanase E (Swissprot ID: P10477) and Chitinase-3 like protein-1 (PDB ID: 1HJX) does not give a good indication about their similarity, as is shown in Figure 3.5. The statistics of the alignment is shown in Table 3.3.

TKB discovers that a putative active site in Endoglucanase E is similar to the active site of Chitinase-3 like protein-1, as shown in Figure 3.6 thereby revealing that Endoglucanase E is potentially a candidate to be transformed to Chitinase-3 like protein-1.

```
1HJX_C      --------------YKLVCYYTSWSQYREGDGSCFPDALDRFLCT--------------- 31
GUNE_CLOTM  IDEAWLNRVEEVVNYVLDCGMYAIINLHHDNTWIIPTYANEQRSKEKLVKVWEQIATRFK 180
                        * * *   :  : :..:   :*   :.  ..
1HJX_C      ----HIIYSFANISNDHIDTWEWN------------------------------------ 51
GUNE_CLOTM  DYDDHLLFETMNEPREVGSPMEWMGGTYENRDVINRFNLAVVNTIRASGGNNDKRFILVP 240
              *:::.  * ..:   .. **
1HJX_C      ---DVTLYGMLNTLKNRNPNLKTLLSVGGWN---------------------------- 79
GUNE_CLOTM  TNAATGLDVALNDLVIPNNDSRVIVSIHAYSPYFFAMDVNGTSYWGSDYDKASLTSELDA 300
              . *   ** *   * : :.::*: .:.
1HJX_C      ----------------FGSQRFSKIAS-----NTQSRRTFIKSVPPFLRTHGFDG----- 113
GUNE_CLOTM  IYNRFVKNGRAVIIGEFGTIDKNNLSSRVAHAEHYAREAVSRGIAVFWWDNGYYNPGDAE 360
                          **:   .:::*     :  :*.:. :.:. *   :*: .
1HJX_C      -------LDLAWLYPGRRDKQHFTTLIKEMKAEFIKEAQPGKKQLLLSAALSAGKVTIDS 166
GUNE_CLOTM  TYALLNRKTLSWYYPEIVQALMRGAGVEPLVSPTPTPTLMPTPSPTVTANILYGDVNGDG 420
                  *:* **   :     : :: : :   . :   . .  ::* :  *.*. *.
1HJX_C      SYDIAKISQHLDFISIMTYDFHG---------------------------------------- 189
GUNE_CLOTM  KINSTDCTMLKRYILRGIEEFPSPSGIIAADVNADLKINSTDLVLMKKYLLRSIDKFPAE 480
            . : :. :    :*    :* .
1HJX_C      ----------------------------AWRG--------------------------T 194
GUNE_CLOTM  DSQTPDEDNPGILYNGRFDFSDPNGPKCAWSGSNVELNFYGTEASVTIKSGGENWFQAIV 540
                                       ** *                          .
1HJX_C      TGHHSPLFRGQEDASPDRFSN--------------TDYAVGYMLRLG---------APAS 231
GUNE_CLOTM  DGNPLPPFSVNATTSTVKLVSGLAEGAHHLVLWKRTEASLGEVQFLGFDFGSGKLLAAPK 600
             *:  * *  :  :*. :: .              *: ::* :  **          *...
1HJX_C      KLVMGIPTFGRSFTLASSETGVG-----------------------------APISGPG 261
GUNE_CLOTM  PLERKIEFIGDSITCAYGNEGTSKEQSFTPKNENSYMSYAAITARNLNASANMIAWSGIG 660
             *   *  :* *:* * .: *..                              . ** *
1HJX_C      IPGRFTKEAGTLAYYEICDFLRGATVHRILGQQVPYATKGNQ-----WVGYDDQESVKSK 316
GUNE_CLOTM  LTMNYGGAPGPLIMDRYPYTLPYSGVRWDFSKYVPQVVVINLGTNDFSTSFADKTKFVTA 720
            :. .:   .*.*   .     *  : *:  :.: ** ..  *        ..: *: .. :
1HJX_C      VQYLKDRQ-----------LAGAMVWALDLD---------------------------- 336
GUNE_CLOTM  YKNLISEVRRNYPDAHIFCCVGPMLWGTGLDLCRSYVTEVVNDCNRSGDLKVYFVEFPQQ 780
             : * ..           .*.*:*. .**
1HJX_C      --------DFQGSFCGQDLRFP-LTNAIKDALA--- 360
GUNE_CLOTM  DGSTGYGEDWHPSIATHQLMAERLTAEIKNKLGWAT 816
                *:: *:. ::*    **  **: *.
```

Figure 3.5: Sequence Alignment of Chitinase-3 like protein-1 and Endoglucanase E



(a) Chitinase Active Site (b) Putative Endoglucanase E Active Site

Figure 3.6: Similarity between active sites of Chitinase and Endoglucanase E

3.4 Research Issues

TKB has provided an engineering solution to a widely acknowledged problem of analyzing information from various resources by combining several off-the-shelf software tools and developing an integrated work-flow that offers biologists with the ability to analyze the nature of toxins. It also provides information to users if a non-toxin protein can be potentially transformed into a toxin using simple substitution of amino-acid residues at their active sites. It is also the single largest resource on information regarding toxins, where biologists can easily synthesize and disseminate knowledge about toxins.

Apart from our engineering processes, our research effort has focused on the development of methods to classify toxins into families based on profiles (using profile hidden Markov models [20, 30]). These models take into account information about variations even across distant homologues and can thus identify remotely related proteins and toxins. We have also investigated methods to build profiles of structures to compare active site information of proteins. The development of these methods of classification allows for a faster decision system that can correctly predict whether an input protein is potentially virulent.

Chapter 4

Protein Profiling with Partially Aligned Sequences

4.1 Introduction

The success of genomic work on various species has resulted in an enormous multitude of biological sequence information. This has created a rich research area centered around the development of automated techniques for analysis of these sequences. An effective means of understanding the characteristics of a new biological polymer from its sequence is through homology – whereby the sequence is compared to other similar sequences with known rich biological information. Another important application of homology search is toxicology tools such as TKB [31, 84], for detecting potential virulent factors.

A number of techniques for homology search have appeared in the bioinformatics research literature. Included among them are techniques based on pairwise comparison of sequences using either dynamic programming as in Needleman-Wunsch [46] and Smith-Waterman [62] or heuristic-based database search as in BLAST [4] and FASTA [50].

Detecting remote homologues often requires the use of additional information that is

usually missing in techniques that do pairwise sequence comparison. One such source of additional information is the knowledge that a set of sequences belong to a *family* i.e. they are homologues of each other. The simplest way of using family information, as done in FPS [23], is to perform multiple pairwise comparisons of the new sequence against the family members and collect aggregate statistics over all the comparisons. More sophisticated methods involve constructing a statistical profile of the family from its known member sequences. Profiles can be constructed only from the most conserved regions of sequences belonging to the family, as in MEME [6] and PROTOMAT [26], or from sequences as in PSI-BLAST [5] and profile hidden Markov models (PHMMs) [20, 30].

PHMMs are recognized as a powerful technique for probabilistic modeling of sequences of biological families. The two dominant approaches for training PHMMs differ mainly in the way training sequences are utilized. At one extreme is training from *completely aligned* sequences where all the residues in every sequence are mapped to a column representation taking into account insertions and deletions. In contrast, (inexpensive) training from *completely unaligned* sequences uses no such information. Not surprisingly, PHMMs trained from completely aligned sequences (which we will refer to as A_PHMMs) have been shown to identify remote homologues with a much higher degree of accuracy than those trained from unaligned sequences (which we will refer to as U_PHMMs). However, producing the information about alignments is a labor intensive process involving expensive structural analysis of entire sequences. The contrasting trade-offs at the two ends of the alignment spectrum gives rise to the question: Can we develop techniques for learning profile PHMMs that trade the accuracy of remote homology identification for alignment information? Using the notion of *partially aligned* sequences where only parts of sequences are aligned against each other, we formulate this problem as one of estimating PHMM parameters from such sequences. We will refer to PHMMs trained with such partially labeled sequences as P_PHMMs.

While the effort needed to derive complete alignment in principle is based on structural

analysis of entire sequences, partial alignment knowledge can be obtained by doing such an analysis over limited fragments of the sequences. Moreover, there exists databases like PROSITE [21] which contain *signatures* of families. These signatures represent residues which are conserved over sequences in the family. Consequently, they provide an inexpensive means of obtaining partial alignment information for their respective families.

The essence of our approach for training PHMMs from partially aligned sequences (referred to as P_PHMM from now on) rests on the observation that a consecutive string of unaligned residues between two aligned residues can be generated only from the sequence of states lying between the match states for the aligned residues in the P_PHMM structure. Based on this observation, the algorithm decomposes P_PHMM into sub-models whose parameters are separately estimated and then composed together to produce the original P_PHMM parameters. The technique is *parameterized* w.r.t. the alignment information in the sense that by varying the alignment information we can estimate the parameters of PHMMs spanning the entire spectrum from aligned PHMM at one end to unaligned PHMM (U_PHMM) at the other end.

4.2 Techniques

Building P_PHMMs rests on the use of *partial alignment* information to *decompose* a PHMM structure into sub-models and *compose* parameters computed from these sub-models into the PHMM's parameters.

4.2.1 Partially Aligned Sequences

In a set of partially aligned sequences, alignment information is known only for a subsequence of residues in every individual sequence in the set. This corresponds to a situation between complete alignment and zero alignment. C_1, C_2, and C_3 in Figure 4.1 show three aligned columns in the ten sequences of the immunoglobin (ig) family. The alignment C_1

Figure 4.1: Partial alignment information for ten ig sequences

spans the residues $A, V, L, I, L, A, K, V, M$, and A in the ten sequences respectively and is illustrated by the leftmost solid line. Similarly, the alignment C_3 spans the Y residues in each of the ten sequences as indicated by the rightmost solid line. As illustrated in C_2, where the residues S, D, F, T, and D in only the last five sequences are aligned, it is not necessary that an alignment information has to cover all the sequences in the set. Observe that in the first sequence, $1LTK$, alignment information is known only for the subsequence of two residues A (in C_1) and Y (in C_3). In the event of alignment being known for all the residues in every sequence, partial alignment collapses to complete alignment while total absence of any alignment information reduces to a set of unaligned sequences.

Given a set of partially aligned training sequences, a PHMM structure is built from them ignoring the partial alignment information. As discussed in Section 2.2, the only parameter which has to be estimated in order to build PHMM structure is the model length. We have used the simple heuristic of taking the average length of the sequences to estimate the model length. For instance, for the ten ig family members in Figure 4.1, the model length computed by averaging over the size of the ten sequences is 74. By the definition of alignment, all residues aligned at a particular column are generated from the same state

in the PHMM. We estimate this state by averaging over the positions of the residues, belonging to the alignment, in their corresponding sequences. For instance, for the alignment C_1 in Figure 4.1, the mean position where a residue in the alignment occurs in a sequence is 12. Consequently, all the ten residues in C_1 are generated from the match state M_{12}. Similarly, the ten residues in C_3 and the five residues in C_2 are generated from the match states M_{65} and M_{21} respectively.

4.2.2 Model Decomposition

The key to using partial alignment information for estimating PHMM parameters is the observation that a substring of unaligned residues between any two aligned residues can only be generated from the sequence of states in model positions between those corresponding to the aligned residues. For instance, in the first sequence 1LTK in Figure 4.1, the residues A (C_1) and Y (C_3) belong to match states M_{12} and M_{65}. The substring of unaligned symbols from R to K between the two aligned residues can be generated only from states in model positions 13 to 64 and the insert state I_{12}. This observation lets us decompose the PHMM structure into sub-models where each sub-model generates substrings from the original sequence. In what follows, we have *ignored gaps* in alignment information for simplicity of exposition of our technique.

In our decomposition framework, aligned residues are generated from singleton match states while substrings of unaligned residues are generated from PHMMs consisting of states in sequences of consecutive positions in the original model. We construct these PHMMs, or sub-models, from the appropriate states in the original model and add begin and end states to complete the sub-model structure. The PHMM P_1 in Figure 4.2 illustrates an example sub-model. During decomposition, for a sequence with aligned residues α_n, α_m generated at match states M_i, M_j respectively and with the intermediate unaligned substring $\alpha_{n+1} \cdots \alpha_{m-1}$ generated from the sub-model P, transitions are created from M_i to

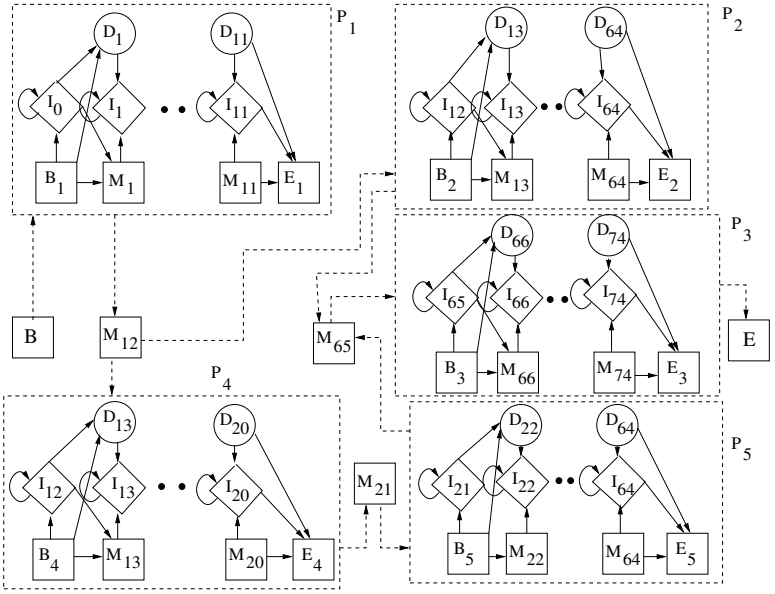


Figure 4.2: Decomposition of a 74 length PHMM structure using the partially aligned sequences in Figure 4.1

P and from P to M_j. In the event of consecutive aligned residues (i.e. $\alpha_m = \alpha_{n+1}$), the sub-model P does not exist and M_i directly transitions to M_j.

We illustrate the decomposition procedure on an initial PHMM structure, as shown in Figure 2.6, with model length 74 using the ten sequences of the ig family and the partial alignment information shown in Figure 4.1. The resultant sub-models created are illustrated in Figure 4.2. In the sequence 1LTK, the first aligned residue A (C_1) occurs at the match state M_{12}. The substring from the first residue I till the residue A before the alignment column C_1 can only be generated from a sub-model having states from initial model positions 0 to 11. The PHMM P_1 in Figure 4.2 denotes such a sub-model. The second aligned residue in 1LTK (C_3) occurs at match state M_{65}. The substring of unaligned residues between the C_1 and C_3 alignment columns can only be generated from the sub-model, P_2, with states from initial model positions 13 to 64. Observe that P_2 also includes the insert state in model position 12. The match state M_{65} is used for the aligned column C_3 and the sub-model P_3 for the rest of the unaligned residues in the sequence. The sub-models are

connected by transitions between B, the begin state in the original model, and P_1, between P_1 and M_{12}, M_{12} and P_2, P_2 and M_{65}, M_{65} and P_3, and finally between P_3 and E, the end state in the original model. The sub-models P_1, P_2, and P_3 and the match states M_{12} and M_{65} are reused for the next four sequences as they possess identical alignment information. The last five sequences contain an additional column (C_2) of residues aligned in the match state M_{21}. In NCA2_HUMAN1, the sequence of unaligned residues between C_1 and C_2 can only be generated from a PHMM with the states from original model positions 13 to 20 including I_{12}. This sub-model is constructed in P_4 and connected with transitions from M_{12} and to M_{21}. A separate sub-model, P_5, is constructed for the unaligned residues between C_2 and C_3. The match state M_{65} and the sub-model P_3 are reused for the rest of the sequence. The last four sequences also follow a similar pattern as NCA2_HUMAN1.

Note that in the presence of gaps in partial alignment, instead of singleton match states, we would have a *single column of match, insert, and delete states* with PHMM-style transitions between them. Additionally, there would be transitions from a begin state to all three and to an end state from each of them.

4.2.3 Parameter Composition

The essence of our composition technique is to estimate the original PHMM parameters from expectations of transition and emission events computed from sub-models and singleton match states.

Recall that a sub-model generates a set of unaligned residue substrings. For instance, the sub-model P_1 in Figure 4.2 generates the first eleven residues of all the ten ig family sequences shown in Figure 4.1. This allows individual sub-model parameters to be estimated by Baum-Welch training.

Central to Baum-Welch parameter estimation is the computation of expectations of transition and emission events. Given a sequence, we denote the transition expectation

between states s_i and s_j for the t^{th} residue by $\xi_t(s_i, s_j)$. In a PHMM, s_i, s_j are the delete, insert, and match states and j is either i or $i+1$. The transition expectation between s_i, s_j for the entire sequence is given by $A_{s_i,s_j} = \sum_{t=1}^{t=N} \xi_t(s_i, s_j)$, where N is the length of the sequence.

The singleton match states corresponding to aligned columns generate a set of residues. For instance, in Figure 4.2, M_{65} emits only the Y residue while M_{12} emits the residues A, V, L, I, K, and M. The emission probabilities of residues in these match states are estimated by smoothed maximum likelihood frequency counting. Also, transition probabilities between sub-models and neighboring match states and vice-versa are estimated using a smoothed maximum likelihood approach. For instance, in Figure 4.2, if n_{M_{12},P_2} and n_{M_{12},P_4} denote the number of sequences where transitions from M_{12} to P_2 and from M_{12} to P_4 occur respectively, the probability of transition from M_{12} to P_2, $p_{M_{12},P}$, is computed as $\frac{n_{M_{12},P_2}+1}{n_{M_{12},P_2}+n_{M_{12},P_4}+2}$.

The partial alignment information in a sequence can be such that:

1. Alignment occurs at the match states M_k and M_l, where $k < i$ and $j < l$, for the residues α_n and α_m respectively.

2. Alignment occurs at the match state M_i for the residue α_n but not at M_{i+1}.

3. Alignment occurs at the match state M_{i+1} for the residue α_m but not at M_i (the converse of the above).

4. Alignment occurs at both M_i and M_{i+1} for residues α_n and α_{n+1} respectively.

Given a sequence, these four scenarios influence the computation of transition expectations for the three kinds of states in a PHMM. Let A_{S_1,S_2} denote the transition expectation between state S_1 and S_2. Apparently scenario 4 only contributes to $A_{M_i,M_{i+1}}$ which in this case is just the count of the number of times a transition is made between the singleton

Scenario	1	2	3
$A_{D_i,D_{i+1}}$	BW	N/A	N/A
A_{D_i,I_i}	BW	N/A	BW
$A_{D_i,M_{i+1}}$	BW	N/A	$A^P_{D_i,E_P} \times p_{P,M_{i+1}}$
A_{I_i,I_i}	BW	N/A	N/A
$A_{I_i,D_{i+1}}$	BW	N/A	N/A
$A_{I_i,M_{i+1}}$	BW	N/A	$A^P_{I_i,E_P} \times p_{P,M_{i+1}}$
A_{M_i,I_i}	BW	$p_{M_i,P} \times A^P_{B_P,I_i}$	BW
$A_{M_i,D_{i+1}}$	BW	$p_{M_i,P} \times A^P_{B_P,D_{i+1}}$	N/A
$A_{M_i,M_{i+1}}$	BW	$p_{M_i,P} \times A^P_{B_P,M_{i+1}}$	$A^P_{M_i,E_P} \times p_{P,M_{i+1}}$

Table 4.1: Transition Expectations

match states M_i and M_{i+1}. For the other three scenarios, Table 4.1 summarizes how the transition expectations are estimated.

In Table 4.1, all entries marked by 'BW' means that the expectation is estimated from Baum-Welch on the appropriate sub-model. For example, the expectation $A_{D_i,D_{i+1}}$ from delete state D_i to D_{i+1} for scenario 1 is given by $\sum_{t=n+1}^{t=m-1} \xi_t(D_i, D_{i+1})$.

Scenario 2 and 3 require considering a neighboring singleton match state. Let us work out scenario 3 for $A_{D_i,M_{i+1}}$. In such a situation, D_i makes a transition to the end state E_P of the sub-model P which generates the unaligned substring preceding the aligned residue in M_{i+1}. Thus $A_{D_i,M_{i+1}}$ is estimated as $A^P_{D_i,E_P} \times p_{P,M_{i+1}}$, where $p_{P,M_{i+1}}$ is the probability of transition between P and M_{i+1}.

Finally, the sum of the expectations for any event over all the sequences are used to estimate its probability using a smoothed maximum likelihood technique. For instance, the probability of transition between M_i, M_{i+1} is given by:

$$p_{M_i,M_{i+1}} = \frac{\sum A_{M_i,M_{i+1}} + 1}{\sum A_{M_i,M_{i+1}} + \sum A_{M_i,I_i} + \sum A_{M_i,D_{i+1}} + 3}$$

where the summation denotes the cumulative value of the expectation over all the sequences.

ID	M	P_PHMM		U_PHMM		SAM		TCOF.		MME.		RE	
		T	F	T	F	T	F	T	F	T	F	T	F
ps00012	221	130	3	115	0	165	6	127	2	139	967	169	143
ps00475	80	70	8	71	6	72	2	66	0	60	851	57	11
ps00622	100	55	3	45	0	85	21	85	0	84	1225	77	4
ps00675	91	86	11	81	19	86	49	73	1	86	1412	70	138
ps01330	96	82	5	80	2	90	0	90	0	24	164	59	0

Figure 4.3: Experimental data with 15% training set on the 5 Prosite families

Emission expectations of residues in states are estimated from sub-models, by Baum-Welch, and from singleton match states by frequency counting. Smoothed maximum likelihood is used to compute the emission probabilities from these expectations.

Note that in the presence of gaps, parameter composition would involve considering the expectations of events in *single columned match, insert, and delete* states as well as probabilities of transitions to and from their begin and end states, respectively, from other sub-models.

4.3 Evaluation

Experiments were conducted to compare the performance of P_PHMM against vanilla PHMM (U_PHMM), SAM which is a state of the art PHMM tool, an advanced multiple alignment tool TCoffee, and metaMEME. Our approach was compared against TCoffee as it can be used to generate a multiple alignment (from which a family model can be learned) from partial alignments. Evaluation of performance was based upon comparison of recall and precision metrics for these different models over the same set of protein families. The effect of varying training set size as well as alignment information on the performance were also investigated.

4.3.1 Experimental Setup

Regular expression based signature information available for families in the PROSITE database[21] were used to generate partially aligned sequences. Matches of a family's signature in sequences which belong to it constitute the partial alignment information for these sequences. To demonstrate the effectiveness of our technique in homology identification, 5 PROSITE families, each having at least 50 members, were chosen where RE-based pattern signatures were not very effective in identifying family members. The first column in Figure 4.3 shows the families used while the second column shows the number of members of each family in the Swiss-Prot database [7]. The models trained with P_PHMM, U_PHMM, SAM, and TCoffee were used with `hmmsearch` of HMMER [20] to detect homologues in Swiss-Prot while for metaMEME its own search tool, mhmms, was used. Default cutoff values were used in both the cases.

4.3.2 Recall and Precision

Figure 4.3 tabulates the results of the experiments for the five models on the five families using 15% of the members of each family as the training set. The columns T and F for each model reflect the number of true and false positives respectively in the test set. Figure 4.4 and Figure 4.5 summarize the results w.r.t. precision and recall. Observe from Figure 4.4 that the precision of P_PHMM is significantly better than metaMEME for all the families. The recall of P_PHMM is better than metaMEME for 3 of the 5 families as shown in Figure 4.3. The precision of P_PHMM is significantly better than SAM for ps00622 and ps00675 while being comparable for the other 3 families. Figure 4.5 illustrates the recall of P_PHMM against U_PHMM and TCoffee. P_PHMM has better recall for 4 of the families compared to U_PHMM and, apart from ps00622, has better or similar recall compared to TCoffee.

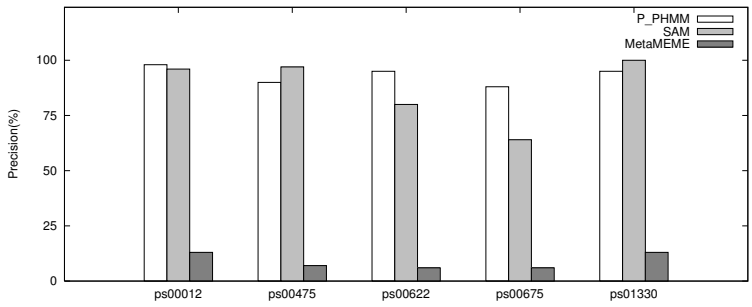


Figure 4.4: P_PHMM precision against SAM and metaMEME

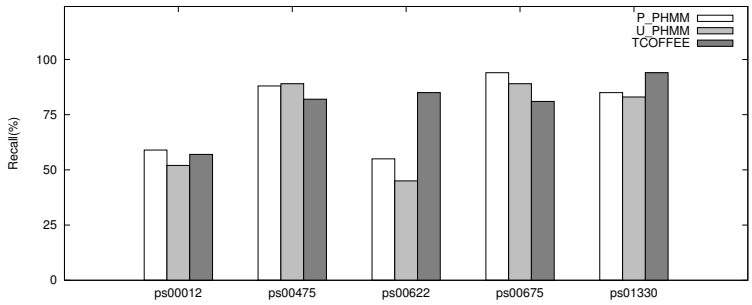


Figure 4.5: P_PHMM recall against U_PHMM and TCoffee

4.3.3 Effect of Varying Training Set

A desirable property of any supervised learning algorithm is the improvement in performance with increased training. Figures 4.6 shows the change in recall with increasing training set size for P_PHMM, U_PHMM, and metaMEME. Observe that for all the four families the recall of P_PHMM increases with training set size. In contrast, vanilla PHMM or U_PHMM does not always show a increase as evident in ps00012 and ps01330. This is even more true for metaMEME which, in spite of having similar recall numbers as P_PHMM in Figure 4.3, does not demonstrate better performance with more training.

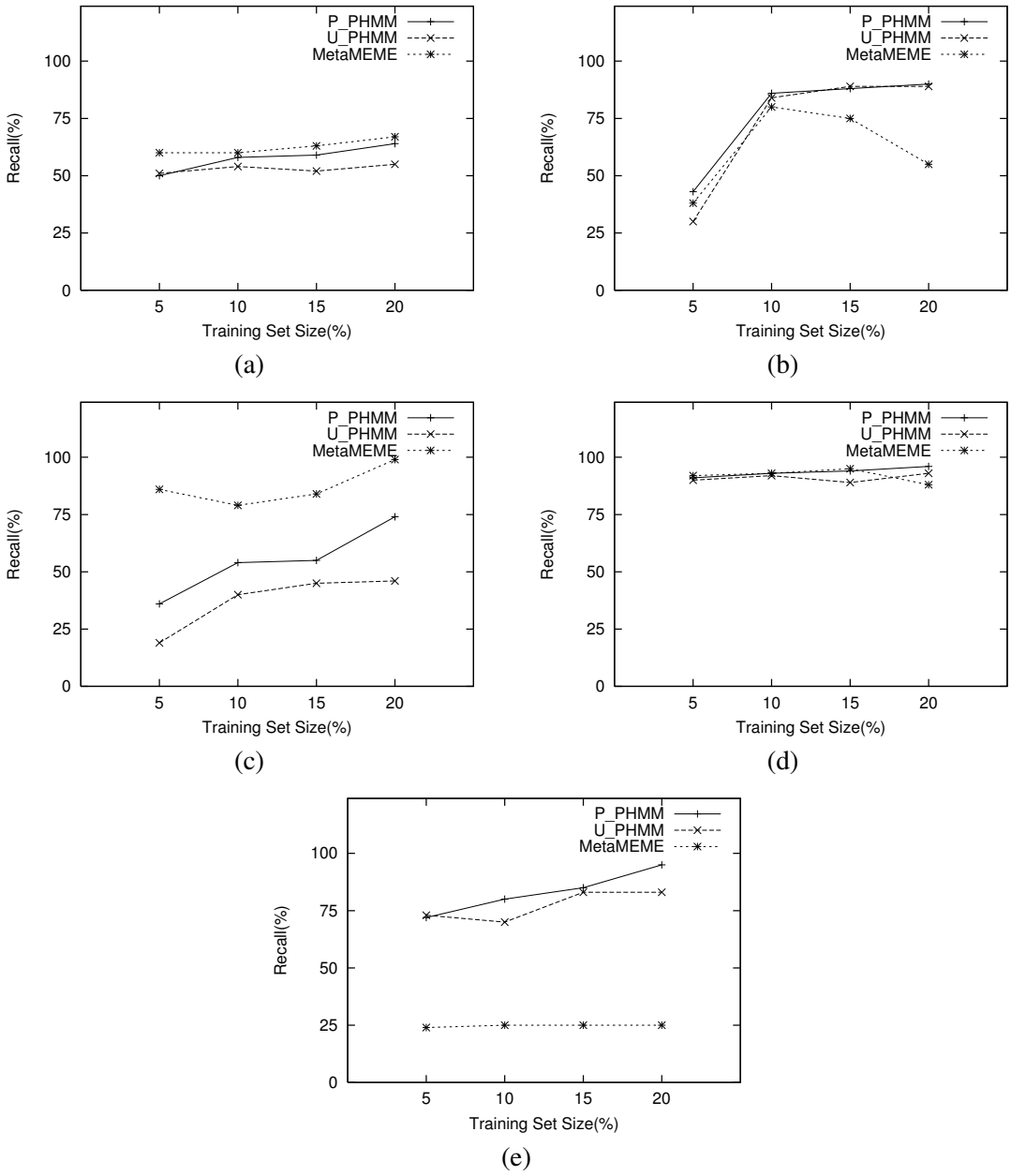


Figure 4.6: Comparing recall of P_PHMM with U_PHMM and metaMEME for (a) ps00012, (b) ps00475, (c) ps00622, (d) ps00675, and (e) ps01330

4.3.4 Effect of Varying Alignment Information

Since P_PHMM training is parameterized w.r.t. the alignment information, experiments were conducted to investigate whether increasing this knowledge results in concomitant

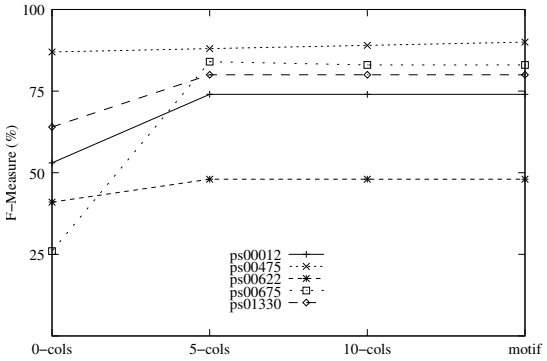


Figure 4.7: Impact on P_PHMM F-measure of varying alignment information for the 5 PROSITE families

improvement in performance.

PROSITE:

For the 5 PROSITE families alignment was varied by selecting different subsequences from the match of the family's signature on the training sequences. Figure 4.7(a) shows the impact on F-measure[1] when alignment is varied from 0 to 5 to 10 columns and finally ending with the entire match (motif). Observe that, apart from ps00475, there is almost no improvement in performance with increasing information from 5 columns to the motif. This is due to the fact the motifs of these families, apart from that of ps00475, are substrings rather than subsequences. Consequently, neighboring aligned columns are close to each other and this makes the improvement in P_PHMM parameters extremely localized.

Ig Family

Experiments on a different family, ig, lent more credence to the above hypothesis. 579 sequences belonging to the ig family and 400 non-ig sequences were used for this experiment. The partial alignment information was obtained from a manual alignment of 10 ig family member subsequences. Figure 5.1(b) shows a significant increase in recall, precision, and F-measure with increasing alignment. This is due to the larger spread between aligned columns (observe distance between $C1$ and $C3$ in Figure 4.1).

[1]The harmonic mean of recall and precision

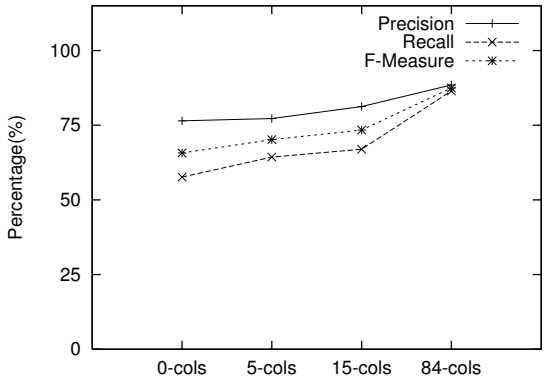


Figure 4.8: Impact on P_PHMM F-measure of varying alignment information for ig family

ID	Members	Aligned Columns
a.1.1.2	60	186
b.1.1.2	59	122
b.34.2.1	26	176
c.47.1.5	31	101
d.169.1.1	28	173

Figure 4.9: The 5 SCOP families

SCOP

The parameterized nature of the P_PHMM algorithm was also borne out by experiments conducted on families from the SCOP [45] database. The SCOP database provides detailed and comprehensive description of the structural and evolutionary relationship between proteins. For the purposes of our experiments, we selected 5 SCOP families each having at least 25 members. Multiple alignment of these families was derived from the PALI [77] database which provides alignments of proteins in the SCOP database. Column 1 in Figure 4.9 lists the ids of these 5 families, while Columns 2 and 3 show the number of family members and the number of aligned columns in their multiple alignment.

P_PHMMs were trained for each of these 5 families. The training set size, for each family, was fixed at a randomly chosen set of 25% of its total members. The amount of alignment information was successively varied from the use of 0% (completely unaligned),

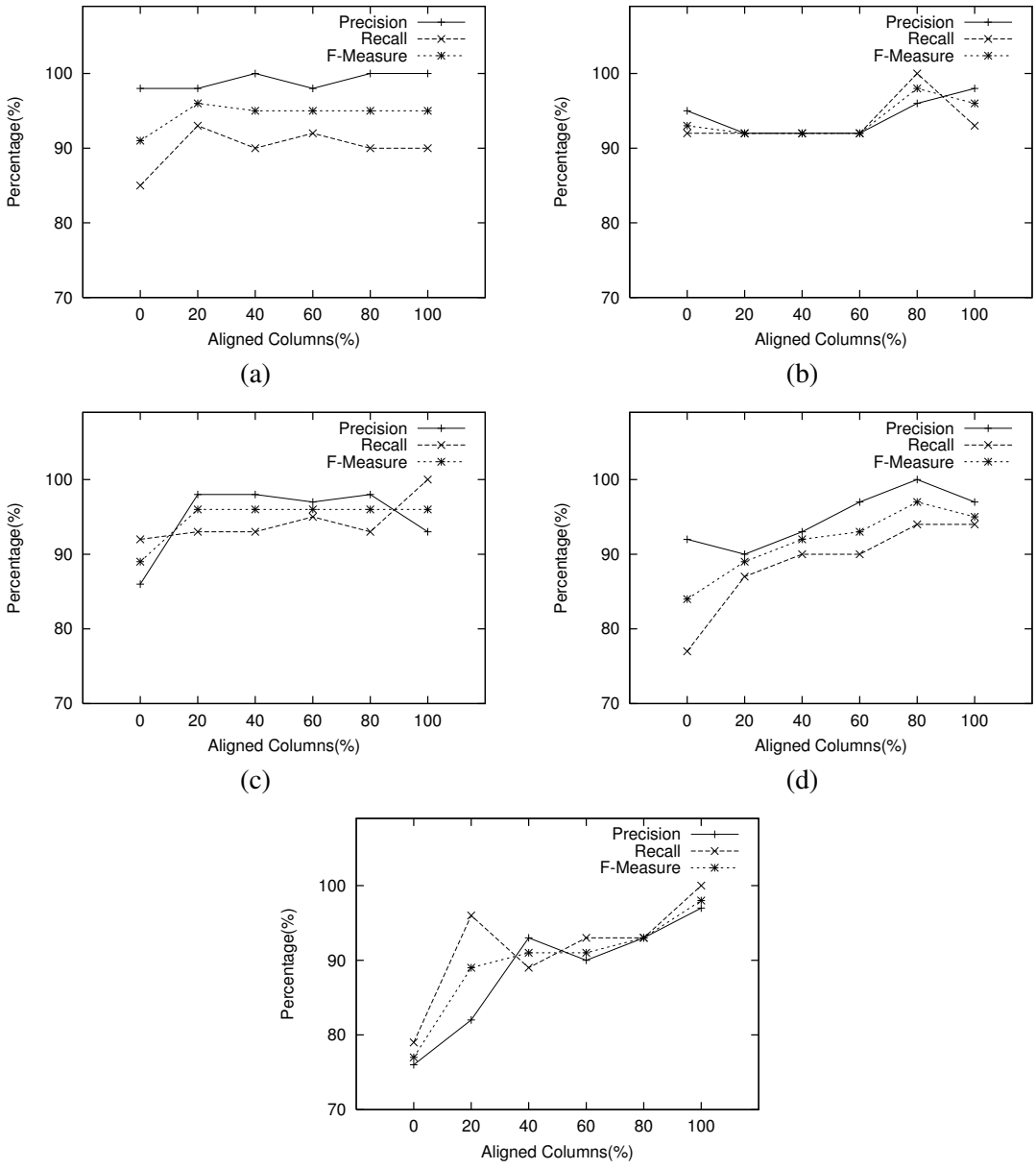


Figure 4.10: Impact on P_PHMM performance of varying alignment information for the SCOP families (a) a1.1.2, (b) b1.1.2, (c) b.34.2.1, (d) c.47.1.5, and (e) d.169.1.1

to 20%, 40%, 60%, and 80% of the number of aligned columns in the multiple alignment of the family. For instance, in the multiple alignment of the training sequences of family

$a.1.1.2$, using a 20% alignment information corresponds to using the first 37 (recall from Figure 4.9 that $a.1.1.2$ contains 186 aligned columns) columns as aligned with the remaining columns being treated as unaligned. The test set for each of these families consisted of all the 5179 domains from all the 1029 families in PALI release 2.3. Recall, precision, and F-measure of homology detection were calculated for them. Figure 4.10 graphically illustrates the impact on the three metrics with varying alignment information on all the 5 families. While all the 5 families show increase in the values of the three metrics with alignment information, this is especially perceptible in the SCOP families $c.47.1.5$ and $d.169.1.1$ in Figure 4.10(d) and (e) respectively.

4.4 Related Work

The idea of combining PHMMS has been explored by MetaMEME[22]. However, our approach uniformly models both motif and non-motif regions as full PHMMs with match, insert, and delete states. This leads to more precise results especially when motifs do not cover significant portions of the sequences. Another closely related work is TCoffee[48] which can be used to generate a multiple alignment (from which a family model can be learned) from partial alignments.

There exist two bodies of homology search work that are related to our technique. The first describes methods to improve PHMMs while the second describes methods for homology search using motifs.

Among the works addressing the improvement of PHMMs, [61] introduces empirically derived Dirichlet mixture priors into model construction so as to reduce the number of training sequences needed. Eddy et al. [19] compensate biased representation in sequence data sets by iteratively updating training sequence weights so that more weights are given to divergent training sequences. A different class of approach, exemplified by ATP [40] and

SVM-Fisher [29], incorporate negative training sequences to improve PHMM discrimination. ATP starts with estimating PHMM parameters from a multiple alignment of positive training sequences and then adjusts transition probability parameters based on the paths that positive training sequences and negative training sequences follow through the model. SVM-Fisher maps protein sequences to vectors in a high-dimensional space and determines a "good" separator plane in that space to separate the positive and negative training examples. All these PHMM improvement methods are orthogonal to our approach and in fact can be incorporated to improve the performance of P_PHMM.

PROSITE [21] is one of the earliest databases containing manually and semi-automatically constructed motifs. Blocks+ [24, 25] is another comprehensive database which also contains motifs that are automatically extracted by tools such as PROTOMAT [26]. A motif can be represented as a regular expression and thus deciding whether a sequence contains it amounts to regular expression matching. Another form of motifs is PSSM which requires programs such as BLIMPS [68] to match sequences against the motif. [11] describes a homology search approach where motifs are used as features to map sequences into vectors which are then classified using SVMs or nearest neighbor techniques. MetaMEME [22] is another tool which combines PHMMs for motifs to build a PHMM for the whole sequence. However, in contrast to metaMEME, our approach uniformly models both motif and non-motif regions as full PHMMs with match, insert, and delete states. This leads to more precise results especially when motifs do not cover significant portions of the sequences.

It is worthwhile noting that tools such as ClustalW and TCoffee [27, 48], that automatically generate multiple alignments from a set of sequences, can be used for partial alignment information in our P_PHMM framework. Furthermore, sophisticated model refinements as in SAM [30] can also be incorporated into our P_PHMM for even better performance.

Chapter 5

Protein Active Site Profiling

5.1 Introduction

Active sites are key areas in proteins' three-dimensional structures. Biochemical reactions at these sites with other proteins or other chemical substances cause the protein to perform a function of one type or another.

A problem of significant importance in computational biology is this: *Are active sites of different proteins similar?* i.e., do they share similar physico-chemical and geometric properties. Active sites with such shared properties may perform similar functions. Answer to the aforementioned similarity question drives a number of important biological applications. For instance it can be used to predict the function of a protein with a substructure similar to the active site of another protein whose function is known. Another important application is toxicology tools such as the Toxin Knowledge Base (TKB) system that we have developed [31, 84], for automated diagnosis of bioengineered pathogens. In such pathogens the virulent domains of toxins can be hidden in otherwise non-toxic proteins. Specifically, the active site of a non-toxic protein that is similar to that of a toxin, has the potential to become toxic by suitably altering the residues in the site.

State-of-the-art techniques for determining active site similarity are exemplified by the

SPASM tool [32, 83]. Its inputs include a protein's structure; the 3-D coordinates of the residues in the active site of another protein whose function is known, substitutions for these residues and a RMSD (root mean square distance) cutoff value. SPASM attempts to identify 3-D substructure(s) of the former protein that are isomorphic to the active site within the specified RMSD cutoff.

There are two problems with the pairwise similarity testing approach embodied in SPASM. Firstly, although there are general guidelines for choosing RMSD values such as "If you use only a few residues (3-5), an RMSD less than one Å tends to be obtained for similar arrangements of residues,"[1] in general it is a laborious trial and error process. However, the more serious problem is that similarity tests are done separately with one active site at a time. Consequently, it does not exploit the common physico-chemical and structural features that can exist amongst the *family* of active sites of proteins. A family here means that the active sites of all of its members exhibit similar functionality and can also include evolutionarily unrelated proteins that share no overall sequence or fold similarities. Pairwise comparisons may use features that may not be common to all the family members and hence can fail to identify family members, especially "remote"[2] members. For instance, SPASM fails to find the similarity between the active sites of BOVINE RIBONUCLEASE (PDB ID: 3RN3)[3] and a variant of RIBONUCLEASE (PDB ID: 1RBC) for reasonable RMSD cutoffs because atoms not directly related to the protein's function differ a lot in these two structures. Note however that a "*profile*" of the common features in a collection of active sites belonging to a family would have revealed the irrelevance of such atoms and hence would have been excluded as a shared feature. So a principal benefit of profile based methods is that they capture the essential features shared by all of the family members thereby making it possible to determine the similarity of remote members.

[1] Å denotes an angstrom which is the distance measure between atoms. One angstrom is 1.0×10^{-10} meters.

[2] These are active sites that have few features in common with the other family members.

[3] PDB –`http://www.rcsb.org` – is the Protein Data Bank of 3-D protein structures uniquely indexed by an ID

Automated construction of active site family profiles to discern common features is a fairly unexplored problem. In this dissertation we formulate a solution to this problem inspired by the successful profile-based search methods for homologous protein sequences[4] [49].

We adapt PHMM for profiling the three dimensional active sites in proteins. To begin with, the adaptation requires choosing a representative set of active site features. Whereas only residue types (such as Histidine, Glutanmate, etc) are used as features in PHMMs for protein sequences we will now have to contend with the structural (i.e., geometric) features of active sites also. So in addition to using the atoms' types in the active site residues we also use their distances from their center of mass as the structural features. Furthermore these distances are assumed to be drawn from a probability distribution. Next we adapt the training phase of PHMM to learn the parameters of this distribution and finally modify the scoring phase to assign a similarity score to the input data.

5.2 Techniques

Adapting PHMMs for active site profiling is not entirely straightforward. Let us examine the underlying issues. Firstly, observe that PHMM is a sequential model in the sense that it was developed to handle protein sequences which are simply 1-D strings of amino acids. On the other hand active sites are 3-D structures. So the immediate problem is one of serializing these 3-D structures in such a way that salient aspects of their physico-chemical and geometric properties are still retained. Secondly, each state in a traditional PHMM emits only one discrete symbol (i.e., an amino acid) at a time. For active sites these emissions must include both physico-chemical features such as the discrete valued residue types as well as geometric features. So emissions are tuples ranging over the physico-chemical

[4] A protein sequence is simply a linear string of amino acids that constitute the primary structure of a protein. Sequences that are similar are referred to as homologues.

and geometric feature set. A robust description of geometric configurations of active sites is best done using continuous measures. Hence in contrast to traditional PHMMs where only discrete probabilities of emission symbols are estimated we will now need to estimate the joint distribution of physico-chemical and geometric features. In the rest of this section we describe how we address these issues.

5.2.1 Serializing Active Sites

Since PHMM is a sequential model the task now is to identify a set of 3-D features and serialize them. This serialization will represent the observation sequence corresponding to an active site.

The primary issue in serialization is inventing an ordering for the sequence. For primary protein sequences of amino acid chains this is simply the position of the residue in the chain. For 3-D active sites there is no such obvious ordering. Let us first examine the desiderata for such an ordering. Ideally, if a and b are two conserved atoms in one active site, a' and b' are atoms in another active site corresponding to a and b, respectively, then the order of a and b in the serialized sequence derived from the former active site should be consistent with that of a' and b' in the sequence derived from the latter. A candidate for such an ordering is the distance of the atoms in the active site from their center of mass. Given a set of n atoms with coordinates $(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_n, y_n, z_n)$, their center of mass is the expression:

$$(\frac{1}{n}\sum_{i=1}^{n} x_i, \frac{1}{n}\sum_{i=1}^{n} y_i, \frac{1}{n}\sum_{i=1}^{n} z_i)$$

In other words the center of mass is the average over each of the coordinate positions of the atoms. For illustration, suppose an active site contains only one residue D260 with atoms whose coordinates are listed in the first four columns of Table 5.1. The 3-D coordinate of their center of mass is $(35.719, 107.377, 22.470)$. Distances of each atom from the center of mass are shown in the last column of Table 5.1. The ordering of atoms arranged

Atom Name	X-Cord	Y-Cord	Z-Cord	Distance
N	36.729	107.613	20.276	2.427
CA	35.813	107.722	21.395	1.133
C	36.031	109.051	22.149	1.732
O	37.157	109.446	22.496	2.519
CB	35.875	106.405	22.220	1.016
CG	34.949	106.290	23.394	1.622
OD1	33.858	106.833	23.442	2.169
OD2	35.341	105.659	24.387	2.602

Table 5.1: Example Active Site Atoms

in ascending order of their distances from the center of mass is: < CB, CA, CG, C, OD1, N, O, OD2 >.

To capture physico-chemical feature, we adopt the atom classification in [43] which classifies all non-hydrogen atoms in proteins into 40 classes according to the atom location (side-chain or backbone), connectivity, and chemical nature. We denote the atom type by $ResidueName.AtomName$, which can be unambiguously mapped to an atom type in [43]. For example, the type of the first atom in Table 5.1 is represented by D.N.

As far as geometric feature is concerned, an obvious idea is to use an atom's 3-D coordinate. However, the coordinates of atoms from two active sites are comparable only after those active sites are superposed. Typically, superposing algorithms take two point sets with each point represented by its (x, y, z) coordinate, and perform rigid transformations such as translation and rotation to minimize the RMSD of these two point sets. Since these points are assumed to be typeless, any two points are always superposable. But the problem here is that superposed positions may not be compatible with the atom types at those positions (e.g., in general nitrogen and oxygen atoms cannot be superposed). There are tools such as SPASM [32] that allow users to define superposable atom types. The main problem with this is that knowledge about what are superposable atom types varies from family to family. A desiderata of geometric feature is that it be preserved under serialization. Features that use relative instead of absolute positions can satisfy such a requirement.

Observe that distances of atoms to their center of mass are relative quantities and hence can serve as a geometric feature.

In summary, our feature set is the pair

$$\langle AtomType,\ Distance\_To\_CenterOfMass \rangle,$$

where the first element is the physico-chemical feature and the second is the geometric feature. The general form of an observation sequence corresponding to an active site following serialization using our feature set will be: $\langle t_1,d_1 \rangle$, $\langle t_2,d_2 \rangle$, . . ., $\langle t_n,d_n \rangle$ where n is the number of atoms in the active site, t_i is the atom type and d_i is the distance to the center of mass for i = 1, . . ., n, and $d_i < d_i + 1$ for i=1, . . ., n-1. For our example active site, it is $\langle$D.CB,1.016$\rangle$, $\langle$D.CA,1.133$\rangle$, $\langle$D.CG,1.622$\rangle$, $\langle$D.C,1.732$\rangle$, $\langle$D.OD1,2.169$\rangle$, $\langle$D.N,2.427$\rangle$, $\langle$D.O,2.519$\rangle$, $\langle$D.OD2,2.602$\rangle$.

5.2.2 PHMM for Active Sites

When observation sequences of multiple active sites with similar function are put together, one can identify which atoms are conserved by aligning them. Figure 5.1 shows a segment of the alignment of three similar active sites[5] , namely, acetylcholinesterase (PDB ID: 1ACE) with residues S200, E327, and H440 ; chymotrypsin (PDB ID: 1CHO) with residues H57, D102, and S195; haloalkane dehalogenase (PDB ID: 2DHC) with residues D124, D260, and H289. Although their constituting residues are different, all of them perform similar catalytic function.

This alignment reveals that the consensus sequence has six atoms (see columns marked by '*' in the figure). The atoms appearing in the non-starred columns are insertions. Observe also that the sequence of 2DHC goes through a deletion between the first match and the third match; 1CHO goes through two deletions: one between the third match and the fifth match and the other after the fifth match.

[5]Because of width constraints the sequences in the figure run over to multiple lines.

```
2DHC    <D.OD2,2.91>     -           -
1CHO    <D.OD2,3.08>     -      <H.CB ,3.17>
1ACE    <E.OE1,2.32><H.CD2,2.32><H.CB ,2.45>
             *                       *

2DHC         -      <H.CA ,3.10><H.O  ,3.20>
1CHO    <D.CG ,3.22><H.CA ,3.63>     -
1ACE         -      <H.CA ,3.31><E.CD ,3.46>
                         *           *

2DHC    <H.C  ,3.46><D.CG ,3.47><D.OD1,3.50>
1CHO    <H.C  ,4.06>     -           -
1ACE    <H.C  ,3.94>     -      <E.OE2,4.06>
             *                       *
```

Figure 5.1: A Segment of a Multiple Alignment

We can learn the PHMM parameters (transition and emission probabilities) from such multiple alignments. However, it is labor intensive to come up with such a multiple alignment. One can also learn these parameters from unaligned sequences. First, the number of match states (i.e. the length of the PHMM) is estimated by taking the average length of the training sequences. Then the Baum-Welch algorithm is applied to estimate the transition probabilities and emission probabilities.

We adapt this process for learning PHMM parameters from training data consisting of unaligned serialized active site sequences belonging to a family. First, we estimate the length of the PHMM from the training sequences. This is the average length of the sequences. For example, the average length for the sequences in Figure 5.1 without the dashes is six.

To learn the other two PHMM parameters, we modify the Baum-Welch algorithm. Since emission symbols are pairs $\langle atomtype, distance \rangle$, we will need to compute the joint distribution of these pairs for each state. Making the standard independence assumption done in HMMs, namely, that the random variables in the joint distribution are independent,

the probabilities of the atom types and their distances are computed separately. Let us define the probability of atom type t in a state as $P(t)$ and the probability of the distance d from center of mass as $P(d)$. We calculate the emission probability $P(b)$ of the emission symbol $b = \langle t, d \rangle$ to be $P(t) \times P(d)$.

The distance from the center of mass is a continuous feature. We assume that its probability distribution is generated by a multivariate Gaussian distribution whose probability density function is:

$$\frac{e^{-\left(\frac{(d-\mu)^2}{2\sigma^2}\right)}}{\sigma\sqrt{2\pi}}$$

where d is the distance, μ is the mean and σ is the standard deviation of distances to the center of mass. Suppose the distances to the center of mass from atoms that are emitted by a state are $d_1, \ldots, d_m$. We compute μ and σ at this state using the expressions:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} d_i$$

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(d_i - \mu)^2 + \epsilon}$$

The small constant ϵ is added so that σ is always positive even when $n = 1$.

Recall that we need 42 parameters to describe the emission distribution for each state. Forty of these parameters correspond to the emission probabilities of the 40 atom types and they must sum up to 1. The remaining two are μ and σ that represent the distribution of the distances of atoms emitted from the state to their center of mass.

For a set of unaligned sequences, Baum-Welch algorithm iteratively updates the parameters of the model to increase the overall probability of the set of training sequences to be generated by the model. We modify the Baum-Welch algorithm to take into account the new emission parameter set and the joint emission probability. At each step of iteration, we calculate the individual probabilities of atom type and distance from center of

mass and multiply these probabilities to get the joint probability. For a family of observation sequences of active sites, this modified Baum-Welch algorithm is used to estimate the parameters of the PHMM that profiles this family.

Armed with a PHMM M trained on a family S of serialized active site sequences we can now answer questions about similarity of active sites. To determine if a protein has substructures similar to the active sites in S we proceed as follows: First we find candidate substructures in the protein structure. This can be done with tools such as MOE Active Site Finder [82] and Q-SiteFinder [36]. Then a serialized observation sequence is generated for each candidate substructure. Those are the candidate observation sequences for the protein. For each such observation sequence, we apply the Viterbi algorithm to compute the probability of its most likely path in the PHMM M. To compute the probability of observing a pair $\langle t, d \rangle$ at a state, the Viterbi algorithm computes the probabilities of observing atom type t and distance d separately using the emission distribution parameters of that state, and then multiply them to get the emission probability of the pair.

The step that remains is computation of the log-odds ratio (see Section 2.2). For the PHMM M we define a random model M' whose length and transition probabilities are identical to those in M. The emission parameters are assumed to be uniform for all the insert and match states. These state-independent parameters are computed as follows:

1. The emission probability for atom type a is $\sum_{r\ where\ a \in r} \frac{q(r)}{num\ of\ atoms\ in\ r}$, where r is a residue and $q(r)$ is the frequency of r (see Section 2.2).

2. Randomly sample substructures from PDB, each of which contains the same number of residues as the training examples.

3. For each such substructure, compute the center of mass and the distances of the atoms to this center.

4. Compute the mean μ and the standard deviation σ over all distances and over all

Protein Families	Avg # of Active Site Atoms	Training Set Size	Test Set Size	# of Positive Test Examples	# of Negative Test Examples
Ribonuclease A	20	25	30	10	20
Ribonuclease T1	24	24	35	10	25
Eukaryotic Lysozyme	29	30	20	8	12
Prokaryotic Lysozyme	21	35	31	15	16
Nu:-His-Elec catalytic triad	24	103	105	40	65

Table 5.2: Data Statistics for Different Protein Families

Protein Families	Avg # of Active Site Atoms	Training Set Size	Test Set Size	# of Positive Test Examples	# of Negative Test Examples
sub1	20	18	32	12	20
sub2	22	20	35	15	20
sub3	26	15	22	10	12
sub4	28	18	25	13	12
sub5	24	21	21	11	10

Table 5.3: Data Statistics for Subfamilies of Nu:-His-Elec Catalytic Triad

substructures.

Armed with M and M', we can compute the bit score of an observation sequence in M and decide similarity as was described in Section 2.2.

5.3 Evaluation

We implemented our PHMM-based profiling of active sites. In this section we report on its experimental performance. It is organized into the following subsections: The experimental setup for the evaluation; the performance metrics measured; the experimental results; and discussion of the results.

5.3.1 Experimental Setup

The evaluation was conducted over different sets of protein families detailed below.

Protein Families

We developed PHMM profiles for five different protein families, namely, *Ribonuclease A, Ribonuclease T1, Eukaryotic Lysozyme, Prokaryotic Lysozyme, Nu:-His-Elec catalytic triad*. The Nu:-His-Elec family is further divided into five subfamilies according to the residues that comprise the catalytic triads, which are Ser-His-Asp, Ser-His-Glu, Asp-His-Asp, Ser-His-Trp, and Cys-His-Asn. They are denoted by sub1, sub2, sub3, sub4, and sub5, respectively. We also built profiles for these five subfamilies.

Training and Test Data

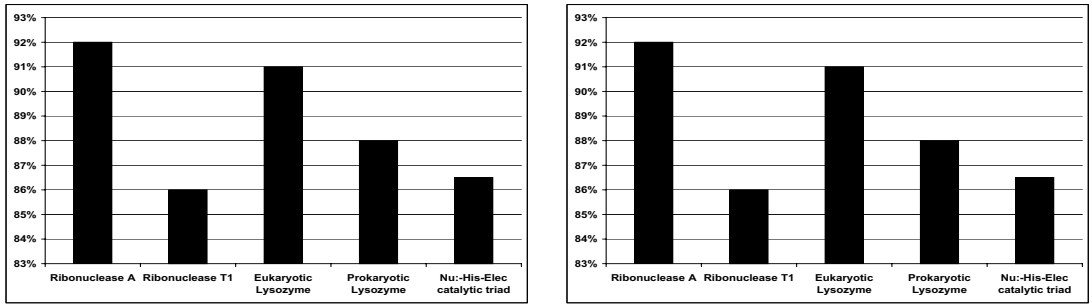
We used 35, 34, 30, 35 and 153 members respectively of Ribonuclease A, Ribonuclease T1,Eukaryotic Lysozyme, Prokaryotic Lysozyme and Nu:-His-Elec catalytic triad families. For profiling the subfamilies we used 30, 35, 25, 31, 32 members of sub1, sub2, sub3, sub4, and sub5 respectively.

The active sites per family were divided into two mutually exclusive training and test sets. The active sites of a family included in the test set associated with the family were labeled as positive test examples. For each family, we augmented its test set with a subset of active sites belonging to other four families. These augmented active sites were labeled as negative test examples.

Statistics associated with the experimental data used are listed in table 5.2 for the five families and in table 5.3 for the subfamilies of Nu:-His-Elec catalytic triad.

From these statistics observe that on the average we used around 43 and 44 active sites respectively for training and testing each family.

We built a separate PHMM per family. The parameters were learned using the training set associated with the family. The global threshold for the log-odds ratio was set to 0.



(a) (b)

Figure 5.2: Precision Performance of (a) 5 Protein Families and (b) Subfamilies of Nu:-His-Elec

5.3.2 Performance Metrics

We evaluated the PHMM with respect to three performance metrics: recall, precision and f-measure [6] using the test data set constructed above.

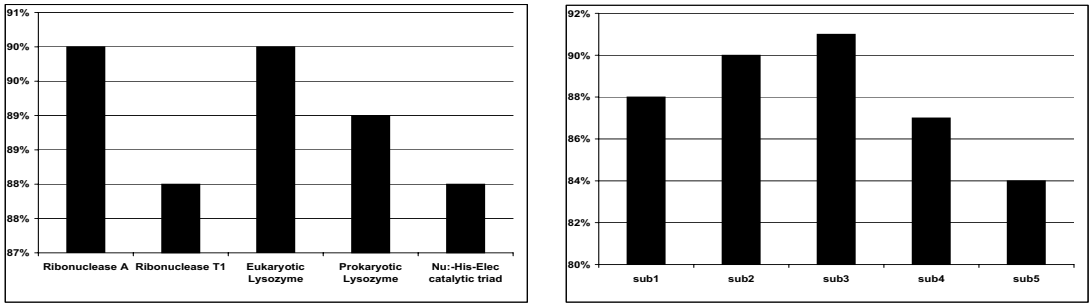
Observe that an active site in the test data for a family was uniquely labeled as a positive or negative test example. These labels are used to classify the similarity results produced by PHMM on the test data into true positives, false positives, true negatives and false negatives. Based on these classifications the recall/precision/F-measures are directly computed from their definitions.

5.3.3 Experimental Results

The results of the experimental evaluation are shown in Figure 5.2, 5.3 and 5.4.

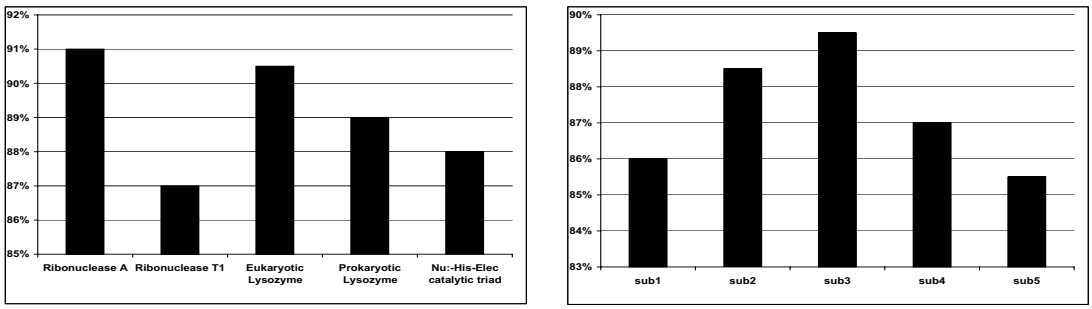
Figure 5.2(a) shows the precision performance for each of the protein families. They range from 86% (for Ribonuclease T1) to 92% (for Ribonuclease A). Figure 5.2(b) shows the precision performance for each sub-families of Nu:-His-Elec catalytic triad.

[6] Recall value for a protein family is defined as the ratio of correctly identified proteins (which are members of the family) over the total number of family members present in the test set. For precision, the denominator is taken as the total number of proteins (both positive as well as negative test examples) present in the test which are identified as members of the family. F-measure is defined as the harmonic mean of recall and precision



(a) (b)

Figure 5.3: Recall Performance of (a) 5 Protein Families and (b) Subfamilies of Nu:-His-Elec



(a) (b)

Figure 5.4: F-Measure Performance of (a) 5 Protein Families and (b) Subfamilies of Nu:-His-Elec

Figure 5.3(a) shows the recall performance for each of the protein families. They range from 88% (for Ribonuclease T1 and Nu:-His-Elec catalytic triad) to 90% (for Ribonuclease A and Eukaryotic Lysozyme). Figure 5.3(b) shows the recall performance for the subfamilies of Nu:-His-Elec catalytic triad.

We also calculated f-measure for each of the families. Figure 5.4(a) shows the f-measure for each of the families. It ranges from 87% (for Ribonuclease T1) to 91% (for Ribonuclease A). Figure 5.4(b) shows the f-measure performance for the sub-families of Nu:-His-Elec catalytic triad.

5.3.4 Discussions

The experimental performance suggests that PHMM-based methods described in this report for determining similarity of active sites works well in practice. The PHMM constructed for each family exhibits reasonably high recall, precision and f-measure values. A high degree of shared features by family members results in higher performance metrics. For instance the active sites of Ribonuclease A shares many atom types along with their geometric configuration. This is reflected by its high recall, precision and f-measures (90%, 92%, 91%). On the other hand the low degree of shared features observed in Ribonuclease T1 has translated into low recall, precision and f-measure values (86%, 88%, 87%).

5.4 Related Work

We review here computational tools and techniques related to the problem of determining similarity of active sites.

On the tool front the best known system is SPASM [32, 83]. It takes the pair $\langle$ protein structure, target active site $\rangle$ as the input and finds substructures in the protein that are similar to the active site. As we had discussed earlier comparing a substructure to an active site independently of other members of the active site's family fails to exploit the commonality amongst them. Consequently, it can fail to establish similarity with some family members, especially remote ones. A profile based approach as we have done addresses this problem.

The idea of profiling active sites was first explored in the context of building the PROCAT database [80, 67, 8], in which the term "functional template" was used for what we refer to as the active site profile in this dissertation. In PROCAT, functional templates are manually defined for several enzyme families. For example, it includes templates for Ribonuclease A and the five subfamilies of Histidine-based catalytic triad (see Tables 5.2 and 5.3). These templates consist of only a subset of atoms in the active site residues.

For instance, only the O^{γ} atom is included in the template for the Ser-His-Asp subfamily. The decision of which atoms to include is done manually through close inspection of the structures and functional mechanisms of all the proteins in the family. The template so constructed captures the features shared by the family members. Th problem here is that template construction is a manual process thereby limiting scalability. In contrast our approach to "learn the templates" is highly automated.

A more recent work is Catalytic Site Atlas (CSA) [66, 74], a database documenting enzyme active sites and catalytic residues present in enzymes with 3-D structures. The active sites are labeled either original or derived. The former are extracted from scientific literature while the latter are associated with proteins whose primary sequences are homologous to the primary sequences of proteins containing the original active sites. An original active site and all of its derived sites constitutes a family. Templates with shared features are again constructed manually for each family.

MultiBind is yet another recent work that takes a set of active sites and automatically aligns all of them [60, 75]. The multiple alignment reveals what are the subset of atoms that are conserved among all the active sites in the set. Firstly, this approach is not statistical unlike ours. But the more important difference is that multiple alignment alone does not provide any quantitative measure of how close an active site is to the aligned sites. Without such measures it is not possible to algorithmically deduce similarity.

PHMMs were used for profiling entire protein structures in [3]. The 3-D structure is serialized into a sequence of 3-D coordinates. In other words this work uses only one geometric feature. Such an approach is useful for determining similarity of entire protein structures whose superposition has the lowest RMSD value. As we had discussed earlier (see Section 5.2.1) 3-D coordinates alone may not adequately capture the salient shared features of the family. Good superpositions in terms of low RMSD values may produce incompatible atom types at the superposed positions. Factoring in both physico-chemical and geometric features as is done in our approach can result in more accurate determinations

of similarity and our experimental results seem to validate this hypothesis. Furthermore, as discussed below, there is no correlation between similarity of entire structures as is done in this work and active site similarity.

Finally, we remark that protein functions can also be predicted based on sequence homology or overall structure similarity. However it has been observed that there is no significant correlation between conservation of sequences, structures and active sites [33]. Hence function prediction by detection of substructures in proteins that are similar to active sites of proteins with known functions complement those based on sequence homology and structural similarity methods.

Chapter 6

Automatic Text Segmentation with CRF

6.1 Introduction

Text segmentation is the process of converting information in plain text strings into structured records. Given a schema consisting of n attributes and an input string, the problem of segmenting the input string can be informally defined as partitioning the string into n contiguous sub-strings and assigning each sub-string a unique attribute from the n attributes. For example, given the address schema consisting of the five attributes $\langle$COMPANY, STREET, CITY, STATE, PHONE$\rangle$ and the input string "1 2 3 Convenience Store (516)538-0854 144 Hempstead Tpke W Hempstead NY", the task of text segmentation is to convert the string into the address record: $\langle$1 2 3 Convenience Store, 144 Hempstead Tpke, W Hempstead, NY, (516)538-0854$\rangle$.

In the World Wide Web, data (such as product, bibliographic and address data) exists as unstructured text strings. They have to be segmented into structured records to facilitate efficient query processing and analysis. Therefore accurate text segmentation methods are important.

Extant techniques for text segmentation either use rules for identifying attributes in the text or employ statistical models. Rule-based approaches require domain experts to create

and maintain a set of rules for each application domain. It is difficult to anticipate all possible variations in the text strings to be segmented and design rules accordingly. This difficulty is further compounded by the presence of noise in the data. Therefore rule-based approaches are neither scalable nor robust. In contrast statistical approaches automatically learn a statistical model for each application domain. The variability and noise in the input text data are elegantly dealt with by the statistical characteristics inherent in such approaches.

Hidden Markov Model (HMM) [53] is a dominant statistical model used in text segmentation. HMM is a generative model in the sense that it captures the probability distribution of observations (e.g. the input strings in the case of text segmentation). There are two main problems with HMM-based text segmentation. Firstly, in order to estimate the distributions of observations HMMs will need to enumerate all possible observations. This may not always be possible. Secondly, as articulated in [34], HMMs have to make strict independence assumptions to achieve computational tractability and hence cannot capture long-range dependences in the input data.

To address the above two shortcomings of HMMs, Conditional Random Fields (CRFs) were introduced in [34] for sequential labeling problems. Note that text segmentation is an application of this problem. There have been a number of recent works on CRFs (see [59, 65, 52, 42, 38]). CRF is a discriminative model that directly computes the conditional probability of a label sequences given an observation sequence. Therefore, it does not need to capture the probability distribution of observations. The other important aspect of CRFs is that they can capture long range dependence in data [34]. Implementations of CRFs have been shown to out-perform generative models like HMMs [13, 58] for text segmentation, especially when the data exhibits long range dependencies.

Text segmentation using statistical models are typically supervised, that is, the model is supplied with manually labeled training data. This, in general, is a labor-intensive process. Unsupervised learning techniques eliminate the need for manually labeled training data.

BUSINESS	STREET	CITY	STATE	PHONE
1 Hour Auto Glass Inc	403 West St	New York	NY	(212)691-3344
1 Hundred 60 4th St Auto	8412 164th St	Jamaica	NY	(718)523-9018
10 Minute Oil Change	1156 Hempstead Tpke	Uniondale	NY	(516)486-0060
A Salerno Realty Crop	11 Mill	Rhinebeck	NY	(914)876-5551
Circuit City	111 E El Camino Real	Sunnyvale	CA	(408)720-1043

Table 6.1: A Segment of a Reference Table

Recently a fully automatic, unsupervised text segmentation system is described in [1]. This system exploits structured reference tables consisting of clean tuples. In other words the attributes in these tables are already labeled.

Table 6.1 shows a fragment of a reference table. Reference tables can contain a large number of records thereby providing a rich source of labeled training data. Note that the order in which the attributes appear in the table might be different from that in which they appear in the input sequences. For example, the order in which attributes appear in one bibliography data source may be [AUTHOR, TITLE, PUBLISHER, PAGE, YEAR] while in another source it may be [TITLE, AUTHOR, PUBLISHER, PAGE, YEAR]. So an unsupervised text segmentation system will have to deal with differences in the attribute order of the input sequences. Assuming that a batch of text sequences to be segmented share the same total attribute order (e.g. publications in a researcher's home page), the technique in [1] first trains an HMM model for each attribute using the reference table data. Next it uses these trained attribute recognition models to identify the best starting positions for every attribute in every input sequence. Then it uses these positions to infer the common total order. Finally the total order is used to construct a global HMM to segment the input text data sequences.

Although CRFs have recently been used for text segmentation, they are by and large supervised approaches. A recent work on CRF-based text segmentation [39] focuses on reducing the training data using reference tables but does not completely eliminate their use.

Inspired by the work in [1], we have developed a CRF-based unsupervised text segmentation technique using reference tables. Using CRFs for this problem poses some challenges. The main difficulty is inferring the total order. In HMMs this is not an issue. Being a generative model HMM can easily compute the marginal distribution of observations. In particular suppose $P(\mathbf{o})$ is the marginal distribution of an observation sequence $\mathbf{o}$. Given two substrings $\mathbf{s}_1$ and $\mathbf{s}_2$ and the attribute HMM trained to recognize instances of $Attr$, we can readily determine which one is more likely to be an instance of $Attr$ by simply comparing $P(\mathbf{s}_1)$ and $P(\mathbf{s}_2)$.

Since CRFs do not model distributions of observations we will have to develop a new technique for inferring the total order. In this chapter we present such a technique. We introduce negative labels and include negatively labeled examples in the training of attribute CRF models. An important aspect of our approach is the process underlying the generation of these examples so as to ensure that the attribute CRF will assign low likelihood scores to incorrect starting positions of an attribute in the input sequence.

6.2 Techniques

6.2.1 Reference Table

A reference table is a relational table whose columns, particularly column names, correspond to labels that are to be assigned to tokens in test sequences. Let us assume $\langle a_1, a_2, \ldots, a_n \rangle$ are the columns. We use columns and column names interchangeably. Our method of exploiting the reference table for unsupervised text segmentation with CRF is based on the observation that test token sequences usually come in batches with an attribute order common to all the sequences in a batch. Examples include product data description in an online vendor catalog such as OfficeMax and Staples.

Let $\{\mathbf{s_1}, \mathbf{s_2}, \ldots, \mathbf{s_m}\}$ denote a batch consisting of m test sequences with $\mathbf{s_i}$ being the

$i'th$ token sequence. Let us further assume that the common attribute order for this batch is: $a'_1 \rightarrow a'_2 \rightarrow \cdots \rightarrow a'_n$ where $\langle a'_1, a'_2, \ldots, a'_n \rangle$ is a permutation of $\langle a_1, a_2, \ldots, a_n \rangle$. The overall idea of segmenting this batch is as follows:

1. We first build a CRF for each column $a_i (1 \leq i \leq n)$ of the reference table. We will call it the *attribute CRF*. The training of attribute CRFs is *fully automatic*.

2. The attribute CRFs are used to compute the most likely starting position (in a probabilistic sense) of an attribute instance in each test sequence.

3. The results obtained in the previous step are combined to infer the common order $a'_1 \rightarrow a'_2 \rightarrow \cdots \rightarrow a'_n$ of the sequences in the batch.

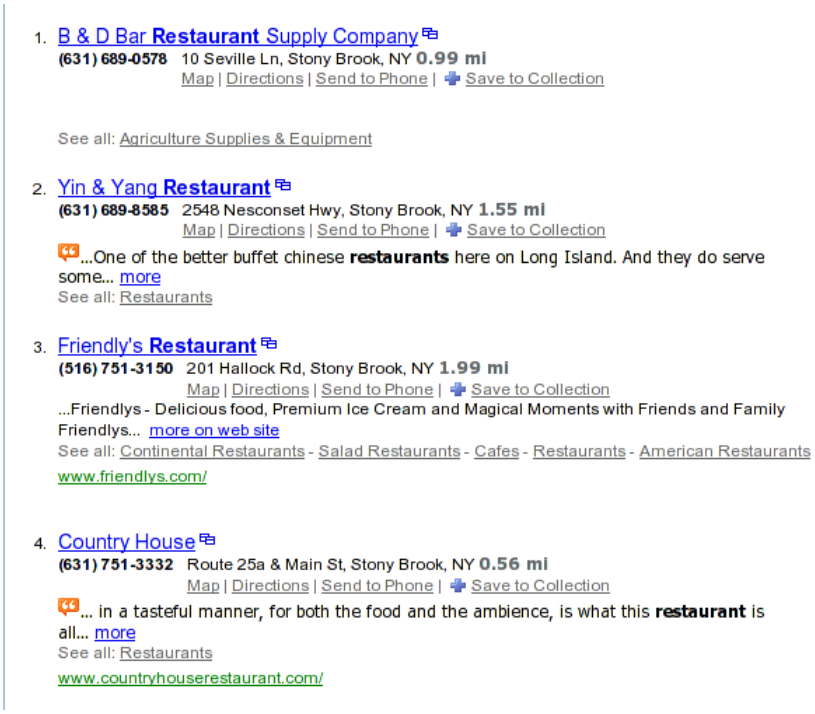
4. In the final step we derive labeled training examples from the reference table according to the inferred common order and train a global CRF model for text segmentation using the known techniques [34, 59].

From now on, we will refer to the attribute CRF corresponding to the attribute Attr by Attr CRF.

The aforementioned process is illustrated by the following example. Figure 6.1 shows a fragment of the search results returned by Yahoo Local in response to a query about restaurants in the neighborhood of zip code 11790.

The first entry in Figure 6.1 starts with the name of a BUSINESS "B & D Bar Restaurant Supply Company", followed by its PHONE "(631)689-0578", STREET "10 Seville Ln", CITY "Stony Brook", and STATE "NY". The common order of the attributes for this search result fragment is: BUSINESS → PHONE → STREET → CITY → STATE.

We assume that the search result is preprocessed and tokenized. Our task now is to segment the four token sequences shown in Figure 6.2 and assign labels to the segments from the label set {BUSINESS, PHONE, STREET, CITY, STATE}.


1. B & D Bar **Restaurant** Supply Company
(631) 689-0578 10 Seville Ln, Stony Brook, NY **0.99 mi**
Map | Directions | Send to Phone | Save to Collection

See all: Agriculture Supplies & Equipment

2. Yin & Yang **Restaurant**
(631) 689-8585 2548 Nesconset Hwy, Stony Brook, NY **1.55 mi**
Map | Directions | Send to Phone | Save to Collection
...One of the better buffet chinese **restaurants** here on Long Island. And they do serve some... more
See all: Restaurants

3. Friendly's **Restaurant**
(516) 751-3150 201 Hallock Rd, Stony Brook, NY **1.99 mi**
Map | Directions | Send to Phone | Save to Collection
...Friendlys - Delicious food, Premium Ice Cream and Magical Moments with Friends and Family Friendlys... more on web site
See all: Continental Restaurants - Salad Restaurants - Cafes - Restaurants - American Restaurants
www.friendlys.com/

4. Country House
(631) 751-3332 Route 25a & Main St, Stony Brook, NY **0.56 mi**
Map | Directions | Send to Phone | Save to Collection
... in a tasteful manner, for both the food and the ambience, is what this **restaurant** is all... more
See all: Restaurants
www.countryhouserestaurant.com/

Figure 6.1: Restaurant Addresses Presented in the Same Attribute Order

```
{
  [B, &, D, Bar, Restaurant, Supply, Company, 631,
   689, 0578, 10, Seville, Ln, Stony, Brook, NY],
  [Yin, &, Yang, Restaurant, 631, 689, 8585, 2548,
   Nesconset, Hwy, Stony, Brook, NY],
  [Friendly's, Restaurant, 516, 751, 3150, 201,
   Hallock, Rd, Stony, Brook, NY],
  [Country, House, 631, 751, 3332, Route, 25a, &,
   Main, St, Stony, Brook, NY]
}
```

Figure 6.2: Example Test Token Sequences

An example reference table that can be used to segment token sequences like those in Figure 6.2 is shown in Table 6.1. We use the reference table to automatically train the 5 attribute CRFs: one each for BUSINESS, STREET, CITY, STATE, and PHONE. Those attribute CRFs will be deployed in tandem to infer the common attribute order in the batch

```
{
  <[1, Hour, Auto, Glass, Inc, 212, 691, 3344, 403,
    West, St, New, York, NY],
   [B, B, B, B, B, P, P, P, S, S, S, C, C, T]>,
  <[1, Hundred, 60, 4th, St, Auto, 718, 523,
    9018, 8412, 164th, St, Jamaica, NY],
   [B, B, B, B, B, B, P, P, P, S, S, S, C, T]>,
  <[10, Minute, Oil, Change, 516, 486, 0060, 1156,
    Hempstead, Tpke, Uniondale, NY],
   [B, B, B, B, P, P, P, S, S, S, C, T]>,
  <[A, Salerno, Realty, Crop, 914, 876, 5551, 11,
    Mill, Rhinebeck, NY],
   [B, B, B, B, P, P, P, S, S, C, T]>,
  <[Circuity, City, 408, 720, 1043, 111, E, El,
    Camino, Real, Sunnyvale, CA],
   [B, B, P, P, P, S, S, S, S, S, C, T]>
}
```

Figure 6.3: Example Training Data Derived from Reference Table

consisting of the sequences in Figure 6.1, namely: BUSINESS → PHONE → STREET → CITY → STATE. Then labeled training examples are derived automatically from the reference table, Table 6.1, according to this inferred order (see Figure 6.3). For brevity, we use B for BUSINESS, P for PHONE, S for STREET, C for CITY, and T for STATE. A global CRF is then trained from the examples in Figure 6.3 and used to segment the token sequences in Figure 6.2 using standard CRF techniques [34, 59].

The idea of exploiting reference tables for unsupervised text segmentation was first explored in [1] using HMMs. Applying this idea to CRF is not entirely straight-forward. In the following section we first discuss the challenges and then present our solution.

6.2.2 Attribute CRF

6.2.2.1 Issues in Training Attribute CRFs

Recall that the primary objective for building attribute CRFs is to infer the total order of attributes that is common to all the test sequences in a batch. For each test sequence,

we compute the most likely starting position for its attributes. These positions impose a local precedence relation $\prec_{local}$ on pairs of attributes. Specifically, we say that $a_i \prec_{local} a_j$ whenever p and q are the most likely starting positions for a_1 and a_j respectively and $p < q$. If $a_i \prec_{local} a_j$ in a majority of the test sequences then we say that a_i precedes a_j in the common total order, denoted $\prec_{global}$.

Thus, an attribute CRF for the attribute *Attr* should be able to identify the most likely token sub-sequence corresponding to *Attr's* occurrence in the test sequence. However, training an attribute CRF to do such an identification is not as simple as taking all instances of *Attr* in the reference table and training the Attr CRF model using standard CRF training algorithms.

For an exposition of the underlying issues, let us revisit the example in Section 2.3 and examine the training of CITY CRF. If we train it only from the attribute instances of CITY, then no matter what the input token sequence is, the only way of labeling is to assign the label CITY to every token. Therefore, the conditional probability of labeling the token sub-sequence [Huntington] with CITY is the same as that of labeling [NY] with CITY, both of which are 1. It is hence impossible to decide whether [Huntington] or [NY] is the more likely instance of the attribute CITY.

In HMM-based attribute models as described in [1], this is not a problem. HMMs model the joint probability $P(\mathbf{x}, \mathbf{y})$, $\mathbf{x}$ being the token sequence and $\mathbf{y}$ the label sequence. Note that a label sequence corresponds to a state sequence in HMM terminology. The marginal probability $P(\mathbf{x})$=$\sum_y' P(\mathbf{x}, \mathbf{y}')$, computed by the classic forward algorithm [53], can be used for the purpose of deciding the best starting position of an attribute instance. Specifically, given an attribute HMM trained from the instances of the attribute in the reference table, deciding whether the token sub-sequence $\mathbf{x_1}$ or $\mathbf{x_2}$ fits the attribute better is simply done by comparing their marginal probabilities $P(\mathbf{x_1})$ and $P(\mathbf{x_2})$, and picking the one with the higher probability value.

In contrast CRF is discriminative model, i.e., given the token sequence $\mathbf{x}$, it directly

models the conditional probability $P(\mathbf{y}|\mathbf{x})$ of the label sequence $\mathbf{y}$. Since CRFs do not model probability distributions of observation (i.e. token) sequences, one cannot compute their marginal probabilities. In particular $\sum_{\mathbf{y}'} P(\mathbf{y}|\mathbf{x})$ is always 1 no matter what $\mathbf{x}$ is and therefore it cannot serve as a criteria for deciding which token sub-sequence best fits an attribute.

6.2.2.2 Negative Labels: A Substitution for Marginalization

To solve the problem of using CRFs for identifying the most likely token sub-sequence as an attribute instance, we introduce negative labels and include negatively labeled examples in the training of attribute CRF models.

We associate each attribute CRF with two labels – positive and negative. For example the labels associated with the CITY CRF are CITY and ¬CITY. Instances of an attribute in the reference table are assigned the positive label. So instances of CITY constitute the positive examples for the CITY CRF.

We assign the negative label to instances of all the other attributes in the reference table. So instances of BUSINESS, STREET, STATE and PHONE in Table 6.1 will be assigned the label ¬CITY.

However, these alone will not suffice as negatively labeled examples. Recall that an attribute CRF is required to choose from among all the token sub-sequences of a test sequence the most likely instance of the attribute. So for instance given the token sequence [B, &, D, Bar, Restaurant, Supply, Company, 631, 689, 0578, 10, Seville, Ln, Stony, Brook, NY], ideally CITY CRF should assign the label ¬CITY not only to tokens in the token sub-sequence [10, Seville, Ln] which is an instance of STREET, but also to the three token sub-sequences [Stony], [Stony, Brook, NY], and [Company, 631, 689, 0578, 10, Seville]. These are readily characterized by missing tokens (such as Brook in [Stony]) or with extra tokens (e.g. NY and Selville in the second and third token sub-sequences respectively). Therefore we should include different kinds of negative examples so that the attribute CRF

By (Rule 1)	[York]
By (Rule 2)	[New]
By (Rule 3)	[Inc, New, York], [St, New York], [NY, New, York], [3344, New, York]
By (Rule 4)	[New, York, 1], [New, York, 403], [New, York, NY], [New, York, 212]

Table 6.2: Negatively Labeled Examples

can assign low attribute association likelihood to the token sub-sequences in the aforementioned cases.

Taking into consideration the discussion of the issues above, the process of generating negatively labeled examples for an attribute CRF is as follows: Suppose Attr is an attribute. A straightforward way to generate negatively labeled examples for Attr CRF is to simply form all possible combinations of token sub-sequences from the reference table and eliminate those that are instances of Attr. Clearly this is an expensive proposition. Instead we use the following simple but efficient heuristic rules:

- *Rule 1*: token sequences obtained by deleting the first token from any Attr instance.
- *Rule 2*: token sequences obtained by deleting the last token from any Attr instance.
- *Rule 3*: token sequences obtained by prefixing any Attr instance with the last token of any instance of any attribute other than Attr.
- *Rule 4*: token sequences obtained by suffixing any Attr instance with the first token of any instance of any attribute other than Attr.

Table 6.2 shows examples assigned the label ¬CITY. They are generated by the application of the above heuristic to the first tuple in Table 6.1. The first column in Table 6.2 indicates the rule used to generate the sequences in the row.

The positive and negative examples generated by the process described above is used to train an attribute CRF using well known CRF training algorithms [34, 59].

Armed with the trained CRF for an attribute say Attr, the probability of a token sequence $\mathbf{s}$ being an instance of Attr is:

$$\frac{P(p|\mathbf{s})}{P(p|\mathbf{s}) + P(n|\mathbf{s})} \tag{6.1}$$

where $P(p|\mathbf{s})$ is the conditional probability, computed by Attr CRF, for labeling all the tokens in $\mathbf{s}$ with the positive label Attr and $P(n|\mathbf{s})$ is the conditional probability for labeling all the tokens in $\mathbf{s}$ with the negative label ¬Attr.

The idea of using negative examples in CRFs was first explored in [39]. But there are two differences with our approach. Firstly, CRFs with negative labels in [39] are used to define additional feature functions. Manually labeled examples, although fewer in number, are still required. In our case, we use such CRFs to infer a total order of attributes shared by a batch of test sequences, based on which we develop an unsupervised approach for training CRFs from reference tables. Secondly, as described previously in this section, the negative examples of our attribute CRFs are constructed subtly (see Rules 1, 2, 3 and 4) so as to ensure that incorrect token sub-sequences are given a very low score.

6.2.3 Unsupervised Text Segmentation with Attribute CRF

Following [1], we also assume that the batch of test sequences share a common attribute order. The first step in inferring this order is to compute, with attribute CRFs, the most likely starting position (in a probabilistic sense) of an attribute instance in each test sequence.

6.2.3.1 Computing Most Likely Starting Positions of Attribute Instances

We associate two vectors $\mathbf{v_k}$ and $\mathbf{score_k}$ with the k-th test sequence in the batch $[s_1, \ldots, s_m]$. The test sequence s_i is denoted by $[t(i)_1, \ldots, t(i)_{i_m}]$. Given that $a_1, \ldots, a_n$ are the attributes in the reference table, both vectors have length n. The i-th element in $\mathbf{v_k}$ is the most likely starting position in the test sequence for attribute a_i, and $score_{k_i}$ denotes

the probability of its likelihood. Algorithm *BestStartingPosition* is a sketch of how the most likely starting positions are computed. In the algorithm $[t(k)_1, \ldots, t(k)_{k_m}]$ denotes the test sequence.

Algorithm *BestStartingPosition*

1. **for** $i \leftarrow 1$ **to** n
2. **do** $v_{k_i} \leftarrow -1$;
3. $score_{k_i} \leftarrow 0$
4. **for** $i \leftarrow 1$ **to** k_m
5. **for** $j \leftarrow i$ **to** k_m
6. **do** $\mathbf{s} \leftarrow [t(k)_i, \ldots, t(k)_j]$
7. **for** $l \leftarrow 1$ **to** n
8. **do** $p \leftarrow$probability of $\mathbf{s}$ being an instance of attribute a_l by Equation 6.1 using a_l's attribute CRF
9. **if** $p > score_{k_l}$
10. **then**
11. $v_{k_l} \leftarrow i$;
12. $score_{k_l} \leftarrow p$;

Line 1 to 3 is initializing $\mathbf{score_k}$ and $\mathbf{v_k}$. Line 4-5 forms a loop over all token sub-sequences of the test sequence. The token sub-sequence is assigned to $\mathbf{s}$ in Line 6. Line 7 loops over all attributes. Line 8-12 updates the most likely starting position in $\mathbf{v_k}$ and the associated probability in $\mathbf{score_k}$ for each attribute and each token sub-sequence.

6.2.3.2 Inferring the Common Total Order

For a batch of m test sequences, we say that $a_i \prec_{local} a_j$ in the k-th test sequence if $v_{k_i} < v_{k_j}$, otherwise $a_j \prec_{local} a_i$. Once the best starting positions are computed by Algorithm *BestStartingPosition*, we associate a global vote count $votes_{i,j}$ for each pair of attributes a_i and a_j ($1 \leq i, j \leq n$ and $i \neq j$). This vote count corresponds to the number

of test sequences in which $a_i \prec_{local} a_j$. If in a majority of test sequences $a_i \prec_{local} a_j$ holds then we say that $a_i \prec_{global} a_j$. The rationale is that, in most cases the attribute CRFs can approximately recognize the correct starting position for an attribute among all possible token sub-sequences of a test sequence. Notice that for each pair of attributes a_i and a_j, we either have $a_i \prec_{global} a_j$ or $a_j \prec_{global} a_i$. This $\prec_{global}$ relation forms a directed graph. When there is no cycle in the graph, we can find a total order of the attributes. When a cycle exists, we break the cycle by removing an arbitrary edge in the cycle.

Algorithm *InferOrder* below sketches these ideas. In the algorithm, R is the $\prec_{global}$ relation, S is the set of attributes, and $LIST$ is a list that represents the total order of attributes.

Algorithm *InferOrder*

1. **for** $i \leftarrow 1$ to n
2. **for** $j \leftarrow 1$ to n
3. $votes_{i,j} \leftarrow 0$
4. **for** $k \leftarrow 1$ **to** m
5. compute $\mathbf{v_k}$ by Algorithm *BestStartingPosition*
6. **for** $i \leftarrow 1$ to $n-1$
7. **for** $j \leftarrow i+1$ to n
8. **if** $v_{k_i} < v_{k_j}$
9. **then** $votes_{i,j}++$
10. **else** $votes_{j,i}++$
11. $R \leftarrow$empty set
12. **for** $i \leftarrow 1$ to $n-1$
13. **for** $j \leftarrow i+1$ to n
14. **if** $votes_{i,j} > votes_{j,i}$
15. **then** add $a_i \prec_{global} a_j$ to R
16. **else** add $a_j \prec_{global} a_i$ to R
17. $S \leftarrow \{a_1, \ldots, a_n\}$

18. $LIST \leftarrow$empty list
19. **repeat**
20. **if** there is an attribute a so that there is no $a' \prec_{global} a$ for any attribute a' in R
21. **then**
22. append a to $LIST$
23. delete a from S
24. delete all $a \prec_{global} a'$ for any attribute a' from R
25. **else** there is a cycle in the precedence relation; break the cycle by removing an arbitrary $a \prec_{global} a'$ that is part of the cycle
26. **until** S is empty

Line 1-3 initializes $\mathbf{votes}$. Line 4-10 computes $votes_{i,j}$ for each pair of attributes a_i and a_j. Line 11 initializes R. Line 12-16 computes R, i.e., the $\prec_{global}$ relation. Line 17 and Line 18 initializes S and $LIST$, respectively. Line 19-20 finds the total attribute order.

There are more sophisticated ways of inferring the total order. For example, the precedence relation is associated with probabilities in [1] and a brute-force search over all possible total orders is used to compute the most probable total order. A greedy algorithm for deciding the total order is described in [35]. However, as we will show in Section 6.3.3.2 on evaluation, our simple majority vote algorithm works well in practice. Note that cycles in the precedence relation are broken arbitrarily. Interestingly, cycles were a rare occurrence in our experiments.

We illustrate the algorithm with a simple example. The best attribute starting positions computed by attribute CRFs for the four test sequences in Figure 6.2 is shown in Table 6.3.

In the first test sequence, we can see that STREET $\prec_{local}$ BUSINESS, STREET $\prec_{local}$ PHONE, STREET $\prec_{local}$ CITY, STREET $\prec_{local}$ STATE, BUSINESS $\prec_{local}$ PHONE, BUSINESS $\prec_{local}$ CITY, BUSINESS $\prec_{local}$ STATE, PHONE $\prec_{local}$ CITY, PHONE $\prec_{local}$ STATE, and CITY $\prec_{local}$ STATE. The $\prec_{local}$ relation other sequences are determined similarly. The number of sequences in which STREET $\prec_{local}$ BUSINESS holds is 1 while

	BUSINESS	STREET	CITY	STATE	PHONE
#1	5	4	14	16	8
#2	1	8	11	13	5
#3	1	6	9	11	3
#4	1	8	10	12	3

Table 6.3: Estimated Starting Positions

that of BUSINESS $\prec_{local}$ STREET is 3. Therefore for these two attributes, BUSINESS $\prec_{global}$ STREET. We do the same thing for other pairs of attributes. This leads to the $\prec_{global}$ relation R = { BUSINESS $\prec_{global}$ PHONE, BUSINESS $\prec_{global}$ STREET, BUSINESS $\prec_{global}$ CITY, BUSINESS $\prec_{global}$ STATE, PHONE $\prec_{global}$ STREET, PHONE $\prec_{global}$ CITY, PHONE $\prec_{global}$ STATE, STREET $\prec_{global}$ CITY, STREET $\prec_{global}$ STATE, CITY $\prec_{global}$ STATE}. When applying Algorithm *InferOrder*, we first add BUSINESS to the list $LIST$, then R = {PHONE $\prec_{global}$ STREET, PHONE $\prec_{global}$ CITY, PHONE $\prec_{global}$ STATE, STREET $\prec_{global}$ CITY, STREET $\prec_{global}$ STATE, CITY $\prec_{global}$ STATE}. Then we add PHONE to $LIST$. And so on and so forth. Finally the total order of attributes shared by the batch of test sequences in Figure 6.2 is computed as BUSINESS $\rightarrow$ PHONE $\rightarrow$ STREET $\rightarrow$ CITY $\rightarrow$ STATE.

6.2.3.3 Putting it All Together

We use the attribute order, determined in the previous step to generate labeled training examples from the reference table. Specifically, for each row in the reference table, we concatenate attribute instances according to that order to form a token sequence. Each token is labeled with the attribute label associated with the column from where the token is drawn. These labeled sequences are used to train a global CRF for segmenting the test sequences.

As an example, Figure 6.3 shows the labeled training examples derived from the reference table in Table 6.1 according to the total order computed in the previous step, i.e.,

BUSINESS → PHONE → STREET → CITY → STATE.

Unsupervised Text Segmentation

Once we get the labeled training examples, we use the standard training and inference techniques described in [34, 59, 73] to train a global CRF and use it to segment the test sequences in the batch (see Algorithm *UnsupervisedSeg* below).

Algorithm *UnsupervisedSeg*

1. Infer total order by Algorithm *InferOrder* (Section 6.2.3.2)
2. Generate labeled examples
3. Train a global CRF
4. Segment test sequences in the batch

In the last two steps, we can just use any standard CRF training and inferencing techniques such as [34, 59, 51].

6.3 Evaluation

We implemented Algorithm *UnsupervisedSeg*, our structured-reference-tables driven unsupervised text segmentation technique with CRFs. Our implementation extended the freely available CRF source code [73]. In this section we report on its experimental performance. We begin with the experiment setup.

6.3.1 Experimental Setup

We discuss here the datasets used for the experiments including training and test data.

Datasets

Table 6.4 lists the three different structured datasets that we used in our experiments, namely, address, product and bibliographic data. The address and bibliographic data were

Dataset	Schema	Source	No. of Tuples
Address	*Name, Street, City, State, Zip*	RISE repository [81]	4300
Product	*Thickness, Manufacturer, Model, RingType, Category, Quantity*	Staples, OfficeMax, and Office Depot	510
Bibliography	*Author, Title, Publisher, Page, Year*	personal .bib file [72]	80

Table 6.4: Datasets Used for Experiments

obtained from [81] and [72] respectively. The product data was extracted from the Web. The data schema corresponding to the three datasets is shown in the second column of Table 6.4.

Each dataset represents a structured reference table where individual columns represent attributes. The entry in a particular column of the dataset represents an instance of the attribute corresponding to that column. We use the attribute name to label its instances. Observe that these reference tables serve as labeled training data.

Training and Test Datasets

Each of the three datasets were divided into two mutually exclusive sets: a training set and a test set.

As noted above, the training data drawn from the structured reference table already comes in labeled form. We trained a attribute CRF model for each attribute in a dataset. Thus for the address dataset we build NAME CRF, STREET CRF, and so on To construct an attribute CRF for say *Attr* we used two kinds of data - positive labeled data corresponding to instances of *Attr* and negatively labeled data generated as described in Section 6.2.2.

For illustration: Consider the segment of the bibliographic reference table shown in table 6.5. The text segments in the AUTHOR column constitute positively labeled examples for the AUTHOR CRF. Instances of such examples will include S.W Zucker, P.N. Johnson and G. Nelson, all drawn from the AUTHOR column.

Each column such as AUTHOR is augmented with another column ¬AUTHOR populated with instances that are not in the AUTHOR column using the techniques described

AUTHOR	TITLE	YEAR
S.W. Zucker.	Childhood and Adolescence	1976
P.N. Johnson.	Mental Models	1983
G. Nelson	Systems programming	1991

Table 6.5: Bibliography Reference Table Fragment

in 6.2.2.2. The instances in ¬AUTHOR so constructed constitute the negatively labeled example set for constructing the AUTHOR CRF. Examples of text segments labeled with ¬AUTHOR are: Childhood and Adolescence, 1976, Johnson, Zucker.

We set aside a fraction of the reference table data not used for training, as the test data set. We choose a priori a consistent attribute order for all the sequences in the test set and generate them from the reference table. So for instance, if the third row of Table 6.5 is used to generate a test sequence, with the attribute order fixed as TITLE → AUTHOR → YEAR, then the generated test sequence will be: <Systems programming G. Nelson 1991>.

6.3.2 Brute Force Segmentation

As is done in [1], Algorithm *UnsupervisedSeg* in Section 6.2.2 also assumes that there is a consistent global total order among the attributes in the input (test) data sequences. To assess the impact of this assumption we compare *UnsupervisedSeg* with *BruteforceSeg*, a brute force text segmentation algorithm that makes no such assumptions.

Given a test sequence, *BruteforceSeg* tries all possible ways of labeling it in a brute-force way. Suppose we have n labels to label a test sequence of m tokens. First, *Bruteforce-Seg* divides the input sequence into i segments, with i from 1 to n. There are C_{m-1}^{i-1} ways of such segments. Then it chooses i labels from those n labels and permutes them. There are P_n^i permutations. Therefore in total there are $\sum_{i=1}^{n} C_{i-1}^{m-1} P_n^i$ ways of labeling the input sequence. Each labeling corresponds to a particular segmentation of the test sequence. For each segmentation, the following computation is done: For each attribute CRF, *Bruteforce-Seg* computes the conditional probability of labeling the corresponding sub-string with only

positive labels.

We illustrate this with an example. For the segmentation which labels "Stony Brook" as CITY and "NY" as STATE, *BruteforceSeg* computes the probability for "Stony Brook" with the CITY CRF and for "NY" with STATE CRF. After computation of individual conditional probabilities from each CRF, joint probability value is computed. Assuming independence among the attribute CRFs, an intuitive way of computing the joint probability of labeling "Stony Brook" with CITY and "NY" with STATE is to take the product. However, we have variable number of attributes and simply taking the product favors fewer attributes since each probability is between 0 and 1. So, *BruteforceSeg* computes the geometric mean of the probabilities returned by individual attribute CRFs as the joint probability. Once the joint probabilities of attribute CRFs of all possible segmentations are computed, the one with the highest joint probability is taken as the segmentation of the test sequence.

Note that, *BruteforceSeg* does not use the total order assumption. Therefore, it can be used as a baseline system to measure the goodness of *UnsupervisedSeg* as is shown in following subsection.

6.3.3 Experimental Results

We evaluated the performance of our algorithm with respect to three performance metrics: recall, precision and f-measure [1] . We report on the performance of:

- Algorithm *UnsupervisedSeg* with clean data, noisy data and varying training size,
- Algorithm *InferOrder*, and
- Algorithm *BruteforceSeg* and contrast it with *UnsupervisedSeg*.

[1] Recall value for an attribute *Attr* is defined as the ratio of tokens correctly assigned the label *Attr* over the total number of instances of *Attr* in the test set. For precision, the denominator is taken as the total number of tokens assigned the label *Attr* either correctly or incorrectly. The f-measure is calculated by taking the harmonic mean of recall and precision.

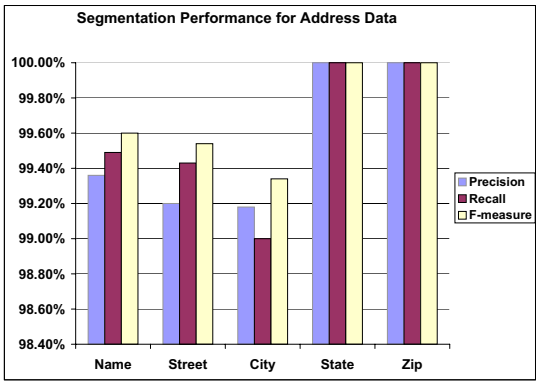


Figure 6.4: Performance for Address Dataset

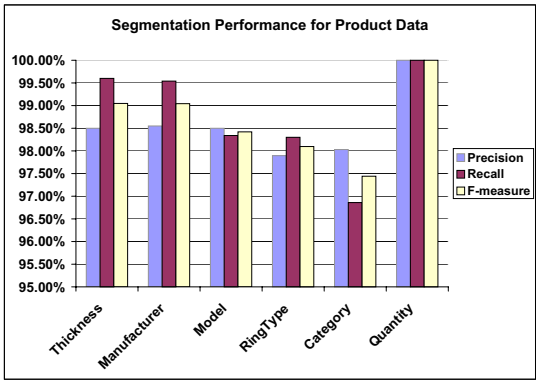


Figure 6.5: Performance for Product Dataset

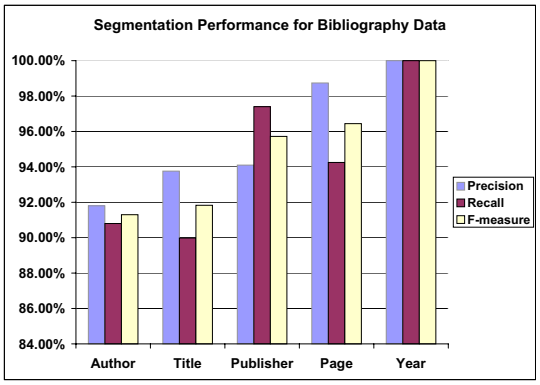


Figure 6.6: Performance for Bibliography Dataset

6.3.3.1 Performance of UnsupervisedSeg

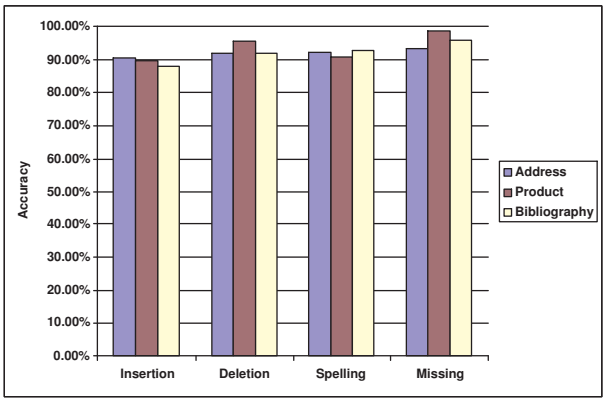


Figure 6.7: Accuracy of UnsupervisedSeg with Noisy Data

With Clean Data

Figure 6.4, 6.5, and 6.6 illustrate precision, recall and f-measure performance for each of the attributes in address, product and bibliography dataset respectively.

For address dataset, precision value ranges from 99.18% (for City) to 100% (for State and Zipcode). Recall performances range from 99% (for City) to 100% (for State and Zipcode). F-measure value ranges from 99.09% (for City) to 100% (for State and Zipcode).

For product dataset, precision value ranges from 94.50% (for Model) to 100% (Quantity). Recall performances range from 95% (for Model) to 100% (for Quantity) and f-measure performances range from 95.44% (for Category) to 100% (for Quantity).

For bibliography dataset, precision performances range from 91.80% (for Author) to 100% (for Year). Recall value ranges from 89.98% (for Title) to 100% (for Year). F-measure values range from 91.30% (for Author) to 100% (for Year).

These results suggest that unsupervised CRF-based text segmentation with reference tables works well in practice. Uniform performance is observed over all the three datasets. We also note that the experimental results are comparable to the unsupervised HMM-based approach with reference tables in [1].

With Noisy Data

Noise Type	Noise Description	Accuracy
Insertion	Insert a random token	80.81
Deletion	A randomly chosen token is deleted	92.63
Spelling	A randomly chosen token is corrupted	87.83
Missing	Replace randomly chosen attribute with null	96.08

Table 6.6: Performance of UnsupervisedSeg with Noisy Data

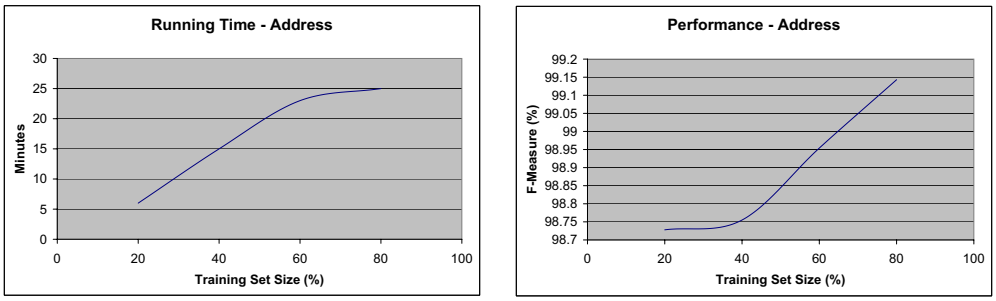
We characterize test data with missing attribute values, one or more insertion, deletion, or spelling errors as noisy data [1]. Figure 6.7 shows accuracy (in terms of precision) of *UnsupervisedSeg* for address, product and bibliography dataset with noisy test data. Address and product datasets exhibit over 90% accuracy for any kind of noise present in the test data while for bibliography dataset the accuracy is at least 88%. Table 6.6 summarizes experimental results for each kind of noise. These results suggest that *UnsupervisedSeg* works well even with noisy data and are comparable to those shown in [1].

With Varying Training Size

Figure 6.8, 6.9 and 6.10 show the relationship between training set size and performance on address, product and bibliography dataset respectively. 20% of the dataset was set aside for testing. From the remaining 80%, the size of the training data was increased from 25% to 100% in steps of 25%. Observe the longer training times and accuracy improvement with training size increase. The CRFs trained from 80% of data have an F-Measure of 99.1%, 99.3%, and 95.6% with training time of 25 minutes, 2 minutes, and 35 seconds for address, product, and bibliography respectively. Observe that, we can trade training time for high accuracy.

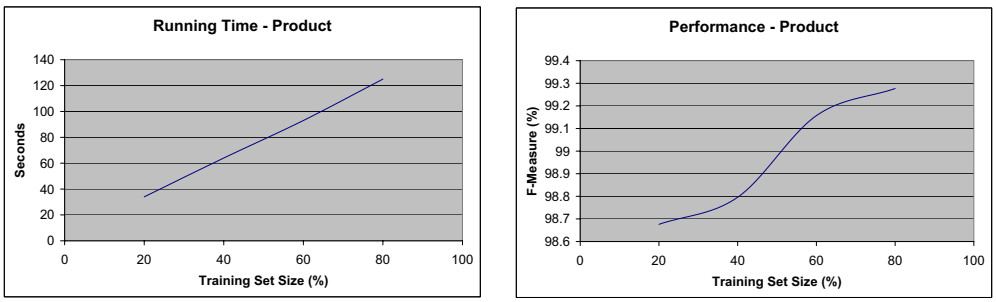
To get a sense of how many training examples are needed to get a reasonable result, observe that 20% of training data amounts to 860, 102, and 16 examples for address, product and bibliography dataset respectively.

For the address dataset, the training time is 6 minutes and the accuracy of the resulting CRF in segmenting the test data is 98.7%. For the product dataset, the training time is 34



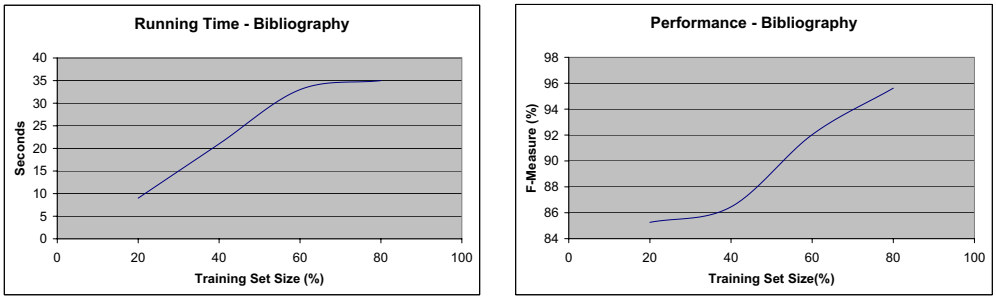
(a) (b)

Figure 6.8: Varying Training Set Size of Address Dataset (a) Running Time (b) F-Measure



(a) (b)

Figure 6.9: Varying Training Set Size of Product Dataset (a) Running Time (b) F-Measure



(a) (b)

Figure 6.10: Varying Training Set Size of Bibliography Dataset (a) Running Time (b) F-Measure

seconds and the accuracy is also 98.7%. For the bibliography dataset, the numbers are 9 seconds and 94.6%. Therefore, we can achieve high accuracy even with a small number of training examples (which are not manually labeled). So these results seem to indicate that

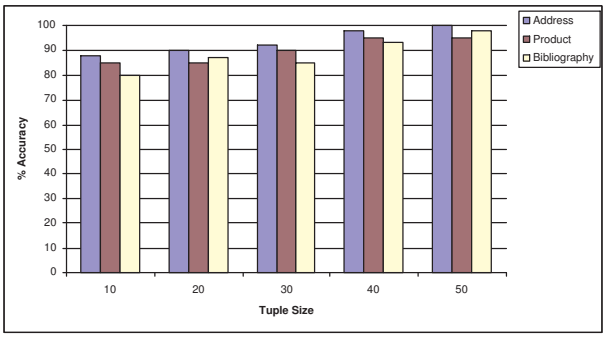


Figure 6.11: Accuracy of InferOrder

training a highly accurate CRF is not necessarily time consuming.

6.3.3.2 Performance of InferOrder

We evaluated the accuracy as defined in [1] [2] of InferOrder, for all three datasets. We took sets of sequences from each dataset. The size of each such set was varied from 10 to 50 tuples. Figure 6.11 shows the accuracy of our algorithm. Note that, our algorithm exhibits high accuracy even for small batch size.

UnsupervisedSeg vs BruteforceSeg

Figure 6.12, 6.13, and 6.14 show their comparative performance. For all of the three datasets, we get higher precision, recall and f-measure values with the *UnsupervisedSeg*.

On the average, its precision, recall and f-measure values are 98.2%, 98.4% and 98.3% respectively. In contrast, *BruteforceSeg* achieves 96.20% precision, 96.3% recall and 96.2% in f-measure.

Observe that, *UnsupervisedSeg* performs better than *BruteforceSeg*. This is because the former can capture dependence between attributes actually present in the input data sequences assuming a attribute total order common to all of them.

[2]Given test sequences with a consistent total order, accuracy is defined as the fraction of attributes whose positions in the order were identified correctly

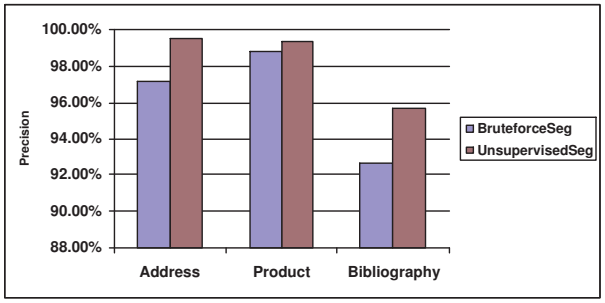


Figure 6.12: Comparative Performance (Precision)

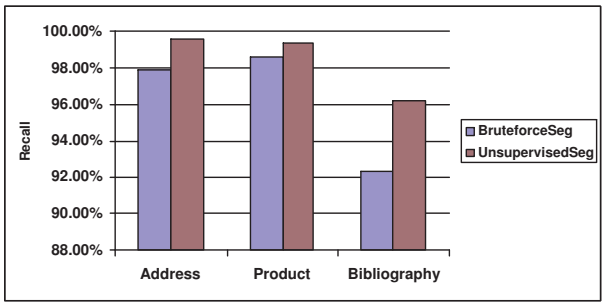


Figure 6.13: Comparative Performance (Recall)

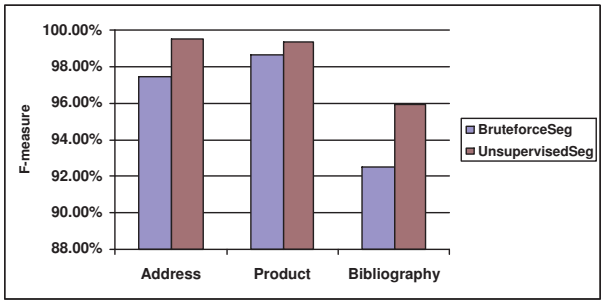


Figure 6.14: Comparative Performance (F-Measure)

6.4 Related Work

The work described in this chapter is broadly related to rule based, and supervised/unsupervised statistical text segmentation research.

Rule-based Approaches

Several rule based text segmentation works have been reported in the research literature (see [2, 63] for example). They are typically based on manually specified rules for identifying attribute instances in text. A limitation of such rule-based systems is that developers need to specify these rules and they may vary for different applications. With the use of reference tables our approach is fully automatic.

Statistical Approaches

Statistical based text segmentation approaches fall into two broad categories – supervised and unsupervised. The characteristic aspect of supervised methods, exemplified by a number of works [13, 41, 14, 10, 34], is that the segmentation models are trained with manually labeled training set. Among the dominant statistical models used for segmentation are HMMs (e.g. the DATAMOLD system [13]), Maximum Entropy Models (e.g. [41]) and Conditional Random Fields (e.g. [34]). A recent work on CRF-based text segmentation [39] focuses on reducing the training data using reference tables. There are two main differences of this work with our approach. First, CRFs with negative labels are used to define additional feature functions in [39]. Manually labeled examples, although fewer in number, are still required. In our case, we use such CRFs to infer a total order of attributes shared by a batch of test sequences, based on which we develop an unsupervised approach for training CRFs from reference tables. Second, the negative examples of our attribute CRFs are subtly constructed (as described in 6.2.2.2) to ensure that incorrect token sub-sequences are given a very low score.

Fully automatic unsupervised approaches to text segmentation is relatively less explored. Recently [1] describe CRAM, a fully automatic text segmentation system based on HMMs. The training data in this work is drawn from structured reference tables, obviating the need for manual labeling. Our unsupervised text segmentation approach based on CRFs coupled with reference tables is inspired by this work.

In their work, reference tables are used to train HMM-based attribute recognition models (ARMs). The transition structures in ARMs require some knowledge of the application domain. So different application domains may require changes to this structure. In CRFs accommodating differences in application domain is accomplished by simply adding or removing features.

But the fundamental difference between CRAM and our approach is in the nature of the underlying statistical model. ARMs are based on HMMs which are generative models. Such models capture probability distributions of observations which are stored in their states. To accommodate observations that were not seen in training CRAM uses a generalized dictionary, which is taxonomy of symbols such as words, numbers, delimiters, etc. Building a generalized dictionary is dependent on the application domain. In contrast CRFs are discriminative models, i.e. they do not capture the distribution of observations and hence dictionaries are not needed.

Chapter 7

Conclusion

Bioinformatics and text segmentation have attracted enormous research efforts in recent years. Classification techniques, especially profile hidden Markov model (PHMM) and conditional random field (CRF), are widely used in these two fields to extract useful information from the vast amount of data resulted from rapid progress in molecular biology and ever increasing World Wide Web activities.

In this dissertation, we have made several important contributions to advance PHMM and CRF based techniques in bioinformatics and text segmentation.

Our bioinformatics research has been performed in the context of Toxin Knowledge Base (TKB), in which protein sequence homology search and protein active site similarity search are among the most important research issues.

For protein sequence homology search, we have developed a parameterized technique for learning PHMMs from partially aligned sequences. Our technique is based upon decomposing a PHMM structure into sub-PHMMs and composing these sub-PHMMs' parameters into those of the PHMM. Our experimental results show improved classification accuracy over existing PHMM approaches when completely aligned training sequences are not available.

For protein active site similarity search, we have developed computational techniques

for statistically profiling active sites in proteins. Specifically we have adapted the successful PHMM based approach for analysis of linear sequences to encode the profiles of 3-D active sites from the same family. Our experience with our implementation indicates that it is effective in practice.

For text segmentation, we have developed a novel technique to remove labeled training for CRF models by exploiting reference tables. Assuming that a batch of input sequences to be segmented conform to the same attribute ordering, we use attribute CRFs trained from individual columns of a reference table to decide the ordering. Labeled training data is then derived from the reference table based on that ordering. A global CRF model is trained from the training data, which is used to segment all sequences in the batch. Our experimental results show that our technique is highly accurate in text segmentation in terms of both precision and recall measures.

In summary, we have developed innovative techniques to exploit partial information in training classifiers for bioinformatics and text segmentation applications. We have shown that these techniques are practical through experimental evaluations. Moreover, we have demonstrated that partial information can be used to greatly improve PHMM classification accuracy, expand PHMM applicability on 3-D structures, and reduce manual efforts required to label training sequences for CRF.

Bibliography

[1] E. Agichtein and V. Ganti. Mining reference tables for automatic text segmentation. In *KDD'04*, pages 20–29, August 2004.

[2] B. Aldelberg. Nodose: A tool for semi-automatically extracting structured and semistructured data from text documents. In *In SIGMOD*, 1998.

[3] V. Alexandrov and M. Gerstein. Using 3d hidden markov models that explicitly represent spatial coordinates to model and compare protein structures. *BMC Bioinformatics*, 5, 2004.

[4] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. A basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[5] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped blast and psi-blast: A new generation of protein database search programs. *Nucleic Acids Research*, 25:3389–3402, 1997.

[6] T. L. Bailey and C. Elkan. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In *Intl. Conf. on Intelligent Systems for Molecular Biology (ISMB)*, pages 28–36, 1994.

[7] A. Bairoch and B. Boeckmann. The swiss-prot protein sequence data bank. *Nucleic Acids Research*, 20:2019–2022, 1992.

[8] J. Barker and J. Thornton. An algorithm for constraint-based structural template matching: application to 3d templates with statistical analysis. *Bioinformatics*, 19:1644–1649, 2003.

[9] L. Baum. An inequality and associated maximization technique in statistical estimation of probabilistic functions of a markov process. *Inequalities*, 3:1–8, 1972.

[10] D. Beeferman, A. Berger, and J. Lafferty. Statistical models for text segmentation. *Machine Learning*, 34(1-3), 1999.

[11] A. Ben-Hur and D. Brutlag. Remote homology detection: a motif based approach. *Bioinformatics*, 19:26–33, 2003.

[12] A. Berger. The improved iterative scaling algorithm: A gentle introduction, 1997.

[13] V. Borkar, K. Deshmukh, and S. Sarawagi. Automatic text segmentation for extracting structured records. In *ACM SIGMOD International Conf. on Management of Data*, 2001.

[14] M. E. Califf and R. J. Mooney. Relational learning of pattern-match rules for information extraction. In *Sixteenth National Conference on Artificial Intelligence*, 1999.

[15] W. W. Cohen and S. Sarawagi. Exploiting dictionaries in named entity extraction: Combining semi-markov extraction processes and data integration methods. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, USA*, 2004.

[16] J. Darroch and D. Ratchliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43:1470–1480, 1972.

[17] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J. Roy. Stat. Soc.*, 39:1–38, 1977.

[18] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

[19] S. Eddy, G. Mitchison, and R. Durbin. Maximum discrimination hidden markov models of sequence consensus. *Journal of Comput. Biol.*, 3:9–23, 1995.

[20] S. R. Eddy. Profile hidden markov models. *Bioinformatics*, 14:755–763, 1998.

[21] L. Falquet, M. Pagni, P. Bucher, N. Hulo, C. Sigrist, K. Hofmann, and A. Bairoch. The prosite database, its status in 2002. *Nucleic Acids Research*, 30:235–238, 2002.

[22] W. Grundy, T. Bailey, C. Elkan, and M. Baker. Meta-meme: Motif-based hidden markov models of protein families. *Computer Applications in Biosciences*, 13(4):397–406, 1997.

[23] W. N. Grundy. Homology detection via family pairwise search. *Journal of Computational Biology*, 5(3), 1998.

[24] J. Henikoff, S. Pietrokovski, C. McCallum, and S. Henikoff. Blocks-based methods for detecting protein homology. *Electrophoresis*, 21:1700–1706, 2000.

[25] S. Henikoff, J. Henikoff, and S. Pletrokovski. Blocks+: a non-redundant database of protein alignment blocks derived from multiple compilations. *Bioinformatics*, 15:471–479, 1999.

[26] S. Henikoff and J. G. Henikoff. Automated assembly of protein blocks for database searching. *Nucleic Acids Research*, 19:6565–6572, 1991.

[27] D. Higgins, J. Thompson, and T. Gibson. Clustalw: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific

gap penalites and weight matrix choice. *Nucleic Acids Research*, 22:4673–4680, 1994.

[28] R. Hughey and A. Krogh. Hidden markov models for sequence analysis: Extension and analysis of the basic method. *Comput. Appl. Biosci.*, 12:95–107, 1996.

[29] T. Jaakkola, M. Diekhans, and D. Haussler. Using fisher kernel method to detect remote protein homologies. In *Intl. Conf. on Intelligent Systems for Molecular Biology (ISMB)*, 1999.

[30] K. Karplus, C. Barrett, and R. Hugher. Hidden markov models for detecting remote protein homologies. *Bioinformatics*, 14:846–856, 1999.

[31] M. Kifer, I. Ramakrishnan, A. Ramanathan, C. Zhao, S. Jayaraman, and S. Swaminathan. Tkb: Toxin knowledge base for discovering bio-engineered threats. In *ISMB 2005*, 2005. Tool Demo and Poster.

[32] G. J. Kleywegt. Recognition of spatial motifs in protein structures. *Journal of Molecular Biology*, 285:1887–1897, 1999.

[33] D. Korkin, F. Davis, and A. Sali. localization of protein-binding sites within families of proteins. *Protein Science*, 14:2350–2360, 2005.

[34] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001.

[35] M. Lapata. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of annual meeting of ACL*, 2003.

[36] A. Laurie and R. Jackson. Q-sitefinder: an energy-based method for the prediction of protein-ligand binding sites. *Bioinformatics*, 21(9):1908–1916, 2005.

[37] J. Li, A. Najmi, and R. M. Gray. Image classification by a two-dimensional hidden markov model. *IEEE Transactions on Signal Processing*, 48(2):517–533, 2000.

[38] W. Li and A. McCallum. Rapid development of hindi named entity recognition using conditional random fields and feature induction. In *ACM Transactions on Asian Language Information Processing*, volume 2, pages 290–294, 2003.

[39] I. R. Mansuri and S. Sarawagi. Integrating unstructured data into relational databases. In *ICDE*, page 29, 2006.

[40] W. Markus and E. L. L. Sonnhammer. Improving profile hmm discrimination by adapting transition probabilities. *Journal of Molecular Biology*, 338:847–854, 2004.

[41] A. McCallum, D. Freitag, and F. C. N. Pereira. Maximum entropy markov models for information extraction and segmentation. In *ICML*, pages 591–598, 2000.

[42] A. McCallum and W. Li. Early results for named entity recognition with conditional random fields. In *Proceedings of the seventh conference on Natural Language Learning (CoNLL-2003)*, 2003.

[43] F. Melo and E. Feytmans. Novel knowledge-based mean force potential at atomic level. *J. Mol. Biol.*, 267:207–222, 1997.

[44] S. Mukherjee, C. Zhao, and I. Ramakrishnan. Profiling protein families from partially aligned sequences. In *SIAM Conference on Data Mining*, 2006.

[45] A. Murzin, S. Brenner, T. Hubbard, and C. Chothia. Scop: A structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247:536–540, 1995.

[46] S. Needleman and C. Wunsch. A general method applicable to the search for similarities in the amino acid sequences of two proteins. *Journal of Molecular Biology*, 48:444–453, 1970.

[47] K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification. In *IJCAI-99 Workshop on Machine Learning for Information Filtering*, 1999.

[48] C. Notredame, D. Higgins, and J. Heringa. T-coffee: A novel method for fast and accurate multiple sequence alignment. *Journal of Molecular Biology*, 302:205–217, 2000.

[49] J. Park, K. Karplus, C. Barrett, R. Hugher, D. Haussler, T. Hubbard, and C. Chothia. Sequence comparisons using multiple sequences detect three times as many remote homologues as pairwise methods. *J. Mol. Biol.*, 284:1201–1210, 1998.

[50] W. Pearson. Rapid and sensitive sequence comparison with fastp and fasta. *Methods in Enzymology*, 183:63–98, 1985.

[51] S. D. Pietra, V. D. Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:380–393.

[52] D. Pinto, A. McCallum, X. Lee, and W. Croft. Table extraction using conditional random fields. In *Proceedings of 26th ACM SIGIR*, 2003.

[53] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of IEEE*, 77(2), 1989.

[54] S. Raychaudhuri, J. Chang, P. Sutphin, and R. Altman. Associating genes with gene ontology codes using a maximum entropy analysis of biomedical literature. *Genome Research*, 12:203–214, 2002.

[55] R. Russell. Detection of protein three-dimensional side-chain patterns: new examples of convergent evolution. *Molecular Biology*, 279:1211–1227, 1998.

[56] S. safavian and D. Landgrebe. A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man and Cybernetics*, 21:660–674, 2002.

[57] A. Sali and T. L. Blundell. Comparative protein modelling by satisfaction of spatial restraints. *Journal of Molecular Biology*, 234:779–815, 1993.

[58] K. Seymore, A. McCallum, and R. Rosenfeld. Learning hidden markov model structure for information extraction. In *In Papers from the AAAI-99 Workshop on Machine Learning for Information Extraction*, 1999.

[59] F. Sha and F. C. N. Pereira. Shallow parsing with conditional random fields. In *HLT-NAACL*, 2003.

[60] M. Shatsky, A. Shulman-Peleg, R. Nussinov, and H. Wolfson. recognition of binding patterns common to a set of protein structures. In *RECOMB*, pages 440–455, 2005.

[61] K. Sjölander, K. Karplus, M. Brown, R. Hughey, A. Krogh, I. S. Mian, and D. Haussler. Dirichlet mixtures: a method for improving detection of weak but significant protein sequence homology. *Comput. Appl. Biosci.*, 12:327–345, 1996.

[62] T. Smith and M. Waterman. Identifying of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[63] S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3), 1999.

[64] A. Stark, S. Sunyaev, and R. Russell. A model for statistical significance of local similarities in structure. *Molecular Biology*, 326:1307–1316, 2003.

[65] C. Sutton, K. Rohanimanesh, and A. McCallum. Dynamic conditional random fields: factorized probabilistic models for labeling and segmenting sequence data. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 99, 2004.

[66] J. Torrance, G. Bartlett, C. Porter, and J. Thornton. Using a library of structural templates to recognize catalytic sites and explore their evolution in homologous families. *J. Mol. Biol.*, 347:565–581, 2005.

[67] A. Wallace, N. Borkakoti, and J. Thornton. Tess: A geometric hashing algorithm for deriving 3d coordinate templates for searching structural databases. application to enzyme active sites. *Protein Science*, 6:2308–2323, 1997.

[68] J. Wallace and S. Henikoff. Patmat: A searching and extraction program for sequence, pattern, and block queries and databases. *CABIOS*, 8:249–254, 1992.

[69] M. Yousef, M. Nebozhyn, H. Shatkay, S. Kanterakis, L. Showe, and M. Showe. Combining multi-species genommic data for micorrna identification using a naive bayes classifier. *Bioinformatics*, 22:1325–1334, 2006.

[70] C. Zhao, J. Mahmud, and I. Ramakrishnan. Exploiting structured reference data for unsupervised text segmentaion with conditional random fields. In *SIAM Conference on Data Mining*, 2008.

[71] C. Zhao, J. Mahmud, I. Ramakrishnan, and S. Swaminathan. Computing statistical profiles of active sites in proteins. In *SIAM Conference on Data Mining*, 2007.

[72] http://www.it.iitb.ac.in/˜ sunita/data/personalBib.tar.gz.

[73] http://crf.sourceforge.net/.

[74] http://www.ebi.ac.uk/thornton-srv/databases/CSA/.

[75] http://bioinfo3d.cs.tau.ac.il/MultiBind/.

[76] http://www.ncbi.nlm.nih.gov.

[77] http://pauling.mbu.iisc.ernet.in/ pali.

[78] http://www.rcsb.org/pdb.

[79] http://pfam.cgb.ki.se/help/scores.html.

[80] http://www.biochem.ucl.ac.uk/bsm/PROCAT/PROCAT.html.

[81] http://www.isi.edu/info-agents/RISE/.

[82] http://www.chemcomp.com/journal/sitefind.htm.

[83] http://alpha2.bmc.uu.se/usf/spasm.html.

[84] http://www2.all.cs.sunysb.edu.

[85] http://www.expasy.org.