Posterior Decoding Methods for Optimization and Accuracy Control of Multiple Alignments

Ariel Shaul Schwartz

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2007-39
http://www.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-39.html

March 28, 2007

Posterior Decoding Methods for Optimization and Accuracy Control of Multiple Alignments

by

Ariel Shaul Schwartz

B.Sc. (Technion, Israel) 1999

A dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Lior Pachter, Chair
Professor Dan Klein
Professor Marti Hearst

Spring 2007

The dissertation of Ariel Shaul Schwartz is approved.

Chair Date

Date

Date

University of California, Berkeley

Spring 2007

Posterior Decoding Methods for Optimization and

Accuracy Control of Multiple Alignments

by

Ariel Shaul Schwartz

Abstract

Posterior Decoding Methods for Optimization and
Accuracy Control of Multiple Alignments

by

Ariel Shaul Schwartz

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Lior Pachter, Chair

Comparative genomics analyses, such as genome annotation, phylogenetic inference, protein structure prediction and protein function prediction, rely heavily on multiple sequence alignment (MSA), which is essentially a homology prediction problem at the character level over multiple sequences. Alignments are also crucial in other settings, including natural language processing, as seen in word alignment in machine translation, or in the problem of citation entity alignment that is introduced in this dissertation. Fundamental as it is, MSA is a hard computational problem, and producing accurate MSAs remains challenging.

In this thesis we develop new posterior decoding based methods for multiple alignments of discrete characters. We begin by discussing biological sequence alignment, and describe a new alignment accuracy measure (AMA), which is based on a metric for the space of alignments. In the case of pairwise sequence alignment, it is possible to find an alignment that maximizes the expected AMA value using posterior decoding. We then describe an extension of the pairwise alignment method to MSA using

a novel approach that is based on a poset representation of multiple alignments, and an efficient online topological ordering algorithm. This leads to a *sequence annealing* algorithm, which is an incremental method for building MSAs one match at a time. The sequence annealing algorithm outperforms all existing algorithms on benchmark test sets of protein sequence alignments. It simultaneously achieves high recall and high precision, dramatically reducing the number of incorrectly aligned residues in comparison to other programs. The method can adjust the recall/precision trade off, and can reliably identify homologous regions among protein sequences. Finally, we demonstrate that similar posterior decoding methods can be applied to the problem of multiple word alignment of citation sentences (*citances*) in the bioscience literature. We extend the definition of AMA to many-to-many word alignments, and develop a posterior decoding algorithm for multiple citance alignment.

Professor Lior Pachter
Dissertation Committee Chair

To Abba.

Contents

Contents ii

List of Figures iv

List of Tables vii

Acknowledgments x

1 Introduction 1

2 Hidden Markov Models and posterior decoding 8

2.1 Hidden Markov Models 8

2.2 HMM inference algorithms 10

2.3 State and transition posterior decoding 12

2.4 Multiple sequence HMMs 15

2.4.1 The alignment problem 15

2.4.2 Multiple sequences 16

2.5 Discussion 18

3 Alignment Metric Accuracy 19

3.1 Metric based alignment accuracy 23

3.2 AMA based alignments 28

3.2.1 Maximal expected accuracy alignments 28

3.2.2 The AMAP algorithm 29

3.3 Results 33

3.3.1 Performance of existing programs on the SABmark datasets . 33

3.3.2 Controls for multiple alignment experiments 35
3.3.3 Performance of the AMAP algorithm 44
3.4 Discussion 48

4 Multiple Alignment by Sequence Annealing 50
4.1 Alignment posets 53
4.2 Sequence annealing 56
4.3 Results 65
4.4 Summary and Future Directions 72

5 Multiple Alignment of Citation Sentences 74
5.1 Citances 75
5.2 Multiple citance alignments 79
5.3 MCA utility and loss functions 82
5.4 Probabilistic model for MCA 92
5.4.1 Conditional random fields for word alignment 93
5.4.2 The posterior decoding algorithm for MCA 95
5.5 Data sets 97
5.6 Feature engineering 98
5.7 Results 102
5.8 Discussion 106

6 Conclusions 108

Bibliography 110

A Proof that the triangle-inequality holds for the Braun-Blanquet coefficient 116

List of Figures

3.1 **Example of a global pairwise alignment of DNA sequences.** . 21

3.2 **Example of a reference alignment and a predicted alignment of two protein sequences.** 28

3.3 **Example of four different alignments of two unrelated protein sequences.** Light characters represent correctly aligned positions, and dark characters represent incorrectly aligned positions. Since the two sequences are unrelated all aligned positions are wrongly aligned, and all gapped positions are correctly aligned. The fraction of characters that are correctly aligned (light) represent an intuitive notion of accuracy. 30

3.4 **Correlation between the AMA of CLUSTALW (as judged by reference alignments in SABmark), and average similarity to alignment produced by other programs.** Each dot in the plot corresponds to one CLUSTALW alignment in the SABmark Twilight-FP and Superfamilies-FP datasets. The x coordinate represents the average similarity (s) of the CLUSTALW alignment to the alignments produced by three other programs (MUSCLE, ProbCons, T-Coffee). The y coordinate represents the Alignment Metric Accuracy (AMA) of the CLUSTALW alignment as judged by the reference SABmark alignment. 37

3.5 **Comparison of different rankings of the Superfamilies dataset alignments.** The accuracy of the metric-based ranking [$AMA(top(\mathcal{M}_{\mathcal{G}}^{sim}, \psi^{sim}, u))$; blue] is compared to that of an optimal ranking [$AMA(top(\mathcal{M}_{\mathcal{G}}^{sim}, \psi^{R}, u))$; red], and a random ranking of the alignments that have been picked by the first part of the alignment-control protocol [$AMA(top(\mathcal{M}_{\mathcal{G}}^{sim}, \psi^{rand}, u))$; purple]. In addition an optimal ranking of the best original alignments is shown [$AMA(top(\mathcal{M}_{\mathcal{G}}^{max}, \psi^{R}, u))$; green], as well as the similarity threshold used at each filtration level [ψ^{sim}; orange]. 42

4.1 **Example of progressive alignment.** (a) A guide tree is constructed based on the sequence similarity of the four sequences. (b) The two most similar sequences are aligned. (c) A second pair of sequences are aligned according to the guide tree. (d) The two alignments are aligned together to construct the final MSA. 52

4.2 **Alignment posets.** (a) A set of four sequences, an alignment poset together with a linear extension, and a global multiple alignment. The function from the set of sequence elements to the alignment poset that specifies the multiple alignment is not shown, but is fully specified by the diagram on the right. For example, the second element in the first sequence is $\sigma_2^1 = G$, and $\varphi(\sigma_2^1)$ corresponds to the fourth column of the multiple alignment. (b) A different linear extension for the same alignment poset results in a different (but equivalent) global multiple alignment. (c) The null alignment of the same four sequences. 54

4.3 **A general sequence annealing algorithm.** 57

4.4 **Online topological ordering and alignment posets.** (a) During the sequence annealing process, the next candidate *merge* operation of columns 5 and 8 is drawn from the heap. (b) The new edge is added to the poset, and the online topological ordering procedure updates the current linear extension. (c) Since no cycle are found during the online topological ordering procedure the two columns are merged. (d) The next candidate edge connects columns 2 and 7. However, the online topological ordering procedure locates a cycle, and the edge is discarded. 59

4.5 **The first step of the sequence annealing algorithm with weight function w_{tgf}^1 on four protein sequences.** (a) Starting from the null alignment the first candidate column pair with weight of ≈ 10000 is fetched from the heap. (b) The two columns are merged, adding $\delta \approx 2$ to the scoring function. Affected columns in the global multiple alignment are rearranged based on the new liner extension. 63

4.6 **Later steps in the sequence annealing of four protein sequences with weight function w_{tgf}^1.** (a) At temperature $\gamma \approx 52$ different pairs of sequences are aligned at different parts of the multiple alignment. (b) At temperature $\gamma \approx 2.5$ all fully conserved columns are aligned. (c) The final alignment. The sequence annealing process stops at temperature $\gamma \approx 1.04$ with total scoring function improvement of 959.55 over the null alignment. Next candidate column pair has weight < 1, which can only decrease the scoring function. 64

4.7 **Comparison of the recall/precision trade-off of different alignment programs with AMAP on the SABmark datasets with no false-positives** (a) Results averaged over alignments. (b) Results averaged over all positions. 69

4.8 **Comparison of the recall/precision trade-off of different alignment programs with AMAP on the SABmark datasets with false-positives** (a) Results averaged over alignments. (b) Results averaged over all positions. 70

5.1 **Example of three unaligned citances.** 81

5.2 **Example of three normalized aligned citances.** Homologous entities are colored the same. Unaligned entities are black. 81

5.3 **Calculation of different set-agreement functions on a pairwise citance alignment.** A predicted alignment of normalized citances (filled squares) is compared to a reference alignment of the same citances (empty squares). For each word the $U_{set\_agreement}$ scores is calculated using six different set-agreement coefficients. 90

5.4 **Recall/Precision curve of pairwise citance alignments comparing Viterbi to posterior decoding.** 103

5.5 **Recall/Precision curve of MCAs comparing CRF with posterior decoding to normalized-edit-distance baseline.** 105

List of Tables

3.1 **Performance of aligners on the SABmark benchmark datasets.** Entries show the average developer (f_D), modeler (f_M) and alignment metric accuracy (AMA). Best results are shown in bold. All numbers have been multiplied by 100. 34

3.2 **Total distance (d) and average similarity (s) of different aligners on the SABmark dataset.** Values below the diagonal show the total distance (d) between alignments produced by different alignment programs. Values above the diagonal show the average similarity (s) between the different alignments. Distance values have been divided by one million, and similarity values have been multiplied by 100. . . 35

3.3 **AMA of alignments picked by the control protocol on the SABmark datasets.** The first four rows show the average AMA of alignments produced by different alignment programs. The last three rows show the average AMA of all alignments programs (Average), of the best alignments for every group (Best), and of the alignments picked by the control protocol (Picked). AMA values have been multiplied by 100. The Wilcoxon signed ranks test was used to measure the statistical significance of the difference between the picked alignments and the alignments produced by the four programs (P-values are shown in parenthesis. Values > 0.05 are not statistically significant). 40

3.4 **Statistical evaluation of the metric-based ranking protocol on the SABmark datasets.** For every dataset the number of groups (L), the Pearson correlation (r) and the Spearman rank correlation (ρ) are given. The following measures are calculated at different filtration levels, as well as the means over all levels; the accuracy (average AMA multiplied by a 100) of the metric-based and optimal rankings, the percent improvement in AMA of the two rankings compared to the full set, and the ratio between the percent improvement of the metric-based ranking compared to the optimal ranking (improvement ratio). 43

3.5 **Performance of algorithm variants on the SABmark Twilight Zone set.** Entries show the f_D, f_M, and AMA scores of the Viterbi, and AMAP alignments with different gap-factor (γ) values on the SABmark Twilight Zone set, which includes 209 alignment groups. The first three columns show the results using default transition probabilities, and the last three columns show the results using transition probabilities calculated for each group from the reference alignments. All scores have been averaged over groups and multiplied by 100. 45

3.6 **Performance of algorithm variants on the SABmark Superfamilies set.** Entries show the f_D, f_M, and AMA scores of the Viterbi, and AMAP alignments with different gap-factor (γ) values on the SABmark Superfamilies set, which includes 425 alignment groups. The first three columns show the results using default transition probabilities, and the last three columns show the results using transition probabilities calculated for each group from the reference alignments. All scores have been averaged over groups and multiplied by 100. 45

3.7 **Performance of algorithm variants on simulated data.** Entries show the performance of the Viterbi algorithm (Vit), and the AMAP algorithm with different settings of the gap-factor parameter (0, 1, and 2) using three accuracy measures (f_D, f_M, and AMA). The first three columns show the configuration of the pair-HMM parameters e_{match} (match emission probability), δ (gap initiation probability) and ε (gap extension probability), except for the last row for which random unrelated sequences have been aligned. Best results for every parameter configuration and measure are shown in bold. All numbers have been multiplied by 100. 46

4.1 **Comparison of protein alignment programs on the SABmark datasets with no false positives.** Entries show the average developer (f_D), modeler (f_M) and alignment metric accuracy (AMA) scores. Best results are shown in bold. All numbers have been multiplied by 100. 67

4.2 **Comparison of protein alignment programs on the SABmark datasets with false positives.** Entries show the average developer (f_D), modeler (f_M) and alignment metric accuracy (AMA) scores. Best results are shown in bold. All numbers have been multiplied by 100. 68

5.1 **Comparison of different utility function as calculated on the example in Figure 5.3.** The first line shows the values of the six different coefficients, in addition to recall and precision when using pairs of indices as the basic set elements. The second line shows the word-based values of the coefficients when averaged over all word in Figure 5.3. 90

Acknowledgements

First, I would like to thank my advisors Marti Hearst and Lior Pachter. Marti introduced me to the field of computational biology through our work on an abbreviation recognition algorithm for bioscience text, which is still being used by many people. She has been a wonderful advisor, and supported me for most of my graduate studies. We worked together on many successful text mining problems as part of the BioText project. I thank Marti for encouraging me to explore other areas of computational biology. Her many useful comments helped improve the quality of this dissertation. This dissertation would not have been possible without Lior. He has kindly agreed to take Gene Myers' place as my co-advisor when Gene left Berkeley. Since then he has been a constant inspiration, and his continuous guidance and support lead to most of the research in this dissertation. In particular I thank Lior for always believing in me, and pushing me to fulfill my potential.

I thank Dan Klein for serving on my dissertation committee, and for his useful comments that helped improve my work. Thanks to Gene Myers' for introducing me to the field of comparative sequence analysis. Our earlier work on comparative gene finding led later to the posterior decoding algorithms for alignments that are presented here. I thank Bob Edgar and Michael Brudno for many enlightening discussions during Gene's group meetings, in one of which I first learned about posterior decoding methods.

I thank the members of the BioText group. Special thanks are due to Preslav (Presley) Nakov for many fruitful collaborations throughout the years. I thank Anna Divoli for helping with the citance annotations, and for bringing her positive energy to our group. Many thanks to Gaurav Bhalotia for many collaborations during my first years at graduate school, but more importantly for being a true friend.

I thank Phil Blunsom and Trevor Cohn for sharing their CRF-based alignment

program with me, and Chuong (Tom) Do for making is alignment program ProbCons available as open source.

I thank Mani Narayanan, Sourav Chatterji, Colin Dewey, Anat Caspi, Nick Bray, and Sagi Snir for many useful discussion on alignments and other aspects of comparative genomics.

I thank Hanoch for staying such a good friend in spite of the long geographical distance, and my brother Haggai and his wife Michal for always being there for me. Many thanks are due to my mother, who not only raised me to be who I am, but has always kept helping out and supporting me. Most importantly, I thank my wife and best friend Reut for always believing in me, and for joining me on this long journey. Lastly, I thank my two wonderful children for being a true source of joy, and for always helping to remind me what is most important in life.

Chapter 1

Introduction

The alignment problem is to identify related characters among multiple sequences, where the relationship may be evolutionary, semantic, or otherwise determined by the origin of the sequence. In this work we study alignment for sequences of discrete characters, motivated by the problem of identifying evolutionary related portions of biological sequences, and semantically related entities of scientific text. By taking a unified view of the alignment problem, we find that a common set of tools are useful for these two distinct alignment applications. Our methods are therefore applicable to a wide range of pairwise and multiple alignment problems that arise in different fields of computer science.

Alignment is a fundamental tool in comparative analysis of structured sequential data. In particular, pairwise and multiple alignments are a core component of many algorithms and systems in different fields of computer science such as comparative genomics and statistical machine translation. A standard approach for analyzing multiple sequences in such fields is to first align the sequences, and then use the alignment as an input to a higher level analysis tool, such as a program for comparative

functional annotation of genomics sequences, a phylogenetic analysis algorithm, or a statistical machine translation program.

While the input alignments provide only an estimate of the true alignments and can contain errors, in most cases such errors are not modeled in the higher level systems, and the input alignments are treated as essentially correct. The performance of comparative analysis systems can be very sensitive to the quality of their input alignments. It is therefore essential to optimize alignment quality, as well as control and predict its accuracy. In this work we describe posterior decoding based methods for direct optimization and accuracy control of multiple alignments of biological sequences, and citation sentences.

Improving the accuracy of alignments requires carefully defined accuracy measures. We first review the standard performance measures for binary classification tasks. A binary classifier predicts, for every entity, if it belongs to the *positive* or *negative* class. For example, a spam email detection program should classify incoming spam email messages as positives, and legitimate messages as negatives.

Given a set of input entities and a reference classification, the set of predicted classifications can be partitioned into the following four subsets. *True positives* (TP) are the entities that are correctly classified as positives (spam classified as spam). *True negatives* (TN) are the entities that are correctly classified as negatives (non-spam classified as non-spam). *False positives* (FP) are the entities that are incorrectly classified as positives (non-spam classified as spam). *False negatives* (FN) are the entities that are incorrectly classified as negatives (spam classified as non-spam). Note that the positive class of the reference classification includes TP and FN, and the negative class includes TN and FP, while the positive class of the predicted classification includes TP and FP, and the negative class includes TN and FN.

The following performance measures are defined using the above sets. *Sensitivity*

or *recall* is defined as TP / (TP + FN), and is the proportion of correctly classified entities out of the reference's positive class (the likelihood that a spam message is classified as spam). *Positive prediction value* (PPV) or *precision* is defined as TP / (TP + FP), and is the proportion of correctly classified entities out of the predicted positive class (the likelihood that a message classified as spam is in fact spam). *Specificity* is defined as TN / (TN + FP), and is the proportion of correctly classified entities out of the reference's negative class. (the likelihood that a non-spam message is classified as non-spam). *Negative prediction value* (NPV) is defined as TN / (TN + FN), and is measures the proportion of correctly classified entities out of the predicted negative class (the likelihood that a message classified as non-spam is in fact non-spam). *Accuracy* is defined as (TP + TN) / (TP + TN + FP + FP), and is the proportion of correctly classified entities out of all the entities (the likelihood that a message is classified correctly as spam or non-spam). Since specificity is sometimes confused with PPV we use the terms recall and precision rather than sensitivity and PPV throughout this paper in order to avoid ambiguity.

There is an inherent trade-off between some of the different performance measures. In particular, such trade-off exists between recall and precision, since predicting more entities as positives can increase recall, but is likely to lead to lower precision, and vice versa. In our spam detection example, tuning a spam-detection program to increase recall might reduce precision since more non-spam emails are likely to be classified as spam, while increasing precision might hurt recall since more spam message are likely to be classified as non-spam. In general, it is useful to be able to control such a trade-off. While for some applications a higher recall is preferred, other applications might require better precision, and others might need a balance between the two. For example, a spam detection program should typically have very high precision, since it is better to mis-classify some spam messages as non-spam rather than classify (potentially important) messages as spam. In other words, in this case the *loss*

associated with a FP error is much greater than the loss from a FN error. The loss associated with each error can change with the task, and the preference of the user. For example, in document retrieval tasks it is sometimes more important to achieve high recall when the number of relevant documents is small.

The standard performance measures for binary classification cannot be applied directly to more complex learning problems such as alignments, since predicting alignments is not a binary classification task, and the standard positive and negative classes are not well defined. One solution is to use a 0–1 loss function, which assess a loss of one unit for every alignment that is not predicted to be exactly the same as the reference alignment. The problem with such approach is that in most practical cases almost any program is very unlikely to predict the exact same alignment as the reference alignment. While some predicted alignments can be much more similar to the reference alignment than others, the 0–1 loss function does not distinguish between them. Another alternative is to treat the alignment problem as a set of many smaller binary classification tasks and then use standard performance measures such as precision and recall. This is typically done by treating every potential pair of positions as a single entity. The recall of a predicted alignment is then defined as the proportion of pairs that are correctly aligned in the predicted alignment out of all the pairs that are aligned in the reference alignment. The precision of a predicted alignment is the proportion of pairs that are correctly aligned in the the predicted alignment out of all the aligned pairs in that alignment.

While defining recall and precision of alignments is useful, there is a need for a single accuracy measure that can give a balanced assessment of recall and precision. In Chapter 3 we define a new accuracy measure for sequence alignment that is based on a metric for the space of alignments. We term this measure *alignment metric accuracy* (AMA) and show that it has several noted advantages over current alignment performance measures. We also demonstrate that since AMA is based on a metric it

can be used to predict the accuracy of sequence alignments even when the reference alignment is unknown. In Chapter 5 we extend the definition of AMA to multiple alignment of citation sentences.

Viterbi decoding is the most widely used inference method in machine learning. It finds the most likely explanation for the observed data given a probabilistic model, which is equivalent to maximizing the expected 0–1 loss. While the 0–1 loss is typically not used to evaluate the performance of alignment algorithms for the reasons stated above, most alignment algorithms use Viterbi decoding (or its non-probabilistic equivalents) to produce their predicted alignments. This is not necessarily the best strategy when there are many alternative alignments with similar probability. An alternative is to find the alignment that maximizes the expected accuracy. This can be done using posterior decoding methods instead of Viterbi. In Chapter 2 we introduce the concept of *posterior decoding* and show how it can be applied on the most common probabilistic model for sequential data—Hidden Markov Models (HMMs). In Chapter 3 we show how posterior decoding can be applied to the problem of pairwise sequence alignment. In particular we show how given a probabilistic model for pairwise alignment, the alignment with maximal expected AMA score can be found. Additionally, we show that posterior decoding can enable direct control of the precision/recall trade-off using a single parameter. We demonstrate on simulated data and benchmark datasets that the posterior decoding algorithm produces more accurate alignments than Viterbi, and it can much better handle unalignable sequences. In addition, we demonstrate the unique feature of our posterior decoding approach for control of the recall/precision trade-off.

Multiple sequence alignment is a hard problem that cannot be solved directly. Given a scoring function for multiple alignments, alignment algorithms utilize heuristics for searching the space of alignments for a solution that maximizes the value of that function. The most common heuristic is progressive alignment, which builds the

alignment along a guide tree. This approach has several limitations. In particular, it tends to be very sensitive to errors that are introduces in the first step of the search procedure. Posterior decoding provides an infrastructure for a much more refined exploration of the alignment space. In Chapter 4 we describe a new multiple sequence alignment algorithm, which we call *sequence annealing*. Sequence annealing builds multiple alignments using minimal steps in the alignment space. Unlike progressive alignment, sequence annealing allows to first align positions that are more likely to correctly align. It produces a range of alignments, from high precision to high recall alignments. Sequence annealing is implemented efficiently using a poset representation of multiple alignment, and an online topological ordering algorithm. Results on benchmark datasets demonstrate that sequence annealing performs better than existing state of the art alignment algorithms, and show how the recall/precision trade-off can be controlled.

In Chapter 5 we extend the posterior decoding methods we have developed for biological sequence alignment to a new problem—multiple alignment of citation sentences. Citation sentences (citances) are a rich and relatively unexplored resource of comparative data for text mining. Much like sequence analysis, higher level systems will need an aligned input in order to utilize the citance resource. While sequence alignment is a one-to-one alignment, citance alignment, is a many-to-many alignment. We show how AMA can be extended to support many-to-many alignments, and show how other alternative utility functions compare with AMA. In the case of citance alignment two parameters are required to control the recall/precision trade-off. We develop a posterior decoding algorithm for multiple citance alignment, and demonstrate its performance on a hand curated dataset.

The main contributions of this work are summarized in Chapter 6. In particular, we show that the general framework that combines a careful definition of accuracy together with posterior decoding methods for optimization and control of the expected

accuracy can be applied to other problems related to sequence and word alignment. Additionally, we discuss several open research directions for extending the current work.

Chapter 2

Hidden Markov Models and posterior decoding

This chapter demonstrates the concept of *posterior decoding* on *Hidden Markov Model* (HMM), which is the most widely used probabilistic model for structured sequential data. We first compare several decoding methods for single sequence HMMs, and then describe multiple sequence HMMs and the alignment problem that is associated with them.

2.1 Hidden Markov Models

Probabilistic models of structured sequential data, such as biological sequences and sentences, need to model the dependencies between the labeling of different positions. For many applications modeling such dependencies using a Markov assumption is an acceptable approximation to the true dependencies. Hidden Markov Models (HMMs) were first introduced by Baum and Petrie (1966), and later became a very popular model for problems in speech recognition, biological sequences anal-

ysis, natural language processing, and almost any other field that requires modeling sequential data (see Rabiner (1989) for a comprehensive review of HMMs). HMMs and their various extensions model dependencies between contiguous positions with a relatively simple model, but are not well suited for modeling long-distance effects, such as symmetry effects in RNA folding problems. Despite their limitations HMMs are probably the most commonly used generative models for sequential data.

Let $\sigma \triangleq \sigma_1\sigma_2\cdots\sigma_n$ be an observed string of length n. Following the notation of Durbin *et al.* (1998) we define $\pi \triangleq \pi_1\pi_2\cdots\pi_n$ to be a sequence of hidden states of length n or a *path*. The HMM model is parameterized by the transition probabilities a_{kl}, and the emission probabilities $e_k(b)$

$$a_{kl} = P(\pi_i = l|\pi_{i-1} = k) \tag{2.1}$$

$$e_k(b) = P(\sigma_i = b|\pi_i = k). \tag{2.2}$$

The joint probability of an observed string σ and a hidden path π is

$$P(\sigma, \pi) = a_{0\pi_1} \prod_{i=1}^{n} e_{\pi_i}(\sigma_i) a_{\pi_i\pi_{i+1}}, \tag{2.3}$$

where 0 is used to label both the begin and end states, and $\pi_{n+1} = 0$.

There are many extensions to the basic HMM model, of which we mention two. The basic HMM model is also called 1^{st} order HMM, because the hidden states form a 1^{st} order Markov chain. In an n^{th} order HMM each state and emitted character depend on the previous n states. An n^{th} order HMM with K states can be represented by a 1^{st} order HMM with K^n states.

In many applications it is desirable to model the probability of staying in the same state for a certain duration (length). In 1^{st} order HMMs self transition probabilities can be used, but they can only model geometric length distributions. When a geometric distribution is inappropriate, more complex length distributions can be modeled by introducing additional states with identical labels (Durbin *et al.*, 1998).

Generalized HMMs model length distributions explicitly, in the expense of increased time and space complexity.

Our work is concerned mainly with inference (or decoding) techniques, and not much with parameter estimation. We therefore only briefly mention the standard HMM parameter estimation techniques, and refer the reader to Durbin *et al.* (1998) for their detailed descriptions. When labeled training data is available the HMM parameters can be learned directly using *maximum likelihood estimation* (MLE) or alternatively with *maximum a posterior* (MAP) estimation. When only unlabeled or partially labeled data is available the Baum-Welch algorithm (Baum *et al.*, 1970), which is a special case of the *expectation maximization* (EM) algorithm (Dempster *et al.*, 1977), is typically used.

2.2 HMM inference algorithms

Given a sequence σ and a trained HMM model the goal of an inference algorithm is to predict a path π with minimal (maximal) expected loss (utility). The loss or utility functions can be defined differently for different types of tasks and user preferences, but they should encode the cost associated with different types of errors.

The most widely used inference algorithm for HMMs (as well as other probabilistic models) is the Viterbi algorithm (Viterbi, 1967). Viterbi is a dynamic programming algorithm that finds the most likely path given a trained HMM and an input sequence.

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, P(\sigma, \pi) = \underset{\pi}{\operatorname{argmax}}\, a_{0\pi_1} \prod_{i=1}^{n} e_{\pi_i}(\sigma_i) a_{\pi_i \pi_{i+1}}. \tag{2.4}$$

The main recursion equation of the Viterbi algorithm (omitting backtrace pointers, and initialization and termination conditions) is

$$v_l(i) = e_l(\sigma_i) \max_k (v_k(i-1) a_{kl}), \tag{2.5}$$

where $v_l(i)$ is the probability of the most probable path ending in state l with observation σ_i. While Viterbi is typically viewed as a maximum likelihood algorithm, it is important to realize which loss function's expectation Viterbi is minimizing. Predicting the most likely path is equivalent to minimizing the 0–1 loss function, which assigns a cost of 1 to any path that is not exactly the same as the true path. Given the true path π^t we define the 0–1 loss of a predicted path π^p as

$$L_{0-1}(\pi^t, \pi^p) \triangleq \mathbf{1}\{\pi^t \neq \pi^p\}, \tag{2.6}$$

where $\mathbf{1}\{\cdot\}$ is an indicator function that evaluates to 1 when the condition $\cdot$ is true, and to 0 otherwise. Taking the expectation of L_{0-1} we get

$$\begin{aligned} E_{\pi^t} L_{0-1}(\pi^t, \pi^p) &= \sum_{\pi^t} P(\pi^t|\sigma)\mathbf{1}\{\pi^t \neq \pi^p\} \\ &= 1 - P(\pi^p|\sigma) \\ &= 1 - \frac{P(\pi^p, \sigma)}{P(\sigma)}. \end{aligned} \tag{2.7}$$

It is clear that π^* minimizes the expression in Equation (2.7).

A simple modification of the Viterbi algorithm leads to the *forward* algorithm, which calculates the total probability of a string σ marginalizing over all paths

$$P(\sigma) = \sum_{\pi} P(\sigma, \pi) = \sum_{\pi} a_{0\pi_1} \prod_{i=1}^{n} e_{\pi_i}(\sigma_i) a_{\pi_i \pi_{i+1}}. \tag{2.8}$$

In its main recursion the forward algorithm calculates the *forward variables*

$$\alpha_l(i) = P(\sigma_1 \sigma_2 \cdots \sigma_i, \pi_i = l) = e_l(\sigma_i) \sum_k \alpha_k(i-1) a_{kl}, \tag{2.9}$$

where $\alpha_l(i)$ is the probability of $\sigma_1 \cdots \sigma_i$, requiring that $\pi_i = l$. The *Backward* algorithm is similar to the forward algorithm, but uses a backward recursion instead. It calculates the following *backward variables*

$$\beta_k(i) = P(\sigma_{i+1} \cdots \sigma_n | \pi_i = k) = \sum_l a_{kl} e_l(\sigma_{i+1}) \beta_l(i+1), \tag{2.10}$$

where $\beta_k(i)$ is the probability of $\sigma_{i+1} \cdots \sigma_n$ given that $\pi_i = k$. The marginal posterior probability $P(\pi_i = k|\sigma)$—that in the path that generated σ, σ_i was emitted from state k—is computed using the forward and backward variables.

$$P(\pi_i = k|\sigma) = \frac{P(\pi_i = k, \sigma)}{P(\sigma)} = \sum_{\pi|\pi_i=k} \frac{P(\pi, \sigma)}{P(\sigma)} = \frac{\alpha_k(i)\beta_k(i)}{\sum_l \alpha_l(n)a_{l0}}. \tag{2.11}$$

It is important to notice that $P(\pi_i = k|\sigma)$ is a marginalized over all possible paths, and it equals to the probability mass of all paths that emit σ_i from state k.

2.3 State and transition posterior decoding

Posterior probabilities provide an alternative to the Viterbi algorithm. When the probability of the most likely path is low, and when many different paths have probability that is close to the best one, the Viterbi path is not very likely the correct path. In such cases it is better to use a different loss function than the 0–1 loss, rather than maximize the probability of predicting the entire path correctly. One alternative is to define a loss function that is the sum of the 0–1 losses of individual states in the predicted path. This is also called the Hamming loss (Hamming, 1950).

$$L_{Hamming}(\pi^t, \pi^p) \triangleq \sum_i \mathbf{1}\{\pi_i^t \neq \pi_i^p\}. \tag{2.12}$$

Minimizing the expected hamming loss leads to the following *state posterior decoding* path (Durbin *et al.*, 1998),

$$\hat{\pi} = \{\hat{\pi}_i : \hat{\pi}_i = \operatorname*{argmax}_k P(\pi_i = k|\sigma)\}. \tag{2.13}$$

One of the problems with the state posterior decoding approach is that it ignores the structure of the HMM model, and can produce illegal paths, i.e. paths that include transitions between states k and l for which $a_{kl} = 0$. Even if this problem is corrected by disallowing such transitions, the posterior decoding approach ignores

the dependencies between contiguous states, which are fundamental to HMM based models.

We propose an extension to state posterior decoding. While Equation (2.11) defines posterior probabilities over states, it is also possible to define posterior probabilities over transitions between states,

$$\begin{aligned}
\gamma_{kl}(i) &= P(\pi_i = k, \pi_{i+1} = l|\sigma) \\
&= \frac{\alpha_k(i) a_{kl} e_l(\sigma_{i+1}) \beta_l(i+1)}{P(\sigma)}, \qquad (2.14) \\
\tau_{kl}(i) &= \frac{\gamma_{kl}(i)}{\sum_j \gamma_{kj}(i)} \\
&= P(\pi_{i+1} = l|\pi_i = k, \sigma) \\
&= \frac{\alpha_k(i) a_{kl} e_l(\sigma_{i+1}) \beta_l(i+1)}{P(\pi_i = k, \sigma)} \\
&= \frac{\alpha_k(i) a_{kl} e_l(\sigma_{i+1}) \beta_l(i+1)}{\alpha_k(i) \beta_k(i)} \\
&= \frac{a_{kl} e_l(\sigma_{i+1}) \beta_l(i+1)}{\beta_k(i)}. \qquad (2.15)
\end{aligned}$$

$\gamma_{kl}(i)$ is the joint posterior probability that σ_i is emitted from state k amd σ_{i+1} is emitted from state l. We term $\tau_{kl}(i)$ the *transition posterior probability* from state k in position i to state l in position $i+1$ given σ. It is instructive to note that the state posterior probabilities can be derived from the γ variables by marginalization

$$P(\pi_i = k|\sigma) = \sum_l \gamma_{kl}(i). \qquad (2.16)$$

Another important observation is that the transition posterior probabilities can be used to find the most likely path π^*, which is typically found using the Viterbi

algorithm (2.4)

$$\begin{aligned}
\prod_{i=1}^{n} \tau_{\pi_{i-1}\pi_i}(i-1) &= \prod_{i=1}^{n} \frac{a_{\pi_{i-1}\pi_i} e_{\pi_i}(\sigma_i)\beta_{\pi_i}(i)}{\beta_{\pi_{i-1}}(i-1)} \\
&= \frac{\beta_{\pi_n}(n)}{\beta_0(0)} \prod_{i=1}^{n} a_{\pi_{i-1}\pi_i} e_{\pi_i}(\sigma_i) \\
&= \frac{a_{\pi_n}(0)}{P(\sigma)} \prod_{i=1}^{n} a_{\pi_{i-1}\pi_i} e_{\pi_i}(\sigma_i) \\
&= \frac{P(\sigma,\pi)}{P(\sigma)} \\
&= P(\pi|\sigma), \qquad (2.17)
\end{aligned}$$

$$\begin{aligned}
\pi^* &= \operatorname*{argmax}_{\pi} P(\sigma,\pi) \\
&= \operatorname*{argmax}_{\pi} P(\pi|\sigma) \\
&= \operatorname*{argmax}_{\pi} \prod_{i=1}^{n} \tau_{\pi_{i-1}\pi_i}(i-1). \qquad (2.18)
\end{aligned}$$

Equations (2.18) and (2.16) show that the γ variables alone can be used to find both the most likely path π^*, and the state posterior decoding path $\hat{\pi}$. They can also be used to find the path $\ddot{\pi}$ that maximizes the expected number of correct transitions

$$\ddot{\pi} = \operatorname*{argmax}_{\pi} \sum_{i=1}^{n} \gamma_{\pi_{i-1}\pi_i}(i-1). \qquad (2.19)$$

Although $\ddot{\pi}$ is likely a valid path, that is not guarantied. An additional constraint can be added to (2.19) that will only allow using transitions with positive γ values.

Transition posterior probabilities can also be used to sample paths from the posterior distribution. Starting from the begin state (0) pick a state π_1 with probability $\tau_{0\pi_1}(1)$. Continue to sample states in the path with probability $\tau_{\pi_{i-1}\pi i}(i)$. The total probability of the sampled path π is

$$\prod_{i=1}^{n} \tau_{\pi_{i-1}\pi_i}(i-1).$$

Sampling paths instead of predicting a single path is useful when more than one path can be true, and it has been applied previously by Cawley and Pachter (2003) to the problem of comparative prediction of alternative splicing of genes.

Finally, the main advantage of both state and transition posterior probabilities is that they enable direct computation and optimization of many useful utility functions. In Chapters 3, 4 and 5 we demonstrate how the expectations of several different utility and loss functions can be optimized using posterior decoding methods.

2.4 Multiple sequence HMMs

So far we have discussed HMMs for single strings. It is possible to extend the HMM framework to model multiple related strings. The natural extension of single sequence HMMs are *pair HMMs*, which emit pairs of strings (Alexandersson *et al.*, 2005).

2.4.1 The alignment problem

When each state in a pair HMM emits a pair of characters then both resulting strings have identical lengths. In such cases the pair HMM is completely equivalent to the basic HMM, and all the algorithms we have described previously for HMMs can be used without modification with pair HMMs. However, typically pair HMMs can have semi-silent states, in which a character is emitted for one of the strings but not the other. Semi-silent states lead to the *alignment problem*. Although the generative model emits pairs of characters, we observe the output as two independent strings. During inference, it is therefore required to first decide which pairs of characters should be aligned before these pairs can be assigned to hidden states. It also means that the total length of the hidden Markov chain is not known, unlike in basic HMMs.

An extension of the Viterbi algorithm to pair HMMs with semi-silent states results in a dynamic programming algorithm that finds the most likely path π^*, which defines an alignment and the path of hidden states. While the time complexity of the Viterbi algorithm for single HMMs with K states and pair HMMs with no semi-silent states is $O(nK)$, it is $O(nmK)$ in the case of a pair HMM for sequences of length n and m. The alignment problem makes the inference problem quadratic instead of linear in the total length of the observed sequences.

It is possible to combine the alignment and decoding in one algorithm. However, when the number of states in the pair HMM is large and when the sequences are long this solution might not be feasible in practice. An alternative approach is to first align the two sequences using a simplified pair HMM, and then decode the path of the more complex model assuming that the alignment is correct. This approach leads to an algorithm of time complexity $O(nm + (n + m)K)$. We discuss algorithms that improve pairwise alignment accuracy using posterior probabilities in Chapter 3.

2.4.2 Multiple sequences

Pair HMMs can be extended to multi HMMs, which emit multiple related strings. The relationship between the sequences can be modeled using a tree, such as phylogenetic trees in the case of genomic sequences, and language trees in the case of translation lexicons. The issues we have described for pair HMMs grow exponentially with number of sequences in multi HMMs. The time complexity of aligning n sequences of maximum length l using a multi HMM is $O(l^n)$. Typically, the number of states K also grows when modeling multiple sequences. In the general case of a Cartesian product of HMMs the number of states in the resulting multi HMM is $O(K^n)$, where K is the number states in the single HMM.

Because of the complexity of the problem, multi HMMs almost always use an

existing multiple sequence alignment as their input. Several heuristics are typically used to produce multiple sequence alignments. *Progressive alignment* algorithms first align a pair of sequences, then align a third sequence to the pairwise alignment to produce a three-way alignment, and so forth until all the sequences are aligned.

One problem with progressive alignment algorithms is that once a subset of sequences are aligned their sub-alignment is fixed, and cannot change with additional evidence from the other sequences. *Iterative refinement* methods remove a subset of sequences from an existing multiple sequence alignment, and realign the to the remaining aligned sequences. Both these and other heuristics have no guarantee on the quality of the alignment they produce, however they have been used extensively in practice to produce alignments that are then used to model more complex problems, such as gene finding and phylogenetic tree construction. In Chapter 4 we show how using posterior decoding methods helps to explore the space of multiple alignments with much more refined steps.

Another way of solving the multiple sequence alignment problem is to avoid using it in the first place. We propose to combine posterior probabilities calculated from pair HMMs to combine evidence from all available sequences and then use them to decode the state path in each sequence separately. As we have shown the γ variables can be used to calculated π^*, $\hat{\pi}$ or $\ddot{\pi}$. The way γ variables should be combined is still an open problem. The simplest approach would be to take a weighted average of γ values for every pair of states in every position

$$\gamma_{kl}^{\sigma^1}(i) = \sum_s w(\sigma^s)\gamma_{kl}^{\sigma^1\sigma^s}(i). \tag{2.20}$$

2.5 Discussion

Posterior decoding methods have several advantages over Viterbi decoding. Unlike Viterbi, posterior decoding can be designed to minimize the expectation of a specific loss function. In the next chapter we show how such loss (or utility) functions can be defined in the case of pairwise sequence alignmet, and demonstrate the advantages of posterior decoding over Viterbi.

Chapter 3

Alignment Metric Accuracy

Sequence alignment is probably the most basic sequence analysis task, yet it is still being actively researched more than forty years after the definition of the *edit distance* by Levenshtein (1966), and the introduction of the dynamic algorithm for global alignment of Needleman and Wunsch (1970). This has to do with the fact that while sequence alignment is a basic component of almost any comparative genomics analysis, producing accurate alignments is a hard problem.

The tertiary (three dimensional) structures of molecules in living cells, such as DNA, RNA, and proteins are essential for their function. While knowing the structure of each molecule would greatly help understand how they interact and perform their roles, obtaining these structures is still a very slow, expensive, and painstaking task. On the other hand, obtaining the primary structures of these molecules, which can be represented as sequences of characters, has become an almost trivial task. It is therefore not surprising that the amount of data in sequence databases, such as GeneBank, has been growing exponentially over the past two and a half decades.[1]

We focus our discussion on alignment of biological sequences; although align-

[1] http://www.ncbi.nlm.nih.gov/Genbank/genbankstats.html

ments of other types of sequences are also important for problems such as spelling-correction, and speech recognition. Moreover, while there are different types of sequence-alignments, such as local alignments, whole-genome alignments, and alignments with repeats and rearrangements, we only consider the problem of global-alignment. We defer the discussion on global multiple sequence alignment to the next chapter, and concentrate on global pairwise alignment in this chapter.

Since all organisms on earth are part of the *Tree of Life*, any two organisms have a common ancestor. Therefore, given two biological sequences, a valid and important questions is—which pairs of positions in these two sequences are homologous, i.e. originate from the same position in their common ancestral sequence? In essence, a sequence alignment is an answer to the above question. It is important to remember that while the question refers only to homologous positions, it implies that we also want to know which positions are not homologous. Therefore, given two unrelated sequences (i.e. sequences that were derived from completely distinct positions in the common ancestor) a correct answer will be an alignment in which no two positions are aligned.

More formally, we define pairwise alignment using the notation in Pachter and Sturmfels (2005). An alignment of a pair of sequences $\sigma^1 \triangleq \sigma_1^1\sigma_2^1\cdots\sigma_n^1$ and $\sigma^2 \triangleq \sigma_1^2\sigma_2^2\cdots\sigma_m^2$ can be represented by an *edit string* h over the *edit alphabet* $\{H, I, D\}$, where H stands for homology, I for insertion, and D for deletion. Equivalently, if $\mathcal{A}_{n,m}$ is the set of all alignments of a sequence of length n to a sequence of length m, and $h \in \mathcal{A}_{n,m}$, then h can be represented by a path in the *alignment graph*, or a sequence of pairs of the form $(\sigma_i^1 \diamond \sigma_j^2)$, $(\sigma_i^1 \diamond -)$, or $(- \diamond \sigma_j^2)$ where the symbol $\diamond$ is used to indicate the alignment of two characters. Figure 3.1 shows a pairwise alignment of two DNA sequences `ACGTAGC` and `ACCGAGACC`. The edit string of the alignment in the figure is $HHIHIHHIDH$.

```
AC-G-TA-GC
ACCGAGAC-C
```

Figure 3.1. **Example of a global pairwise alignment of DNA sequences.**

A recent survey on sequence alignment (Batzoglou, 2005) discusses a number of important problems and challenges that need to be overcome in order to facilitate large-scale comparative analysis of the multiple genomes currently being sequenced. Among these, the following two problems are highlighted:

1. "As suggested (Miller, 2000), methods to evaluate alignment accuracy. This goes at the core of the problem: which regions are alignable, and what is a correct alignment?"

2. "A definition of *alignability* – at what point is it no longer possible to do meaningful sequence alignment. Or rather, at what point can one conclude that two sequences are no longer related?"

Furthermore, the development of "rigorous methods for evaluating the accuracy of an alignment" and the need for "improved pairwise alignment with a statistical basis" are singled out as the most pressing challenges for the alignment community. We address some of these points in this chapter.

Sequence alignments are typically used in computational biology as an input to higher level comparative analyses of the sequences, such as comparative annotation of functional elements, like genes, and transcription factor binding sites; phylogenetic and evolutionary analysis; and reconstruction of ancestral sequences. Such analyses typically assume the input alignments are correct, without incorporating possible alignment errors into their models. However, different alignment algorithms, or even changing parameters of the same algorithms, can produce very different alignments, especially when the input sequences are evolutionary distant. In this chapter we

discuss how the accuracy of pairwise sequence alignments can be optimized directly using posterior decoding. We also show how the accuracy of alignments can be estimated when the correct alignment is unknown.

In Section 3.1 we define an alignment metric, and a new accuracy measure, called *AMA*, based on this metric. Next, we propose an algorithm which maximizes the expected AMA value given an underlying probabilistic model for mutation of sequences. We term this alignment strategy AMAP. The algorithm is explained in detail in Section 3.2, where we show that a single parameter we term *gap-factor* (γ) can be used to adjust the recall/precision trade-off. A special case of AMAP, where we set $\gamma = 0$, is the Maximal Expected Accuracy (MEA) alignment algorithm introduced in (Durbin *et al.*, 1998; Holmes and Durbin, 1998) and used in ProbCons (Do *et al.*, 2005), and Pecan (Paten, 2005).

In Section 3.3 we show how existing algorithms perform when judged using AMA, and contrast this with the developer score, which is the standard measure used in most papers (Do *et al.*, 2005; Edgar, 2004; Sze *et al.*, 2005). Since the developer score is a measurement of recall of aligned pairs, and algorithms have traditionally been judged by it, we find that existing algorithms are heavily biased in favor of recall, rather than precision. We show that in extreme cases existing algorithms align large fractions of completely unrelated sequences. We also see that multiple alignments produced by different programs differ considerably from each other, even though they may appear to perform similarly when judged only by the developer score.

In a different application of the alignment metric, we show that in the typical case where reference alignments are not available for judging the success of multiple alignment experiments, the metric can be used as a control by measuring the distances between alignments predicted by different programs.

Finally, we analyze the performance of the new AMAP algorithm, using the SAB-

mark dataset (Van Walle *et al.*, 2005) and simulated datasets, and show that AMAP compares favorably to other programs.

3.1 Metric based alignment accuracy

Two commonly used alignment accuracy measures are the *developer* (f_D) and the *modeler* (f_M) measures (Sauder *et al.*, 2000). These measures correspond to evaluating the recall (number of correctly matched pairs divided by the number of matched pairs in the reference alignment) and precision (number of correctly matched pairs divided by the number of matched pairs in the predicted alignment) of matched pairs in the predicted alignment respectively. These measures have several problems. First, there is an inherent trade-off between recall and precision, and therefore maximizing one measure typically reduces the other measure. Second, both measures do not account for gap columns. Blanchette *et al.* (2004) defined the *agreement* score, as the fraction of pairwise columns in the predicted alignment that agree with the reference alignment. While the agreement score is a single accuracy measure that does consider gap columns, it is not symmetric, since the number of columns in the predicted alignment can differ from the number of columns in the reference alignment.

We use the following notation to formally define the different alignment accuracy measures. For $h \in \mathcal{A}_{n,m}$ let

- $h_H \triangleq \{(i,j) : (\sigma_i^1 \diamond \sigma_j^2) \in h\}$,
- $h_D \triangleq \{i : (\sigma_i^1 \diamond -) \in h\}$,
- $h_I \triangleq \{j : (- \diamond \sigma_j^2) \in h\}$.

Less formally, h_H is the set of position pairs in σ^1 and σ^2 that are aligned according to h, h_D is the set of position in σ^1 that are gapped, and h_I is the set

of position in σ^2 that are gapped. For example, for the alignment in Figure 3.1 $h_H = \{(1,1),(2,2),(3,4),(4,6),(5,7),(7,9)\}$, $h_D = \{6\}$, and $h_I = \{3,5,8\}$. Note that for any $h \in \mathcal{A}_{n,m}$

$$|h_H| + |h_D| = n \text{ and } |h_H| + |h_I| = m. \tag{3.1}$$

Two alignments are equivalent if they align the same character pairs, while the order of insertions and deletions between any two consecutive aligned pairs is redundant. We therefore define:

$$\forall h^i, h^j \in \mathcal{A}_{n,m}\ h^i \equiv h^j \text{ if and only if } h^i_H = h^j_H. \tag{3.2}$$

Note that $h^i_H = h^j_H$ if and only if $h^i_I = h^j_I$ and $h^i_D = h^j_D$. We can therefore use the following equivalent definition:

$$\forall h^i, h^j \in \mathcal{A}_{n,m}\ h^i \equiv h^j \text{ if and only if } h^i_I = h^j_I \text{ and } h^i_D = h^j_D. \tag{3.3}$$

We say that two alignments are *distinct* if they are not equivalent. The number of distinct alignments is in bijection with lattice paths from the origin $(0,0)$ to (m,n) in the square grid:

Proposition 1. *The number of distinct alignments in* $\mathcal{A}_{n,m}$ *is* $\binom{n+m}{m}$.

Proof: Two equivalent alignments differ by definition only by the order of insertions and deletions between adjacent homology states. Therefore, every alignment has a canonical edit string, in which insertions always precede deletions between adjacent homology states. In other words, a canonical edit string does not include the substring 'DI'. There is a bijective mapping between canonical edit strings and lattice paths from the the origin $(0,0$ to (m,n) in the square grid. Let 'I' represent a one point move in the first dimension, and 'D' a one point move in the second dimension. Replacing every 'H' with 'DI' results in a path of length $m+n$ with m moves in the first dimension and n moves in the second dimension, which is exactly a path from

$(0,0)$ to (m,n). In the other direction, every such path can be mapped to a unique canonical edit string by replacing every 'DI' with 'H', since no 'DI' substring exist in the canonical edit string. Since every path is of length $n+m$, and every path can be uniquely represented by the m positions with $'I'$ moves (or the n positions with $'D'$ moves) there are $\binom{n+m}{m}$ unique paths from $(0,0)$ to (m,n) and the same number of distinct alignments in $\mathcal{A}_{n,m}$. □

Given a predicted alignment h^p and a reference alignment h^r, the developer (f_D) and modeler (f_M) measures are defined:

$$f(h^i, h^j) \triangleq \frac{|h^i_H \cap h^j_H|}{|h^i_H|} \tag{3.4}$$

$$f_D(h^r, h^p) \triangleq f(h^r, h^p) = \frac{|h^r_H \cap h^p_H|}{|h^r_H|} \tag{3.5}$$

$$f_M(h^r, h^p) \triangleq f(h^p, h^r) = \frac{|h^r_H \cap h^p_H|}{|h^p_H|} \tag{3.6}$$

Note that both measures do not explicitly use the I and D characters in h^r and h^p, and are not well defined when h^r or h^p do not include any H characters.

What properties should an alignment accuracy measure satisfy? Generally, it should assign higher accuracy values to alignments that are "closer" to the correct alignment, and lower accuracy to alignments that are "farther" from the truth. If the accuracy is to be used as a utility function that is optimized with posterior-decoding, then it should decompose well over the basic elements for which posterior-probabilities are available. We would like to formalize the notion of similarity and distance between alignments. If h^i and h^j are two alignments and we denote their distance by $d(h^i, h^j)$, we would like to have:

$$d(h^i, h^j) \geq 0 \qquad \forall h^i, h^j \in \mathcal{A}_{n,m}, \tag{3.7}$$

$$d(h^i, h^j) = 0 \text{ if and only if } h^i \equiv h^j \qquad \forall h^i, h^j \in \mathcal{A}_{n,m}, \tag{3.8}$$

$$d(h^i, h^j) = d(h^j, h^i) \qquad \forall h^i, h^j \in \mathcal{A}_{n,m}, \tag{3.9}$$

$$d(h^i, h^j) + d(h^j, h^k) \geq d(h^i, h^k) \qquad \forall h^i, h^j, h^k \in \mathcal{A}_{n,m}. \tag{3.10}$$

The first condition specifies that the distance between two alignments should be non-negative. The second condition requires that the distance should be 0 if, and only if, the two alignments are equivalent. The third requirement specifies that the distance should be symmetric. For example, comparing a prediction with a reference alignment should be the same as comparing the reference alignment to the prediction. The fourth requirement ensures a certain consistency: the distance between two predictions should be less than the sum of the distances from the predictions to a reference alignment (the triangle inequality). In other words, an accuracy measure should be based on a *metric*. Furthermore, the accuracy measure should account for unalignable sequence. For example, if two sequences are unrelated, the true alignment contains only gaps (regardless of order), and a good accuracy measure should reflect that. Note that although metrics on the space of *sequences* have been constructed (Spiro and Macura, 2004), a metric for alignment accuracy should be defined on the space of *sequence alignments*, and should measure the distance between alignments not sequences.

While the metric requirements are not satisfied by (3.4), they are satisfied by the following:

$$\begin{aligned} d(h^i, h^j) &\triangleq 2|h^i_H| + |h^i_I| + |h^i_D| - 2|h^i_H \cap h^j_H| - |h^i_I \cap h^j_I| - |h^i_D \cap h^j_D| \\ &= 2|h^j_H| + |h^j_I| + |h^j_D| - 2|h^i_H \cap h^j_H| - |h^i_I \cap h^j_I| - |h^i_D \cap h^j_D| \\ &= n + m - 2|h^i_H \cap h^j_H| - |h^i_I \cap h^j_I| - |h^i_D \cap h^j_D|. \end{aligned} \tag{3.11}$$

Proposition 2. *$d(h^i, h^j)$ is a finite metric for $\mathcal{A}_{n,m}$.*

Proof: It is easy to see that $d(h^i, h^j)$ satisfies requirements (3.7), (3.8), and (3.9). We need to show that it satisfies the triangle inequality (3.10). Let $\bigcap_{ij} \triangleq 2|h^i_H \cap h^j_H| + |h^i_I \cap h^j_I| + |h^i_D \cap h^j_D|$, and $\bigcap_{ijk} \triangleq 2|h^i_H \cap h^j_H \cap h^k_H| + |h^i_I \cap h^j_I \cap h^k_I| + |h^i_D \cap h^j_D \cap h^k_D|$. Using the fact that $\bigcap_{ik} - \bigcap_{ijk} \geq 0$ and $\bigcap_{ij} + \bigcap_{jk} - \bigcap_{ijk} \leq n + m$, we have that

$d(h^i, h^j)+d(h^j, h^k)-d(h^i, h^k) = n+m-\bigcap_{ij} - \bigcap_{jk} + \bigcap_{ik} = n+m-(\bigcap_{ij} + \bigcap_{jk} - \bigcap_{ijk})+ \bigcap_{ik} - \bigcap_{ijk} \geq 0.$ □

Example 3 (Metric for $\mathcal{A}_{2,2}$)**.** *By Proposition 1, there are six distinct alignments in* $\mathcal{A}_{2,2}$*. The metric is:*

	HH	HDI	DIH	IHD	DHI	$DDII$
HH	0	2	2	4	4	4
HDI	2	0	4	3	3	2
DIH	2	4	0	3	3	2
IHD	4	3	3	0	4	2
DHI	4	3	3	4	0	2
$DDII$	4	2	2	2	2	0

Intuitively, the distance between two alignments is the total number of characters from both sequences that are aligned differently in the two alignments. Alternatively, the quantity

$$s(h^i, h^j) \triangleq 1 - \frac{d(h^i, h^j)}{n+m} = \frac{2|h^i_H \cap h^j_H| + |h^i_I \cap h^j_I| + |h^i_D \cap h^j_D|}{n+m} \tag{3.12}$$

is a convenient similarity measure that can be interpreted as the fraction of characters that are aligned the same in both alignments. We therefore define the *Alignment Metric Accuracy (AMA)* of a predicted alignment h^p given a reference alignment h^r to be $s(h^r, h^p)$. The intuitive motivation for this accuracy measure is that it represents the fraction of characters in σ^1 and σ^2 that are correctly aligned, either to another character or to a gap.

Figure 3.2 shows a reference alignment and a predicted alignment of the same two protein sequences. The f_D score for the predicted alignment is $3/3 = 1$, since all three pairs of aligned positions from the reference alignment exist in the predicted alignment; the f_M score is $3/4 = 0.75$, since out of four aligned pairs in the predicted alignment only three exist in the reference alignment; the AMA score is $9/11 = 0.82$, since out of eleven positions in both sequences, nine are aligned the same.

Reference alignment	Predicted alignment
EL-IGKPQ	ELIGKPQ
SLK----Q	SL--K-Q

Figure 3.2. **Example of a reference alignment and a predicted alignment of two protein sequences.**

AMA can easily be extended to multiple sequence alignments (MSA) by using the sum-of-pairs approach. Let $\mathcal{A}_{n_1,n_2,\ldots,n_k}$ be the space of all MSAs of k sequences of lengths n_1 to n_k. Given two MSAs $h^i, h^j \in \mathcal{A}_{n_1,n_2,\ldots,n_k}$,

$$d(h^i, h^j) \triangleq \sum_{s^1=1}^{k-1} \sum_{s^2>s^1}^{k} d(h^i_{s^1,s^2}, h^j_{s^1,s^2}), \tag{3.13}$$

where $h^i_{s^1,s^2}$ is the pairwise alignment of sequences s^1, s^2 as projected from the MSA h^i with all-gap columns removed. The similarity of two MSAs is defined to be

$$AMA(h^r, h^p) \triangleq s(h^r, h^p) = 1 - \frac{d(h^r, h^p)}{(k-1)\sum_{i=1}^{k} n_i}. \tag{3.14}$$

Unlike standard sum-of-pairs scoring, our definition follows from the requirement that our accuracy measure should be based on a metric, and the multiple AMA retains the desirable properties of the pairwise AMA.

3.2 AMA based alignments

3.2.1 Maximal expected accuracy alignments

Given a probabilistic model for alignments, such as a pair-HMM, an alignment of a pair of sequences is typically obtained by the Viterbi algorithm (Viterbi, 1967), which finds the global alignment with highest probability. In the case of a pair-HMM

with three states, the Viterbi algorithm is equivalent to the standard Needleman-Wunsch algorithm with affine gap scores (Durbin *et al.*, 1998). In effect, the Viterbi algorithm minimizes the expected 0-1 loss, or maximizes the expected number of times that a predicted alignment is equivalent to the reference alignment ($h^p \equiv h^r$). However, when the probability of the most likely alignment is low, there might be many candidate alignments with similar probability. In such cases it might be more desirable to predict alignments that are likely to align the most number of characters correctly on average even if they are less likely to be identical to the correct alignment.

An alternative to Viterbi alignment is the *optimal accuracy* alignment (Holmes and Durbin, 1998; Durbin *et al.*, 1998), also called *maximal expected accuracy* (MEA) alignment (Do *et al.*, 2005), which maximizes the expected f_D score. The MEA alignment is calculated using a dynamic programming algorithm that finds the alignment h^{MEA} that maximizes the expected number of correctly aligned character pairs:

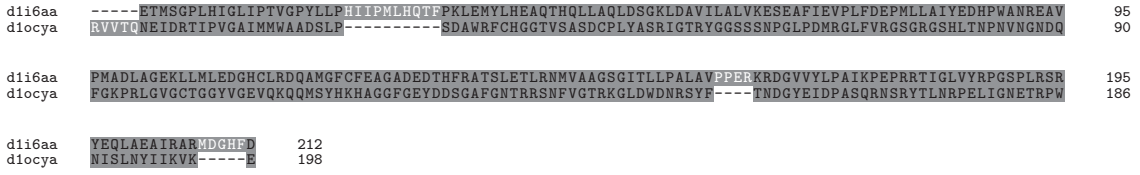
$$h^{MEA} \triangleq \underset{h \in \mathcal{A}_{n,m}}{\operatorname{argmax}} \sum_{(i,j) \in h_H} P(\sigma_i^1 \diamond \sigma_j^2 | \sigma^1, \sigma^2, \theta), \tag{3.15}$$

where $P(\sigma_i^1 \diamond \sigma_j^2 | \sigma^1, \sigma^2, \theta)$ is the posterior probability that σ_i^1 is homologous to σ_j^2 given σ^1, σ^2 and the parameters of the model θ. In the case of a pair-HMM, these posterior probabilities can be computed in $O(nm)$ time using the *forward-backward* algorithm (Durbin *et al.*, 1998).

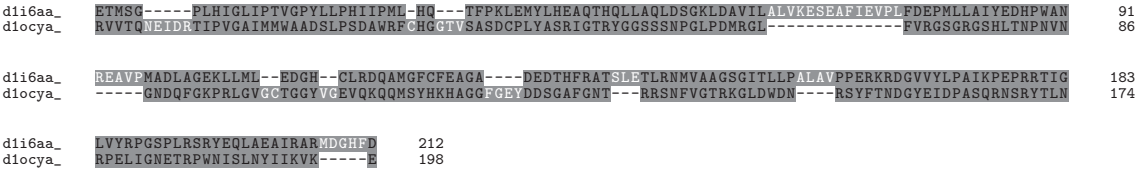
3.2.2 The AMAP algorithm

While the MEA algorithm maximizes the expected f_D score it can perform very poorly on the f_M score when the reference alignment contains many unaligned characters (gaps), since it tends to over-align characters. Figure 3.3 shows four predicted alignments of two unrelated proteins. While the f_D and f_M scores are undefined in such case, the AMA measure provides a qualitative assessment of the accuracy of each

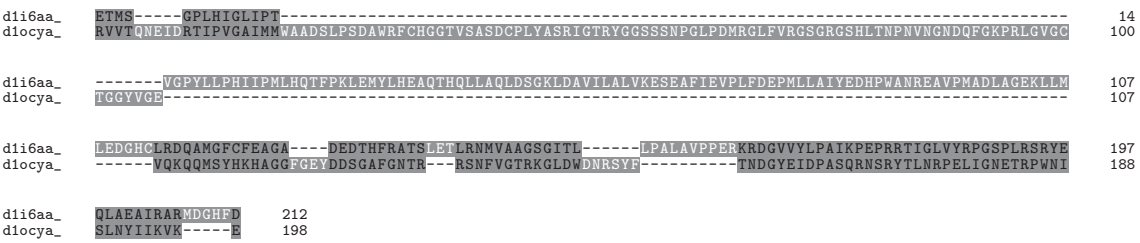
Viterbi



MEA, or AMAP with $\gamma = 0$



AMAP with $\gamma = 1$



AMAP with $\gamma = 8$

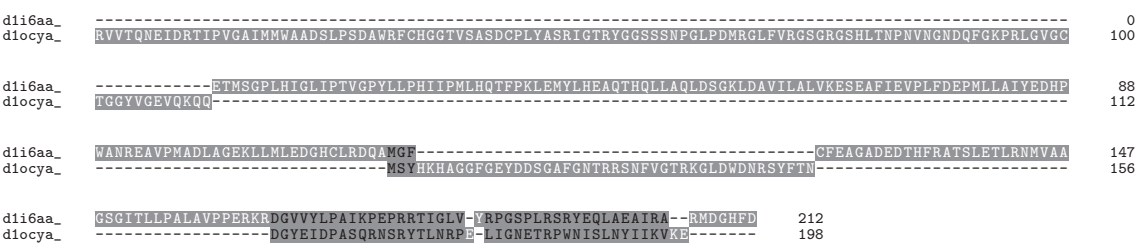


Figure 3.3. **Example of four different alignments of two unrelated protein sequences.** Light characters represent correctly aligned positions, and dark characters represent incorrectly aligned positions. Since the two sequences are unrelated all aligned positions are wrongly aligned, and all gapped positions are correctly aligned. The fraction of characters that are correctly aligned (light) represent an intuitive notion of accuracy.

alignment. The first alignment is the Viterbi alignment, which incorrectly aligns all but 24 positions, and achieves an AMA score of $24/410 = 0.059$. The MEA alignment is slightly better at $48/410 = 0.117$; although it could be worse than Viterbi in other cases. Note that the worst possible AMA score on this example is $14/410 = 0.034$, since the first sequence is longer than the second by 14 characters. It is clear that both Viterbi and MEA suffer from the over-alignment phenomena, which can greatly affect their accuracy when aligning unrelated sequences, or sequences that include unrelated regions (a very common scenario). This is a major drawback of these popular procedures, since a user cannot distinguish between alignment columns that are correct and those that are spurious, simply by inspecting the predicted alignment.

Maximizing the expected f_M score can be done easily by only aligning the pair of characters with highest posterior probability to be homologous. This will result in an alignment with only one H character, $n-1$ D characters, and $m-1$ I characters, which in most cases will result in a poor f_D score. There is currently no alignment algorithm that enables adjustment of the recall/precision trade-off (f_D / f_M trade-off). However, we show that it is possible to maximize the expected AMA value using an algorithm similar to the original MEA algorithm. By maximizing the expected AMA, we avoid the problems of MEA alignment. In addition to maximizing the expected AMA value, the new algorithm, which we call AMAP, has one free parameter, which we term *gap-factor*,or γ, that controls the f_D/f_M trade-off.

Let $P(\sigma_i^1 \diamond - |\sigma^1, \sigma^2, \theta) \triangleq 1 - \sum_{j=1}^{m} P(\sigma_i^1 \diamond \sigma_j^2 | \sigma^1, \sigma^2, \theta)$ be the posterior probability that σ_i^1 is not homologous to any character in σ^2, and $P(- \diamond \sigma_j^2 | \sigma^1, \sigma^2, \theta) \triangleq 1 - \sum_{i=1}^{n} P(\sigma_i^1 \diamond \sigma_j^2 | \sigma^1, \sigma^2, \theta)$ the posterior probability that σ_j^2 is not homologous to any character in σ^1. AMAP should find the alignment h^{AMA} that maximizes the expected AMA score, which is equivalent to the expected number of characters that

are correctly aligned to another character or to a gap:

$$
\begin{aligned}
h^{AMA} &\triangleq \underset{h^p \in \mathcal{A}_{n,m}}{\operatorname{argmax}} E_{h^t}\left(AMA(h^t, h^p)\right) \\
&= \underset{h^p \in \mathcal{A}_{n,m}}{\operatorname{argmax}} \sum_{h^t \in \mathcal{A}_{n,m}} P(h^t|\sigma^1, \sigma^2, \theta) \frac{2|h^t_H \cap h^p_H| + |h^t_I \cap h^p_I| + |h^t_D \cap h^p_D|}{n+m} \\
&= \underset{h^p \in \mathcal{A}_{n,m}}{\operatorname{argmax}} \sum_{h^t \in \mathcal{A}_{n,m}} P(h^t|\sigma^1, \sigma^2, \theta) \\
&\qquad \left(2 \sum_{(i,j) \in h^p_H} \mathbf{1}\{(i,j) \in h^t_H\} + \sum_{j \in h^p_I} \mathbf{1}\{j \in h^t_I\} + \sum_{i \in h^p_D} \mathbf{1}\{i \in h^t_D\}\right) \\
&= \underset{h^p \in \mathcal{A}_{n,m}}{\operatorname{argmax}} \; 2 \sum_{(i,j) \in h^p_H} \sum_{h^t \in \mathcal{A}_{n,m}} P(h^t|\sigma^1, \sigma^2, \theta) \mathbf{1}\{(i,j) \in h^t_H\} + \\
&\qquad \sum_{j \in h^p_I} \sum_{h^t \in \mathcal{A}_{n,m}} P(h^t|\sigma^1, \sigma^2, \theta) \mathbf{1}\{j \in h^t_I\} + \\
&\qquad \sum_{i \in h^p_D} \sum_{h^t \in \mathcal{A}_{n,m}} P(h^t|\sigma^1, \sigma^2, \theta) \mathbf{1}\{i \in h^t_D\} \\
&= \underset{h^p \in \mathcal{A}_{n,m}}{\operatorname{argmax}} \; 2 \sum_{(i,j) \in h^p_H} P(\sigma^1_i \diamond \sigma^2_j | \sigma^1, \sigma^2, \theta) + \\
&\qquad \sum_{j \in h^p_I} P(- \diamond \sigma^2_j | \sigma^1, \sigma^2, \theta) + \sum_{i \in h^p_D} P(\sigma^1_i \diamond - | \sigma^1, \sigma^2, \theta). \qquad (3.16)
\end{aligned}
$$

h^{AMA} can be computed efficiently using a dynamic programming algorithm similar to the Needleman-Wunsch, and MEA algorithms in time $O(nm)$. The fact that the AMA function decomposes well over the basic elements, for which posterior-probabilities are easy to obtain, is instrumental for the computation in Equation (3.16). We will see in Section 5.3 that this is not a trivial requirement. For example, the F_1 measure does not decompose well, and therefore it is not clear how to maximize its expected value using posterior-decoding.

h^{AMA} is the alignment with the maximal expected AMA score, which provides a balanced assessment of recall and precision. However, there are cases when better control of the recall/precision trade-off is desired. In particular, some sequence analysis tasks like phylogenetic analysis can use partial alignments, and might benefit from

using alignments with higher precision, while tasks such as comparative functional annotation, might require higher recall. Such control of recall and precision can be achieved by addition of a single parameter γ to the calculation in Equation (3.16). We term this parameter *gap-factor* since it adjusts the weight of the gap-posterior probabilities with respect to the match-posterior probabilities.

$$\begin{aligned} h^{\gamma} &\triangleq \underset{h^p \in \mathcal{A}_{n,m}}{\operatorname{argmax}} \sum_{h^t \in \mathcal{A}_{n,m}} P(h^t|\sigma^1,\sigma^2,\theta) 2|h^t_H \cap h^p_H| + \gamma(|h^t_I \cap h^p_I| + |h^t_D \cap h^p_D|) \\ &= \underset{h^p \in \mathcal{A}_{n,m}}{\operatorname{argmax}} \; 2 \sum_{(i,j) \in h^p_H} P(\sigma^1_i \diamond \sigma^2_j | \sigma^1, \sigma^2, \theta) + \\ &\qquad \gamma \left(\sum_{j \in h^p_I} P(-\diamond \sigma^2_j | \sigma^1, \sigma^2, \theta) + \sum_{i \in h^p_D} P(\sigma^1_i \diamond - | \sigma^1, \sigma^2, \theta) \right). \end{aligned} \tag{3.17}$$

h^{γ} is the alignment that maximizes the expected value of the utility function $U_{\gamma}(h^r, h^p) \triangleq 2|h^r_H \cap h^p_H| + \gamma(|h^r_I \cap h^p_I| + |h^r_D \cap h^p_D|)$, where $\gamma \in [0, \infty)$ The neutral value for γ is 1, in which the algorithm maximizes the expected AMA value, while when $\gamma = 0$ the expression in (3.17) is equal to the expression in (3.15), and the algorithm is identical to the original MEA algorithm, which maximized the expected f_D score. Setting γ to higher values than 1 results in better f_M scores in the expense of lower f_D scores.

3.3 Results

3.3.1 Performance of existing programs on the SABmark datasets

We begin by assessing the performance of existing programs on the SABmark 1.65 (Van Walle *et al.*, 2005) datasets with the goal of comparing alignment metric

	Twilight			Superfamilies			Twilight-FP			Superfamilies-FP		
Program	f_D	f_M	AMA	f_D	f_M	AMA	f_D	f_M	AMA	f_D	f_M	AMA
Align-m	21.6	**23.6**	**51.7**	49.2	**45.6**	**56.9**	17.8	**6.4**	**81.5**	44.8	**16.8**	**77.5**
CLUSTALW	25.6	14.7	24.9	54.0	38.1	43.8	20.4	2.4	35.5	50.9	7.4	37.0
MUSCLE	27.3	16.4	27.6	56.3	40.3	46.4	19.4	2.3	37.1	49.7	7.5	38.9
ProbCons	**32.1**	21.1	37.3	**59.8**	44.4	51.8	**26.7**	4.4	55.7	56.0	10.9	55.0
T-Coffee	29.4	19.6	35.6	58.4	43.7	50.9	26.5	4.2	54.1	**57.0**	11.0	54.4

Table 3.1. **Performance of aligners on the SABmark benchmark datasets.** Entries show the average developer (f_D), modeler (f_M) and alignment metric accuracy (AMA). Best results are shown in bold. All numbers have been multiplied by 100.

accuracy with previously used measures. SABmark includes two sets of pairwise reference alignments with known structure from the ASTRAL (Brenner *et al.*, 2000) database. The Twilight Zone set contains 1740 sequences with less than 25% identity divided into 209 groups based on SCOP folds (Murzin *et al.*, 1995). The Superfamilies set contains 3280 sequences with less than 50% identity divided into 425 groups. Additionally, each dataset has a "false positives" version, which contains unrelated sequences with the same degree of sequence similarity in addition to the related sequences.

Table 3.1 shows the performance of a number of existing alignment programs as measured by the developer, modeler, and AMA accuracy measures on the four SABmark 1.65 datasets. Methods tested include Align-m 2.3 (Van Walle *et al.*, 2005), CLUSTALW 1.83 (Thompson *et al.*, 1994), MUSCLE 3.52 (Edgar, 2004), ProbCons 1.1 (Do *et al.*, 2005) and T-Coffee 2.49 (Notredame *et al.*, 2000). The results highlight the inherent recall/precision trade-off. While ProbCons and T-Coffee have the best developer scores, Align-m has the best modeler scores. It is not clear which program outperforms the others. Programs with higher recall tend to over-align unalignable regions, which results in lower precision. We would like to answer the question, which program produces alignments that are the closest to the reference alignments? This is exactly the interpretation of the new AMA measure. Using this measure it is clear

	Align-m	CLUSTALW	MUSCLE	ProbCons	T-Coffee	Reference
Align-m		37.3	39.9	52.8	50.4	**67.0**
CLUSTALW	45.0		38.2	38.1	39.1	**37.0**
MUSCLE	43.8	49.4		43.2	42.3	**39.2**
ProbCons	32.0	48.7	47.1		52.0	**51.1**
T-Coffee	30.1	47.0	45.9	38.7		**50.1**
Reference	**13.7**	**44.1**	**43.0**	**31.1**	**28.6**	

Table 3.2. **Total distance (d) and average similarity (s) of different aligners on the SABmark dataset.** Values below the diagonal show the total distance (d) between alignments produced by different alignment programs. Values above the diagonal show the average similarity (s) between the different alignments. Distance values have been divided by one million, and similarity values have been multiplied by 100.

that Align-m is the most accurate alignment program among the ones tested on the SABmark benchmark datasets.[2]

3.3.2 Controls for multiple alignment experiments

A recurring question has been how to judge the accuracy of alignments in the absence of a reference. To demonstrate how AMA is useful for that, we compare the total distance (d) and average similarity (s) between the alignments produced by four alignment programs, and the reference alignments. Table 3.2 shows these values averaged over the entire SABmark dataset. An interesting observation is that there seem to be a correlation between the following two distances: (i) the distance $d(M_l^v, M_l^p)$ between any predicted alignment M_l^v and the corresponding reference alignment M_l^R; (ii) the average distance $d(\sum_{u|u\neq v} d(M_l^v, M_l^u)/(V-1))$ between the predicted alignment M_l^v and the other $V-1$ alignments produced by the other programs for the same group of sequence $\{M_l^u | u \neq v\}$. To check for such correlation, we calculated the Pearson correlation between AMA and the average similarity for different datasets. Figure 3.4 shows the correlation between the accuracy (AMA) of

[2]Note that the SABmark benchmark dataset was compiled by the same authors as Align-m.

CLUSTALW alignments of the Twilight-FP and Superfamilies-FP datasets, and their average similarity to any of the alignments produced by the other three programs. It is evident that there is a strong correlation between the two values (Pearson correlation of $r = 0.87$). We observed similar correlations for the other three alignment programs (0.86, 0.49, 0.57 for MUSCLE, ProbCons, and T-Coffee correspondingly). For the datasets without false-positives (Twilight and Superfamilies) the correlations are even stronger (0.89, 0.86, 0.74, 0.78 for CLUSTALW, MUSCLE, ProbCons, and T-Coffee correspondingly). Overall, when considering the four programs together the correlation between AMA and the average similarity is 0.68 for the false-positive sets, and 0.82 for the no-false-positive sets.

The strong correlation observed suggests that when a multiple alignment of a given set of sequences is needed, one could select the best alignment out of several predicted alignments produced by different programs, by comparing the average similarities of the different alignments. This is an alternative to the standard practice of deferring to a single alignment program that is predicted to be more accurate on average, based on evaluations on benchmark datasets. For example, ProbCons, which has the best *average* AMA score on the SABmark datasets compared to the other three programs (Table 3.2), produces the most accurate alignment in only 54% of the cases. Moreover, it is not clear that ProbCons produces the most accurate alignments on average when a different set is considered. In addition to picking the best alignment out of a set of predicted alignments the alignment metric can be used to rank alignments based on their predicted accuracy, filtering out alignments of low quality.

We use the following definitions in the design of a metric-based protocol for ranking and filtering of alignments. Let $\mathcal{G} \triangleq \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_L\}$ be a dataset of L groups of sequences, where $\mathcal{G}_l \triangleq \{\mathcal{G}_l^1, \mathcal{G}_l^2, \ldots, \mathcal{G}_l^{K_l}\} \triangleq \{\sigma^1, \sigma^2, \ldots, \sigma^{K_l}\}$. Aligning each group of sequences with V different alignment programs $\mathcal{P} \triangleq \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_V\}$, we get the following set of $V \times L$ multiple alignments,

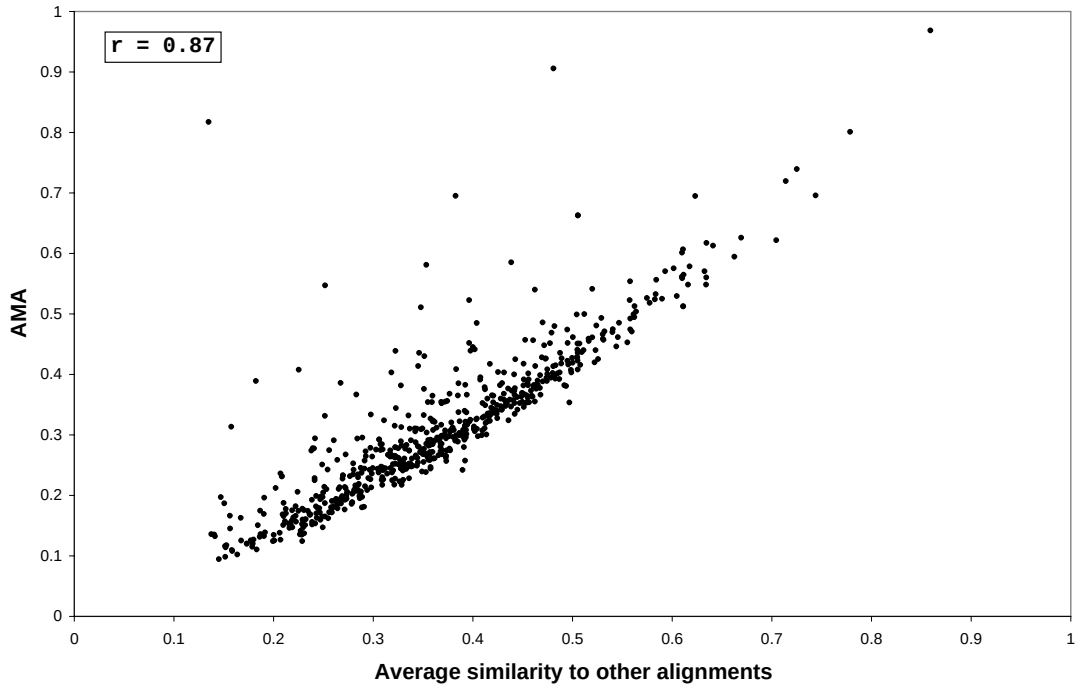


Figure 3.4. **Correlation between the AMA of CLUSTALW (as judged by reference alignments in SABmark), and average similarity to alignment produced by other programs.** Each dot in the plot corresponds to one CLUSTALW alignment in the SABmark Twilight-FP and Superfamilies-FP datasets. The x coordinate represents the average similarity (s) of the CLUSTALW alignment to the alignments produced by three other programs (MUSCLE, ProbCons, T-Coffee). The y coordinate represents the Alignment Metric Accuracy (AMA) of the CLUSTALW alignment as judged by the reference SABmark alignment.

$\mathcal{M}(\mathcal{G},\mathcal{P}) \triangleq \left\{ M_l^v \in \mathcal{A}_{|\mathcal{G}_l^1|,|\mathcal{G}_l^2|,\ldots,|\mathcal{G}_l^{K_l}|} | v \in \{1,2,\ldots,V\}, l \in \{1,2,\ldots,L\} \right\}$. Given a reference set of multiple alignments $\left\{ M_l^R | l \in \{1,2,\ldots,L\} \right\}$, the *accuracy* (average AMA) of any subset of alignments $\mathcal{M}_s \subseteq \mathcal{M}(\mathcal{G},\mathcal{P})$ is defined as

$$AMA(\mathcal{M}_s) \triangleq \frac{\sum_{M_l^v \in \mathcal{M}_s} AMA(M_l^v)}{|\mathcal{M}_s|} = \frac{\sum_{M_l^v \in \mathcal{M}_s} s(M_l^v, M_l^R)}{|\mathcal{M}_s|}. \tag{3.18}$$

The *average similarity* of a multiple alignment M_l^v to a set of multiple alignments is defined to be

$$s\left(M_l^v, \mathcal{M}(\mathcal{G},\mathcal{P})\right) \triangleq \frac{\sum_{M_l^{v'} | v' \neq v} s(M_l^v, M_l^{v'})}{V-1}. \tag{3.19}$$

A *ranking* of a set of multiple alignments $\mathcal{M}_s$ is defined by a function $\psi : \mathcal{M}_s \rightarrow \mathbb{R}$.

Given a ranking function it is natural to define the *top* u multiple alignments in $\mathcal{M}_s$.

$$top(\mathcal{M}_s, \psi, u) \triangleq \Big\{ \; M_l^v \in \mathcal{M}_s | \left(\forall l' v' \; M_{l'}^{v'} \in top(\mathcal{M}_s, \psi, u) \vee \psi(M_l^v) > \psi(M_{l'}^{v'}) \right) \\ \wedge \; |top(\mathcal{M}_s, \psi, u)| = u \qquad \Big\} \tag{3.20}$$

An optimal ranking of $\mathcal{M}_s$ is defined by $\psi^R \triangleq AMA$. The *percent improvement* of ranking ψ using top u alignments, is defined as

$$\mathcal{I}(\mathcal{M}_s, \psi, u) \triangleq \frac{AMA(top(\mathcal{M}_s, \psi, u))}{AMA(\mathcal{M}_s)} - 1. \tag{3.21}$$

Comparing a given ranking ψ to the optimal ranking ψ^R using top u alignments is done using the *improvement ratio* measure

$$\mathcal{I}^R(\mathcal{M}_s, \psi, u) \triangleq \frac{\mathcal{I}(\mathcal{M}_s, \psi, u)}{\mathcal{I}(\mathcal{M}_s, \psi^R, u)}. \tag{3.22}$$

Finally, *mean accuracy* ($\overline{AMA}$), *mean percent improvement* ($\overline{\mathcal{I}}$), and *mean improvement ratio* ($\overline{\mathcal{I}}^R$) are defined as

$$\overline{AMA}(\mathcal{M}_s, \psi) \triangleq \frac{\sum_{u=1}^{|\mathcal{M}_s|} AMA(top(\mathcal{M}_s, \psi, u))}{|\mathcal{M}_s|}, \tag{3.23}$$

$$\overline{\mathcal{I}}(\mathcal{M}_s, \psi) \triangleq \frac{\overline{AMA}(\mathcal{M}_s, \psi)}{AMA(\mathcal{M}_s)} - 1, \tag{3.24}$$

$$\overline{\mathcal{I}}^R(\mathcal{M}_s, \psi) \triangleq \frac{\overline{\mathcal{I}}(\mathcal{M}_s, \psi)}{\overline{\mathcal{I}}(\mathcal{M}_s, \psi^R)}. \tag{3.25}$$

A *candidate alignment set* (CAS) of a sequence group dataset $\mathcal{G}$ is a set of multiple alignments that includes at most one alignment for every alignment group in the dataset. A *complete candidate alignment set* (CCAS) is a CAS of size $|\mathcal{G}|$. For example, given $\mathcal{M}(\mathcal{G}, \mathcal{P})$, one can define the CCAS $\mathcal{M}_{\mathcal{G}}^{\mathcal{P}_v}$ of $\mathcal{G}$ by selecting all the alignments produced by program $\mathcal{P}_v$. However, the subset of $\mathcal{M}(\mathcal{G}, \mathcal{P})$ that is the most accurate CCAS is obtained by selecting the most accurate alignment for every group of sequences in $\mathcal{G}$.

$$\mathcal{M}_{\mathcal{G}}^{max} \triangleq \bigcup_{\mathcal{G}_l \in \mathcal{G}} \operatorname*{argmax}_{\mathcal{P}_v \in \mathcal{P}} AMA(M_l^v). \tag{3.26}$$

Computing $\mathcal{M}_{\mathcal{G}}^{max}$ directly requires to know the accuracy of any given alignment, which is unknown in most practical cases.

We propose a two step protocol for using the alignment metric as a control for alignment accuracy, when the correct alignment is not known. The input to the protocol is the alignment set $\mathcal{M}(\mathcal{G}, \mathcal{P})$. $\mathcal{G}$ is a dataset that includes L groups of sequences (several groups of homologous gene sequences, for example). For every group in $\mathcal{G}$ there are V multiple alignments, generated by the alignment programs in $\mathcal{P}$. The goal of the first step of the protocol is to pick the most accurate multiple sequence alignment for every group of sequences in the dataset. More formally, the goal is to find a CCAS that is a subset of $\mathcal{M}(\mathcal{G}, \mathcal{P})$ with accuracy as close as possible to $\mathcal{M}_{\mathcal{G}}^{max}$. Since the reference alignments are not known, there is no way to directly compute $\mathcal{M}_{\mathcal{G}}^{max}$. Instead, the protocol utilizes the strong empirical correlation between $AMA(M_l^v)$ and $s\left(M_l^v, \mathcal{M}(\mathcal{G}, \mathcal{P})\right)$, which can be computed directly without a reference alignment. The output of the first step of the protocol is the following CCAS:

$$\mathcal{M}_{\mathcal{G}}^{sim} \triangleq \bigcup_{\mathcal{G}_l \in \mathcal{G}} \underset{\mathcal{P}_v \in \mathcal{P}}{\operatorname{argmax}}\, s\left(M_l^v, \mathcal{M}(\mathcal{G}, \mathcal{P})\right). \tag{3.27}$$

Practically, $\mathcal{M}_{\mathcal{G}}^{sim}$ is constructed by repeating the following steps for every sequence group $\mathcal{G}_l \in \mathcal{G}$:

1. Align the sequences in $\mathcal{G}_l$ with all alignment programs in $\mathcal{P}$.
2. Compute the similarity $s(M_l^v, M_l^{v'})$ of each pair of alignments of $\mathcal{G}_l$.
3. Pick the alignment with the maximal average similarity to all other alignments, and add it to $\mathcal{M}_{\mathcal{G}}^{sim}$.

In cases where a group of sequences is hard to align accurately, none of the candidate alignments, including the alignment in $\mathcal{M}_{\mathcal{G}}^{sim}$, might be of high enough quality. For certain applications it is better to not produce any alignment in such cases, rather

	Twilight	Superfamilies	Twilight-FP	Superfamilies-FP	Overall
CLUSTALW	24.9 (< 0.0001)	43.8 (< 0.0001)	35.5 (< 0.0001)	37.0 (< 0.0001)	37.0 (< 0.0001)
MUSCLE	27.6 (< 0.0001)	46.4 (< 0.0001)	37.1 (< 0.0001)	38.9 (< 0.0001)	39.2 (< 0.0001)
ProbCons	37.3 (0.0027)	51.8 (0.3190)	55.7 (0.0149)	55.0 (0.9200)	51.1 (0.3298)
T-Coffee	35.6 (0.1184)	50.9 (0.0022)	54.1 (< 0.0001)	54.4 (< 0.0001)	50.1 (< 0.0001)
Average	31.4	48.2	45.6	46.3	44.4
Picked ($\mathcal{M}_{\mathcal{G}}^{sim}$)	36.1	51.5	56.9	55.2	51.1
Best ($\mathcal{M}_{\mathcal{G}}^{max}$)	39.5	53.9	58.0	57.2	53.3

Table 3.3. **AMA of alignments picked by the control protocol on the SABmark datasets.** The first four rows show the average AMA of alignments produced by different alignment programs. The last three rows show the average AMA of all alignments programs (Average), of the best alignments for every group (Best), and of the alignments picked by the control protocol (Picked). AMA values have been multiplied by 100. The Wilcoxon signed ranks test was used to measure the statistical significance of the difference between the picked alignments and the alignments produced by the four programs (P-values are shown in parenthesis. Values > 0.05 are not statistically significant).

than use a low quality alignment. The second part of the protocol aims at ranking alignments of different groups of sequences, discarding low quality alignments. This is done by defining the ranking function $\psi^{sim}(M_l^v) \triangleq s\left(M_l^v, \mathcal{M}(\mathcal{G}, \mathcal{P})\right)$. It is then possible to increase the average accuracy of the alignment set, by filtering out alignments that are not in $top(\mathcal{M}_{\mathcal{G}}^{sim}, \psi^{sim}, u)$, where u can be adjusted to control the number of alignments that are used. We define *filtration level* to be u/L, which is the percent of alignments that are in $top(\mathcal{M}_{\mathcal{G}}^{sim}, \psi^{sim}, u)$.

The above protocol has no mathematical guarantees, but our empirical results show that it works in practice. Table 3.3 shows the average AMA scores of the four alignment programs on the different SABmark datasets, as well as the average AMA scores of the alignments picked by the first part of the protocol. The upper bound on the AMA scores is given by $AMA(\mathcal{M}_{\mathcal{G}}^{max})$. The statistical significance of the difference between the AMA score of the alignments picked by the protocol ($AMA(\mathcal{M}_{\mathcal{G}}^{sim})$) and the ones produced by each of the alignment programs ($AMA(\mathcal{M}_{\mathcal{G}}^{\mathcal{P}_v})$) is measured using the Wilcoxon signed ranks test. Overall, the alignments picked by the protocol are as accurate as the alignments produced by the most accurate program (Prob-

Cons), and are significantly more accurate than the other three programs. For the Twilight-FP dataset the picked alignments are significantly more accurate than ProbCons, while for the Twilight dataset the ProbCons alignment are significantly more accurate the the picked alignments. There is no statistically significant difference in accuracy between the ProbCons and picked alignments for the other sets. The picked alignments are significantly more accurate than the other three programs on all sets, except for T-Coffee alignments on the Twilight dataset. Overall, the protocol picks 44.6% of the T-Coffee alignments, 42.4% of the ProbCons alignments, 10% of the MUSCLE alignments, and 3.6% of the CLUSTALW alignments.[3] These results show that the first part of the proposed protocol is able to pick alignments that are at least as accurate as the best program, without prior knowledge of which program is more accurate. This means that while different programs might perform differently on different sets of sequences, one can pick the best alignments on average by using the proposed protocol.

Figure 3.5 shows the result of applying the second part of the protocol on the Superfamilies dataset. The protocol is able to rank the alignments relatively well. For example, the accuracy increases from 0.515 to 0.641 when using $top(\mathcal{M}_{\mathcal{G}}^{sim}, \psi^{sim}, u)$, where u is adjusted to use the top 50% alignments, while an optimal ranking (ψ^{R}) increases the accuracy to 0.671 using the top 50% alignments in $\mathcal{M}_{\mathcal{G}}^{sim}$ or to 0.687 when using the top 50% alignments in $\mathcal{M}_{\mathcal{G}}^{max}$.

Table 3.4 summarizes the percent improvement in accuracy achieved at different filtration levels (u/L) for the different datasets, as well as the mean percent improvement over all filtration levels. To quantify the quality of the ranking procedure we measured the Pearson correlation (r) between the average similarity value (ψ^{sim}), and the average AMA score of the alignments above that threshold. Additionally, we

[3]Due to ties (more than one alignment with the highest average similarity to other alignments) these percentages sum to slightly more than 100%.

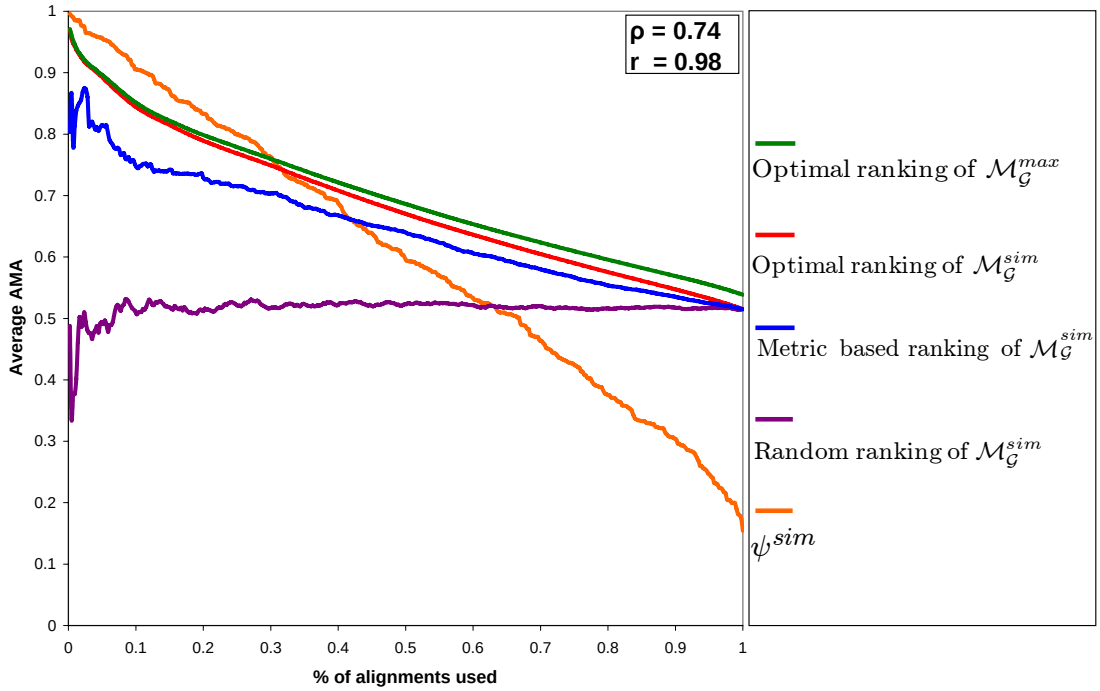


Figure 3.5. **Comparison of different rankings of the Superfamilies dataset alignments.** The accuracy of the metric-based ranking [$AMA(top(\mathcal{M}_{\mathcal{G}}^{sim}, \psi^{sim}, u))$; blue] is compared to that of an optimal ranking [$AMA(top(\mathcal{M}_{\mathcal{G}}^{sim}, \psi^{R}, u))$; red], and a random ranking of the alignments that have been picked by the first part of the alignment-control protocol [$AMA(top(\mathcal{M}_{\mathcal{G}}^{sim}, \psi^{rand}, u))$; purple]. In addition an optimal ranking of the best original alignments is shown [$AMA(top(\mathcal{M}_{\mathcal{G}}^{max}, \psi^{R}, u))$; green], as well as the similarity threshold used at each filtration level [ψ^{sim}; orange].

measured the Spearman rank correlation (ρ) between the average similarity and the AMA score of each alignment. Both tests show a statistically significant correlation (p-value < 0.05), however the ranking protocol is less powerful on the sets with false-positives. For example, on the Superfamilies-FP set, the average accuracy increases by only 12% when using the top 20% of the alignments, which is only 48% of the improvement possible with an optimal ranking, while on the Superfamilies set, there is a much more substantial improvement for the same filtering level (48%, which is 78% of the possible improvement). Generally, it seems that using the top 20% align-

% of alignments used (u/L)	Metric-based ranking (ψ^{sim}) accuracy	% improvement ($\mathcal{I}$)	Optimal ranking (ψ^{R}) accuracy	% improvement ($\mathcal{I}$)	Improvement ratio ($\mathcal{I}^{R}$)
Twilight [$L = 209, r = 0.97, \rho = 0.39$]					
10%	51.3	42%	63.0	74%	57%
20%	48.9	35%	56.4	56%	63%
50%	40.6	12%	46.7	29%	43%
100%	36.1	0%	36.1	0%	–
Means	42.8	18%	48.7	35%	53%
Superfamilies [$L = 425, r = 0.98, \rho = 0.74$]					
10%	75.1	46%	84.5	64%	71%
20%	72.9	42%	78.9	53%	78%
50%	64.1	24%	67.1	30%	81%
100%	51.5	0%	51.5	0%	–
Means	64.5	25%	68.5	33%	76%
Twilight-FP [$L = 209, r = 0.95, \rho = 0.51$]					
10%	65.3	15%	74.3	31%	48%
20%	64.0	13%	71.1	25%	50%
50%	61.4	8%	65.4	15%	53%
100%	56.9	0%	56.9	0%	–
Means	61.4	8%	66.0	16%	50%
Superfamilies-FP [$L = 425, r = 0.81, \rho = 0.43$]					
10%	63.8	16%	72.5	31%	50%
20%	62.0	12%	69.5	26%	48%
50%	58.7	6%	63.8	16%	41%
100%	55.2	0%	55.2	0%	–
Means	59.1	7%	64.5	17%	42%

Table 3.4. **Statistical evaluation of the metric-based ranking protocol on the SABmark datasets.** For every dataset the number of groups (L), the Pearson correlation (r) and the Spearman rank correlation (ρ) are given. The following measures are calculated at different filtration levels, as well as the means over all levels; the accuracy (average AMA multiplied by a 100) of the metric-based and optimal rankings, the percent improvement in AMA of the two rankings compared to the full set, and the ratio between the percent improvement of the metric-based ranking compared to the optimal ranking (improvement ratio).

ments results in a substantial improvement in absolute accuracy as well as a good improvement ratio compared to the optimal ranking.

3.3.3 Performance of the AMAP algorithm

Next, we investigated using pairwise alignments whether the AMAP algorithm can improve on the Viterbi and the Maximal Expected Accuracy (MEA) alignment algorithms for maximizing the AMA.

We first evaluated the algorithms with the same default parameters used in ProbCons ($\delta = 0.01993$, $\varepsilon = 0.79433$, $\pi_{match} = 0.60803$, and emission probabilities based on the BLOSUM62 matrix). Table 3.5 shows the results of the Viterbi algorithm and the AMAP algorithm with different gap-factor values on the SABmark Twilight Zone set, and Table 3.6 show the results on the SABmark Superfamilies set.

The results on both sets show the expected correlation between the gap-factor value and the f_M score, and the negative correlation between the gap-factor value and the f_D score. This validates the prediction that the gap-factor can be used as a tuning parameter for the recall/precision trade-off of matched characters.

When the gap-factor is set to 1 or higher the alignment accuracy is significantly better than Viterbi alignments. When the original MEA algorithm (AMAP with gap-factor set to 0) is used the alignment accuracy is almost identical to that of the Viterbi algorithm. The most accurate alignments were achieved by setting the gap-factor to values higher than 1 (40 in the Twilight Zone set and 16 in the Superfamilies set). We suspected that this is due to the fact that the default pair-HMM parameters underestimate the probability of insertion and deletions. To validate this hypothesis, we calculated the "true" transition probabilities for each alignment group using the reference alignments, and repeated the experiment.

The performance of the algorithms using the "correct" transition probabilities are shown in the right hand columns of Tables 3.5 and 3.6. As expected, with the correct parameters, the accuracy of the alignments achieved when the gap-factor is set to 1 are

Default transition probabilities			
Algorithm	f_D	f_M	AMA
Viterbi	27.2	16.3	28.0
$\gamma = 0$	**29.6**	17.7	29.0
$\gamma = 1$	28.1	19.7	37.4
$\gamma = 2$	25.8	22.8	45.2
$\gamma = 4$	22.4	27.6	51.5
$\gamma = 8$	18.9	33.1	54.8
$\gamma = 16$	15.9	38.9	56.4
$\gamma = 24$	14.3	43.3	56.7
$\gamma = 32$	13.1	46.1	56.8
$\gamma = 40$	12.4	48.0	**56.8**
$\gamma = 56$	11.3	**51.5**	56.7

"Correct" transition probabilities			
Algorithm	f_D	f_M	AMA
Viterbi	18.8	17.0	46.7
$\gamma = 0$	**25.9**	17.5	37.2
$\gamma = 1$	17.2	34.7	56.3
$\gamma = 2$	14.1	42.2	**57.3**
$\gamma = 4$	11.3	52.2	57.3
$\gamma = 8$	8.9	59.3	56.7
$\gamma = 16$	7.0	68.7	56.0
$\gamma = 24$	6.1	74.5	55.6
$\gamma = 32$	5.5	77.3	55.4
$\gamma = 40$	5.1	80.2	55.2
$\gamma = 56$	4.6	**83.1**	55.0

Table 3.5. **Performance of algorithm variants on the SABmark Twilight Zone set.** Entries show the f_D, f_M, and AMA scores of the Viterbi, and AMAP alignments with different gap-factor (γ) values on the SABmark Twilight Zone set, which includes 209 alignment groups. The first three columns show the results using default transition probabilities, and the last three columns show the results using transition probabilities calculated for each group from the reference alignments. All scores have been averaged over groups and multiplied by 100.

Default transition probabilities			
Algorithm	f_D	f_M	AMA
Viterbi	53.1	38.1	44.2
$\gamma = 0$	**54.8**	39.3	45.2
$\gamma = 1$	53.6	42.0	49.8
$\gamma = 2$	51.6	46.2	54.5
$\gamma = 4$	48.1	52.1	58.2
$\gamma = 8$	44.1	58.5	59.9
$\gamma = 12$	41.8	62.0	**60.2**
$\gamma = 16$	40.2	**64.3**	60.1

"Correct" transition probabilities			
Algorithm	f_D	f_M	AMA
Viterbi	46.8	41.3	53.3
$\gamma = 0$	**52.4**	40.1	49.2
$\gamma = 1$	45.4	56.1	60.5
$\gamma = 2$	41.8	63.4	**61.5**
$\gamma = 4$	37.9	70.9	61.2
$\gamma = 8$	34.1	75.9	60.0
$\gamma = 12$	31.9	78.4	59.2
$\gamma = 16$	30.5	**79.9**	58.6

Table 3.6. **Performance of algorithm variants on the SABmark Superfamilies set.** Entries show the f_D, f_M, and AMA scores of the Viterbi, and AMAP alignments with different gap-factor (γ) values on the SABmark Superfamilies set, which includes 425 alignment groups. The first three columns show the results using default transition probabilities, and the last three columns show the results using transition probabilities calculated for each group from the reference alignments. All scores have been averaged over groups and multiplied by 100.

very close to the best ($\gamma = 2$). Note that we did not modify the emission probabilities, which might be the reason $\gamma = 1$ did not maximize the actual accuracy. These results show that the AMAP algorithm significantly outperforms the Viterbi and

Parameters			f_D				f_M				AMA			
e_{match}	δ	ε	0	1	2	Vit	0	1	2	Vit	0	1	2	Vit
30	10.0	90	**7.1**	1.1	0.8	0.6	4.0	61.7	**73.7**	2.2	9.5	**51.0**	50.9	44.7
50	10.0	90	**23.6**	3.9	2.1	12.5	15.7	77.0	**83.5**	10.8	27.2	**52.2**	51.4	30.3
60	10.0	90	**33.6**	17.7	12.2	24.6	23.2	69.5	**82.9**	23.8	34.5	**57.2**	55.9	41.2
80	10.0	90	**61.2**	53.4	46.6	53.6	45.9	72.1	**81.9**	51.7	57.3	**70.8**	70.7	62.4
80	5.0	90	**83.2**	80.8	77.5	81.1	76.3	85.3	**89.9**	77.7	78.8	**82.4**	82.2	78.6
80	2.0	98	**94.4**	93.5	92.7	92.9	89.0	95.0	**96.2**	93.6	93.0	**95.4**	95.3	94.6
80	0.9	98	**97.4**	97.2	96.6	96.8	95.2	97.5	**98.2**	96.6	96.2	**97.2**	**97.2**	96.7
30	0.1	98	**94.5**	**94.5**	93.8	90.1	93.7	93.8	**94.1**	89.3	**93.1**	**93.1**	92.6	88.5
50	0.1	98	**99.4**	**99.4**	**99.4**	99.3	**99.4**	**99.4**	**99.4**	99.3	**99.2**	**99.2**	**99.2**	99.0
unrelated			100	100	100	100	0.0	0.0	0.0	0.0	72.2	96.8	**98.4**	94.3

Table 3.7. **Performance of algorithm variants on simulated data.** Entries show the performance of the Viterbi algorithm (Vit), and the AMAP algorithm with different settings of the gap-factor parameter (0, 1, and 2) using three accuracy measures (f_D, f_M, and AMA). The first three columns show the configuration of the pair-HMM parameters e_{match} (match emission probability), δ (gap initiation probability) and ε (gap extension probability), except for the last row for which random unrelated sequences have been aligned. Best results for every parameter configuration and measure are shown in bold. All numbers have been multiplied by 100.

MEA algorithms (61.5 AMA compared to 53.3 and 49.2 AMA on the Superfamilies dataset, and 57.3 AMA compared to 46.7 and 37.2 on the Twilight Zone dataset). Moreover, with the adjusted parameters, the Viterbi algorithm outperforms the MEA algorithm ($\gamma = 0$) on both datasets. This is due to the over-alignment problem of the MEA algorithm, which uses the expected f_D score as its objective function at the expense of the f_M and AMA scores. Note that the best AMA scores achieved with the default transition probabilities are very close to those of the correct probabilities, demonstrating that adjustment of the gap-factor parameter is able to compensate for bad estimation of the parameters of the underlying probabilistic model.

In order to further analyze the performance of the AMAP algorithm compared to the Viterbi and MEA algorithms, we also conducted simulation studies. Table 3.7 compares the performance of the Viterbi, MEA, and AMAP variants on different sets of simulated pairs of related and unrelated DNA sequences.

Data was simulated using a pair-HMM to generate aligned pairs of nucleotide sequences. The pair-HMM parameters included the transition probabilities δ (gap initiation) and ε (gap extension). For simplicity we fixed the initial probability π_{match} of starting in a Match state to be $1 - 2\delta$. For the emission probabilities we used a simple model that assigns equal probability ($\frac{1}{4}$) to any nucleotide in the Insert or Delete states, $\frac{e_{match}}{4}$ probability for a match in the Match state, and $\frac{1-e_{match}}{12}$ probability for a mismatch in the Match state, where e_{match} is the probability to emit a pair of identical characters in the Match state.

For every setting of the parameters we generated 10 reference alignments with $\min(n, m) = 1000$. An identical pair-HMM with the same parameters was then used to compare the performance of the Viterbi algorithm and MEA algorithm with gap-factor values of (0, 1, and 2). We treat every set of 10 alignments as one big alignment, and calculate the accuracy ($s(h^p, h^r)$) of the predicted alignments, the f_D, and f_M scores. In addition to the simulated reference alignment generated from the pair-HMM, we also generated 10 pairs of unrelated random sequences of length 1000 each with equal probability for every character. All algorithms have been evaluated on the resulting reference alignments, which include no H characters, using the probabilities 0.8, 0.1, and 0.9 for the e_{match}, δ, and ε parameters respectively.

The simulation results demonstrate that the AMAP algorithm produces alignments that are more accurate than the Viterbi and MEA alignment algorithms on both closely related and distant sequences. As expected the best f_D scores are achieved using the MEA algorithm ($\gamma = 0$), the best f_M scores when $\gamma = 2$, and the best AMA when $\gamma = 1$.

It is interesting to note that for distant sequences with larger gap initiation probability (δ), the Viterbi algorithm has better AMA score than the MEA algorithm ($\gamma = 0$). This again emphasizes the main weakness of MEA, which tends to over-align

unalignable regions. This problem is even more pronounced when aligning unrelated sequences. The MEA algorithm performs poorly compared to the AMAP algorithm and even the Viterbi algorithm, achieving a mere 72.2 AMA scored compared to 96.8 and 94.8 respectively. This is due to the fact that the MEA algorithm wrongly aligns 2781 character pairs, compared to 157, 316, and 572 in the case of $\gamma = 2$, $\gamma = 1$, and Viterbi alignment respectively.

3.4 Discussion

We have proposed a metric for alignment space, and shown how it can be used both to judge the accuracy of alignments, and as the basis for an optimization criteria for alignment. The importance of the metric lies in the fact that if two alignments are far from each other, we can conclude that at least one of them is inaccurate. This is a direct consequence of the triangle inequality. More importantly, we show that when alignments made by widely used software programs are compared to each other they are far apart, thus quantitatively confirming that multiple alignment is a difficult problem. Although we see that the recall of many programs is high, i.e., many of the residues that should be aligned together are correctly aligned, it is also the case that many residues are incorrectly aligned. This is particularly evident in results from the Twilight-FP and Superfamilies-FP datasets that contain unrelated sequences. If functional inferences are to be made from sequence alignments, it is therefore important to control for precision, and not only recall. Our alignment algorithm, AMAP, which maximizes the expected AMA, outperforms existing programs on benchmark datasets.

Most existing multiple alignment benchmark datasets include only alignments of "core blocks", and it is therefore only possible to measure the recall of matches (f_D), and not their precision (f_M) or the AMA. However, the fact that it is harder to

construct datasets that can measure the latter two does not mean that alignment algorithms should maximize recall at the expense of precision. The AMAP algorithm is the first to have direct control over the inherent recall/precision trade-off using the gap-factor parameter. In many cases, such as when using MSA for phylogenetic tree reconstruction, or for identification of remote homology, higher precision is preferred over higher recall. In addition, as we have shown, tuning the gap-factor parameter can in some cases compensate for poor parameter estimation of the underlying probabilistic model (pair-HMM). Further work is needed to develop methods for automatic adjustment of this parameter for a given dataset when the probabilistic parameters do not fit the data very well, and a reference alignment is not available.

In the typical case where reference alignments are not available, our empirical observation that the distances between alignments correlate strongly with the accuracy of the programs that generated them can be used to discard inaccurate alignments. It is possible that more sophisticated strategies based on this principle could further help in quantitatively assessing alignment reliability.

In the next chapter we investigate how the basic posterior decoding methods we have developed for pairwise alignments can be extended to the task of multiple sequence alignment.

Chapter 4

Multiple Alignment by Sequence Annealing

In the previous chapter we developed posterior decoding methods for pairwise sequence alignments. In this chapter we extend these methods to the problem of multiple sequence alignment (MSA). We develop an algorithm, which we call *sequence annealing*, and show that it outperforms existing state of the art methods for MSA.[1]

The multiple sequence alignment (MSA) problem is to find all homologous amino acids, or nucleotides, among multiple sequences. This problem is similar in some respects to the protein folding problem: each multiple alignment can be evaluated according to the homology relationships it specifies in the same way that the Gibbs free energy can be computed for a protein conformation. However the differences between the problems reveal distinct fundamental hurdles that lead to very different computational problems. In protein folding the free energy of a conformation is derived from a clear understanding of the underlying physics, whereas in multiple alignment, the mechanisms of evolution are not well understood, leading to uncertainty over how

[1]This chapter is an extended version of Schwartz and Pachter (2007)

to evaluate multiple alignments. There is another fundamental difference: while the space of protein conformations is infinite and difficult to explore completely, there are natural "moves" in the space based on changing backbone torsion angles or side chain conformations. The space of multiple alignments, while also difficult to explore completely, is discrete and finite. However it has been unclear how to perform "small" steps in the space, making it difficult to accurately align single amino acids or nucleotides. In this chapter we show that it is possible to efficiently explore the space of multiple alignments using the smallest possible steps.

Unlike pairwise alignment, MSA is a computationally hard problem, since the search space of possible alignments grows exponentially with the number of sequences, and it has been proven to be NP-hard for many standard formulations of the problem (Wang and Jiang, 1994; Bonizzoni and Vedova, 2001; Just, 2001; Elias, 2006). Existing programs for multiple sequence alignment are mostly based on the idea of *progressive alignment* (Feng and Doolittle, 1987), which is widely used, both by first generation programs such as CLUSTALW (Thompson *et al.*, 1994) and T-Coffee (Notredame *et al.*, 2000), as well as more recent programs such as Align-m (Van Walle *et al.*, 2005), MUSCLE (Edgar, 2004) and ProbCons (Do *et al.*, 2005). In terms of the space of multiple alignments, we can think of progressive alignment as beginning with the "null alignment" where no two sequences are aligned, and proceeding via large steps to arrive at a complete alignment. Each step consists of aligning either a pair of sequences to each other, or the alignment of two partial alignments of subsequences to each other (Figure 4.1). This is a fundamental weakness of progressive alignment. For example, at the very first step, two entire sequences are aligned, possibly incorrectly, because other informative sequences are not used. This has been addressed by *iterative refinement* (Gotoh, 1996), which improves on progressive alignment by iteratively realigning subsets of sequences, but still often fails to correct complex alignment errors involving multiple sequences.

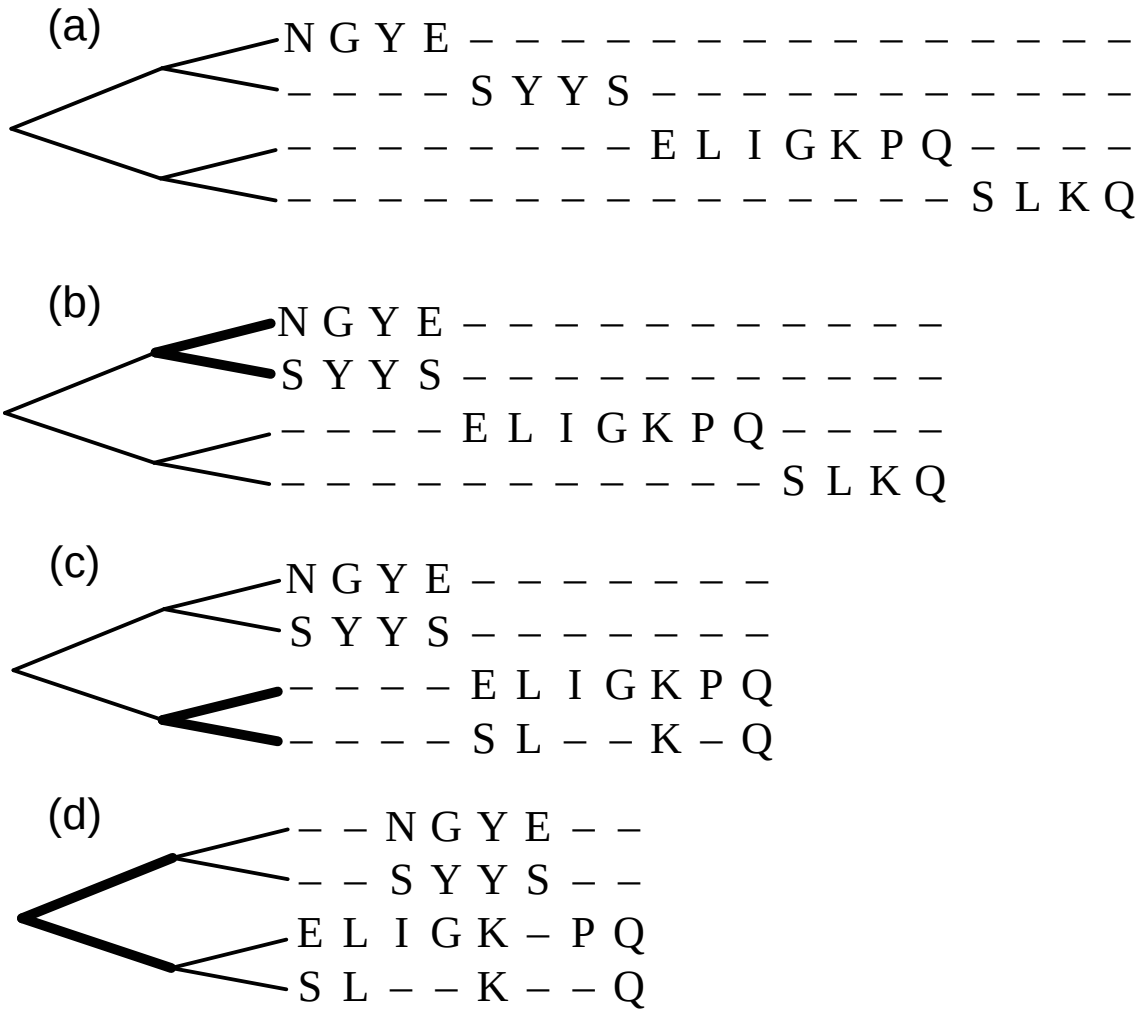


Figure 4.1. **Example of progressive alignment.** (a) A guide tree is constructed based on the sequence similarity of the four sequences. (b) The two most similar sequences are aligned. (c) A second pair of sequences are aligned according to the guide tree. (d) The two alignments are aligned together to construct the final MSA.

Another approach to multiple sequence alignment was introduced by Morgenstern *et al.* (1996) and pursued in a series of papers developing the DIALIGN program (Morgenstern *et al.*, 1998; Subramanian *et al.*, 2005). The main idea was to address the problems of progressive alignment by incrementally aligning matching segments to each other, while preserving the consistency of the alignment. This *segment-to-segment* alignment approach effectively reduced the size of the steps taken by progres-

sive alignment methods. A key ingredient was the careful formulation of the multiple alignment problem via precise definitions of partial and global multiple alignments (Morgenstern *et al.*, 1999). We discuss these ideas in detail in Section 4.1.

In Section 4.2 we introduce the method of sequence annealing. We build alignments one match at a time, "annealing" positions that are more likely to be homologous, first. Using fast algorithms for online topological ordering (Katriel and Bodlaender, 2006), we are able to rapidly construct a consistent global multiple alignment. We remove the requirement of DIALIGN that alignment proceed by segment-to-segment comparison, and allow for segments of size 1. This eliminates the need for many of the heuristics incorporated in DIALIGN. In order to correctly align single residues, we use the posterior-decoding methods for pairwise alignments we introduced in the previous chapter.

In Section 4.3 we show that sequence annealing improves on all existing methods for protein multiple sequence alignment. Not only is sequence annealing more accurate, it is also very fast, even though it is based on performing very small steps in multiple alignment space.

4.1 Alignment posets

We use the notation σ_i^a to denote the i^{th} element of a sequence $\sigma^a \triangleq \sigma_1^a, \ldots, \sigma_n^a$ of length n. By a set of sequence characters $S \triangleq \{\sigma^1, \ldots, \sigma^K\}$ we mean the set of $n_1 + n_2 \cdots + n_K$ sequence characters that form the sequences $\sigma^1, \sigma^2, \ldots, \sigma^K$ of lengths $n_1, n_2, \ldots, n_K$ respectively. A *partial global multiple alignment* of sequence characters $S \triangleq \{\sigma^1, \ldots, \sigma^K\}$ is a partially ordered set $P \triangleq \{c_1, \ldots, c_m\}$ together with a surjective function $\varphi : S \to P$ such that $\varphi(\sigma_i^a) < \varphi(\sigma_j^a)$ if $i < j$. The elements of P correspond to columns of the multiple alignment, and the partial order specifies the

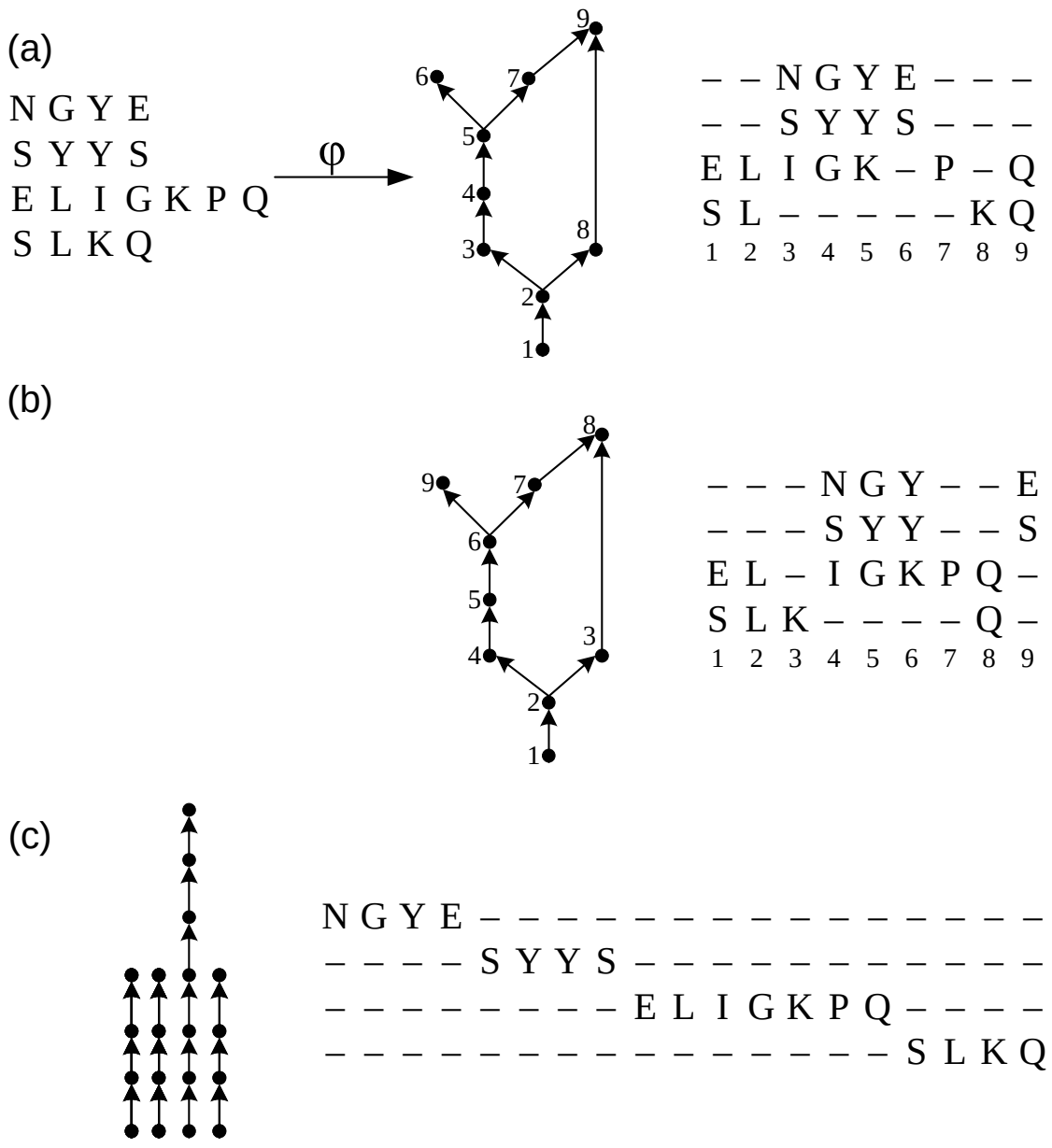


Figure 4.2. **Alignment posets.** (a) A set of four sequences, an alignment poset together with a linear extension, and a global multiple alignment. The function from the set of sequence elements to the alignment poset that specifies the multiple alignment is not shown, but is fully specified by the diagram on the right. For example, the second element in the first sequence is $\sigma_2^1 = G$, and $\varphi(\sigma_2^1)$ corresponds to the fourth column of the multiple alignment. (b) A different linear extension for the same alignment poset results in a different (but equivalent) global multiple alignment. (c) The null alignment of the same four sequences.

order in which columns must appear. We call P an *alignment poset*, and note that unless P is a total order, there are columns of the partial multiple alignment whose order is unspecified. A *linear extension* of a partially ordered set $P \triangleq \{c_1, \ldots, c_m\}$ is a permutation of the elements $c_1, \ldots, c_m$ such that whenever $c_i < c_j$, $i < j$. A *global multiple alignment* is a partial global multiple alignment together with a linear extension of the alignment poset P (Figure 4.2). Note that although different linear extensions of the same alignment poset result in different global multiple alignments, they are all equivalent, since they convey the same homology predictions, with the only difference being order of gapped columns. We therefore claim that the goal of a multiple sequence alignment algorithm is to find the optimal alignment poset, while the specific global multiple alignment that is displayed to the user is simply an artifact of the standard representation of multiple alignments in matrix form. A similar representation using *directed acyclic graphs* instead of posets is used by the Partial Order Alignment (POA) program (Lee *et al.*, 2002), and has also been used in the A-Bruijn alignment (ABA) program (Raphael *et al.*, 2004).

There is an important (trivial) case of a partial multiple alignment. A null global multiple alignment of K sequence $S \triangleq \{\sigma^1, \ldots, \sigma^K\}$ of lengths $n_1, \ldots, n_K$ is a partial global multiple alignment M_{Null} where the alignment poset P has size $\sum_k n_k$. Note that in this case P must be a disjoint union of K chains (Figure 4.2(c)).

Let $\mathcal{M}$ be the set of all partial multiple alignments of a set of sequences S. A *scoring function* for multiple alignments is a function $f : \mathcal{M} \rightarrow \mathbb{R}$ which assigns a "score" to each partial multiple alignment. In what follows, we make use of the AMA-based utility function (U_γ) we defined in Section 3.1 for pairs of sequences, by defining the scoring function to be the expected utility ($f \triangleq E(U_\gamma)$). The scoring function of a multiple alignment is defined to be the sum-of-pairs score for all of the pairwise alignments.

4.2 Sequence annealing

The goal of global multiple alignment is to find $\text{argmax}_{M \in \mathcal{M}}(f(M))$ where M ranges over all partial multiple alignments and f is a scoring function. In principle, f can be evaluated at every partial multiple alignment but this is not feasible in practice, as the set $\mathcal{M}$ is large (Dress *et al.*, 1998).

We formalize a hill climbing approach to multiple alignment as follows: Let S be a collection of K sequences of length $n_1, \ldots, n_K$ and let $L \triangleq \sum_i n_i$. A *sequence annealing* for S with scoring function f is a nesting of partial global multiple alignments $M_L \supset M_{L-1} \cdots \supset M_r$ where the alignment poset P_i associated to M_i has $|P_i| = i$ for all i, P_L is a null multiple alignment of S, and $f(M_{i-1}) \geq f(M_i)$. Note that each multiple alignment M_i consists of a function $\varphi^{M_i} : S \to P$.

By definition, the sequence annealing process can only proceed by minimal steps that reduce the number of columns in the (partial) alignment by one. This can only be done by *merging* two existing columns $c_k^{M_{i+1}}, c_l^{M_{i+1}}$ into $c_k^{M_i}$, such that $\forall \sigma_j^a$, $\varphi^{M_i}(\sigma_j^a) = c_k^{M_i}$ if $\varphi^{M_{i+1}}(\sigma_j^a) = c_l^{M_{i+1}}$, or $= \varphi^{M_{i+1}}(\sigma_j^a)$ otherwise. The difficulty in finding a sequence annealing is that not all pairs of columns can necessarily be merged in a partial multiple alignment. However it is natural to consider the algorithm in Figure 4.3. The algorithm starts with the null alignment, and proceeds by merging columns until no legal *merge* operation that increases the scoring function exists.

The time complexity of the algorithm depends on (a) the number of iterations of the main loop, (b) the time it takes to check the condition in line 3, and (c) the time it takes to merge the two columns. The time for (a) is simply $L - r + 1$. (c) can be done in $O(K)$ time, since there are at most $K - 1$ sequence elements that are affected by each *merge* operation. The challenging step of the algorithm is (b).

In order to perform step (b) efficiently a weight $w(c_k^{M_L}, c_l^{M_L})$ is assigned to each

1: $M_L \leftarrow M_{Null}$

2: $i \leftarrow L$

3: **while** $\exists c_k^{M_i}, c_l^{M_i}$ such that
$c_k^{M_i}$ and $c_l^{M_i}$ can be merged to produce M' and
$f(M') \geq f(M_i)$ **do**

4: $M_{i-1} \leftarrow M'$

5: $i \leftarrow i - 1$

6: **end while**

Figure 4.3. **A general sequence annealing algorithm.**

pair of columns in M_L. Candidate pairs with positive weights are placed in a heap in $O(Lk \log(Lk))$ time.[2] At every iteration of the algorithm the candidate pair (also referred to as *edge*) p with the highest weight is drawn from the heap in $O(1)$ time. The weight of p is recalculated to account for *merge* operations that involved the positions in p. If the new weight w' is lower than the weight of the current top candidate in the heap the edge is reinserted into the heap with the new weight w' in $O(\log(Lk))$ time, otherwise $merge(p)$ is performed. If $merge(p)$ fails because it introduces a cycle into the poset then p is discarded and the next candidate pair is considered.

The *merge* operation can be done efficiently using an *online topological ordering* algorithm. Given a directed acyclic graph G, the *topological ordering problem* is to find a function $ord : V \rightarrow \mathbb{N}$ such that if $i \rightarrow j$ then $ord(i) < ord(j)$. Directed acyclic graphs are equivalent to partially ordered sets, and the topological ordering problem is just the problem of finding a linear extension of the poset (the former terminology is used in computer science, and the latter terminology in mathematics). It is easy to see that the topological ordering problem is trivial (Tarjan, 1972). The

[2]Here we assume that the number of candidate pairs with positive weights per sequence position is of the order $O(K)$.

online topological ordering problem is the topological ordering problem where edges appear one at a time. This problem was first considered in Alpern *et al.* (1990); Marchetti-Spaccarnela *et al.* (1996). Significantly for our application, the problem admits efficient algorithms. We omit a detailed complexity analysis and refer the reader to Ajwani *et al.* (2006). In our implementation we adopted the algorithm of Pearce and Kelly (2004), which has good time complexity and is relatively easy to implement. The online topological ordering is used to quickly identify whether a candidate pair is valid (does not introduce a cycle), and to update the linear extension of the current global multiple alignment after each *merge* operation. Figure 4.4 shows an example of a successful *merge* operation, followed by a failed *merge*.

The correctness of the algorithm requires that the weight function $w(p)$ has the property that merging a pair with positive weight will result in M_{i-1} for which $f(M_{i-1}) \geq f(M_i)$. Additionally, a good weight function should be correlated with $\delta \triangleq f(M_{i-1}) - f(M_i)$, such that pairs that have a higher potential contribution to the scoring function will be merged earlier in the sequence annealing process. Since w can change after each *merge* operation, a naive algorithm will have to update the weights for all the affected pairs after every such operation, and reinsert them to the heap with the updated weights. To reduce the complexity of the algorithm we currently restrict w to be independent of *merge* operations, or to have the property that the weight of a pair can only be reduced after a *merge* operation. The second property enables a rapid evaluation of w. Only when a pair p_l is fetched from the top of the heap, its weight $w'(p_l)$ is recalculated. Since $w'(p_l) \leq w(p_l)$, then $w'(p_l) = \max_{p_k \in H}\{w'(p_k)\}$ if and only if $w'(p_l) \geq w(p_h)$, where p_h is next top candidate pair in the heap H.

In order to specify a weighting function w, we first need to define a utility function for the global multiple alignment problem. Such a utility function should be derived from the quality measures used to evaluate alignments. Like with pairwise alignments, the most widely used accuracy measure for multiple sequence alignment is the

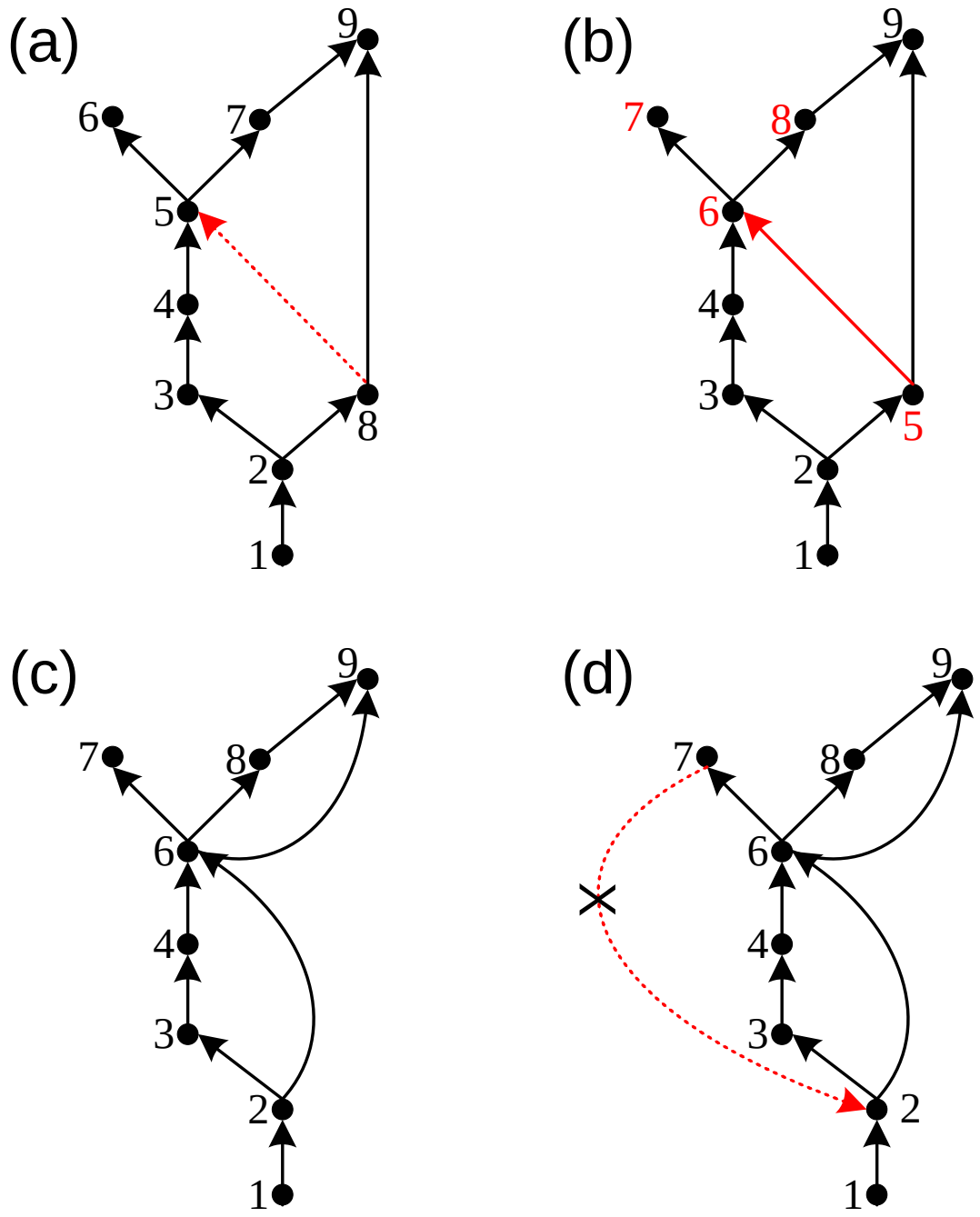


Figure 4.4. **Online topological ordering and alignment posets.** (a) During the sequence annealing process, the next candidate *merge* operation of columns 5 and 8 is drawn from the heap. (b) The new edge is added to the poset, and the online topological ordering procedure updates the current linear extension. (c) Since no cycle are found during the online topological ordering procedure the two columns are merged. (d) The next candidate edge connects columns 2 and 7. However, the online topological ordering procedure locates a cycle, and the edge is discarded.

developer score (f_D) (Sauder *et al.*, 2000), which is equivalent to the *sum-of-pairs* score (SP) (Thompson *et al.*, 1994). We have seen that using f_D to derive the utility function leads to the *over alignment* problem, in which many unrelated positions are aligned without much support, resulting in low precision, especially when aligning unrelated sequences, or sequences with unalignable regions. Moreover, since most alignment programs are compared based on f_D alone, programs that are considered to be very accurate can actually produce poor alignments that do not distinguish between related (homologous) and unrelated sequence-regions. To solve this problem we extend the AMA measure and the U^γ utility function, which we developed for pairwise alignment, to MSA using a sum-of-pairs scheme.

Defining a complete probabilistic model for MSA is infeasible for more than a few sequences, since not only the search space is huge, the parameter space also grows exponentially with the number of sequences. An alternative, is to use a pairwise probabilistic model, like the pair-HMM we described in Section 2.4. Given such a probabilistic model for pairwise alignments, it is possible to compute the pairwise posterior probabilities for every pair of sequences in quadratic time with respect to the number of sequences. The utility function is then defined as a sum of the pairwise utility functions (Equation (4.1)).

$$U_\gamma(M^r, M) \triangleq \sum_{\sigma^a,\sigma^b|a\neq b} \left(2 \sum_{\left\{(j,k)|\varphi^M(\sigma_j^a)=\varphi^M(\sigma_k^b)\right\}} \mathbf{1}\left\{\varphi^{M^r}(\sigma_j^a) = \varphi^{M^r}(\sigma_k^b)\right\} \right.$$
$$\gamma \left(\sum_{\left\{j|\forall\sigma_k^b\varphi^M(\sigma_j^a)\neq\varphi^M(\sigma_k^b)\right\}} \mathbf{1}\left\{\forall\sigma_k^b\varphi^{M^r}(\sigma_j^a) \neq \varphi^{M^r}(\sigma_k^b)\right\} \right.$$
$$\left. \left. \sum_{\left\{k|\forall\sigma_j^a\varphi^M(\sigma_j^a)\neq\varphi^M(\sigma_k^b)\right\}} \mathbf{1}\left\{\forall\sigma_j^a\varphi^{M^r}(\sigma_j^a) \neq \varphi^{M^r}(\sigma_k^b)\right\} \right) \right), \quad (4.1)$$

The gap-factor (γ) parameter serves the same role as in the pairwise alignment case, which is to control the recall/precision trade-off. The objective of a MSA algorithm

is to maximize the scoring function (expected utility) using the pairwise posterior probabilities (Equation (4.2)). However, a naive approach that optimizes each pairwise alignment separately, is bound to fail since the pairwise alignments are coupled by the poset constraints.

$$f_\gamma(M) \triangleq E_{M^t}\left(U_\gamma(M^t, M)\right) = \sum_{\sigma^a,\sigma^b|a\neq b}\left(2\sum_{\left\{(j,k)|\varphi^M(\sigma_j^a)=\varphi^M(\sigma_k^b)\right\}} P(\sigma_j^a\diamond\sigma_k^b|\sigma^a,\sigma^b,\theta) + \gamma\left(\sum_{\left\{j|\forall\sigma_k^b\varphi^M(\sigma_j^a)\neq\varphi^M(\sigma_k^b)\right\}} P(\sigma_j^a\diamond -|\sigma^a,\sigma^b,\theta) + \sum_{\left\{k|\forall\sigma_j^a\varphi^M(\sigma_j^a)\neq\varphi^M(\sigma_k^b)\right\}} P(-\diamond\sigma_k^b|\sigma^a,\sigma^b,\theta)\right)\right). \tag{4.2}$$

Good weight functions should be correlated with the change in the scoring function that results from a *merge* operation, $\delta \triangleq f_\gamma(M_{i-1}) - f_\gamma(M_i)$. When a pair of columns (c_k, c_l) are merged, the match posterior probabilities of newly aligned pairs contributes positively to the scoring function, while the gap posterior probabilities of these positions are subtracted from the scoring function. More formally,

$$\text{Let } \delta_{match}(c_k,c_l) \triangleq 2\sum_{\left\{\sigma_i^a\in\varphi^{-1}(c_k)\right\}}\sum_{\left\{\sigma_j^b\in\varphi^{-1}(c_l)\right\}} P(\sigma_i^a\diamond\sigma_j^b|\sigma^a,\sigma^b,\theta),$$

$$\text{and let } \delta_{gap}(c_k,c_l) \triangleq \sum_{\left\{\sigma_i^a\in\varphi^{-1}(c_k)\right\}}\sum_{\left\{\sigma_j^b\in\varphi^{-1}(c_l)\right\}} P(\sigma_i^a\diamond -|\sigma^a,\sigma^b,\theta) + \sum_{\left\{\sigma_i^a\in\varphi^{-1}(c_k)\right\}}\sum_{\left\{\sigma_j^b\in\varphi^{-1}(c_l)\right\}} P(-\diamond\sigma_j^b|\sigma^a,\sigma^b,\theta),$$

$$\text{then } \delta(c_k,c_l) \triangleq \delta_{match}(c_k,c_l) - \gamma\delta_{gap}(c_k,c_l). \tag{4.3}$$

We propose two weight functions that are derived from $f_\gamma(M)$. Both have a static and a dynamic version. The static weights are computed once using the null align-

ment, and stay constant during the sequence annealing process, while the dynamic weights are recomputed (rapidly) for each top candidate column pair (c_k, c_l):

$$w^{\gamma}_{maxstep}(c_k, c_l) \triangleq \begin{cases} \frac{\delta(c_k,c_l)}{|\varphi^{-1}(c_k)||\varphi^{-1}(c_l)|} & \text{if } c_k \neq c_l \\ -\infty & \text{otherwise} \end{cases}, \tag{4.4}$$

and

$$w^{\gamma}_{tgf}(c_k, c_l) \triangleq \begin{cases} \frac{\delta match(c_k,c_l)}{\delta_{gap}(c_k,c_l)} - \gamma & \text{if } c_k \neq c_l \\ -\infty & \text{otherwise} \end{cases}. \tag{4.5}$$

The first weight function $w^{\gamma}_{maxstep}$ assigns the highest weight to the pair of columns p_{max}, for which the increase in the scoring function divided by the number of newly matched pairs is maximal. It is easy to see that this weight function satisfies the requirement that weights can only decrease following a *merge* operation, since it is calculated by *averaging* over all newly matched residue pairs.

While $w^{\gamma}_{maxstep}$ is a good hill-climbing heuristic, and is useful for finding a local maxima of f_γ at the final alignment M_r, it has no guarantees for the alignments produced during the sequence annealing process. The second weight function w^{γ}_{tgf} addresses this issue. It is motivated by the empirical observation that when the optimal pairwise alignment is found using dynamic programming, 99.8% of the pairs that are matched in alignments that use higher gap-factor values also appear in alignments that use lower gap-factor values. Sequence annealing with w_{tgf} emulates the process of slowly reducing the temperature (gap-factor in our analogy), allowing pairs of columns whose weights become positive to align. Therefore at any step of the sequence annealing process the scoring function that is being optimized is $f_{temp}(M)$, where $temp \triangleq w^{\gamma}_{tgf}(p_{max}) + \gamma = \frac{\delta match(c_k,c_l)}{\delta_{gap}(c_k,c_l)}$.

The following lemma shows that w^{γ}_{tgf} can only decrease after a *merge* operation.
Lemma: If $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ are positive numbers then

$$\max_k \frac{x_k}{y_k} \geq \frac{\sum_k x_k}{\sum_k y_k}.$$

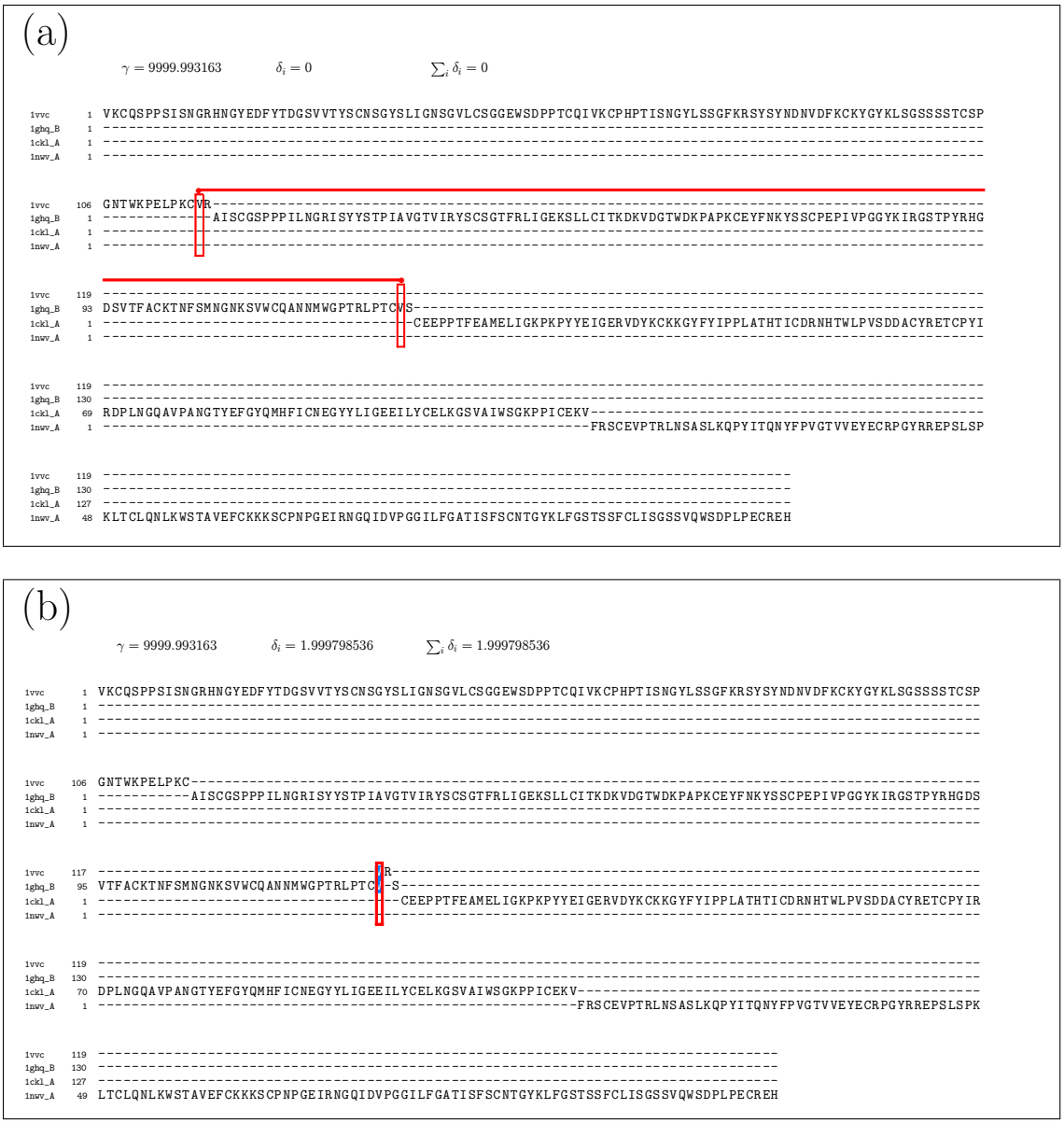


Figure 4.5. **The first step of the sequence annealing algorithm with weight function w^1_{tgf} on four protein sequences.** (a) Starting from the null alignment the first candidate column pair with weight of ≈ 10000 is fetched from the heap. (b) The two columns are merged, adding $\delta \approx 2$ to the scoring function. Affected columns in the global multiple alignment are rearranged based on the new liner extension.

Proof: We may assume without loss of generality that $\sum_k x_k = \sum_k y_k = 1$. If $x_k < y_k$ for all k, then $\sum_k x_k < \sum_k y_k$ which is a contradiction. □

Figures 4.5 and 4.6 demonstrate the behavior of the sequence annealing algorithm with the w^1_{tgf} weight function. Figure 4.5 shows the first step in a sequence annealing

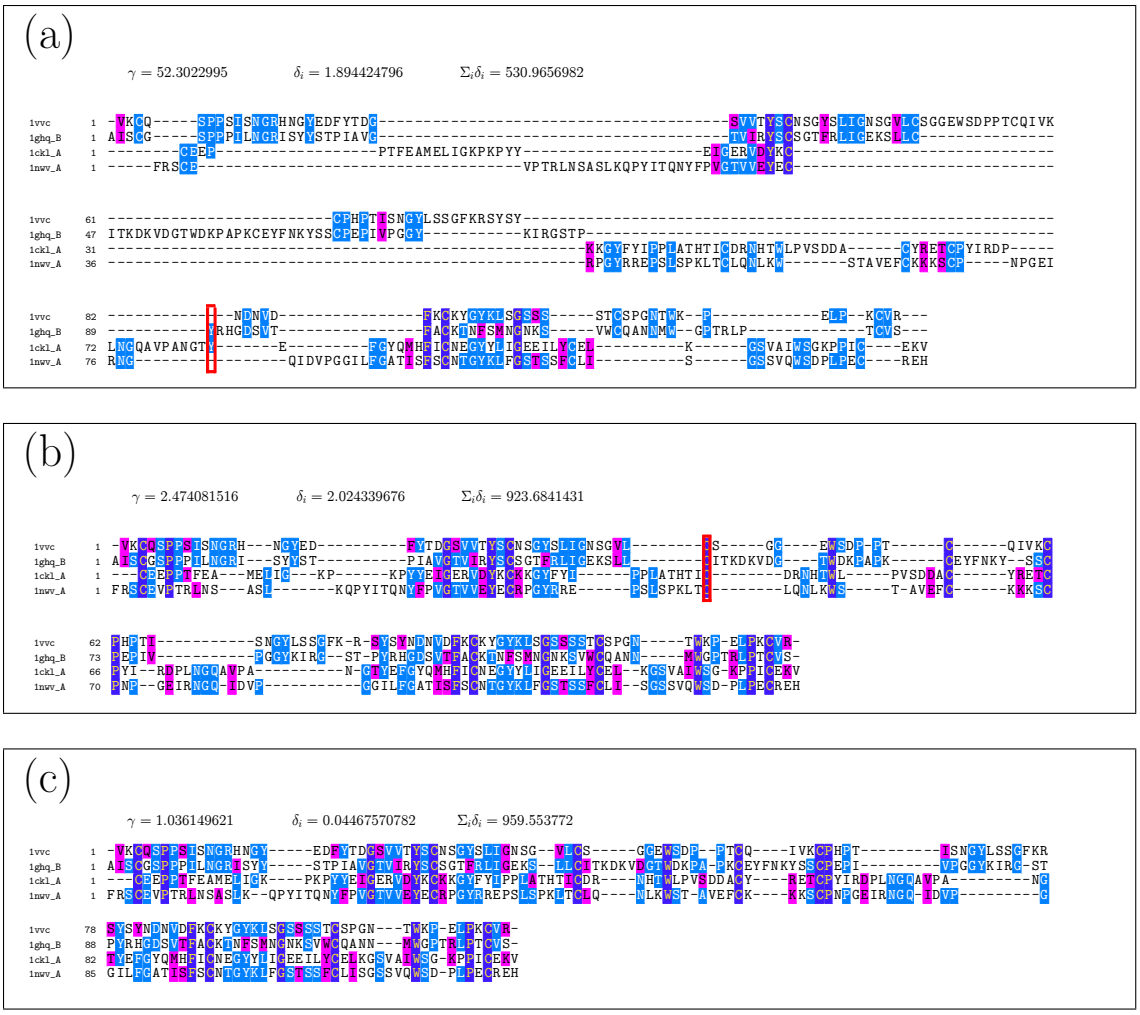


Figure 4.6. **Later steps in the sequence annealing of four protein sequences with weight function w^1_{tgf}.** (a) At temperature $\gamma \approx 52$ different pairs of sequences are aligned at different parts of the multiple alignment. (b) At temperature $\gamma \approx 2.5$ all fully conserved columns are aligned. (c) The final alignment. The sequence annealing process stops at temperature $\gamma \approx 1.04$ with total scoring function improvement of 959.55 over the null alignment. Next candidate column pair has weight < 1, which can only decrease the scoring function.

alignment of four protein sequences. Starting from the null alignment, the temperature (effective gap factor) is reduced until the first pair of columns "anneal" (merge). Note how other columns rearrange in the global multiple alignment, based on the new linear extension that is produced by the online topological ordering procedure. Figure 4.6 (a) is a global multiple alignment of the same four sequences later in the sequence annealing process (temperature $\gamma \approx 52$). Unlike progressive alignment, with

the sequence annealing process different columns can align residues from different sequences. In this example there are columns that align residues from the first and second sequences; third and fourth sequences; second and third sequences; first, second and fourth sequences; and so forth. Another important property of the sequence annealing process is that a valid global multiple alignment exists at every step of the algorithm. The alignments in the earlier stages of the sequence annealing process have very high expected precision with lower expected recall, while later alignments increase the expected recall at the expense of expected precision. In Figure 4.6 (b) all the fully conserved columns are aligned when the temperature is at $\gamma \approx 2.47$, and the overall gain in scoring function compared to the null alignment is $\Sigma_i \delta_i \approx 923.68$. The sequence annealing process stops (Figure 4.6 (c)) when the top candidate column pair requires a temperature smaller than the target gap-factor (1 in our example) to get aligned, since at this point any additional merge operation can only reduce the value of the scoring function. Note, that in our example there are still valid pairs of columns that can be merged without introducing a cycle. when using a gap-factor of 0 the process stops only when no such column pairs exist. However, this results in over-alignment and reduced AMA and f_M scores.

4.3 Results

We implemented sequence annealing by extending our pairwise posterior decoding alignment program, AMAP. The new program, AMAP 2.0, is abbreviated by AMAP in this chapter [3].

The following variations of AMAP were tested:

[3]AMAP uses the ProbCons parameters with a single pair of gap states to generate pairwise posterior probabilities. The latest version of ProbCons uses two additional gap states, which we do not use in the current version of AMAP. We also slightly modified the initial state probabilities to $\{\pi_{match},\pi_{insert},\pi_{delete}\} = \{0.4,0.3,0.3\}$.

- AMAP - the program with its default settings. $w = w^1_{tgf}$ with dynamic weights.

- AMAP$_{rec}$ - adjusted to maximize recall. $w = w^0_{maxstep}$ with static weights, and two rounds of consistency transformations. This version of the program is best suited for comparison with existing multiple alignment programs that focus on maximizing recall.

- AMAP$_{prec}$ - adjusted for better precision. $w = w^8_{tgf}$ with dynamic weights.

We compared the performance of AMAP with other alignment programs on the SABmark 1.65 datasets (see Section 3.3.1). Programs tested include Align-m 2.3 (Van Walle *et al.*, 2005), CLUSTALW 1.83 (Thompson *et al.*, 1994), DIALIGN-T 0.2.1 (Subramanian *et al.*, 2005), MUSCLE 3.52 (Edgar, 2004), ProbCons 1.1 (Do *et al.*, 2005) and T-Coffee 3.84 (Notredame *et al.*, 2000). Our main results are:

- The recall of AMAP$_{rec}$ averaged over all positions in both SABmark datasets (with and without false-positives) is higher than all tested programs, and is achieved with the highest precision of all tested programs. Remarkably, the precision of AMAP$_{rec}$ is almost 3 times higher than the closest competitors in recall.

- Both AMAP and AMAP$_{prec}$ get better modeler and AMA scores than all other programs, including Align-m, which is designed to optimize precision rather than recall.

- The running time of AMAP is comparable to (and in some cases faster) than that of DIALIGN-T, MUSCLE, ProbCons and T-Coffee. This was achieved without any optimization for speed in the current prototype.

These results are detailed in Tables 4.1 and 4.2 that show the performance of all the alignment programs we tested measured with the developer, modeler and AMA

	Twilight by alignment			Superfamilies by alignment			Overall by alignment			Overall by position			Time
Program	f_D	f_M	AMA	f_D	f_M	AMA	f_D	f_M	AMA	f_D	f_M	AMA	Sec.
Align-m	21.6	23.6	51.7	49.2	45.6	56.9	40.1	38.3	55.2	35.2	45.4	56.6	12.7
CLUSTALW	25.6	14.7	24.9	54.0	38.1	43.8	44.7	30.4	37.6	33.6	19.5	28.2	**0.4**
DIALIGN-T	21.3	19.8	45.5	49.9	44.9	54.8	40.4	36.6	51.7	33.9	38.6	52.5	1.4
MUSCLE	27.3	16.4	27.6	56.3	40.3	46.4	46.8	32.4	40.2	37.5	22.5	31.7	2.1
ProbCons	**32.1**	21.1	37.4	**59.8**	44.4	51.8	**50.7**	36.7	47.0	43.0	34.3	47.0	4.5
T-Coffee	26.7	18.1	35.2	56.5	42.8	50.3	46.7	34.7	45.3	39.4	31.5	44.5	11.3
AMAP$_{rec}$	30.9	22.4	40.9	58.8	45.3	53.3	49.6	37.8	49.2	**43.3**	39.1	51.9	2.4
AMAP	24.0	28.3	51.2	52.8	54.6	59.5	43.3	45.9	56.8	32.5	59.7	**59.6**	1.7
AMAP$_{prec}$	14.5	**41.5**	**56.5**	38.7	**69.4**	**60.2**	30.7	**60.2**	**59.0**	20.7	**78.1**	58.9	1.4

Table 4.1. **Comparison of protein alignment programs on the SABmark datasets with no false positives.** Entries show the average developer (f_D), modeler (f_M) and alignment metric accuracy (AMA) scores. Best results are shown in bold. All numbers have been multiplied by 100.

accuracy measures on the SABmark 1.65 datasets without and with false positives respectively. The developer score (f_D) measures recall and the modeler score (f_M) measures precision. The AMA measure, provides a balanced assessment of the overall accuracy of an alignment.

It is important to note that all programs except for Align-m aim at maximizing recall at the expense of precision. It is therefore not surprising that these programs clearly have better f_D scores than f_M scores. On the other hand, AMAP enables control of the recall/precision trade-off, and is able to achieve best results on both measures. This is clear by examining Figure 4.7 which shows recall and precision of AMAP at various stages of the sequence annealing on the SABmark dataset with no false-positives. All the points on the AMAP curve were produced with AMAP, except for the highest recall point, which was produced by AMAP$_{rec}$. The figure depicts one crucial difference between AMAP and all the other current alignment programs. While every other program, produces a single point, the sequence annealing method

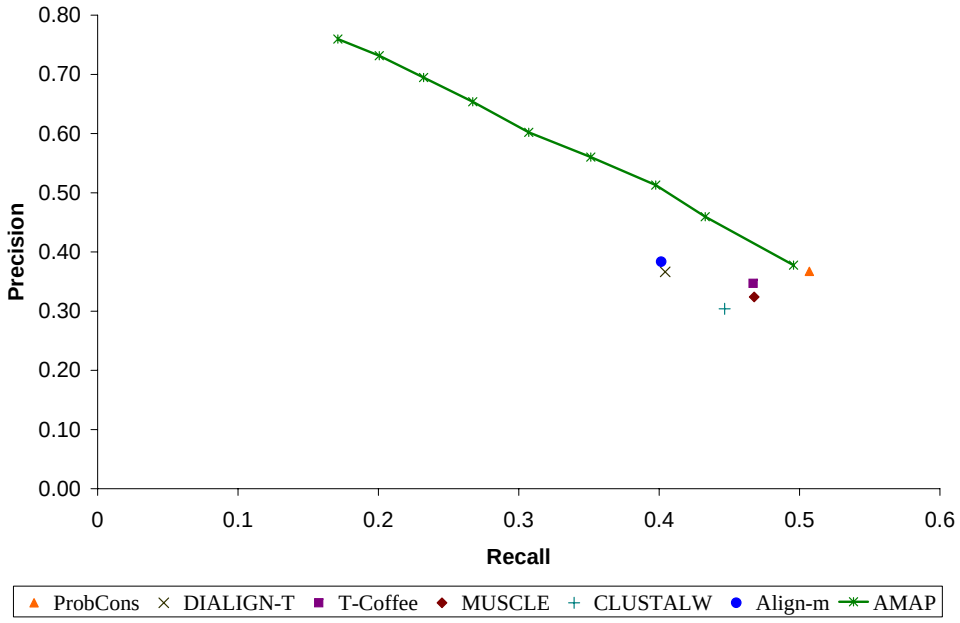
	Twilight-FP by alignment			Superfamilies-FP by alignment			Overall by alignment			Overall by position			Time
Program	f_D	f_M	AMA	f_D	f_M	AMA	f_D	f_M	AMA	f_D	f_M	AMA	Sec.
Align-m	17.8	6.4	81.5	44.8	16.8	77.5	35.9	13.4	78.9	28.6	17.6	83.3	158.9
CLUSTALW	20.4	2.4	35.5	50.9	7.4	37.0	40.8	5.7	36.5	31.2	4.0	34.2	**1.7**
DIALIGN-T	17.0	4.5	74.1	46.7	14.0	71.5	36.9	10.9	72.4	30.2	13.5	78.5	5.7
MUSCLE	19.4	2.3	37.1	49.7	7.5	38.9	39.7	5.8	38.3	30.7	4.1	35.7	19.2
ProbCons	26.8	4.4	55.6	56.0	10.9	55.0	46.4	8.8	55.2	34.8	6.5	54.2	28.5
T-Coffee	13.0	2.3	56.5	42.5	9.3	56.6	32.8	7.0	56.6	26.7	6.6	62.6	61.2
AMAP$_{rec}$	**27.3**	6.6	68.3	**56.1**	14.1	63.8	**46.6**	11.6	65.3	**35.7**	17.7	81.1	13.5
AMAP	19.2	9.8	84.4	46.4	27.0	84.2	37.4	21.4	84.2	27.4	30.1	88.5	11.2
AMAP$_{prec}$	12.7	**17.3**	**91.0**	35.7	**45.9**	**91.1**	28.1	**36.5**	**91.1**	19.1	**50.3**	**91.4**	10.0

Table 4.2. **Comparison of protein alignment programs on the SABmark datasets with false positives.** Entries show the average developer (f_D), modeler (f_M) and alignment metric accuracy (AMA) scores. Best results are shown in bold. All numbers have been multiplied by 100.

used in AMAP produces a series of alignments, starting alignments with high precision and low recall, and ending at alignments with lower precision but higher recall. It is clear that the AMAP line dominates all other programs, since for every alignment produced by some other program, there is a point on the AMAP line that has better recall *and* precision. The advantage of the sequence annealing approach is even more pronounced when considering the sets with false-positives (Figure 4.8), since the sequence annealing avoids fixing erroneous homology prediction at early stages of the algorithm like progressive alignment, and therefore much better suited for identifying un-alignable regions, leaving them unaligned at least until more evidence is obtained from other sequences.

Although we disagree that alignment programs should be evaluated based on recall alone, we will note that the f_D scores of AMAP$_{rec}$ are comparable to ProbCons, which is arguably the most accurate program in terms of recall. This is not surprising since AMAP and ProbCons use similar posterior probabilities. While ProbCons has

(a)



(b)

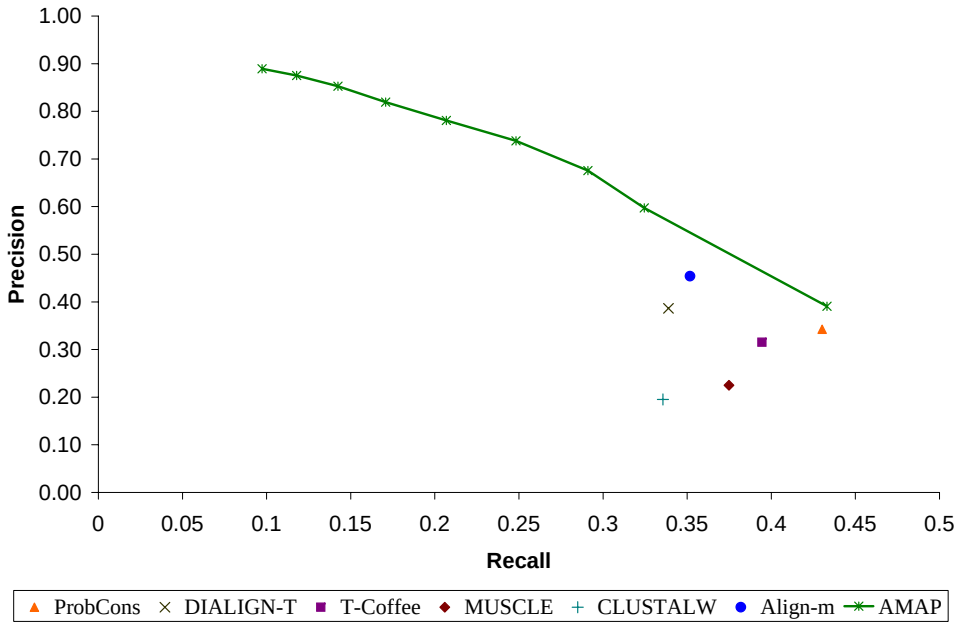
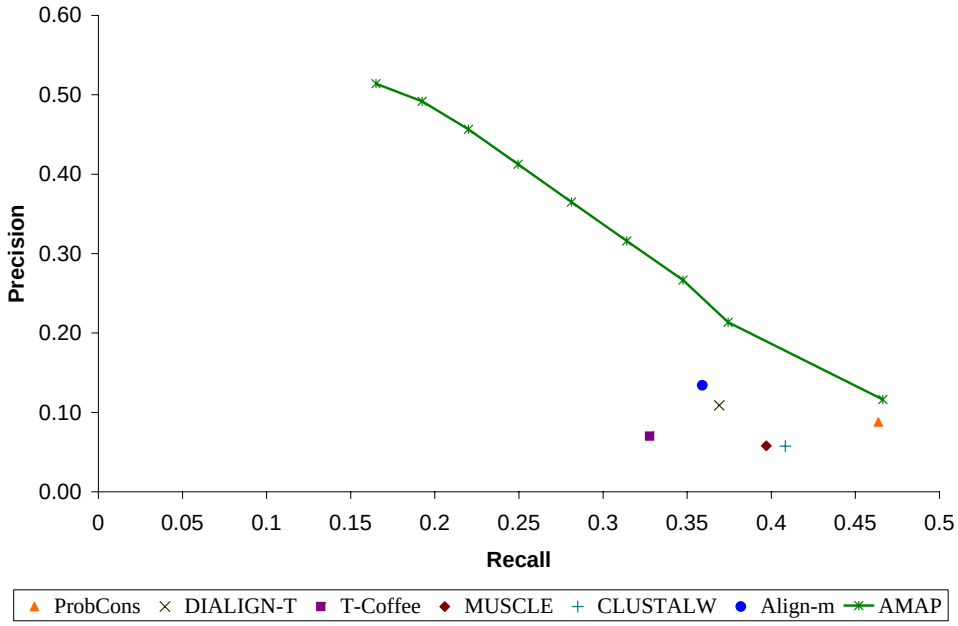


Figure 4.7. **Comparison of the recall/precision trade-off of different alignment programs with AMAP on the SABmark datasets with no false-positives** (a) Results averaged over alignments. (b) Results averaged over all positions.

(a)



(b)

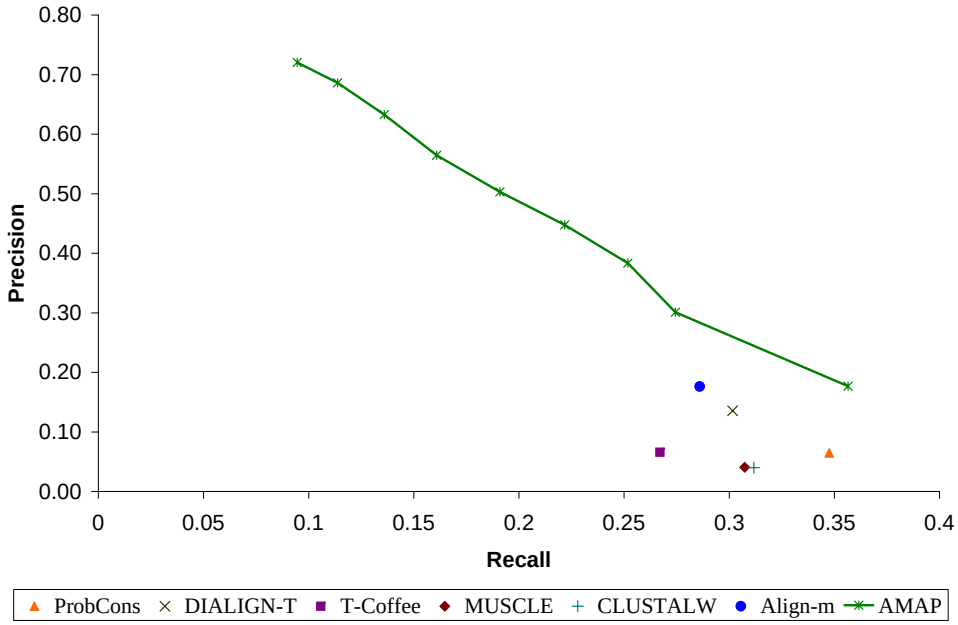


Figure 4.8. **Comparison of the recall/precision trade-off of different alignment programs with AMAP on the SABmark datasets with false-positives** (a) Results averaged over alignments. (b) Results averaged over all positions.

slightly better recall than AMAP$_{rec}$ on the sets without false-positives when scores are averaged over alignments, AMAP$_{rec}$ has a slightly better score when the scores are averaged over all positions in all alignments together. This indicates that the sequence annealing method finds better alignments when the total number of residues in an alignment increases. The same trend is evident in the false-positives sets, in which AMAP$_{rec}$ achieves better scores than ProbCons on all measures and datasets. This is again due to the fact that the false-positives sets are larger and therefore the alignments are less robust to errors made in early stages of the progressive alignment procedure used in ProbCons. Even more striking is the fact that while both programs achieve very similar f_D scores, AMAP$_{rec}$ has much better f_M and AMA scored than ProbCons (81.1 AMA compared to 54.2 on the false-positive sets averaged over all alignments), even though it is tuned to optimized recall alone. Again, this can be explained by the fact that sequence annealing takes the smallest steps in the space of multiple alignments, and fixes matches based on their overall contribution to the scoring function score. In progressive alignment, once two sequences are chosen to be aligned, their entire alignment is determined without regard to the fact that the alignment of some residues may be unclear until later in the procedure.

Both AMAP and AMAP$_{prec}$ get better f_M and AMA scores than all other programs, including Align-m, which is designed to optimize precision rather than recall. AMAP achieves high precision while still producing respectable recall scores. On most datasets, AMAP$_{prec}$ has better AMA scores than AMAP, despite the fact that AMAP is tuned to optimize the AMA scores. This can be explained by the fact that the parameters used to produce the posterior probabilities do not fit the SABmark data very well, and probably under-estimate the gap probabilities.

AMAP compares favorably with most other programs in terms of running time, as well. It takes only 13.5 seconds on average to align a group of sequences from the false-positives sets with current version of AMAP, which is still a prototype that has not yet

been optimized for speed, compared to 19.2 seconds per query with MUSCLE, which is known to be optimized for fast running time. In fact, most of AMAP's running time is spent on computing the posterior probabilities and not on the actual sequence annealing. Using sequence annealing also enables AMAP to produce in one run of the algorithm a range of multiple sequence alignments, ranging from alignments with high precision to those with high recall, since at every point of the algorithm a valid multiple sequence alignment can be produced from the current (partial) alignments. This allows users to see how the alignment is being built, and decide which alignment looks best for their purpose.

4.4 Summary and Future Directions

We have shown that sequence annealing is a practical alternative to progressive alignment that can be used for multiple alignment of protein sequences. In addition to being more accurate than existing methods, sequence annealing has the advantage that the intermediate alignments produced during the annealing process are of use in identifying reliable portions of the multiple alignment. They provide the user with the ability to explore the trade-off between recall and precision.

Although we have only considered multiple alignment of protein sequences in this chapter there is no reason to expect that the methods we have developed cannot be applied to DNA multiple alignment. In the case of DNA, it may be necessary to further develop the sequence annealing approach to allow for melting (in analogy with simulated annealing algorithms). Indeed, by revisiting certain annealing steps, it may be possible to avoid local maxima of the scoring function. Indeed, we do not, at this time, have a clear understanding of the landscape of the scoring function we have proposed. This is a promising direction for future research.

Sequence annealing can also be used for local multiple alignment. A *partial local multiple alignment* of sequence characters $S \triangleq \{\sigma^1, \ldots, \sigma^K\}$ is a partially ordered set $P \triangleq \{c_1, \ldots, c_m\}$ together with a surjective function $\varphi : S \to P$ such that if $x, y \in P$ with $x < y$ then if $\sigma_i^a \in \varphi^{-1}(x)$ and $\sigma_j^a \in \varphi^{-1}(y)$, $i < j$. Note that the every partial global multiple alignment is a partial local multiple alignment (but not vice versa). A null local multiple alignment is the poset consisting of a disjoint union of singletons. Sequences can be annealed with the added "move" of connecting components of the alignment posets without collapsing elements.

In summary, sequence annealing opens up the possibility of exploring the space of multiple alignments, and enables global optimization methods to be applied directly to the multiple alignment problem.

Chapter 5

Multiple Alignment of Citation Sentences

In the previous chapters we have shown how posterior-decoding methods can be applied to the problem of multiple sequence alignment. In particular, posterior-decoding methods can directly maximize the expected value of a specific utility function rather than minimizing the expected value of the standard 0–1 loss function using Viterbi decoding. In addition, posterior-decoding enables direct control of the recall/precision trade-off that is inherent in most machine-learning problems.

In this chapter we demonstrate how techniques similar to the ones we applied to the MSA problem can be applied to a different but related problem—multiple alignment of citation sentences at the word level.

5.1 Citances

In Nakov, Schwartz, and Hearst (2004) we first introduced the concept of *citances*. This section summarizes the motivation for using citances in the biosciences domain as we have described it in that publication.

The scientific literature of biomedicine, genomics, and other biosciences is a rich, complex, and continually growing resource. With appropriate information extraction and retrieval tools, bioscience researchers can use the contents of the literature to further their research goals. In recent years the interest in automatic tools for information extraction and retrieval from bioscience literature has increased considerably. Evidence for that trend is the addition of the genomics track to the Text Retrieval Conference (TREC)[1], and the BioCreAtIvE (Critical Assessment of Information Extraction systems in Biology) competition [2].

As part of the BioText project [3] we are interested in utilizing the large volume of available bioscience text when designing information extraction and retrieval tools. For example, instead of analyzing each document separately, multiple related documents can be analyzed together, thus increasing the accuracy of tools for tasks such as entity recognition, relation extraction, synonym disambiguation, and automatic summarization. So far, most of the Natural Language Processing (NLP) work in the bioscience domain has been done on MEDLINE abstracts. However, full text is becoming more available, providing new opportunities for automatic text processing. One such opportunity lies in the text around citations in full text papers.

We suggest using the sentences that surround the citations to related work as the data from which to build semantic interpretation models. We also introduce

[1] http://trec.nist.gov/
[2] http://www.mitre.org/public/biocreative/
[3] http://biotext.berkeley.edu/

a neologism, *citances*, to mean the sentence(s) surrounding the citation within a document.

Citations are used in every scientific literature, but they are particularly abundant in biosciences. Nearly every statement is backed up by at least one citation, and, conversely, it is quite common for papers in the bioscience domain to be cited by 30–100 other papers. The text around citations tends to state known biological facts with reference to the original papers that discovered them. The cited facts are typically stated in a more concise way in the citing papers than in the original papers. As the same facts are repeatedly stated in different ways in different papers, statistical models can be trained on existing citances to identify similar facts in unseen text. Figure 5.1 shows an example of three different citances to the same target paper.

With the availability of full text articles, and the nature of citation in bioscience literature, traditional citation analysis work can be greatly expanded. We believe that citations have great potential to be a valuable resource in mining the bioscience literature. In particular, we identify the following promising applications of citation analysis:

- **A source for unannotated comparable corpora**. *Comparable corpora*, which are typically generated from news articles on related events, are a useful resource for the development of NLP tools for question answering (Lin and Pantel, 2001) and summarization (Barzilay and Lee, 2003). Most domains outside of news do not contain many articles discussing the same events, but bioscience citances have some of the requisite characteristics in that they include redundancies that enable identification of comparable sentences. In the case of news articles, dates and named entities help link related sentences.

- **Summarization of the target papers**. The set of citances that refer to a specific paper can be viewed as an indication of the important facts in the paper

as seen by the scientific community in that field. This is an excellent resource for summarization. In fact, we believe that a paper that is cited enough times can be summarized using only the citances pointing to it. Instead of showing the user all the citances pointing to a paper (as is done in CiteSeer and in Nanba *et al.* (2000)), we propose to first cluster related citances, and then display to the user only a summary of each cluster. The facts expressed by each cluster can be extracted and stored in a database in a normalized form. This could facilitate answering advanced queries on facts, such as "retrieve all documents that describe which genes upregulate gene G".

- **Synonym identification and disambiguation**. Bioscience literature is rife with abbreviations and synonyms. Citances referring to the same article may enable synonyms to be identified and recorded. On the flip side, in many cases the same terms have multiple meanings. Again, a collection of related citances can help disambiguate these meanings, since in some of the citances an unambiguous form of the term might be present.

- **Entity recognition and relation extraction**. Citances in bioscience literature are more likely to state biological facts than arbitrarily chosen sentences in an article. They also tend to be more concise, since the authors try to summarize previous related work, which has already been described in detail in the original paper. Language presents a myriad number of ways to express the same or similar concepts. Citances provide us a way to build a model of many of the different ways to express a relationship type R between entities of type A and B. We can seed learning algorithms with several examples using concepts that are semantically similar to A and similar to B, for which relation R is known to hold. Then we can train a model to recognize this kind of relation for situations for which the relation is not known. Since the results may extend to

sentences that are not citances as well, citances-based corpora should provide a good collection for building NLP tools for recognizing entities and relations in unseen text.

- **Targets for curation**. We hypothesize that citances contain the most important information expressed in the cited document, and therefore contain the information that curators would want to make use of.

- **Improved citation indexes for information retrieval**. In addition to supporting advance queries over facts as just described, citation indexes can be improved by combining methods that use citances' *context* (e.g. Mercer and Marco, 2004) with methods that use citances' *content* (e.g. Bradshaw, 2003). For example, indexing terms can be taken from citances referring to a target paper, weighting them both by their relative frequency and the type of citations they appear in.

Several issues must be addressed in order to effectively use citances in various applications.

- **Text span.** The span or scope of the text that should be included with the citation must be determined. The appropriate span can be a phrase, a clause, a sentence, or several sentences or their fragments. Furthermore, citations themselves must be parsed, as they can be shown as lists or groups (e.g., "[22-25]").

- **Identifying the different topics.** The different reasons a given paper is cited must be determined, and citances that cite a document for a similar reason must be grouped together.

- **Normalizing or paraphrasing citances.** Once the citances with the same meaning are grouped together, they will convey essentially the same information in different ways, or express different subsets of the same information.

Thus it is important to be able to "normalize" or paraphrase the citances for many applications, including indexing in a database or an IR system, document summarization (Barzilay and McKeown, 2001; Barzilay and Lee, 2003), learning synonyms (Grefenstette, 1992, 1994), building a model of the different expressions of the same relationship for IE (Shinyama *et al.*, 2002; Shinyama and Sekine, 2003), extracting patterns for question answering (Lin and Pantel, 2001), and machine translation (Pang *et al.*, 2003).

5.2 Multiple citance alignments

Most of the applications of citance analysis described in the previous section require the identification of equivalent entities across and within citances. For example, when using citances to a target paper as an input to a summarization system, entities that are semantically related in the context of the cited paper should be identified as such to help in the identification of the main cited facts. The citances alignment problem is partially analogous to the problem of multiple alignment of biological sequences. In both cases the goal is to *align* homologous entities that are derived from the same ancestral entity. While in biology homology is well defined in the molecular level, in the citances case it is defined in the semantic level, which is much more subjective. Given a group of citances that cite the same target paper, we loosely define *semantic homology* as a symmetric, transitive, and reflexive relation between two entities (words or phrases) in the same or different citance that have similar semantics in the context of the cited paper.

Figure 5.1 shows an example of three citances that cite the same target paper (Falck *et al.*, 2001). A multiple alignment of the entities in the same citances (after removal of stop-words) is shown in Figure 5.2. Homologous entities are colored the same. This small example illustrates some of the main challenges of multiple ci-

tance alignment (MCA). While orthographic similarity can help to identify semantic homology (e.g. *phosphorylate* and *phosphorylation*), it can also be misleading (e.g. *cell cycle* and *U3A cells*). In addition, semantic homology might not include any orthographic clues (e.g. *genotoxic stress* and *DNA damage*).

Unlike global MSA where each character can be aligned to at most one character in every other sequence, in MCA each word can be aligned to any number of words in other sentences. Another major difference between the two problems is the fact that while the sequential ordering of characters must be maintained in MSA, this is not the case for MCA. However, MCA retains the transitivity requirement of MSA (i.e. that if $\sigma_i^a \diamond \sigma_j^b$ and $\sigma_j^b \diamond \sigma_k^c$ then $\sigma_i^a \diamond \sigma_k^c$).

Formally, we define MCA as follows. Let $\mathcal{G} \triangleq \{C^1, C^2, \ldots, C^K\}$ be a group of K citances that cite the same target paper, where the i^{th} citance is a sequence of words $C^i \triangleq C_1^i C_2^i \cdots C_{n^i}^i$, and $c^i \triangleq \{c_1^i, c_2^i, \ldots, c_{n^i}^i\}$ is the set of word indices of C^i. A *pairwise citance alignment* of C^i and C^j is an equivalence (symmetric, reflexive, and transitive) relation $\sim_{ij}$ on the set $c^i \cup c^j$. The expression $c_k^i \sim_{ij} c_l^j$ means that according to the pairwise alignment $\sim_{ij}$ word k in citance C^i and word l in citance C^j are aligned. A *multiple citance alignment* (MCA) is an equivalence relation $\sim \triangleq \left(\bigcup_{ij} \sim_{ij}\right)^+$ on the set $\bigcup_i c^i$, which is the transitive closure of the union of all pairwise alignments of citance pairs in $\mathcal{G}$. Taking the transitive closure and not only the union of all pairwise alignments ensures that the MCA is an equivalence relation as well.

Since $\sim$ is an equivalence relation, it includes also pairs of word-indices (c_k^i, c_l^j) from the same citances where $i = j$. For notation convenience we define the set $\sim_{\neq} \triangleq \sim \setminus \bigcup_{ikl} \{(c_k^i, c_l^i)\}$, which includes all indices pairs that align words from different citances. We also add the following notation for two special cases of MCAs. $\sim^{null} \triangleq \bigcup_{ik} \{(c_k^i, c_k^i)\}$ is the null alignment where each word aligns only to itself. $\sim^{all} \triangleq \bigcup_{ij|i \neq j} c^i \times c^j$ is the complete alignment where all words align to all words.

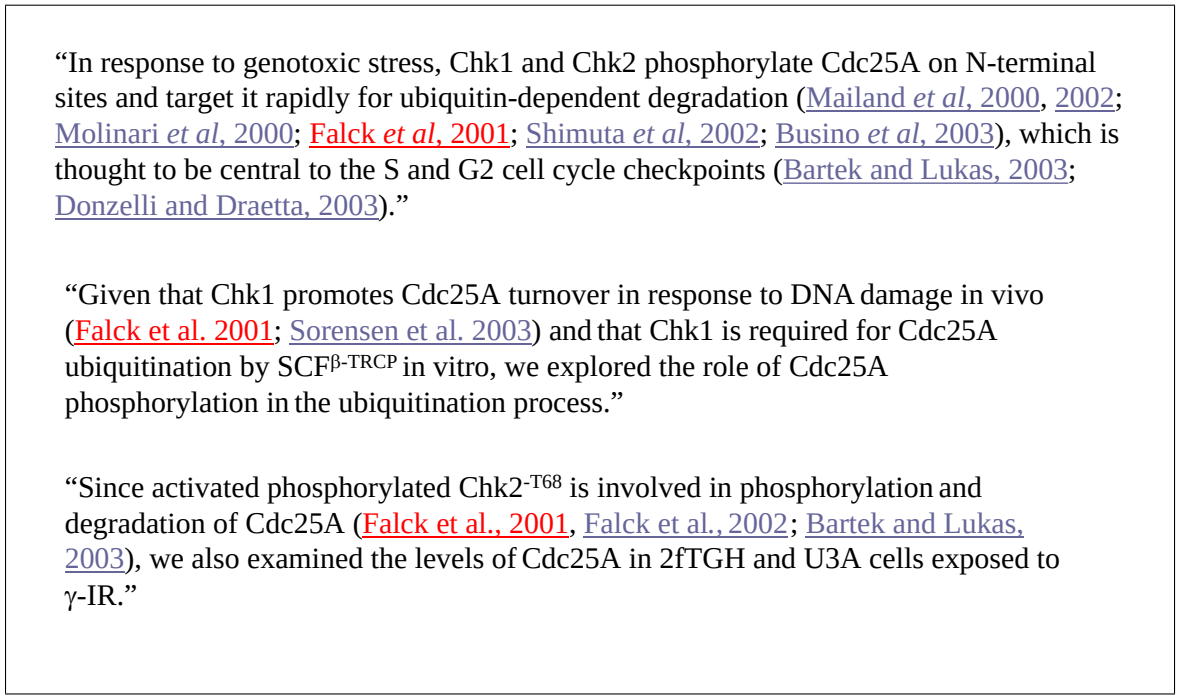


Figure 5.1. **Example of three unaligned citances.**

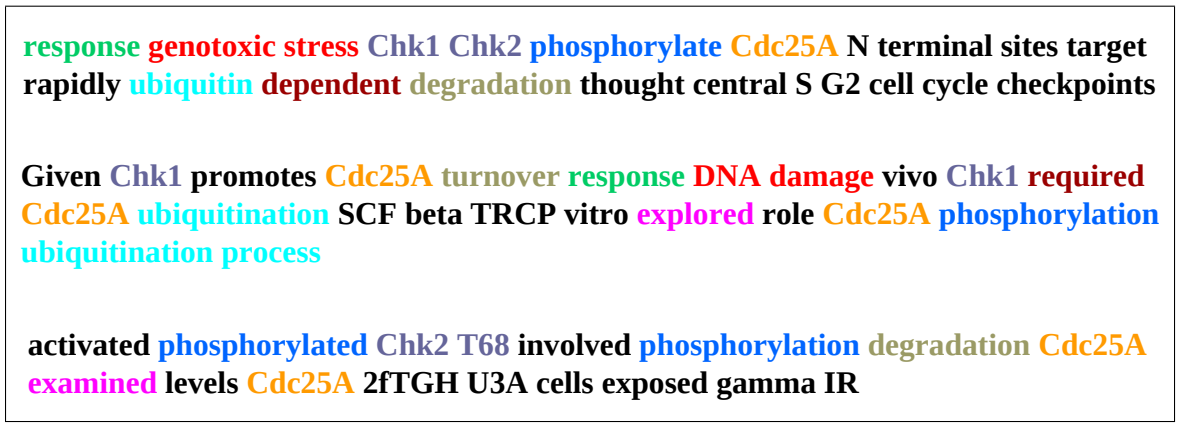


Figure 5.2. **Example of three normalized aligned citances.** Homologous entities are colored the same. Unaligned entities are black.

Note that $|\sim_{\neq}^{null}| = 0$, and $|\sim_{\neq}^{all}| = \sum_{i,j|i\neq j} n^i n^j = \left(\sum_i n^i\right)^2 - \sum_j \left(n^j\right)^2$. Another convenient notation is $\nsim \triangleq \sim_{\neq}^{all} \setminus \sim$, which is a relation that includes all word indices pairs (from different citances) that are not aligned.

An MCA $\sim$ defines a partition of the set of all word indices $c \triangleq \bigcup_{ik} \{c_k^i\}$, which is of size $n \triangleq |c| = \sum_i n^i$. Therefore, the number of distinct MCAs of $\mathcal{G}$ is the number of partitions of a set of size n. This number is called the n^{th} Bell number (Rota, 1964)

$$B_n \triangleq \frac{1}{e} \sum_{k=0}^{\infty} \frac{k^n}{k!}. \tag{5.1}$$

Asymptotically, B_n grows faster than an exponential but slower than a factorial. For example $B_{100} \approx 10^{116}$. Obviously, enumerating all possible MCAs is impractical even for small problems.

5.3 MCA utility and loss functions

Unlike the toy example in Figure 5.2, in most practical scenarios useful MCAs include tens or hundreds of citances. Since it is impractical to manually construct an MCA to every target paper of interest, we are interested in algorithms for producing MCA automatically. For automatic MCAs to be useful as an input to further citance analysis, they need to be accurate or "close" enough to the "true" MCAs or to manually created reference MCAs, which are assumed to be closer to the truth. Ultimately, the success of an MCA algorithm should be judged by its effect on the success of the citance analysis systems that use MCAs as their input. However, measuring this effect directly is very difficult, since the higher level tasks, such as summarization are hard to evaluate objectively, and furthermore it is hard to quantify the contribution of the MCA accuracy to the accuracy of the higher level system that uses it. A more practical alternative is to measure the accuracy of MCAs directly using some accuracy measure, under the simplifying assumption that there is a strong correlation between the measured MCA accuracy and the performance of the higher level system.

Given a reference MCA $\sim^r$ and a two predicted MCAs $\sim^{p1}$ and $\sim^{p2}$, a *utility*

function assigns a higher value to the MCA that is closer to $\sim^r$. Conversely, a *loss function* assigns a higher value to the MCA that is farther from $\sim^r$. In the following discussion we will only consider utility and loss functions that assign values in the range $[0, 1]$, where for $\sim^{p1}=\sim^{p2}$ the utility is 1 and the loss is 0. Therefore, for every utility function $U(\cdot, \cdot)$ an equivalent loss function can be defined as $L(\cdot, \cdot) \triangleq 1-U(\cdot, \cdot)$. Given a utility (loss) function the goal of an MCA algorithm is to produce MCAs that maximize (minimize) the expected value of that function.

We argue that a useful utility function should be correlated (or even identical) to the accuracy measure used to evaluate the performance of an algorithm. In addition, the utility function should be easily decomposable, to enable direct optimization using posterior-decoding. Although any accuracy measure that is acceptable as a single performance measure can be used to guide the design of the utility function, metric-based accuracy measures have several noticeable advantages. First, a metric formalizes the intuitive notion of distance. Hence, an accuracy measure which is based on a metric follows the intuition that reducing the distance to the correct answer should increase the accuracy of the predicted answer. Therefore, defining a metric space for the objects of a given problem leads to a natural definition of accuracy. Another advantage of using a metric-based accuracy measure is the ability to provide bounds in the search space using the triangle inequality. For example, while searching for the answer with the optimal (metric-based) expected utility, a step of length x can only change the expected utility as well as the actual utility by $\pm x$ units. Examples of more complex bounds using metric loss functions are described in Schlüter *et al.* (2005) and Domingos (2000).

Using the requirements from a good utility function described above, it is possible to judge existing utility functions and design new ones. There are numerous alternative loss functions that have been described before for related problems to MCA. We

will only consider a small number of similarity measures that are commonly used in binary classification problems to compare two sets (predicted versus reference).

The 0–1 loss function is very simple, but is widely used as an objective function for binary classification. In the context of MCA the 0–1 loss is defined as:

$$L_{0-1}(\sim^r, \sim^p) \triangleq \begin{cases} 0 & \text{if } \sim^r = \sim^p \\ 1 & \text{otherwise} \end{cases} \tag{5.2}$$

The main problem with the 0–1 loss function is that it is too strict. While it works well for many binary classification problems, it is not as well suited for more complex problems, such as MCA, where it is very difficult to achieve perfect agreement between the reference alignment and the predicted alignment. The 0–1 loss does not distinguish between different types of errors. Intuitively, the loss of a predicted MCA that is closer to the reference MCA should be smaller than the loss of a predicted MCA that is very different from the reference.

A standard loss-function used in a closely related problem to MCA, namely pairwise word alignment for machine translation (Brown *et al.*, 1990), is the Alignment Error Rate (AER) (Och and Ney, 2003). AER is defined for reference alignments that include both "Possible" and "Sure" word-indices pairs. When all pairs are "Sure" pairs the Dice coefficient (Dice, 1945) between $\sim^p_{\neq}$ and $\sim^r_{\neq}$, and the F_1-measure of word-indices pairs are the equivalent utility functions to the the AER loss function.

$$U_{Dice}(\sim^r, \sim^p) \triangleq \frac{2|\sim^r_{\neq} \cap \sim^p_{\neq}|}{|\sim^r_{\neq}| + |\sim^p_{\neq}|} \tag{5.3}$$

In a recent paper, Fraser and Marcu (2006) show that AER has only moderate correlation to the final translation quality even without the "Possible" pairs. Moreover, $1-Dice$ is not a metric, since it does not obey the triangle inequality (Gower and Legendre, 1986). To see that, consider the following example; let $x = \{1\}, y = \{2, 3\}, z = \{1, 2, 3\}$, then $1 - Dice(x, z) + 1 - Dice(y, z) = 2/4 + 1/5 < 1 - Dice(x, y) = 1$.

A similar utility function is based on the Jaccard coefficient (Jaccard, 1901), which, unlike Dice, has an equivalent loss function that is a metric (Lecandowsky and Winter, 1971).

$$U_{Jaccard}(\sim^r, \sim^p) \triangleq \frac{\mid \sim^r_{\neq} \cap \sim^p_{\neq} \mid}{\mid \sim^r_{\neq} \cup \sim^p_{\neq} \mid} \tag{5.4}$$

The main problem with using the Jaccard coefficient (as well as Dice) as the utility function for MCA is that $U_{Jaccard}$ does not decompose well over the word-indices pairs. Any change in the predicted MCA $\sim^p$ affects both the numerator and the denominator in Equation (5.4). It is therefore difficult to directly maximize the expected utility using posterior decoding given the posterior probabilities for every candidate aligned word indices pair $P(c^i_k \sim c^j_l)$.

$$\begin{aligned} E_{\sim^t}(U_{Jaccard}(\sim^t, \sim^p)) &= \sum_{\sim^t} P(\sim^t)\ U_{Jaccard}(\sim^t, \sim^p) \\ &= \sum_{\sim^t} P(\sim^t)\ \frac{\sum_{c^i_k c^j_l \mid i \neq j} \mathbf{1}\{c^i_k \sim^t c^j_l \vee c^i_k \sim^p c^j_l\}}{\sum_{c^i_k c^j_l \mid i \neq j} \mathbf{1}\{c^i_k \sim^t c^j_l \wedge c^i_k \sim^p c^j_l\}} \end{aligned} \tag{5.5}$$

As can be seen in Equation (5.5), there is no simple way to avoid enumerating all possible MCAs (B_n; see Equation (5.1)) using the posterior probabilities, which is impractical.

Another loss function that is based on a metric is the Hamming loss (Hamming, 1950), which is defined as

$$L_{Hamming}(\sim^r, \sim^p) \triangleq \frac{\mid \sim^p_{\neq} \setminus \sim^r_{\neq} \mid + \mid \nsim^p \setminus \nsim^r \mid}{\mid \sim^{all}_{\neq} \mid}. \tag{5.6}$$

The Hamming loss is simply the fraction of word indices pairs that are different (aligned in one MCA and not the other) between the predicted MCA and the reference MCA. Unlike the Dice and Jaccard based utility functions, the Hamming loss decomposes well over the word indices pairs, since its denominator is a constant.

Therefore the expected loss can be computed directly given the posterior probabilities of word indices pairs.

$$
\begin{aligned}
& E_{\sim^t}(L_{Hamming}(\sim^t, \sim^p)) \\
&= \sum_{\sim^t} P(\sim^t)\ L_{Hamming}(\sim^t, \sim^p) \\
&= \sum_{\sim^t} P(\sim^t)\ \frac{\sum_{c_k^i c_l^j | i \neq j} \mathbf{1}\left\{(c_k^i \sim^t c_l^j \wedge c_k^i \nsim^p c_l^j) \vee (c_k^i \nsim^t c_l^j \wedge c_k^i \sim^p c_l^j)\right\}}{\sum_{c_k^i c_l^j | i \neq j} 1} \\
&= \frac{\sum_{c_k^i c_l^j | i \neq j} \sum_{\sim^t} P(\sim^t) \mathbf{1}\left\{(c_k^i \sim^t c_l^j \wedge c_k^i \nsim^p c_l^j) \vee (c_k^i \nsim^t c_l^j \wedge c_k^i \sim^p c_l^j)\right\}}{|\sim_{\neq}^{all}|} \\
&= \frac{\sum_{c_k^i c_l^j | i \neq j} E_{\sim^t}\left(\mathbf{1}\left\{(c_k^i \sim^t c_l^j \wedge c_k^i \nsim^p c_l^j) \vee (c_k^i \nsim^t c_l^j \wedge c_k^i \sim^p c_l^j)\right\}\right)}{|\sim_{\neq}^{all}|} \\
&= \frac{\sum_{c_k^i c_l^j | i \neq j} P(c_k^i \sim^t c_l^j) \mathbf{1}\{c_k^i \nsim^p c_l^j\} + \left(1 - P(c_k^i \sim^t c_l^j)\right) \mathbf{1}\{c_k^i \sim^p c_l^j\}}{|\sim_{\neq}^{all}|}
\end{aligned}
\tag{5.7}
$$

The main drawback of the Hamming loss is that it gives equal weight to false positives and false negatives. Since in large MCAs the majority of word indices pairs are likely to be negative, minimizing the expected Hamming loss can lead to very sparse (or even the null) predicted alignments. To mitigate this effect Lacoste-Julien *et al.* (2006) proposed to use a weighted Hamming loss (for pairwise word alignments), which assigns a higher weight to false negatives[4].

$$
L_{wHamming}(\sim^r, \sim^p) \triangleq \frac{w^+ |\sim_{\neq}^p \setminus \sim_{\neq}^r| + w^- |\nsim^p \setminus \nsim|}{w^+ |\sim_{\neq}^p| + w^- |\nsim^p|}. \tag{5.8}
$$

The expected weighted Hamming loss can be computed directly using posterior probabilities as well. However, it is not a metric, since it is not symmetric. More importantly, the weighted Hamming loss suffers from a limitation shared by all pre-

[4]The loss function in Lacoste-Julien *et al.* (2006) was not normalized. We present here a normalized version of the weight Hamming loss function to be consistent with our requirement that all utility and loss functions range between zero and one. However, note that the normalization might reduce the desired effect of the weights.

vious utility (loss) functions we have discussed. Since all the utility functions are defined over the set of word-pair indices $\sim_{\neq}^{all}$ they over-emphasize the contribution of words-indices that are part of large equivalence classes in the reference MCA $\sim^{r}$. Large equivalence classes occur when the words that belong to the same semantic homology class are common in the citances of $\mathcal{G}$, and when the homology class includes multi-word phrases rather than single words. For example, a phrase of length l that appears in k citances is represented by $l^2(k^2 - k)$ word indices pairs, and therefore its affect on the utility function grows quadratically with the number of occurrences and with its length, while a linear growth seems to be a more reasonable choice. This problem is significant also for pairwise alignments, since misaligning a multi-word phrase of length l still incurs a quadratic loss, while misaligning l single words incurs a linear loss.

In order to devise a solution to the over-representation of large classes, let us revisit the alignment metric accuracy (AMA), the utility function for MSA. When we first described it in Section 3.1 we emphasized the fact that AMA is based on a metric unlike the other accuracy measures for MSA. However, another important distinction is that the basic elements of AMA are the individual character position indices, and not the pairs of indices. This difference is subtle in the case of MSA since it is a one-to-one alignment, where every position in one sequence can align to at most one position in every other sequence. Using individual positions as the basic elements leads to the consideration of unaligned positions, which are ignored when only pairs of positions are considered. 1 - AMA for pairwise sequence alignments can be viewed as a Hamming loss over character positions, where the loss is the number of character positions in both sequences that are aligned differently.

Using the insight from the definition of AMA for MSA, we would like to define a utility function for MCA that uses word indices as its basic elements rather than word indices pairs. This should solve the over-representation problem, since each

word position is treated equally whether it is part of a large equivalence class or not. However, applying AMA directly to MCA would not yield the desired result, because MCA is a many-to-many alignment. Using a strict Hamming loss over word indices, a word position that aligns to multiple word positions in another citance will be rewarded only if it aligns to all these word positions correctly, while a good utility function for MCA should give partial credit to word positions that align to some of the correct word positions while penalizing for aligning to wrong word positions.

To help define such a utility function we define the following. Let $m_h^{ij}(c_l^j) \triangleq \{c_k^i \in c^i | c_k^i \sim c_l^j\}$ be the set of all word positions in citance C^i that align to word position l in citance C^j according to MCA $\sim$. We can then define the following utility function for the MCA $\sim^p$ of the citance group $\mathcal{G}$ given a reference MCA $\sim^r$:

$$U_{AMA}(\sim^r, \sim^p) \triangleq \frac{\sum_{ijl|i\neq j} U_{set\_agreement}\left(m_{\sim^r}^{ij}(c_l^j), m_{\sim^p}^{ij}(c_l^j)\right)}{n(K-1)}, \tag{5.9}$$

where n is the number of word indices in $\mathcal{G}$, $K \triangleq |\mathcal{G}|$ is the number of citances in the group, and $U_{set\_agreement}$ is any utility function for agreement between sets that assigns values in the range $[0, 1]$. $U_{set\_agreement}$ can be viewed as a "score" assigned to each word position based on the agreement between the two alignments with regards to the other word positions that align to it. Using a 0–1 loss as the set agreement score is equivalent to the original AMA.

Any of the utility functions we have discussed earlier, such as Dice, Jaccard and Hamming can be used as $U_{set\_agreement}$. However, only metric-based utility functions will result in a metric-based U_{AMA} utility function. It is easy to see that $1 - U_{AMA}$ satisfies all the requirements of a metric, i.e. it is non-negative, equals to zero if and only if $\sim^r = \sim^p$, symmetric, and obeys the triangle inequality, since if the triangle inequality holds for $U_{set\_agreement}$, it must hold for a sum of $U_{set\_agreement}$ values. The Dice coefficient is not based on a metric, while the Hamming and weighted Hamming loss functions have similar problems to those we have discussed earlier. Since Jaccard

is based on a metric it can be a good candidate to be used as the set agreement function, and unlike when it is defined on word indices pairs, when it is defined as a score for each word position computing its expected value is much more feasible.

$$U_{Jaccard}\left(m^{ij}_{\sim^r}(c^j_l), m^{ij}_{\sim^p}(c^j_l)\right) \triangleq \begin{cases} 1 & \text{if } m^{ij}_{\sim^r}(c^j_l) = \emptyset \\ & \text{and } m^{ij}_{\sim^p}(c^j_l) = \emptyset \\ \frac{|m^{ij}_{\sim^r}(c^j_l) \cap m^{ij}_{\sim^p}(c^j_l)|}{|m^{ij}_{\sim^r}(c^j_l) \cup m^{ij}_{\sim^p}(c^j_l)|} & \text{otherwise} \end{cases} \tag{5.10}$$

The distance measure based on the Jaccard coefficient can be viewed as a normalized edit-distance between two sets when the only allowed moves are insertion and deletions of elements from the sets. Comparing sets A and B the edit distance between the two sets is $|A \cup B| - |A \cap B|$, since starting from A we can remove all the elements that are not in B to get to the intersection ($|A \setminus B|$ steps) and then add the elements that are in B but not in A ($|B \setminus A|$ steps). An alternative is to consider an edit-distance that allows substitutions. In this case the distance between the sets is $max\{|A|, |B|\} - |A \cap B|$. Assuming A is the smaller set, we can replace all the elements that are in A but not in B with elements from B, and then add the rest of the elements that are still missing from B. Overall we used $|B| - |A \cap B|$ moves. Converting to a normalized similarity measure leads to a utility function based on the Braun-Blanquet coefficient (Braun-Blanquet, 1932).

$$U_{Braun-Blanquet}\left(m^{ij}_{\sim^r}(c^j_l), m^{ij}_{\sim^p}(c^j_l)\right) \triangleq \begin{cases} 1 & \text{if } m^{ij}_{\sim^r}(c^j_l) = \emptyset \\ & \text{and } m^{ij}_{\sim^p}(c^j_l) = \emptyset \\ \frac{|m^{ij}_{\sim^r}(c^j_l) \cap m^{ij}_{\sim^p}(c^j_l)|}{\max\left\{|m^{ij}_{\sim^r}(c^j_l)|, |m^{ij}_{\sim^p}(c^j_l)|\right\}} & \text{otherwise} \end{cases} \tag{5.11}$$

Caillez and Kuntz (1996) showed that the Braun-Blanquet coefficient is based on a metric as part of a more complex proof on a family of similarity coefficients.

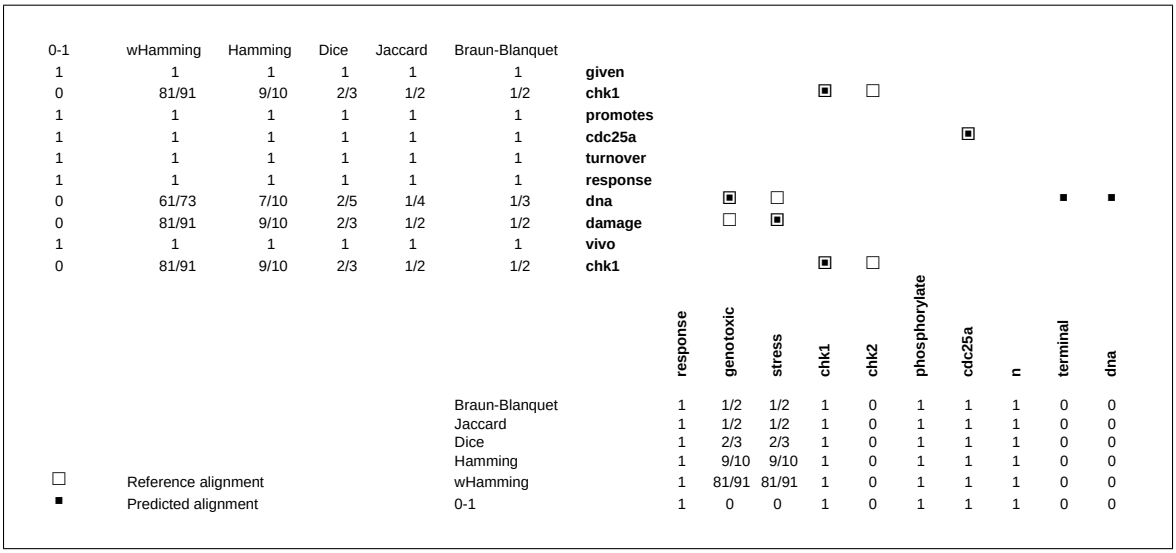


Figure 5.3. **Calculation of different set-agreement functions on a pairwise citance alignment.** A predicted alignment of normalized citances (filled squares) is compared to a reference alignment of the same citances (empty squares). For each word the $U_{set\_agreement}$ scores is calculated using six different set-agreement coefficients.

Function Type	Recall	Precision	0–1	Dice	Jaccard	Hamming	wHamming	Braun-Blanquet
Word indices pairs	0.556	0.714	0	0.625	0.455	0.940	0.955	0.556
AMA (word-based)			0.550	0.737	0.688	0.810	0.814	0.692

Table 5.1. **Comparison of different utility function as calculated on the example in Figure 5.3.** The first line shows the values of the six different coefficients, in addition to recall and precision when using pairs of indices as the basic set elements. The second line shows the word-based values of the coefficients when averaged over all word in Figure 5.3.

We provide an alternative direct proof in Appendix A. There is another important observation about the Braun-Blanquet coefficient—it can be viewed as the minimum of recall and precision. The following inequalities hold between Dice, Jaccard, and Braun-Blanquet coefficients: $U_{Dice}(A, B) \geq U_{Braun-Blanquet} \geq U_{Jaccard}$.

Figure 5.3 demonstrates the calculation of set-agreement functions on a simple example of a pairwise citance alignment. The 0–1 and Hamming loss functions were replaced by their corresponding utility functions $(1 - L)$. For the weighted Hamming utility functions w^- was set to 10 times w^+. Note that the different coefficients differ

only for words for which there is partial agreement between the two alignments. For example, the predicted alignment of the word 'dna' in the vertical citance is scored $2/5, 1/4$, and $1/3$ by the Dice, Jaccard, and Braun-Blanquet coefficients, while the word 'dna' in the horizontal citance is scored 0 by all coefficients. Table 5.1 compares the values of the different utility functions on the example in Figure 5.3 when used directly on word indices pairs (first line), or averaged over words using the U_{AMA} utility of Equation (5.9).

The expected value of the AMA utility function can be approximated using the posterior probabilities of word indices pairs.

$$
\begin{aligned}
&E_{\sim^t}\left(U_{AMA}(\sim^t, \sim^p)\right) = \\
&= \sum_{\sim^t} P(\sim^t) \frac{1}{n(K-1)} \sum_{ijl|i\neq j} U_{set\_agreement}\left(m^{ij}_{\sim^t}(c^j_l), m^{ij}_{\sim^p}(c^j_l)\right) \\
&= \frac{1}{n(K-1)} \sum_{ijl|i\neq j} \sum_{\sim^t} P(\sim^t) U_{set\_agreement}\left(m^{ij}_{\sim^t}(c^j_l), m^{ij}_{\sim^p}(c^j_l)\right) \\
&= \frac{1}{n(K-1)} \sum_{ijl|i\neq j} E_{\sim^t} U_{set\_agreement}\left(m^{ij}_{\sim^t}(c^j_l), m^{ij}_{\sim^p}(c^j_l)\right) \\
&= \frac{1}{n(K-1)} \sum_{ijl|i\neq j} E_{m^{ij}_{\sim^t}(c^j_l)} U_{set\_agreement}\left(m^{ij}_{\sim^t}(c^j_l), m^{ij}_{\sim^p}(c^j_l)\right) \\
&= \frac{1}{n(K-1)} \sum_{ijl|i\neq j} \sum_{c^i_* \in \mathcal{P}(c^i)} P(m^{ij}_{\sim^t}(c^j_l) = c^i_*) U_{set\_agreement}\left(c^i_*, m^{ij}_{\sim^p}(c^j_l)\right) \\
&\approx \frac{1}{n(K-1)} \sum_{ijl|i\neq j} \sum_{c^i_* \in \mathcal{P}(c^i)} \prod_{c^i_k} \left(P(c^i_k \sim^t c^j_l)\mathbf{1}\{c^i_k \in c^i_*\} + (1 - P(c^i_k \sim^t c^j_l))\mathbf{1}\{c^i_k \notin c^i_*\}\right) \\
&\qquad\qquad U_{set\_agreement}\left(c^i_*, m^{ij}_{\sim^p}(c^j_l)\right) \qquad (5.12)
\end{aligned}
$$

The expression $\mathcal{P}(c^i)$ in Equation (5.12) is the power-set (set of all subsets) of c^i, which means that the summation $\sum_{c^i_* \in \mathcal{P}(c^i)}$ ranges over 2^{n^i} possible combinations of word positions in citance C^i, which word position c^j_l can align to. This can slow down the computation of the expected value of the AMA utility function when C^i is

long. In practice, it is possible to approximate this summation by considering only combinations that include word positions that are likely to align to c_l^j ($P(c_k^i \sim^t c_l^j) > Const$).

The last step in Equation (5.12) is based on the assumption that $P(m_{\sim^t}^{ij}(c_l^j) = c_*^i)$ can be approximated by $\prod_{c_k^i} \left(P(c_k^i \sim^t c_l^j)\mathbf{1}\{c_k^i \in c_*^i\} + (1 - P(c_k^i \sim^t c_l^j))\mathbf{1}\{c_k^i \notin c_*^i\}\right)$ using an independence assumption.

5.4 Probabilistic model for MCA

So far we have assumed that the pairwise posterior probabilities $P(c_k^i \sim c_l^j | C^i, C^j)$ are given or can be computed using an arbitrary probabilistic model. However, unlike biological sequences for which pair-HMMs are a natural choice for modeling evolutionary process between two sequences, there is no simple generative model that can be used for modeling pairwise citance alignment. Most of the work on pairwise alignment of sentences at the word level has been done in the statistical machine translation (SMT) community. Och and Ney (2003) present an overview and comparison of the most common models used for SMT word alignments. Out of the models they describe, the HMM models are the most expressive models that can compute posterior probabilities using the forward-backward algorithm. However, unlike sequence-alignments there are no ordering constraints in word-alignments, and the alignments are many-to-many as opposed to one-to-one. Therefore, the SMT HMM models cannot be based on pair-HMMs, which generate two sentences simultaneously. Rather, they are directional models that model the probability of generating a target sentence given a source sentence. In other words they only model one-to-many alignments, recovering the many-to-many alignments in a preprocessing step. Therefore, SMT HMMs can only compute the posterior probabilities $P(c_k^i \rightsquigarrow c_l^j | C^i, C^j)$ and $P(c_l^j \rightsquigarrow c_k^i | C^i, C^j)$, where the relation $\rightsquigarrow$ represents the (directional) event that

a source word is translated into a target word. Nevertheless, recently such posterior probabilities have been used in SMT word alignment system, as an alternative to Viterbi decoding and helped to improve the performance of such systems (Matusov *et al.*, 2004; Liang *et al.*, 2006).

Generative models like HMMs have several limitations. First, they require relatively large training data, which is difficult to attain in case of SMT word alignment, and even more so in the case of MCA. Second, generative models explicitly model the inter-dependence of different features, which reduces the ability to incorporate multiple arbitrary features into the model. Since orthographic similarity is not a strong enough indication for semantic homology in MCA, we would like to be able to incorporate into a single model multiple inter-dependent features, including orthographic, contextual, ontological, and lexical features.

Recently, several authors have described discriminative SMT alignment models (Moore, 2005; Lacoste-Julien *et al.*, 2006; Blunsom and Cohn, 2006). However, to the best of our knowledge only the model of Blunsom and Cohn (2006), which is based on a Conditional Random Field (CRF) (Lafferty *et al.*, 2001), can compute word indices pairs' directional posterior probabilities, like those computed by the HMM models. Therefore, we decided to adopt the CRF-based model to the MCA problem.

5.4.1 Conditional random fields for word alignment

The model of Blunsom and Cohn (2006) is based on a linear chain CRF, which can be viewed as the undirected version of an HMM. The CRF models a one-to-many pairwise alignment, in which every source word can get aligned to zero or one target words, but every word in the target sentence can be the target of multiple source words. CRFs define a conditional distribution over a latent labeling sequence given observation sequence(s). In the case of CRF for word alignment the observed

sequences are the source and target sentences (citances), and the latent labeling sequence is the mapping of source words to target word-indices. Given a source citance C^i of length n^i, and a target citance C^j of length n^j, the one-to-many alignment of C^i to C^j is the relation $\rightsquigarrow$. Since this is a one-to-many alignment, $\rightsquigarrow$ can be represented by a vector a of length n^i. The CRF models the probability of the alignment a conditioned on C^i and C^j as follows:

$$P_\Lambda(a|C^i, C^j) = \frac{\exp\left(\sum_t \sum_k \lambda_k f_k(t, a_{t-1}, a_t, C^i, C^j)\right)}{Z_\Lambda(C^i, C^j)}, \tag{5.13}$$

where $f \triangleq \{f_k\}$ are the model's features, $\Lambda \triangleq \{\lambda_k\}$ are the feature's weights, and $Z_\Lambda(C^i, C^j)$ is the partition (normalization) function which is defined as:

$$Z_\Lambda(C^i, C^j) \triangleq \sum_a \exp\left(\sum_t \sum_k \lambda_k f_k(t, a_{t-1}, a_t, C^i, C^j)\right). \tag{5.14}$$

Parameters are estimated from fully observed data (manually aligned citances) using a maximum a posteriori estimate. The parameter estimation procedure is described in more details in the original paper. Blunsom and Cohn (2006) used Viterbi decoding to find an alignment of two sentences given a trained CRF model, $a^* \triangleq \text{argmax}_a P_\Lambda(a|C^i, C^j)$. However, the posterior probabilities of the labels at each position can be calculated as well using the forward-backward algorithm:

$$P_\Lambda(c^i_l \rightsquigarrow c^j_k|C^i, C^j) = P_\Lambda(a_l = c^j_k|C^i, C^j) = \frac{\alpha_l(c^j_k|C^i, C^j)\beta_l(c^j_k|C^i, C^j)}{Z_\Lambda(C^i, C^j)} \tag{5.15}$$

where α_l and β_l are the forward and backward vectors that are computed with the forward-backward algorithm (Lafferty *et al.*, 2001). Using this procedure the directional pairwise posterior probabilities of every pair of word indices from any pair of citances in both directions can be computed in time $O\left((K^2 - K)\max_{n^i}\{(n^i)^2\}\right)$, where K is the number of citances in the MCA.

5.4.2 The posterior decoding algorithm for MCA

Following our discussion from Section 5.3 we base our MCA utility functions on the AMA utility function with the Braun-Blanquet set agreement coefficient. As with the MSA case a family of utility functions can be defined to enable control of the recall/precision trade-off. Unlike MSA, in the case of MCA two free parameters are needed, in order to have better control of the trade-off using posterior-decoding. In addition to a *gap-factor* that controls the threshold at which unaligned words start to get aligned, a *match-factor* is added to enable control of the number of words-positions each word aligns to.

$$U_{\mu,\gamma}(\sim^r, \sim^p) \triangleq$$
$$\frac{1}{n(K-1)} \sum_{ijl|i\neq j} \left(\mu^{|m^{ij}_{\sim^p}(c^j_l)|} \frac{|m^{ij}_{\sim^r}(c^j_l) \cap m^{ij}_{\sim^p}(c^j_l)|}{\max\left\{|m^{ij}_{\sim^r}(c^j_l)|, |m^{ij}_{\sim^p}(c^j_l)|, 1\right\}} + \gamma \mathbf{1}\{m^{ij}_{\sim^r}(c^j_l) = m^{ij}_{\sim^p}(c^j_l) = \emptyset\} \right), \tag{5.16}$$

where $\gamma \in [0, \infty)$ is a gap-factor, and $\mu \in (0, \infty)$ is a match factor. The neutral value for both parameters is 1, since by setting both parameters to 1, Equation (5.16) recovers the calculation in Equation (5.9) with the Braun-Blanquet set agreement function. Increasing γ results in increased utility to sparser MCAs, while reducing γ increases the utility of denser alignments. However, in the case of MCA the gap-factor only affects the first aligned word position, but it cannot affect the number of word positions each word is aligned to. The match-factor adds this functionality by rewarding MCAs that align words to multiple word positions when $\mu > 1$, and penalizing such MCAs when $\mu < 1$.

Given a group of K citances $\mathcal{G}$ and a trained CRF model, the goal of the MCA algorithm is to find the MCA $\sim^* \triangleq \text{argmax}_{\sim^p} E_{\sim^t} U_{\mu,\gamma}(\sim^t, \sim^p)$ that maximizes the expected utility. Since searching the space of possible MCAs exhaustively is infeasible, we resort to a simple heuristic for predicting an MCA. Instead of searching for a global optimum, the predicted MCA is defined as the equivalence (symmetric transitive)

closure of the union of multiple local optima. For each target word position c_l^j and every source citance C^i the combination of source word positions $c_\circ^i$ that maximize the expected set-agreement score of c_l^j is added to the predicted MCA.

$$
\begin{aligned}
&\rightsquigarrow^p \triangleq \\
&\bigcup_{ijl|i\neq j}\{c_l^j\} \times \text{argmax}_{c_\circ^i \in \mathcal{P}(c^i)} E_{m_{\sim t}^{ij}(c_l^j)} \left(\mu^{|c_\circ^i|} \frac{|m_{\sim t}^{ij}(c_l^j) \cap c_\circ^i|}{\max\left\{|m_{\sim t}^{ij}(c_l^j)|, |c_\circ^i|, 1\right\}} + \gamma \mathbf{1}\{m_{\sim t}^{ij}(c_l^j) = c_\circ^i = \emptyset\} \right) \\
&= \\
&\bigcup_{ijl|i\neq j}\{c_l^j\} \times \text{argmax}_{c_\circ^i \in \mathcal{P}(c^i)} \\
&\qquad \sum_{c_*^i \in \mathcal{P}(c^i)} P(m_{\sim t}^{ij}(c_l^j) = c_*^i) \left(\mu^{|c_\circ^i|} \frac{|c_*^i \cap c_\circ^i|}{\max\left\{|c_*^i|, |c_\circ^i|, 1\right\}} + \gamma \mathbf{1}\{c_*^i = c_\circ^i = \emptyset\} \right) \\
&\approx \\
&\bigcup_{ijl|i\neq j}\{c_l^j\} \times \text{argmax}_{c_\circ^i \in \mathcal{P}(c^i)} \\
&\qquad \sum_{c_*^i \in \mathcal{P}(c^i)} \\
&\qquad \Big(\prod_{c_k^i} \big(P_\Lambda(c_k^i \rightsquigarrow c_l^j | C^i, C^j) \mathbf{1}\{c_k^i \in c_*^i\} + \\
&\qquad\qquad (1 - P_\Lambda(c_k^i \rightsquigarrow c_l^j | C^i, C^j)) \mathbf{1}\{c_k^i \notin c_*^i\} \big) \Big) \\
&\qquad \left(\mu^{|c_\circ^i|} \frac{|c_*^i \cap c_\circ^i|}{\max\left\{|c_*^i|, |c_\circ^i|, 1\right\}} + \gamma \mathbf{1}\{m_{\sim t}^{ij}(c_l^j) = c_\circ^i = \emptyset\} \right)
\end{aligned}
\tag{5.17}
$$

$$
\sim^p \triangleq \left(\rightsquigarrow^p \cup (\rightsquigarrow^p)^{-1} \right)^+ \tag{5.18}
$$

Note that although the directional posterior probabilities are used to generate the predicted MCA, the result is a many-to-many alignment, since the union is done over all pairs of sequences in both directions. The calculation in Equation (5.17) can be computationally intensive in practice, as it requires $|\mathcal{P}(c^i)|^2 = 2^{2n^i}$ operations for each word position c_l^j and citance C^i. This can be overcome by restricting the combinations of source word positions (c_*^i and $c_\circ^i$) to include only the the top MAX_SOURCES source words with a minimum posterior probability of MIN_PROB to align to c_l^j ($P_\Lambda(c_k^i \rightsquigarrow c_l^j | C^i, C^j) \geq$ MIN_PROB). In our implementation we set MAX_SOURCES

to 8 and MIN_PROB to 0.01. Additionally, the probabilities of each combination c^i_* can be calculated only once, since it is independent of $c^i_\circ$. This reduces the total computational complexity of calculating $\sim^p$ to $O\left(2^{16}(K^2 - K)\max_{n^i}\{n^i\}\right)$.

5.5 Data sets

One of the limitations of CRFs and other discriminative models is that they are typically required to be trained on labeled data. Additionally a development set is required for feature selection. Lastly, a held out test set is used for a formal evaluation of the final system's performance. Unlike MSA there has been no previous work done on the problem of MCA, thus there are no publicly available datasets that can be used to develop and evaluate a MCA algorithms. We therefore took upon us the task of creating the first datasets of manually-annotated reference MCAs.

We created two larger MCAs to be used as the development and test set, and four smaller MCAs for our training set. All six target papers where selected from the annotation reference of the Molecular Interaction Map (MIM) (Aladjem *et al.*, 2004) of DNA replication.[5] We decided to restrict the domain of the target papers to molecular interactions, due to the limited size of our date sets, and the limited resources we had for manual annotation. However, it is important to note that this domain is very actively researched in the biosciences text mining community (Hirschman *et al.*, 2002).

For each target paper we downloaded the full text of papers citing it that were available in HTML format. The link structure of the cited references in the HTML documents allowed us to automatically extract citances to a given target paper. For our purpose we define a citance to be the full sentence that contains a citation to

[5]http://discover.nci.nih.gov/mim/

the target paper. Each citance was then tokenized, and normalized by removing all stop-words from a predefined list. For the development set we used 51 citances that cite a single target paper. The test set included 45 citances of a second target paper. To increase the feature diversity in the training set we used 4 different target papers with 40 citances (10 each).

The six groups of citances were manually annotated by Dr. Anna Divoli. Within each group of citances words or phrases that share semantic similarity were annotated with identical identifiers. Using the manually annotated citance groups pairwise word alignments were generated for every source-target pair of citances from every group. That resulted in a training, development, and test sets of 180, 1275, and 990 pairwise alignments respectively. Alignments that were used for development and testing were generated as many-to-many alignments. However, using many-to-many alignments is not suitable for the training the one-to-many CRF alignment model. When a given source word c_k^i aligns to multiple words in the target citance the CRF model arbitrarily treats only one target word as a true-positive, while incorrectly treating the other target words as true-negatives. To alleviate this problem we replaced in such cases all true-positive target words except the first with '*', thus making them real true-negative for the purpose of training. This solution does not solve the inherent limitation of the CRF's one-to-many modeling of a many-to-many alignment, but it prevents learning wrong weights for good features that arbitrarily apply to true-positives that are treated as true-negative by the CRF.

5.6 Feature engineering

The CRF alignment model can combine multiple overlapping features. We evaluated the effectiveness of different features by training models on the training set and evaluating their performance on the development set. We considered variations of

features that were part of the original system of Blunsom and Cohn (2006), and also designed new features that are specific to the problem of MCA, and the bioscience domain.

Orthographic features

We used the following orthographic features from the original system;

- indicator for exact string similarity of source-target words,
- indicator for every possible source-target pair of length 3 word prefixes,
- indicator for exact string match of length 3 prefixes,
- indicator for exact string match of length 3 suffixes,
- absolute difference in word lengths,
- indicator that both words are shorter than 4 characters.

In addition, the following orthographic features were added;

- indicator that both words include capital letters,
- normalized edit-similarity of the two words $(1 - \frac{edit\_distance(c_k^i, c_l^j)}{max\{|c_k^i|, |c_l^j|\}})$.

Due to the small size of our training set we tried to remove features that are too specific and could lead to over-fitting, and unnecessary features that did not improve the performance on the development set. These features include;

- indicator for every possible source-target words,
- indicator for every possible source-target pair of length 3 word suffixes,

- prefix-suffix and suffice match,
- prefix match of lengths 4 and 5,
- un-normalized edit-distance.

Markov features

We used the following Markov features from the original system;

- absolute jump width ($abs(a_t - a_{t-1} - 1)$, which measures the distance between the target words of adjacent source words,
- positive jump width ($max\{a_t - a_{t-1} - 1, 0\}$),
- negative jump width ($max\{a_{t-1} + 1 - a_t, 0\}$),
- indicator for transition from null aligned source-word to non-null aligned source-word,
- indicator for transition from non-null aligned source-word to null aligned source-word,
- indicator for transition from null aligned source-word to null aligned source-word.

In addition we added the following Markov features in order to model the tendency of certain words to be part of longer phrases;

- indicator for every source-word string that is true when it aligns to the same target-word as the previous source-word,
- indicator for every source-word string that is true when it aligns to the same target-word as the next source-word,

- indicator for transition from non-null aligned source-word to non-null aligned source-word.

A general (not word specific) indicator for aligning a source-word to the same target-word as the previous source-word was evaluated but was not included in the final system.

Sentence position

We included the relative sentence position feature from the original system, which is defined as $abs(\frac{a_t}{|c^j|} - \frac{t}{c^i})$. Although it was not expected to be relevant for MCA, since the citances are not expected to align along the diagonal, this feature slightly improved the performance of the development set.

Null

An indicator function for leaving a source-word unaligned was retained from the original system. This is an essential feature since without it the CRF tends to over-align words, and produces meaningless posterior probabilities.

Ontological features

Orthographic and positional features alone do not cover all cases of semantic homology. We therefore included features that are based on domain specific ontologies.

Using an automated script we mapped specific words and phrases in every citance to MeSH[6] terms, Gene identifiers from Entrez Gene,[7] UniProt,[8] and OMIM.[9] We

[6]http://www.nlm.nih.gov/mesh/
[7]http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=gene
[8]http://www.pir.uniprot.org/
[9]http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=OMIM

then added features indicating when the source and target words are annotated with the same MeSH term or the same gene identifier. We tried numerous features that compare MeSH terms based on their distance in the ontology, and other features that indicate whether a word is part of a longer term. However, all these feature were not selected for the final system.

In addition to biological ontologies we added a feature for semantic word similarity between the source and target words, based on the Lin (1998) WordNet similarity measure.

5.7 Results

We modified the CRF alignment system of Blunsom and Cohn (2006) to support MCA, by incorporating the posterior decoding algorithm from Section 5.4.2 into the existing system. The CRF model was trained using the features that were selected using the development set, on a dataset that included the training and development MCAs. All the performance results in this section are reported on the test set, which includes 990 pairs of citances ($45 \times 44/2$), with a total of 34188 words (8547×44). On average, 20% of the source-words are aligned to at least one other target-word in a given reference pairwise alignment. Since the union of all the pairwise alignments results in only a single test MCA, it is hard to make strong arguments about the performance of the system in general. Therefore, we concentrate our discussion on general trends, and do not claim that the specific performance numbers we report here are statistically significant. It is interesting to note that the SMT community has been evaluating performance of word-alignment systems on an even smaller dataset of 447 pairs of (non-overlapping) sentences (Mihalcea and Pedersen, 2003).

We first analyze the performance of the system on pairwise citance alignments.

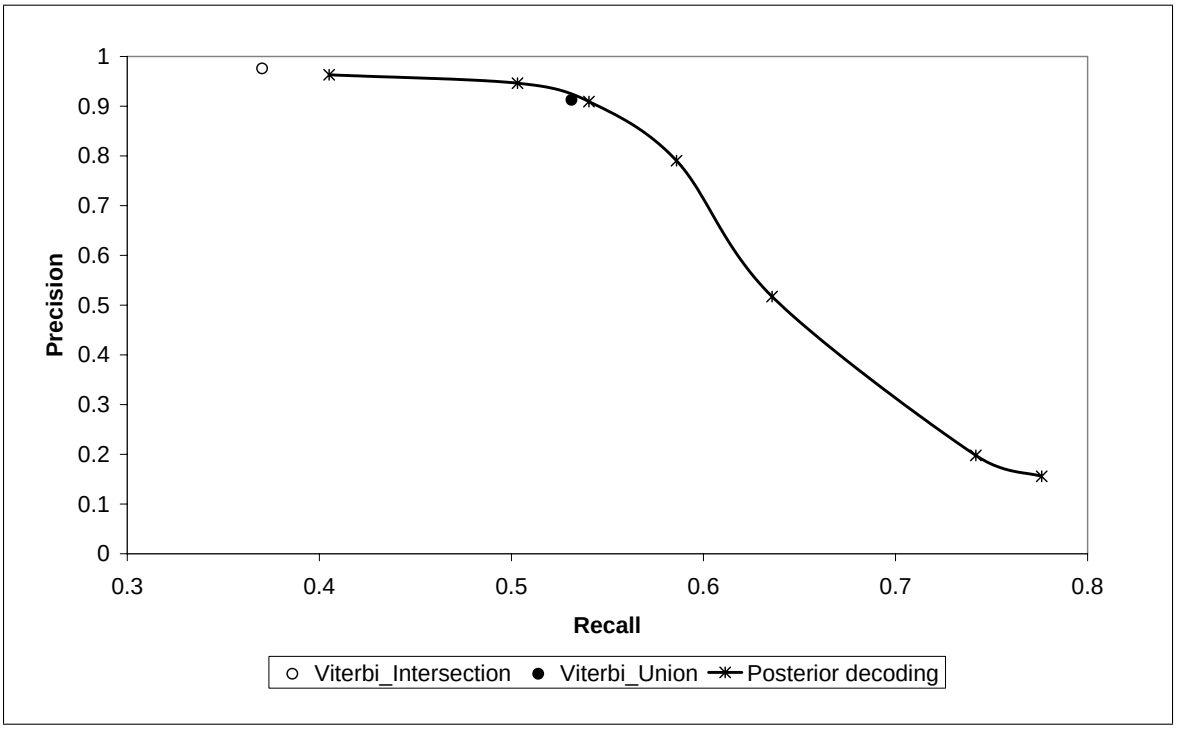


Figure 5.4. **Recall/Precision curve of pairwise citance alignments comparing Viterbi to posterior decoding.**

Instead of taking the equivalence closure of $\rightsquigarrow^p$ we take only the symmetric closure. The result is 990 many-to-many pairwise alignments. In order to evaluate the effectiveness of the posterior-decoding algorithm we generate the Viterbi alignments using the same CRF model. The Viterbi many-to-many pairwise alignments are then generated by combining equivalent pairs of one-to-many alignments using three different standard symmetrization methods for word-alignment—union, intersection, and the refined method of Och and Ney (2003).

Figure 5.4 shows the recall/precision trade-off of the pairwise posterior-decoding and Viterbi alignments. The curve for the posterior-decoding alignments was produced by varying the gap and match factors. For the Viterbi alignments only three results could be generated (one for each symmetrization method). However, since the refined method produced a very similar result to the union, only the union is displayed in the figure. The important observation is that while posterior-decoding

enables refined control over the recall/precision trade-off, the Viterbi decoding generates only three alignments, which cover only a small fraction of the curve at its high precision range. The union of Viterbi alignments achieves 0.531 recall at 0.913 precision, which is similar result to the 0.540 recall at 0.909 precision achieved using posterior-decoding with gap-factor and match-factor set to 1. However, unlike Viterbi, posterior-decoding produces alignments with much higher recall levels, by increasing the match-factor and decreasing the gap-factor. For example setting the gap-factor to 0.1 and match-factor to 1.2 results in alignments with 0.636 recall at 0.517 precision, and setting them to 0.05 and 1.5 results in 0.742 recall at 0.198 precision. Generally, the gap and match factor affect the accuracy of the alignments as expected. In particular, the alignments with the best AMA (0.889) and the best F_1-measure (0.678) are generated when the gap match factor are set to their natural values (1,1), which theoretically should maximize the expected AMA.

The performance of the pairwise alignments validates that the underlying probabilistic model behaves as theoretically expected. However the union of all pairwise alignments is not a valid MCA. For evaluating the MCA posterior decoding algorithm we compared it to baseline MCAs. The baseline MCAs are constructed by using only the normalized-edit-distance $\frac{edit\_distance(c_k^i, c_l^j)}{\max\{|c_k^i|, |c_l^j|\}}$, and defining $c_k^i \rightsquigarrow^\delta c_l^j$ if and only if $normalized\_edit\_distance(c_k^i, c_l^j) \leq \delta$, where δ is a distance threshold. The final baseline MCA is constructed by taking the equivalence closure of all pairwise alignments, $\sim^\delta \triangleq \left(\rightsquigarrow^\delta \cup (\rightsquigarrow^\delta)^{-1}\right)^+$. The δ parameter can be used to control the recall/precision trade-off, since increasing it adds more position-pairs to the alignment, thus increasing recall, while decreasing it increases precision.

Figures 5.5 compares the performance of the CRF posterior-decoding MCAs with the baseline MCAs. The different MCAs were produced by varying the gap and match factors in the case of the posterior-decoding, and δ for the baseline MCAs. The CRF curve clearly dominates the baseline curve. However, they do overlap in range between

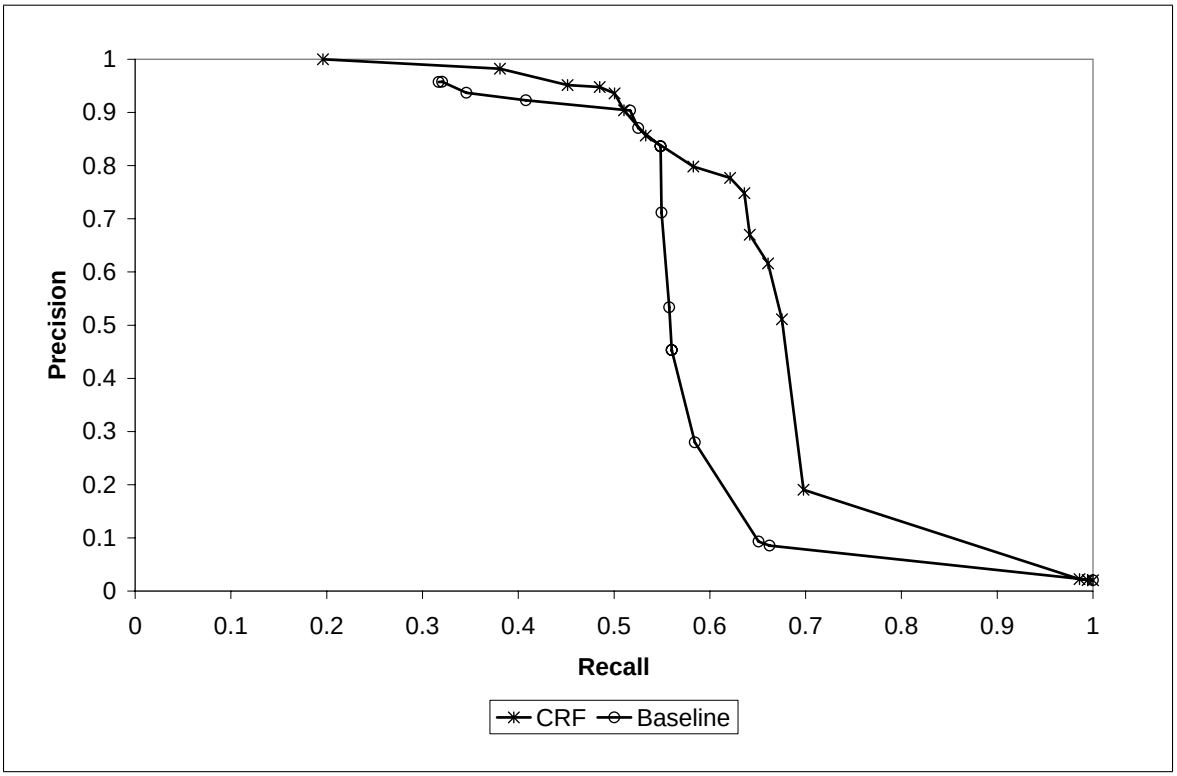


Figure 5.5. **Recall/Precision curve of MCAs comparing CRF with posterior decoding to normalized-edit-distance baseline.**

0.52 and 0.55 recall (0.84 and 0.90 precision). This is probably a range in which for this particular MCA the orthographic similarity is the most dominant feature. While the baseline curve drops sharply after that range, the posterior-decoding curve keeps improving recall up to 0.636 at 0.748 precision, before there is a major drop in precision. The additional recall is due to the ability of the CRF model to incorporate multiple overlapping features. In particular, the domain-specific features are important for aligning words and phrases that have little or no orthographic similarity. At the other end of the overlap range, the posterior-decoding achieves better precision than the baseline for the same recall levels. For example, the posterior decoding gets 0.381 recall at 0.982 precision compared with 0.346 at 0.937 for the baseline.

Unlike the pairwise alignment case, the neutral settings of the gap and match factors did not result in the best AMA score. This is due to the equivalence closure

heuristic that results in MCAs that are too dense, since a single link between two equivalence classes causes them to merge. The best AMA score (0.886) is obtained by reducing the gap-factor to 0.5 and match-factor to 0.45, in order to compensate for the effect of the equivalence closure heuristic. For comparison, the best F_1-measure (0.690) is achieved by setting the gap and match factors to 0.75.

5.8 Discussion

The main purpose of this chapter was to investigate how the posterior-decoding methods that we have developed for MSA can be applied to related, but different domains. We made a first attempt at defining the problem of MCA, which is motivated by recent work in biosciences text mining. By looking at a new problem, we were free to select our accuracy measures and utility functions. We explored the pros and cons of different utility functions, and were able to modify the AMA for one-to-one MSAs to apply to the many-to-many MCAs. The main design principle was that the utility function should decompose well over the basic elements for which we can compute posterior probabilities.

Since the sequence-annealing algorithm for MSA cannot be applied directly to MCA, we showed how to derive a posterior-decoding algorithm that aims at maximizing the expected utility. Adding a gap and match factor to the utility function enabled to control the recall/precision trade-off using posterior-decoding.

Another advantage of optimizing the expected utility with posterior-decoding methods is that they are decoupled from the probabilistic model that generate the posterior probabilities. Therefore, we were able to use CRFs instead of HMMs in the case of MCA with little change to the posterior decoding algorithm.

Our experiments were limited by the size of the training and test labeled data.

However, the results support the theoretical predictions, and demonstrate the advantage of posterior-decoding over Viterbi decoding.

Since citances are still a relatively unexplored resource, it is still unclear whether the formulation we presented here for citance alignment is the most useful for applications that use citances for comparative analysis of bioscience text. Unlike biological sequence alignment, citance alignments are much more subjective since they depend on a loose definition of semantic homology between entities. Even the definition of the basic entities can vary, since in many cases noun-compounds and other multi-word entities seem to be a more natural choice for basic elements of semantic homology and alignment. However, automatic segmentation and entity recognition are still difficult tasks in the bioscience text domain.

Chapter 6

Conclusions

In this work we have explored methods for optimization and control of alignment accuracy. A combination of posterior decoding methods and careful definition of accuracy measures and utility functions for multiple sequence alignment and citance alignment lead to improved overall alignment accuracy, and enables direct control of the recall/precision trade-off.

Although our work focused on multiple alignments, methods similar to the ones we have developed can be applied to other problems in related areas, such as biological network alignment, phylogeny reconstruction, comparative gene fining, alignment for statistical machine translation, and alignment of HTML pages.

Our proposed framework includes the following components: (i) definition of utility functions that decompose well over the basic elements of the problem. Although not a necessary requirement, deriving utility functions from metrics has many advantages over non-metric based functions; (ii) a tuning parameter for control of the recall/precision trade-off; (iii) a probabilistic model that enables efficient computation of marginal posterior probabilities; (iv) a posterior decoding algorithm that uses the posterior probabilities in order to maximize the expected utility.

There are many directions in which the current work can be extended. One direction for future research includes enhancing the utility functions that are being optimized. For example, different sequence elements can get different weights based on their predefined class. In the case of biological sequence alignment, positions that correspond to loop regions in the secondary and tertiary structure of a protein can get lower weight in a utility function without invalidating the metric properties of that function. Similarly, in the case of word alignment, different part-of-speech tags, or entity types can be assigned different weights.

Another extension includes setting the pairwise model parameters differently for different pairs of sequences. In particular, when comparing biological sequences with varying evolutionary distances, one might try to adjust the model's parameters for every pairwise posterior probabilities computation, based on the evolutionary distance of the compared sequences.

Improved methods for exploring the solution space can be developed. For example, when aligning nucleotide sequences, a simple hill climbing procedure might get stuck in a local minima and could benefit from the addition of simulated annealing steps. The search in solution space can also benefit from bounds that can be developed using the metric properties of the utility functions.

Posterior decoding techniques have many advantages over the more common maximum likelihood inference algorithms. In cases when the correct solution is trivial both methods find it. However, in more challenging situations where many plausible solutions exist, posterior decoding methods are more flexible, and can be used to maximize expectations of utility functions that are much more useful than the standard 0–1 loss function.

Bibliography

Ajwani, D., Friedrich, T., and Meyer, U., 2006. An $o(n^{2.75})$ algorithm for online topological ordering. *arXiv:cs.DS/0602073* .

Aladjem, M. I., Pasa, S., Parodi, S., Weinstein, J. N., Pommier, Y., and Kohn, K. W., 2004. Molecular Interaction Maps–A Diagrammatic Graphical Language for Bioregulatory Networks. *Sci. STKE* 2004, pe8–.

Alexandersson, M., Bray, N., and Pachter, L., 2005. Pair hidden Markov models. In Jorde, L. B., Little, P., Dunn, M., and Subramanian, S., eds., *Encyclopedia of Genetics, Genomics, Proteomics and Bioinformatics*.

Alpern, B., Hoover, R., Rosen, B. K., Sweeney, P. F., and Zadeck, F. K., 1990. Incremental evaluation of computational circuits. In *Proceedings 1st Annual ACM-SIAM Symposium on Discrete Algorithms*, 32–42.

Barzilay, R. and Lee, L., 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of HLT-NAACL.*, 16–23.

Barzilay, R. and McKeown, K., 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of ACL.*, 50–57.

Batzoglou, S., 2005. The many faces of sequence alignment. *Brief. Bioinform.* 6, 6–22.

Baum, L. E. and Petrie, T., 1966. Statistical inference for probabilistic functions of finite state markov chains. *Ann.Math.Stat.* 37, 1554–1563.

Baum, L. E., Petrie, T., Soules, G., and Weiss, N., 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics* 41, 164–171.

Blanchette, M., Kent, W. J., Riemer, C., Elnitski, L., Smit, A. F. A., Roskin, K. M., Baertsch, R., Rosenbloom, K., Clawson, H., Green, E. D., Haussler, D., and Miller, W., 2004. Aligning multiple genome sequences with the threaded blockset aligner. *Genome Research* 14, 708–715.

Blunsom, P. and Cohn, T., 2006. Discriminative word alignment with conditional random fields. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, 65–72. Association for Computational Linguistics, Sydney, Australia.

Bonizzoni, P. and Vedova, G. D., 2001. The complexity of multiple sequence alignment with SP-score that is a metric. *Theoretical Computer Science* 259, 63–79.

Bradshaw, S., 2003. Reference directed indexing: Redeeming relevance for subject search in citation indexes. In *Proceedings of the 7th European Conference on Research and Advanced Technology for Digital Libraries*.

Braun-Blanquet, J., 1932. *Plant sociology: the study of plant communities*. McGraw-Hill, New York.

Brenner, S. E., Koehl, P., and Levitt, M., 2000. The ASTRAL compendium for protein structure and sequence analysis. *Nucl. Acids Res.* 28, 254–256.

Brown, P. F., Cocke, J., Della Pietra, S. A., Della Pietra, V. J., Jelinek, F., Lafferty, J. D., Mercer, R. L., and Roossin, P. S., 1990. A statistical approach to machine translation. *Computational Linguistics* 16, 79–85.

Caillez, F. and Kuntz, P., 1996. A contribution to the study of the metric and euclidean structures of dissimilarities. *Psychometrika* 61, 241–253.

Cawley, S. L. and Pachter, L., 2003. HMM sampling and applications to gene finding and alternative splicing. *Bioinformatics* 19, ii36–41.

Dempster, A. P., Laird, N. M., and Rubin, D. B., 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)* 39, 1–38.

Dice, L. R., 1945. *Ecology* 26, 297–302.

Do, C. B., Mahabhashyam, M. S. P., Brudno, M., and Batzoglou, S., 2005. ProbCons: Probabilistic consistency-based multiple sequence alignment. *Genome Res.* 15, 330–340.

Domingos, P., 2000. A unified bias-variance decomposition and its applications. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 231–238. Morgan Kaufmann, Stanford, CA.

Dress, A., Morgenstern, B., and Stoye, J., 1998. The number of standard and of effective multiple alignments. *Appl. Math. Lett.* 11, 43–49.

Durbin, R., Eddy, S., Krogh, A., and Mitchison, G., 1998. *Biological sequence analysis. Probablistic models of proteins and nucleic acids*. Cambridge University Press.

Edgar, R. C., 2004. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucl. Acids Res.* 32, 1792–1797.

Elias, I., 2006. Settling the intractability of multiple alignment. *Journal of Computational Biology* 13, 1323–1339.

Falck, J., Mailand, N., Syljuasen, R. G., Bartek, J., and Lukas, J., 2001. The ATM-Chk2-Cdc25A checkpoint pathway guards against radioresistant DNA synthesis. *Nature* 410, 842–847.

Feng, D. F. and Doolittle, R. F., 1987. Progressive alignment of amino acid sequences as a prerequisite to correct phylogenetic trees. *J. Mol. Evol.* 25, 351–360.

Fraser, A. and Marcu, D., 2006. Measuring word alignment quality for statistical machine translation. Technical Report ISI-TR-616, ISI/University of Southern California.

Gotoh, O., 1996. Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments. *J. Mol. Biol.* 264, 823–838.

Gower, J. C. and Legendre, P., 1986. Metric and euclidean properties of dissimilarity coefficients. *Journal of Classification* 3, 5–48.

Grefenstette, G., 1992. Sextant: Exploring unexplored contexts for semantic extraction from syntactic analysis. In *Proceedings of ACL*, 324–326.

Grefenstette, G., 1994. *Explorations in Automatic Thesaurus Discovery.* Kluwer Academic Publishers.

Hamming, R. W., 1950. Error detecting and error correcting codes. *Bell Systems Technical Journal* 29, 147–160.

Hirschman, L., Park, J. C., Tsujii, J., Wong, L., and Wu, C. H., 2002. Accomplishments and challenges in literature data mining for biology. *Bioinformatics* 18, 1553–1561.

Holmes, I. and Durbin, R., 1998. Dynamic programming alignment accuracy. *J. Comp. Biol.* 5, 493–504.

Jaccard, P., 1901. *Bulletin del la Société Vaudoisedes Sciences Naturelles* 37, 241–272.

Just, W., 2001. Computational complexity of multiple sequence alignment with sp-score. *Journal of Computational Biology* 8, 615–623.

Katriel, I. and Bodlaender, H. L., 2006. Online topological ordering. *ACM Transactions on Algorithms* in press.

Lacoste-Julien, S., Taskar, B., Klein, D., and Jordan, M. I., 2006. Word alignment via quadratic assignment. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, 112–119. Association for Computational Linguistics, New York City, USA.

Lafferty, J., McCallum, A., and Pereira, F., 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, 282–289. Morgan Kaufmann, San Francisco, CA.

Lecandowsky, M. and Winter, D., 1971. Distance between sets. *Nature* 234, 34–35.

Lee, C., Grasso, C., and Sharlow, M. F., 2002. Multiple sequence alignment using partial order graphs. *Bioinformatics* 18, 452–464.

Levenshtein, V. I., 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics - Doklady* 10, 707–710. Translated from Doklady Akademii Nauk SSSR, Vol. 163 No. 4 pp. 845-848, August 1965.

Liang, P., Taskar, B., and Klein, D., 2006. Alignment by agreement. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, 104–111. Association for Computational Linguistics, New York City, USA.

Lin, D., 1998. An information-theoretic definition of similarity. In *Proc. 15th International Conf. on Machine Learning*, 296–304. Morgan Kaufmann, San Francisco, CA.

Lin, D. and Pantel, P., 2001. Discovery of inference rules for question answering. *Natural Language Engineering* 7(4), 343–360.

Marchetti-Spaccarnela, A., Nanni, U., and Rohnert, H., 1996. Maintaining a topological order under edge insertions. *Information Processing Letters* 59, 53–58.

Matusov, E., Zens, R., and Ney, H., 2004. Symmetric word alignments for statistical machine translation. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, 219. Association for Computational Linguistics, Morristown, NJ, USA.

Mercer, R. E. and Marco, C. D., 2004. A design methodology for a biomedical literature indexing tool using the rhetoric of science. In *BioLink workshop in conjunction with NAACL/HLT*, 77–84.

Mihalcea, R. and Pedersen, T., 2003. An evaluation exercise for word alignment. In Mihalcea, R. and Pedersen, T., eds., *HLT-NAACL 2003 Workshop: Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, 1–10. Association for Computational Linguistics, Edmonton, Alberta, Canada.

Miller, W., 2000. comparison of genomic sequences: Solved and unsolved problems. *Bioinformatics* 17, 391–397.

Moore, R. C., 2005. A discriminative framework for bilingual word alignment. In *HLT/EMNLP*, 81–88.

Morgenstern, B., Dress, A., and Werner, T., 1996. Multiple DNA and protein sequence alignment based on segment-to-segment comparison. *PNAS* 93, 12098–12103.

Morgenstern, B., Frech, K., Dress, A., and Werner, T., 1998. DIALIGN: Finding local similarities by multiple sequence alignment. *Bioinformatics* 14, 290–294.

Morgenstern, B., Stoye, J., and Dress, A., 1999. Consistent equivalence relations: a set-theoretical framework for multiple sequence alignment. Technical Report Materialien und Preprints 133, University of Bielefeld.

Murzin, A. G., Brenner, S. E., Hubbard, T., and Chothia, C., 1995. SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.* 247, 536.

Nakov, P. I., Schwartz, A. S., and Hearst, M. A., 2004. Citances: Citation sentences for semantic analysis of bioscience text. In *SIGIR'04 Workshop on Search and Discovery in Bioinformatics*.

Nanba, H., Kando, N., and Okumura, M., 2000. Classification of research papers using citation links and citation types: Towards automatic review article generation. In *American Society for Information Science SIG Classification Research Workshop: Classification for User Support and Learning*, 117–134.

Needleman, S. B. and Wunsch, C. D., 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology* 48, 443–453.

Notredame, C., Higgins, D., and Heringa, J., 2000. T-Coffee: A novel method for multiple sequence alignments. *J. Mol. Biol.* 302, 205–217.

Och, F. J. and Ney, H., 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics* 29, 19–51.

Pachter, L. and Sturmfels, B., eds., 2005. *Algebraic Statistics for Computational Biology.* Cambridge University Press.

Pang, B., Knight, K., and Marcu, D., 2003. Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In *Proceedings of HLT-NAACL*, 181–188.

Paten, B., 2005. Http://www.ebi.ac.uk/∼bjp/pecan/.

Pearce, D. J. and Kelly, P. H. J., 2004. A dynamic algorithm for topologically sorting directed acyclic graphs. In *Proceedings of the Workshop on Efficient and experimental Algorithms*, *Lecture Notes in Computer Science*, volume 3059, 383–398. Springer-Verlag.

Rabiner, L., 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77, 257–286.

Raphael, B., Zhi, D., Tang, H., and Pevzner, P., 2004. A novel method for multiple alignment of sequences with repeated and shuffled elements. *Genome Res.* 14, 2336–2346.

Rota, G.-C., 1964. The number of partitions of a set. *The American Mathematical Monthly* 71, 498–504.

Sauder, J. M., Arthur, J. W., and Dunbrack, R. L., 2000. Large-scale comparison of protein sequence alignment algorithms with structure alignments. *Proteins* 40, 6–22.

Schlüter, R., Scharrenbach, T., Steinbiss, V., and Ney, H., 2005. Bayes risk minimization using metric loss functions. In *Proceedings of the European Conference on Speech Communication and Technology, Interspeech*, 1449–1452. Portugal.

Schwartz, A. S. and Pachter, L., 2007. Multiple alignment by sequence annealing. *Bioinformatics* 23, e24–29.

Shinyama, Y. and Sekine, S., 2003. Paraphrase acquisition for information extraction. In *Proceedings of Second International Workshop on Paraphrasing (IWP2003).*

Shinyama, Y., Sekine, S., Sudo, K., and Grishman, R., 2002. Automatic paraphrase acquisition from news articles. In *Proceedings of HLT*, 40–46.

Spiro, P. A. and Macura, N., 2004. A local alignment metric for accelerating biosequence database search. *Journal of Computational Biology* 11, 61–82.

Subramanian, A. R., Weyer-Menkhoff, J., Kaufmann, B., and Morgenstern, B., 2005. DIALIGN-T: An improved algorithm for segment-based multiple alignment. *BMC Bioinformatics* 6, 66.

Sze, S.-H., Lu, Y., and Yang, Q., 2005. A polynomial time solvable formulation of multiple sequence alignment. *Lecture Notes in Computer Science* 3500, 204–216.

Tarjan, R. E., 1972. Depth first search and linear graph algorithms. *SIAM Journal on Computing* 1, 146–160.

Thompson, J. D., Higgins, D. G., and Gibseon, T. J., 1994. CLUSTALW: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucl. Acids Res.* 22, 4673–4680.

Van Walle, I., Lasters, I., and Wyns, L., 2005. SABmark–a benchmark for sequence alignment that covers the entire known fold space. *Bioinformatics* 21, 1267–1268.

Viterbi, A. J., 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory* IT-13, 260–269.

Wang, L. and Jiang, T., 1994. On the complexity of multiple sequence alignment. *Journal of Computational Biology* 1, 337–348.

Appendix A

Proof that the triangle-inequality holds for the Braun-Blanquet coefficient

Let A, B, and C be three sets. The distance d between sets as derived from the Braun-Blanquet coefficient is

$$d(A,B) = 1 - \frac{|A \cap B|}{\max\{|A|, |B|, 1\}} - \mathbb{1}_{\not<}\{A = \emptyset \wedge B = \emptyset\} \tag{A.1}$$

Without loss of generality let $|A| \geq |B| \geq |C| > 0$.[1] We need to prove:

(i) $d(A,B) + d(B,C) - d(A,C) \geq 0$.

(ii) $d(A,B) + d(A,C) - d(B,C) \geq 0$.

(iii) $d(B,C) + d(A,C) - d(A,B) \geq 0$.

[1] All cases are trivial when one or more of the sets is empty. We therefore assume non-empty sets.

Proof of (i)

$$
\begin{aligned}
& d(A,B) + d(B,C) - d(A,C) = \\
& 1 + \frac{|A \cap C| - |A \cap B|}{|A|} - \frac{|B \cap C|}{|B|} = \\
& 1 + \frac{\overbrace{|A \cap C| - |A \cap C \cap B|}^{\geq 0} - |A \cap B| + |A \cap C \cap B|}{|A|} - \frac{|B \cap C|}{|B|} \geq \\
& 1 - \frac{\overbrace{|A \cap B| - |A \cap C \cap B|}^{\geq 0}}{\underbrace{|A|}_{\geq |B|}} - \frac{|B \cap C|}{|B|} \geq \\
& 1 - \frac{|A \cap B| + |B \cap C| - |A \cap C \cap B|}{|B|} \geq \\
& 1 - \frac{B}{B} = 0. \quad \square
\end{aligned}
$$

Proof of (ii)

$$
\begin{aligned}
& d(A,B) + d(A,C) - d(B,C) = \\
& 1 - \frac{|A \cap B| + |A \cap C|}{|A|} + \frac{|B \cap C|}{\underbrace{|B|}_{\leq |A|}} \geq \\
& 1 - \frac{|A \cap B| + |A \cap C| - |B \cap C|}{|A|} \geq \\
& 1 - \frac{|A \cap B| + |A \cap C| - |A \cap B \cap C|}{|A|} \geq \\
& 1 - \frac{|A|}{|A|} = 0. \quad \square
\end{aligned}
$$

Proof of (iii)

$$
\begin{aligned}
& d(B,C) + d(A,C) - d(A,B) = \\
& 1 + \frac{|A \cap B| - |A \cap C|}{|A|} - \frac{|B \cap C|}{|B|} = \\
& 1 + \frac{\overbrace{|A \cap B| - |A \cap B \cap C|}^{\geq 0} - |A \cap C| + |A \cap B \cap C|}{|A|} - \frac{|B \cap C|}{|B|} \geq \\
& 1 - \frac{\overbrace{|A \cap C| - |A \cap B \cap C|}^{\geq 0}}{\underbrace{|A|}_{\geq |C|}} - \frac{|B \cap C|}{\underbrace{|B|}_{\geq |C|}} \geq \\
& 1 - \frac{|A \cap C| + |B \cap C| - |A \cap B \cap C|}{|C|} \geq \\
& 1 - \frac{|C|}{|C|} = 0. \quad \square
\end{aligned}
$$